ST

TEXAS INSTRUMENTS

Improving Man's Effectiveness Through Electronics

Model 990 Computer

Prototyping System Documentation

VOLUME 1 OF 2

PART NO. 943380-0019

Digital Systems Division

TEXAS INSTRUMENTS
INCORPORATED

DATE 05/25/76

LIST OF MATERIAL

PART NUMBER LM 943518-0019 REV *

PRINT ITEM NUMBER	QUANTITY PER ASSEMBLY	UNIT OF ISSUE	DWG. SIZE	PART NUMBER	DESCRIPTION	VENDOR PART NUMBER
C001	00001.000	EA		943518-9901	LML,PX9MTP,PROTO SYS DEBUG MONITOR-PX990	
0002	00001.000	EA		943519-9901	AL, INITIP,INITIALIZATION,PROTO-PX990	
C002A					PX9MTP-ROOT	
0002B					MUST BE FIRST IN PX9MTP-ROOT	
C003	00001.000	EA		945354-9901	AL, ERRINT,ERROR INTERRUPT PROC-PX990	
C003A					PX9MTP-ROOT	
C004	00001.000	EA		945382-9901	AL, WKSPMG,WORKSPACE MANAGEMENT-PX990	
0004A					PX9MTP-ROOT	
C005	00001.000	EA		945330-9901	AL, CMSTRF,CMD STRING PROC,PROTO-PX990	
0005A					PX9MTP-ROOT	
C006	00001.000	EA		945369-9901	AL, PCOUT,OUTPUT DATA,FRONT PANEL-PX990	
0006A					PX9MTP-ROOT	
0007	00001.000	EA		945356-9901	AL, GTFELD,INPUT CHARACTER STRING-PX990	
0007A					PX9MTP-ROOT	
C008	00001.000	EA		945357-9901	AL, GTHEX,INPUT HEXADECIMAL NUMBER-PX990	
0008A					PX9MTP-ROOT	
C009	00001.000	EA		945381-9901	AL, SUPVSR,SUPERVISOR CALL I/F-PX990	
0009A					PX9MTP-ROOT	
0010	00001.000	EA		945352-9901	AL, CONVRT,ASCII/BINARY CONVERSION-PX990	
C010A					PX9MTP-ROOT	

DRAFTSMAN	DATE	CKD. DRAFTSMAN	DATE	DESIGN ENGINEER	DATE	TITLE
L. GARZA	5/25/76	L. GARZA	5/25/76			SP,PX9MTP LINKABLE PARTSIAL I-PX990

APPD.-MFG.	DATE	APPD. PROJECT ENGINEER	DATE	RELEASED	DATE	PROJECT NO.		PART NUMBER	REV
		M A Mota	6-4-76	D DM	6/15/76	7506		LM 943518-0019	*

T.I. 13849

PRINT ITEM NUMBER	QUANTITY PER ASSEMBLY	UNIT OF ISSUE	DWG. SIZE	PART NUMBER	DESCRIPTION	VENDOR PART NUMBER
0011	00001.000	EA		945346-9901	AL, ASGLUN,ASSIGN LUNO-PX990	
C011A					PX9MTP-RCCT	
0012	00001.000	EA		945348-9901	AL, CHARIA,CHARACTER INPUT-PX990	
C012A					PX9MTP-RCCT	
0013	00001.000	EA		945383-9901	AL, USRPGM,USER PROG SPVSR CALLS-PX990	
C013A					PX9MTP-RCCT	
C014	00001.000	EA		945374-9901	AL, PX9IC,I/O DISPATCHER-PX990	
0014A					PX9MTP-RCCT	
0015	00001.000	EA		945353-9901	AL, CSR733,733ASR DEVICE DRIVER-PX990	
0015A					PX9MTP-RCCT	
0016	00001.000	EA		945377-9901	AL, SB,SET BREAKPOINT-PX990	
0016A					PX9MTP-RCCT	
0017	00001.000	EA		945375-9901	AL, RANGE,EXTRACT RANGE/PARAMETERS-PX990	
0017A					PX9MTP-RCCT	
C018	00001.000	EA		945360-9901	AL, IP,INSPECT MEMORY-PX990	
C018A					PX9MTP-RCCT	
C019	00001.000	EA		945359-9901	AL, IC,INSPECT CRU PROCESSOR-PX990	
0019A					PX9MTP-RCCT	
C020	00001.000	EA		945363-9901	AL, INWKSP,INSPECT WORKSPACE-PX990	
0020A					PX9MTP-RCCT	

DRAFTSMAN	DATE	CKD. DRAFTSMAN	DATE	DESIGN ENGINEER	DATE	TITLE
						SP,PX9MTP LINKABLE PARTS(AL)-PX990

APPD.-MFG.	DATE	APPD. PROJECT ENGINEER	DATE	RELEASED	DATE	PROJECT NO.		PART NUMBER	REV
								LM 943518-0019	*

TEXAS INSTRUMENTS
INCORPORATED

DATE 05/25/76

LIST OF MATERIAL

PART NUMBER LM 943518-0019 REV *

PRINT ITEM NUMBER	QUANTITY PER ASSEMBLY	UNIT OF ISSUE	DWG. SIZE	PART NUMBER	DESCRIPTION	VENDOR PART NUMBER
0021	00001.000	EA		945362-9901	AL, INSPSS,INSPECT SNAPSHOT-PX990	
0021A					PX9MTP-RCCT	
0022	00001.000	EA		945365-9901	AL, MCDME,MODIFY MEMORY-PX990	
0022A					PX9MTP-RCCT	
0023	00001.000	EA		945364-9901	AL, MODCR,MODIFY CRU-PX990	
0023A					PX9MTP-RCCT	
0024	00001.000	EA		945366-9901	AL, MCDREG,MODIFY REGISTERS-PX990	
0024A					PX9MTP-RCCT	
0025	00001.000	EA		945367-9901	AL, MCDWP,MODIFY WORKSPACE-PX990	
0025A					PX9MTP-RCCT	
0026	00001.000	EA		945370-9901	AL, PRTMMC,PRINT MEMORY,CRU RANGES-PX990	
0026A					PX9MTP-RCCT	
0027	00001.000	EA		945371-9901	AL, PRTSSN,PRINT SNAPSHOT-PX990	
0027A					PX9MTP-RCCT	
0028	00001.000	EA		945373-9901	AL, PXLDCR,LOADER DRIVER-PX990	
0028A					PX9MTP-RCCT	
0029	00001.000	EA		945368-9901	AL, OVERLAY,OVERLAY PROCESSOR-PX990	
0029A					PX9MTP-RCCT	
0030	00001.000	EA		945369-9901	AL, PRCMLD,LOAD PROM PROGRAMMER-PX990	
0030A					PX9MTP-RCCT	

DRAFTSMAN	DATE	CKD. DRAFTSMAN	DATE	DESIGN ENGINEER	DATE	TITLE
						SP,PX9MTP LINKABLE PARTS(AL I-PX990

PD.-MFG.	DATE	APPD. PROJECT ENGINEER	DATE	RELEASED	DATE	PROJECT NO.		PART NUMBER	REV
								LM 943518-0019	*

J. 13849

PRINT ITEM NUMBER	QUANTITY PER ASSEMBLY	UNIT OF ISSUE	DWG. SIZE	PART NUMBER	DESCRIPTION	VENDOR PART NUMBER
0031	00001.000	EA		945380-9901	AL, SSNAP,SET SNAPSHOT-PX990	
C031A					PX9MTP-ROCT	
0032	00001.000	EA		945378-9901	AL, SCNTBL,SCAN TABLES-PX990	
C032A					PX9MTP-ROCT	
0033	00001.000	EA		945326-9901	AL, RUNPGP,USER PROG CONTROL,PROTO-PX990	
C033A					PX9MTP-ROCT	
0034	00001.000	EA		945379-9901	AL, SIEPR,SIE DRIVER-PX990	
0034A					PX9MTP-ROCT	
C035	00001.000	EA		945347-9901	AL, BKPPR,BREADPOINT PROCESSOR-PX990	
0035A					PX9MTP-ROCT	
0036	00001.000	EA		945349-9901	AL, CLEAR,CLEAR COMMAND PROCESSOR-PX990	
0036A					PX9MTP-ROCT	
0037	00001.000	EA		945355-9901	AL, FIND,SCAN MEMORY FOR VALUES-PX990	
C037A					PX9MTP-ROCT	
0038	00001.000	EA		945358-9901	AL, HXARTH,HEXADECIMAL ARITHMETIC-PX990	
C038A					PX9MTP-ROCT	
0039	00001.000	EA		943524-9901	AL, WRTPRT,WRITE PROTECT-PX990	
C039A					PX9MTP-ROCT	
0040	00001.000	EA		943523-9901	AL, FPDEFP,FRONT PANEL DEF'S,PROTO-PX990	
C040A					PX9MTP-ROCT	

DRAFTSMAN	DATE	CKD. DRAFTSMAN	DATE	DESIGN ENGINEER	DATE	TITLE
						SP,PX9MTP LINKABLE PARTSIAL 1-PX990

APPD.-MFG.	DATE	APPD. PROJECT ENGINEER	DATE	RELEASED	DATE	PROJECT NO.		PART NUMBER	REV
								LM 943518-0019	*

TEXAS INSTRUMENTS
INCORPORATED

DATE 05/25/76

LIST OF MATERIAL

PART NUMBER LM 943518-0019 REV *

PRINT ITEM NUMBER	QUANTITY PER ASSEMBLY	UNIT OF ISSUE	DWG. SIZE	PART NUMBER	DESCRIPTION	VENDOR PART NUMBER
0041	00001.000	EA		945350-9901	AL, CMDDEF,COMMAND DEFINITIONS-PX990	
0041A					PX9MTP-RCCT	
0041B					MUST BE NEXT-TO-LAST IN	
C041C					PX9MTP-RCCT	
0042	00001.000	EA		945372-9901	AL, PXCATA,READ/WRITE DATA AREA-PX990	
C042A					PX9MTP-ROOT	
0042B					MUST BE LAST IN PX9MTP-ROOT	
0043	00001.000	EA		945388-9901	AL, ABSCMF,DUMP ABSOLUTE-PX990	
0043A					PX9MTP-ABS OVERLAY	
0044	00001.000	EA		945389-9901	AL, ABSLC,ABSOLUTE LOADER-PX990	
0044A					PX9MTP-ABS OVERLAY	
C045	00001.000	EA		945387-9901	AL, TRACEMOD,TRACE INTERPRETER-PX990	
C045A					PX9MTP-TRACE OVERLAY	
0046	00001.000	EA		945385-9901	AL, SETREG,SET TRACE REGION-PX990	
0046A					PX9MTP-TRACE OVERLAY	
0047	00001.000	EA		945386-9901	AL, SETRACE,SET TRACE FORMAT-PX990	
C047A					PX9MTP-TRACE OVERLAY	
0048	00001.000	EA		945390-9901	AL, PXLAL,LINKING LOADER MODULE-PX990	
0048A					PX9MTP-LAL OVERLAY	
0049	00001.000	EA		945391-9901	AL, PROMPG,PROM PROGRAMMER,PART 1-PX990	

DRAFTSMAN	DATE	CKD. DRAFTSMAN	DATE	DESIGN ENGINEER	DATE	TITLE
						SP,PX9MTP LINKABLE PARTS(AL 1-PX990

APPD.-MFG.	DATE	APPD. PROJECT ENGINEER	DATE	RELEASED	DATE	PROJECT NO.		PART NUMBER	REV
								LM 943518-0019	*

T.I. 13849

TEXAS INSTRUMENTS
INCORPORATED

DATE 05/25/76

LIST OF MATERIAL

PART NUMBER LM 943518-0019 *

PRINT ITEM NUMBER	QUANTITY PER ASSEMBLY	UNIT OF ISSUE	DWG. SIZE	PART NUMBER	DESCRIPTION	VENDOR PART NUMBER
C049A					PX9MTP-PRPRG OVERLAY	
0050	00001.000	EA		945357-9901	AL, CMBNPF,DUMP BNPF-PX990	
C050A					PX9MTP-BNPF OVERLAY	
0051	00001.000	EA		945325-9901	AL, CMHL,DUMP HI-LO,PROTO-PX990	
C051A					PX9MTP-HILO OVERLAY	

DRAFTSMAN	DATE	CKD. DRAFTSMAN	DATE	DESIGN ENGINEER	DATE	TITLE
						SP,PX9MTP LINKABLE PARTS(AL)-PX990

APPD.-MFG.	DATE	APPD. PROJECT ENGINEER	DATE	RELEASED	DATE	PROJECT NO.		PART NUMBER	REV
								LM 943518-0019	*

T.I. 13849

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																
SHEET																																
REV STATUS OF SHEETS	REV																															
	SHEET																															

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DATE
DWN	
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

LML, PROTOTYPING SYSTEM DEBUG MONITOR-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	943518-9901

SCALE | REV. | SHEET 1 of 12

TI-0023-C

```
 MEMORY 10000                                                    943518 00000010
 PHASE 0,PX9MTP                                                  943518 00000020
 INCLUDE                                                         943518 00000030
     INITIP          MUST BE FIRST MODULE                        943518 00000040
     ERRINT                                                      943518 00000050
     WKSPMG                                                      943518 00000060
     CMSTPP                                                      943518 00000070
     PCOUT                                                       943518 00000080
     GTFELD                                                      943518 00000090
     GTHEX                                                       943518 00000100
     SUPVSR                                                      943518 00000110
       CONVRT                                                    943518 00000120
     ASGLUN                                                      943518 00000130
     CHARIN                                                      943518 00000140
     USRPGM                                                      943518 00000150
     PX9IO                                                       943518 00000160
     OSR733                                                      943518 00000170
     SB                                                          943518 00000180
     RANGE                                                       943518 00000190
     IM                                                          943518 00000200
     IC                                                          943518 00000210
     INWKSP                                                      943518 00000220
     INSPSS                                                      943518 00000230
     MODME                                                       943518 00000240
     MODCR                                                       943518 00000250
     MODREG                                                      943518 00000260
     MODWP                                                       943518 00000270
     PRTMMCR                                                     943518 00000280
     PRTSSN                                                      943518 00000290
     PXLDDR                                                      943518 00000300
     OVERLAY                                                     943518 00000310
     PROMLD                                                      943518 00000320
     SSNAP                                                       943518 00000330
     SCNTBL                                                      943518 00000340
     RUNPGP                                                      943518 00000350
     SIEPR                                                       943518 00000360
     BKPPR                                                       943518 00000370
     CLEAR                                                       943518 00000380
     FIND                                                        943518 00000390
     HXARTH                                                      943518 00000400
     WRTPRT                                                      943518 00000410
     FPDEFP                                                      943518 00000420
     CMDDEF          MUST BE NEXT TO LAST MODULE                 943518 00000430
     PXDATA          MUST BE LAST MODULE OF PX9MTR (ROOT)        943518 00000440
/*                                                               943518 00000450
 PHASE 1,LAL                                                     943518 00000460
 INCLUDE                                                         943518 00000470
     PX9LAL                                                      943518 00000480
/*                                                               943518 00000490
 PHASE 1,TRACE                                                   943518 00000500
 INCLUDE                                                         943518 00000510
     TRACEMOD                                                    943518 00000520
     SETREGN                                                     943518 00000530
     SETRACE                                                     943518 00000540
/*                                                               943518 00000550
 PHASE 1,ABS                                                     943518 00000560
 INCLUDE                                                         943518 00000570
     ABSDMP                                                      943518 00000580
     ABSLD                                                       943518 00000590
/*                                                               943518 00000600
 PHASE 1,PRPRG                                                   943518 00000610
 INCLUDE                                                         943518 00000620
```

```
    PROMPG PART I                                                    943518 00000630
   WHEN CONSTRUCTING THE TAPE, PROMPG PART II                        943518 00000640
   MUST BE PLACED BEHIND PROMPG PART I.                              943518 00000650
/*                                                                   943518 00000660
 PHASE 1,BNPF                                                        943518 00000670
 INCLUDE                                                             943518 00000680
     OMBNPF                                                          943518 00000690
/*                                                                   943518 00000700
 PHASE 1,HILO                                                        943518 00000710
 INCLUDE                                                             943518 00000720
     OMHL                                                            943518 00000730
/*                                                                   943518 00000740
 END                                                                 943518 00000750
 FOLLOWED BY:UPFRONT LOADER                                          943518 00000760
             PX9ASM                                                  943518 00000770
 FOLLOWED BY:UPFRONT LOADER                                          943518 00000780
             PX9EOT                                                  943518 00000790
 FOLLOWED BY:RELOCATABLE MONITOR                                     943518 00000800
/*                                                                   943518 00000810
```

```
NOSYMT
 PHASE 0,PX9MTP
 INCLUD
 PHASE 1,LAL
 INCLUD
PHASE 1,TRACE
INCLUDE
 PHASE 1,ABS
 INCLUD
 PHASE 1,PRPRG
 INCLUD
 PHASE 1,BNPF
 INCLUD
 PHASE 1,HILO
 INCLUD
 END
*END OF BUILD PHASE.
```

SDSLED 945275 * * 00107126 * PAGE 1
PX9MTP LENGTH 1FFA PROGRAM ORIGIN 0000

MODULE NAME	LENGTH	PROGRAM ORIGIN	DATE	TIME	MODULE SOURCE
INITIA	0030	0000			
ERRINT	001A	0030			
WKSPMG	008C	004A			
CMSTPR	0148	00D6			
PCOUT	0010	021E			
GTFELD	0088	022E			
GTHEX	0080	02B6			
SUPVSR	009E	0336			
CONVRT	011E	03D4			
ASGLUN	003C	04F2			
CHARIN	00D0	052E			
USRPGM	000A	05FE			
PX9IO	0146	0608			
DSR733	0454	074E			
SB	0058	0BA2			
RANGE	0046	0BFA			
IM	0012	0C40			
IC	001C	0C52			
INWKSP	0058	0C6E			
INSPSS	0024	0CC6			
MODME	0030	0CEA			
MODCR	0082	0D1A			
MODREG	0082	0D9C			
MODWP	005C	0E1E			
PRTMMC	0096	0E7A			
PRTSSN	006A	0F10			
PXLDDR	017A	0F7A			
OVERLA	007E	10F4			
PROMLD	002A	1172			
SSNAP	0080	119C			
SCNTBL	0030	124C			
RUNPGM	0098	127C			
SIEPR	005E	1314			
BKPPR	0064	1372			
CLEAR	003A	13D6			
FIND	00A6	1410			
HXARTH	006A	14B6			
WRTPRT	004A	1520			
FPDEFS	0000	156A			
CMDDEF	00D4	156A			
PXDATA	031C	163E			

D E F I N I T I O N S

NAME	VALUE	NAME	VALUE	NAME	VALUE	NAME	VALUE
ACL	0060	ASLUN	04F2	*BDA	000A*	*BHA	000C*
BKPT	17AA	BKPTPR	1372	BKSTR	1726	BKSTRB	172E
BLANK	16F2	BPTAR	1780	BS	16FA	BSFLG	17E4
CBD	164C	CBDSTG	164E	CBH	1644	CBH0	1646
*CBH2	1648	CBP	13D6	CDNDT	063C	*CHB	163E
*CHBVAL	1640	CLDT	067E	CLEARP	1520	CLST	03D4
*CMDMSK	0694	CMDTBL	156A	CNVRT	03E4	COMMA	16F5
CPLSAV	17E0	CR	16F8	CRLF	16FE	CRR	13E2
CRUOFF	1FE0*	CRUPRT	1FA0*	CS	0001*	CSBACK	0000*
CSFWD	0000*	CSOPEN	0790	CSP	00D6	CSRASC	0798
CSRWND	0000*	CSS	13DC	CSUNLD	0000*	CSWASC	085C

SDSLED 945275 * * 00107126 * PAGE 2
PX9MTP LENGTH 1FFA PROGRAM ORIGIN 0000

NAME	VALUE	NAME	VALUE	NAME	VALUE	NAME	VALUE
CSWEOF	08DC	*DAB	000B*	DC7331	074E	DFBIAS	16F0
DMPSPR	1722	*DUM	007F*	EFMASK	0693	EI	0030
ENTSIZ	000A*	EOR	16F8	EQSIGN	1720	ERBUF	170B
ERROR	05AE	ERSTR	1706	ESC	16FB	EXCT	05FE
FBT	1434	*FPSTRT	FE00*	FPWP	F800*	FREMEM	181A
FWD	1410	GETBUF	0050	GETCHR	052E	GETFLD	0232
GETFLN	022E	GETHEX	02BA	GETHXN	02B6	*HAB	000D*
HILIM	17F0	HXAR	14B6	ICP	0C52	IMP	0C40
INCHAR	1660	INIMEM	004C	INIT	0000	INSCNT	17E6
IO	069E	IRP	0DAA	ISRFLG	177A	ISS	0CC6
IWP	0C6E	LDABS	0F84	LDADDR	1678	LDBUF	1682
LDCBA	167E	LDCBBC	1680	LDCC	167C	*LDENTY	16D4
LDFLG	1676	*LDLDPT	16D6	LDLUN	1675	LDNMCC	16D8
LDOPCD	1674	*LDPGNM	16D9	LDPRB	1672	LDPRT	152A
LDTBL	16D4	LDUFL	0F7A	*LF	16F9	LGRASC	08F8
LGWASC	0A12	LOAD	0F7E	LOADOV	0F9A	LOWLIM	17EE
*LWP	0058	MEMWP	17FA	MINUS	16F4	MODCRU	0D1C
MODMEM	0CEA	MODWSP	0E1E	MONCHR	0A86	MRP	0DB2
*NOCELS	000A*	NOCMDS	163C	NUMBPS	0004*	NUMSNP	0004*
NUMTR	0004*	NUMTTE	0004*	*OPEN	0000*	*OVLRET	1128
OVLY	10F4	PCOUT	021E	PERIOD	16F7	PL	1172
PLUS	16F6	PRBBUF	1668	PRBCC	166C	PRBUER	1666
PRCRLF	054E	PRINT	0556	PRNTC	0552	PRNTHB	0570
PRNTHN	0576	PRNTHX	057C	PROMPT	1702	PRTCRU	0E8C
PRTMEM	0E7A	PRTPRD	1724	PRTSS	0F10	PTDIFF	1718
PTSUM	1712	RANGEX	0C02	*RDASC	0009*	RETBUF	0054
RGSTR	1730	RGSTRB	1732	ROMLDR	FFFA*	RR	0064
RUBOUT	16FC	RUN	127C	*RWP	005C	SAVWP	17DE
SBP	0BA4	SCTB	124C	SETLUN	0666	SIE	1314
SIEINT	1344	SIEST	17DA	SIEWP	17D8	SIINPG	17DC
SNAPS	1784	SNPENT	0008*	SNPTAB	178A	SREGN	1604
SSS	119C	SSSTR	1734	SSSTRB	173B	STRPRT	17E2
*SVC	007C*	SVCALT	166E	SVCPRB	1654	SVCSFG	1658
SVCSR	0358	SVCSRA	0354	*SVCUSR	1659	*SVCW10	17F8
*SVCW7	17F2	SVCWP	17E4	SVCWRT	1662	TERM	0604
TERMCR	16FD	TRACE	1774	TRFLGS	0000*	TRHIGH	0004*
TRLNTH	1778	TRLOW	0002*	TRNARA	195A	TRTBL	173C
TRTYPE	0006*	TRV1	0008*	TTTBL	177C	*UEFMSK	0695
UEMASK	0692	USRPC	17EA	USRST	17EC	USRWP	17E8
USRWSP	17CE	*WTASC	000B*	ZERO	16F3		

SDSLED 945275 * * 00107126 * PAGE 3
LAL LENGTH 06A0 PROGRAM ORIGIN 195A

MODULE NAME	LENGTH	PROGRAM ORIGIN	DATE	TIME	MODULE SOURCE
PX9LAL	06A0	195A			

D E F I N I T I O N S

NAME	VALUE	NAME	VALUE	NAME	VALUE	NAME	VALUE
*LALCSR	1A18						

SDSLED 945275 * * 00107126 * PAGE 4
TRACE LENGTH 069C PROGRAM ORIGIN 195A

MODULE NAME	LENGTH	PROGRAM ORIGIN	DATE	TIME	MODULE SOURCE
TRACEM	0558	195A			
SETREG	00A6	1EB2			
SETRAC	009E	1F58			

D E F I N I T I O N S

NAME	VALUE	NAME	VALUE	NAME	VALUE	NAME	VALUE
SR	1EB2	STRACE	1F74	TRACER	1A50		

SDSLED 945275 * * 00107126 * PAGE 5
ABS LENGTH 0456 PROGRAM ORIGIN 195A

MODULE NAME	LENGTH	PROGRAM ORIGIN	DATE	TIME	MODULE SOURCE
ABSDMP	0242	195A			
ABSLD	0214	189C			

D E F I N I T I O N S

NAME	VALUE	NAME	VALUE	NAME	VALUE	NAME	VALUE
*ABSBUT	189C	ABSLDR	1882	*DPRG	19CC		

SDSLED 945275 * * 00107126 * PAGE 6
PRPRG LENGTH 0690 PROGRAM ORIGIN 195A

MODULE NAME	LENGTH	PROGRAM ORIGIN	DATE	TIME	MODULE SOURCE
PROMPG	0690	195A			

D E F I N I T I O N S

NAME	VALUE	NAME	VALUE	NAME	VALUE	NAME	VALUE
PGBIAS	19AA	*PPCSI	1A6E	PROGSZ	19AC	*PSCSI	1EB8

SDSLED 945275 * * 00107126 * PAGE 7
BNPF LENGTH 0382 PROGRAM ORIGIN 195A

MODULE NAME	LENGTH	PROGRAM ORIGIN	DATE	TIME	MODULE SOURCE
DMBNPF	0382	195A			

D E F I N I T I O N S

NAME	VALUE	NAME	VALUE	NAME	VALUE	NAME	VALUE
*DMBNPF	19EA						

SDSLED 945275 * * 00107126 * PAGE 8
HILO LENGTH 02D6 PROGRAM ORIGIN 195A

MODULE NAME	LENGTH	PROGRAM ORIGIN	DATE	TIME	MODULE SOURCE
DMHL	02D6	195A			

D E F I N I T I O N S

NAME	VALUE	NAME	VALUE	NAME	VALUE	NAME	VALUE
*DMHL	19CA						

**** LINKING COMPLETED

A[

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																
SHEET																															
REV STATUS OF SHEETS	REV																															
	SHEET																															

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DWN	DATE
	CHK	
	ENGR	
	QA	
	APVD	
	CONTR NO	
	DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, INITIP, INITIALIZATION, PROTOTYPE-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	943519-9901

SCALE	REV.	SHEET
		1 of 4

TI-0023-C

```
0003                        IDT  'INITIP'
0004                  * TITLE:    INITIP
0005                  * REVISION:
0006                  *           ORIGINAL
0007                  * COMPUTER: 990,ASM
0008                  * ABSTRACT: INITIALIZES THE MONITOR WHENEVER IT IS
0009                  *           LOADED OR RESTARTED. BRANCHES TO
0010                  *           COMMAND STRING PROCESSOR.
0011                  * CALLING SEQUENCE:
0012                  *           LOAD OR RESTART MONITOR
0013                  *                OR
0014                  *           B    @INIT
0015                  *
0016                        DEF  INIT
0017                        REF  CMDTBL
0018                        REF  DC7331
0019                        REF  CSP
0020                        REF  SVCSR
0021                        REF  SVCWP
0022                        REF  EI
0023                        REF  FREMEM
0024                        REF  GETBUF
0025                        REF  INIMEM
0026                        REF  FPWP
0027                        REF  FPSTRT
0028                        REF  CRUPRT
0029                  *
0030        0000      WRTPRT EQU  0
0031                  *
0032        0001      R1     EQU  1
0033        0002      R2     EQU  2
0034        0003      R3     EQU  3
0035        0004      R4     EQU  4
0036        000C      R12    EQU  12
0037        0000'     INIT   EQU  $
0038  0000  02E0             LWPI FREMEM
      0002  0000
0039  0004  0300             LIMI >2
      0006  0002
0040                  *
0041                  *      CLEAR PARITY ERRORS
0042                  *
0043  0008  0360             RSET
0044                  *
0045                  *      FRONT PANEL INTERRUPT VECTOR
0046                  *
0047  000A  0201             LI   R1,EI
      000C  0000
0048  000E  C801             MOV  R1,@>A
      0010  000A
0049                  *
0050                  *      TURN OFF PROTECT VIOLATION FLAG AND CLEAR PROTECT RE
0051                  *
0052  0012  020C             LI   R12,CRUPRT
```

```
        0014  0000
 0053   0016  1D00          SBO   WRTPRT
 0054                 *
 0055                 *         INITIALIZE 733
 0056                 *
 0057          0018' INI733 EQU   $
 0058   0018   C320          MOV   @DC7331,R12       MOD CRU BASE
        001A   0000
 0059   001C   1D09          SBO   >9
 0060   001E   1D0A          SBO   >A
 0061   0020   1D0B          SBO   >B
 0062   0022   1D0C          SBO   >C
 0063                 *
 0064                 *       INITIALIZE MEMORY CHAINS
 0065   0024   0420          BLWP  @INIMEM
        0026   0000
 0066   0028   0420          BLWP  @GETBUF
        002A   0000
 0067   002C   0460          B     @CSP
        002E   0000
 0068                         END   INIT
0000 ERS
```

960 - 980 CONCORDANCE

$		0037	0057
CMDTBL		0017	
CRUPRT		0028	0052
CSP		0019	0067
DC7331		0018	0058
EI		0022	0047
FPSTRT		0027	
FPWP		0026	
FREMEM		0023	0038
GETBUF		0024	0066
INI733	0057		
INIMEM		0025	0065
INIT	0037	0016	0068
R1	0032	0047	0048
R12	0036	0052	0058
R2	0033		
R3	0034		
R4	0035		
SVCSR		0020	
SVCWP		0021	
WRTPRT	0030	0053	

THERE ARE 0021 SYMBOLS

AL

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

NOTE: REFERENCE 945384-FPDEFS.

REV																															
SHEET																															
REV STATUS OF SHEETS	REV																														
	SHEET																														

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, FPDEFP, FRONT PANEL DEFINITIONS, PROTOTYPE-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	943523-9901

SCALE	REV.	SHEET
		1 of 3

TI-0023-C

```
0003                        IDT   'FPDEFP'
0004                        DEF   FPSTRT
0005                        DEF   ROMLDR
0006                        DEF   CRUOFF
0007                        DEF   CRUPRT
0008                        DEF   FPWP
0009        FE00   FPSTRT   EQU   >FE00
0010        FFFA   ROMLDR   EQU   >FFFA
0011        1FE0   CRUOFF   EQU   >1FE0
0012        1FA0   CRUPRT   EQU   >1FA0          CRU PROTECT REGION REG
0013        F800   FPWP     EQU   >F800          FRONT PANEL WORKSPACE
0014                        END
0000 ERS
```

```
960 - 980  CONCORDANCE
CRUOFF   0011       0006
CRUPRT   0012       0007
FPSTRT   0009       0004
FPWP     0013       0008
ROMLDR   0010       0005

THERE ARE    0005  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV STATUS OF SHEETS

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, WRTPRT, WRITE PROTECT-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	943524-9901

SCALE REV. SHEET 1 of 5

TI-0023-C

```
0003                *    PROCEDURE CLEAR REGION
0004                *    DECLARE
0005                *       1 CLEAR REGION BIT 16 ('>8000')
0006                *    CALL SET REGION(CLEAR REGION)
0007                *    END
0008                *    PROCEDURE SET PROTECTED REGION(CPL POINTER)
0009                *    DECLARE
0010                *       1 CPL BIT 48   (COMMAND PARAMETER LIST)
0011                *          2 PARM COUNT BIT 8
0012                *          2 PRESENCE BIT 8
0013                *          2 PARM 1 BIT 16
0014                *          2 PARM 2 BIT 16
0015                *       1 BASE BIT 16 ('>200')
0016                *       1 REGION BIT 16
0017                *          2 LOW BOUND BIT 8
0018                *          2 UPPER BOUND BIT 8
0019                *    IF PRESENCE(1) = 0 THEN CALL ERROR;
0020                *    ELSE
0021                *       IF PRESENCE(2) = 0 THEN CALL ERROR;
0022                *       ELSE
0023                *          LOW BOUND = PARM1 MOD BASE;
0024                *          UPPER BOUND = PARM2 MOD BASE;
0025                *          IF LOWER BOUND .GE. UPPER BOUND THEN CALL ERROR;
0026                *          ELSE
0027                *             CALL SET REGION(REGION);
0028                *    END
0029                *    PROCEDURE SET REGION(REGION)
0030                *       CRU(PROTECT REGISTER) = REGION;
0031                *    END
0032                       IDT  'WRTPRT'
0033                * TITLE:    WRTPRT
0034                *           WRITE PROTECT
0035                * REVISION:
0036                *           ORIGINAL
0037                * COMPUTER: 990,ASM
0038                * ABSTRACT:
0039                *        1) THIS ROUTINE WILL STORE THE BOUNDS FOR
0040                *           THE WRITE PROTECT IN THE CRU REGISTER.
0041                *           ANY BOUND NOT ON A 256 WORD BOUNDARY
0042                *           WILL BE TRUNCATED TO THE NEXT LOWER
0043                *           256 WORD BOUNDARY.
0044                *        2) CLEAR THE PROTECT REGISTER.
0045                * CALLING SEQUENCE:
0046                *           CALLED FROM COMMAND STRING PROCESSOR.
0047                *             ENTRY = R10 POINTS TO PARAMETER LIST
0048                *
0049                *
0050                *WORKSPACE REGISTER DEFINITIONS
0051                *
0052  0000  R0     EQU  0
0053  0001  R1     EQU  1
0054  0002  R2     EQU  2
0055  0003  R3     EQU  3
0056  0004  R4     EQU  4
```

```
0057          0005  R5      EQU  5
0058          0006  R6      EQU  6
0059          0007  R7      EQU  7
0060          0008  R8      EQU  8
0061          0009  R9      EQU  9
0062          000A  R10     EQU  10
0063          000B  R11     EQU  11
0064          000C  R12     EQU  12
0065          000D  R13     EQU  13
0066          000E  R14     EQU  14
0067          000F  R15     EQU  15
0068                *
0069                *
0070                      REF  ACL
0071                      REF  RR
0072                      REF  STRPRT
0073                      DEF  LDPRT
0074                      DEF  CLEARP
0075                      REF  CRUPRT
0076                      REF  ERROR
0077          0200  BASE    EQU  >200          256 WORDS
0078          8000  CLRPRT  EQU  >8000         SET HIGH BIT IN CRU TO CLEAR
0079          0205  MS05    EQU  >0205         ERROR MESSAGE
0080                *
0081                *                          0=PROCEDURE CLEAR REGION
0082                *                          1=DECLARE
0083                *                          2=1 CLEAR REGION BIT 16 (!>800
0084          0000' CLEARP  EQU  $
0085  0000    0420        BLWP @ACL
      0002    0000
0086                *                          1=CALL SET REGION(CLEAR REGION
0087  0004    0204        LI   R4,CLRPRT
      0006    8000
0088  0008    1013        JMP  LDPRT1
0089                *                          0=END
0090                *                          0=PROCEDURE SET PROTECTED REGI
0091                *                          1=DECLARE
0092                *                          2=1 CPL BIT 48   (COMMAND PARA
0093                *                          3=2 PARM COUNT BIT 8
0094                *                          3=2 PRESENCE BIT 8
0095                *                          3=2 PARM 1 BIT 16
0096                *                          3=2 PARM 2 BIT 16
0097                *                          2=1 BASE BIT 16 (!>200')
0098                *                          2=1 REGION BIT 16
0099                *                          3=2 LOW BOUND BIT 8
0100                *                          3=2 UPPER BOUND BIT 8
0101          000A' LDPRT   EQU  $
0102  000A    0420        BLWP @ACL
      000C    0002'
0103  000E    C27A        MOV  *R10+,R9
0104                *                          1=IF PRESENCE(1) = 0 THEN CALL
0105  0010    0A99        SLA  R9,9
0106  0012    1714        JNC  LDPRT2
0107                *                          1=ELSE
0108                *                          2=IF PRESENCE(2) = 0 THEN CALL
```

```
 0109  0014  0A19          SLA   R9,1
 0110  0016  1712          JNC   LDPRT2
 0111                *
 0112                *
 0113  0018  C0FA          MOV   *R10+,R3
 0114  001A  04C2          CLR   R2
 0115  001C  0201          LI    R1,BASE
       001E  0200
 0116  0020  3C81          DIV   R1,R2
 0117                *
 0118  0022  C17A          MOV   *R10+,R5
 0119  0024  04C4          CLR   R4
 0120  0026  3D01          DIV   R1,R4
 0121                *
 0122  0028  8102          C     R2,R4
 0123  002A  1408          JHE   LDPRT2
 0124                *
 0125                *
 0126  002C  0A82          SLA   R2,8
 0127  002E  D102          MOVB  R2,R4
 0128                *
 0129                *
 0130        0030' LDPRT1  EQU   $
 0131                *
 0132  0030  C804          MOV   R4,@STRPRT
       0032  0000
 0133  0034  020C          LI    R12,CRUPRT
       0036  0000
 0134  0038  3004          LDCR  R4,0
 0135  003A  1004          JMP   LDPRT3
 0136                *
 0137        003C' LDPRT2  EQU   $
 0138  003C  020A          LI    R10,MS05
       003E  0205
 0139  0040  06A0          BL    @ERROR
       0042  0000
 0140        0044' LDPRT3  EQU   $
 0141  0044  0420          BLWP  @RR
       0046  0000
 0142  0048  045B          RT
 0143                      END
0000 ERS
```

```
2=ELSE
3=LOW BOUND = PARM1 MOD BASE;

3=UPPER BOUND = PARM2 MOD BASE

3=IF LOWER BOUND .GE. UPPER BO

3=ELSE
4=CALL SET REGION(REGION);

0=END
0=PROCEDURE SET REGION(REGION)

2=CRU(PROTECT REGISTER) = REGI
 SAVE PROTECT REGION

0=END
```

960 - 980 CONCORDANCE

$		0084	0101	0130	0137	0140		
ACL		0070	0085	0102				
BASE	0077	0115						
CLEARP	0084	0074						
CLRPRT	0078	0087						
CRUPRT		0075	0133					
ERROR		0076	0139					
LDPRT	0101	0073						
LDPRT1	0130	0088						
LDPRT2	0137	0106	0110	0123				
LDPRT3	0140	0135						
MS05	0079	0138						
R0	0052							
R1	0053	0115	0116	0120				
R10	0062	0103	0113	0118	0138			
R11	0063							
R12	0064	0133						
R13	0065							
R14	0066							
R15	0067							
R2	0054	0114	0116	0122	0126	0127		
R3	0055	0113						
R4	0056	0087	0119	0120	0122	0127	0132	0134
R5	0057	0118						
R6	0058							
R7	0059							
R8	0060							
R9	0061	0103	0105	0109				
RR		0071	0141					
STRPRT		0072	0132					

THERE ARE 0030 SYMBOLS

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																		
SHEET																																		
REV STATUS OF SHEETS	REV																																	
	SHEET																																	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, DMHL, DUMP HI/LO, PROTOTYPE-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945325-9901

SCALE	REV.	SHEET 1 of 20

TI-0923-C

```
0003                        IDT  'DMHL'
0004                * TITLE:    DMHL
0005                *           DUMP HIGH/LOW FORMAT AND DESCR
0006                * REVISION: 03/15/76
0007                *           ORIGINAL
0008                * ABSTRACT:
0009                *           DUMP 256 4-BIT PATTERNS FROM MEMORY TO CASSETTE
0010                *           TAPE IN TI'S 4 BY 256 HIGH/LOW FORMAT
0011                * COMPUTER: 990,ASSEMBLY
0012                *** EXTERNAL REFERENCES
0013                *
0014                        DEF  DMHL
0015                        REF  ERROR
0016                        REF  RR
0017                        REF  ACL
0018                        REF  PRCRLF
0019                        REF  PRNTC
0020                        REF  PRNTHN
0021                        REF  EQSIGN
0022                        REF  PRNTHX
0023                        REF  SVCALT
0024                        REF  PRTPRD
0025                *
0026  0000    48            TEXT 'HL'
0027  0002  0070'           DATA DMHL
0028  0004  0000            DATA 0
0029                *
0030                *** ERROR MESSAGE EQUATES
0031                *
0032        0205  MS05   EQU  >0205          REQUIRED PARM MISSING
0033        0001  MX01   EQU  >0001          UNRECOVERABLE I/O ERROR
0034        1103  DP03   EQU  >1103          OUT OF BOUNDS
0035        0100  MP00   EQU  >0100          INVALID PARAMETER
0036                *
0037                *** SVC/PRB EQUATES
0038                *
0039        000B  WRITA  EQU  11             WRITE ASCII
0040        0009  READA  EQU  9              READ ASCII
0041        0000  OPEN   EQU  00             OPEN FILE
0042        000D  WEOF   EQU  13             WRITE END-OF-FILE
0043                *
0044                *** PRB.DEFINITIONS
0045                *
0046        0006' PRB    EQU  $
0047  0006  0060  PRBSC  DATA 0              SERVICE TYPE
0048  0008    00  PRBOP  BYTE 0              I/O COMMAND
0049  0009    07  PRBLU  BYTE 7              LOGICAL UNIT NUMBER
0050  000A    00  PRBSF  BYTE 0              SYSTEM COMPLETION FLAGS
0051  000B    00  PRBUF  BYTE 0              USER FLAGS
0052  000C  001A' PRBBF  DATA BFR            OUTPUT BUFFER ADDRESS
0053  000E  0056  PRBLN  DATA 86             OUTPUT BUFFER LENGTH
0054  0010  0050  PRBCC  DATA 80             CHARACTER COUNT
0055                *
0056                *** MISCELLANEOUS DATA
```

```
0057                    *
0058  0012  0000  SAVERG DATA 0
0059        2D00  HYPHON EQU  >2D00
0060        0014' ENDADD EQU  $
0061  0014  0000        DATA 0
0062        0016' CHARCT EQU  $
0063  0016  0050        DATA >0050              CHARACTER COUNT FOR WRITE
0064        014D  MPRT   EQU  >100+'M'           MEMORY WORD TYPE
0065        0154  TPRT   EQU  >100+'T'           TAPE WORD TYPE
0066        0020  BLANK  EQU  ' '                BLANK=' '
0067        0018' AHL    EQU  $
0068  0018  484C        DATA 'HL'
0069  001A  0000  BFR    DATA 0,0,0,0,0,0,0,0,0,0   EIGHTY=
      001C  0000
      001E  0000
      0020  0000
      0022  0000
      0024  0000
      0026  0000
      0028  0000
      002A  0000
      002C  0000
0070  002E  0000        DATA 0,0,0,0,0,0,0,0,0,0     SIX
      0030  0000
      0032  0000
      0034  0000
      0036  0000
      0038  0000
      003A  0000
      003C  0000
      003E  0000
      0040  0000
0071  0042  0000        DATA 0,0,0,0,0,0,0,0,0,0       BYTES
      0044  0000
      0046  0000
      0048  0000
      004A  0000
      004C  0000
      004E  0000
      0050  0000
      0052  0000
      0054  0000
0072  0056  0000        DATA 0,0,0,0,0,0,0,0,0,0         OF
      0058  0000
      005A  0000
      005C  0000
      005E  0000
      0060  0000
      0062  0000
      0064  0000
      0066  0000
      0068  0000
0073  006A  0000        DATA 0,0,0                         ZEROES
      006C  0000
      006E  0000
```

0074 *** REGISTER DEFINITIONS ***

```
0076                *       ALL  REG. REFERENCES IN BNPF DUMP ARE VIA SYMBOLIC
0077                *       ACRONYMS INDICATIVE OF THE CURRENT USE OF THE REGIS=
0078                *       TER.  THESE ACRONYMS ARE DEFINED BELOW.  ANY TIME
0079                *       A REGISTER FREES UP, OR IS RE=ASSIGNED, ONE OF THE
0080                *       FOLLOWING COMMENT CARDS WILL APPEAR IN THE LISTING
0081                *
0082                *** REGISTER REASSIGNMENT: OLD WPX=!PTR1!, NEW WPX=!FREE!
0083                *** REGISTER REASSIGNMENT: OLD WPX=!PTR1!, NEW WPX=!PTR2!
0084                *
0085          0000  R0      EQU  0              SCRATCH
0086          0001  R1      EQU  1              SCRATCH
0087          0002  CPRM    EQU  2              PTR TO COMMAND PARM LIST
0088          0003  HDR     EQU  3              HEADER FROM CMD. PARM. LIST
0089          0004  ACT     EQU  4              DUMP OR COMPARE TOGGLE
0090          0005  STRT    EQU  5              DUMP OR CMPR START ADDRESS
0091          0005  MEM     EQU  5              DUMP OR CMPR STRT+CURRENT IND
0092          0006  END     EQU  6              DUMP OR CMPR END ADDRESS
0093          0006  ROMADD  EQU  6              ROM WORD ADDRESS
0094          0007  BIT     EQU  7              BIT DISPLACEMENT WITHIN WORD
0095          0008  BFRA    EQU  8              OUT BFR BASE + CURR INDEX
0096          0009  BFRE    EQU  9              PTR TO MX DATA ADDR IN BUFFER
0097          000A  PRBA    EQU  10             ADDRESS OF PRB
0098          000A  R10     EQU  10             SCRATCH
0099          000B  R11     EQU  11             RETURN ADDRESS
0100          000D  R13     EQU  13             PTR TO PREVIOUS WORKSPACE
0101          000E  R14     EQU  14             SCRATCH
0102          000F  R15     EQU  15             SCRATCH
```

```
0104          0070' DMHL   EQU  $
0105  0070  C08A          MOV  R10,CPRM
0106  0072  0420          BLWP @ACL                    LINK A NEW WORKSPACE
      0074  0000
0107  0076  C80D          MOV  R13,@SAVERG             SAVE OLD WKSP POINTER
      0078  0012'
0108  007A  C0F2          MOV  *CPRM+,HDR               HEADER WORD
0109                *** PARAMETER VALIDATION
0110                *1  IF(STRTB.AND.ENDB.EQ.0)CALL ABORT(MS05)
0111  007C  0A93          SLA  HDR,9                   SHIFT ACT  PRESENT BIT INTO
0112  007E  1722          JNC  DMH01                    CARRY
0113  0080  020A          LI   R10,MP00
      0082  0100
0114  0084  D032          MOVB *CPRM+,R0               CHAR COUNT TO R0
0115  0086  0990          SRL  R0,9                    TRUNCATE TO EVEN NUM
0116  0088  0A10          SLA  R0,1
0117  008A  D132          MOVB *CPRM+,ACT              TOGGLE
0118  008C  0984          SRL  ACT,8
0119  008E  0284          CI   ACT,'D'                 IF TOGGLE NE
      0090  0044
0120  0092  1303          JEQ  DMH00                    'D' OR 'C'
0121  0094  0284          CI   ACT,'C'                 THEN CALL ERROR(MP00)
      0096  0043
0122  0098  1617          JNE  ABORT
0123          009A' DMH00  EQU  $
0124  009A  A080          A    R0,CPRM                 INCREMENT CPL POINTER TO STRT
0125  009C  0A13          SLA  HDR,1                   SHIFT STRT PRESENT BIT INTO
0126  009E  1712          JNC  DMH01                    CARRY
0127  00A0  C172          MOV  *CPRM+,STRT              DUMP STARTING ADDR
0128  00A2  0915          SRL  STRT,1                  IF STRT NOT EVEN THEN
0129  00A4  1811          JOC  ABORT                   CALL ERROR(MP00)
0130  00A6  0A15          SLA  STRT,1
0131  00A8  0A13          SLA  HDR,1                   START OK, CHECK FOR END
0132  00AA  170C          JNC  DMH01
0133  00AC  C1B2          MOV  *CPRM+,END               DUMP END ADDRESS
0134  00AE  0816          SRA  END,1                   IF END NOT EVEN THEN
0135  00B0  1808          JOC  ABORT                   THEN CALL ERROR
0136  00B2  0A16          SLA  END,1
0137  00B4  04C7          CLR  BIT                     DEFAULT TO 0
0138  00B6  0A13          SLA  HDR,1                   END OK, CHECK FOR BIT
0139  00B8  170A          JNC  DMH01A                   JUMP IF NOT PRESENT
0140  00BA  C1F2          MOV  *CPRM+,BIT              BIT DISPLACEMENT IN WORD
0141  00BC  0287          CI   BIT,12                  IF BIT VALUE OUT OF RANGE
      00BE  000C
0142  00C0  1503          JH   ABORT                   ABORT
0143  00C2  1005          JMP  DMH01A
0144          00C4' DMH01  EQU  $
0145  00C4  020A          LI   R10,MS05                ABORT-REQUIRED PARM MISSING
      00C6  0205
0146          00C8' ABORT  EQU  $
0147  00CA  06A0          BL   @ERROR
      00CC  0000
0148  00CE  104A          JMP  DMHTRM
0149                *
```

```
0150          00CE' DMH01A EQU  $
0151                *1  X=END-START
0152  00CE    C006        MOV  END,R0
0153  00D0    020A        LI   R10,DP03          JUMP IF START IS GREATER
      00D2    1103
0154  00D4    6005        S    STRT,R0
0155                *1  IF(X.LE.0)CALL ABORT(MP06)
0156                *1  IF(X.GT.256)CALL ABORT(DP03)
0157  00D6    0280        CI   R0,>1FE           THAN END OR LEN > 256 WORDS
      00D8    01FE
0158  00DA    1BF5        JH   ABORT
0159                *
0160                ** PARAMETER VALIDATION COMPLETE
0161                * **                        OLD R2=CPRM, NEW R2=FREE
0162                * **                        OLD R3=HDR   NEW R3=FREE
0163                *
0164          0002  R2    EQU  2
0165          0003  R3    EQU  3
0166                *
0167                *1= CALL OPEN(PRBA)
0168                *       (IN LINE EXPANSION)
0169                *
0170          00DC' DMH02  EQU  $
0171  00DC    0201        LI   R1,OPEN*256+WRITA OPEN LUNO 7
      00DE    000B
0172  00E0    C820        MOV  @CHARCT,@PRBCC    SET CHAR. CT. FOR WRITE
      00E2    0016'
      00E4    0010'
0173  00E6    0284        CI   ACT,'C'           IF TOGGLE SET TO COMPARE
      00E8    0043
0174  00EA    1602        JNE  DMH02A            PREPARE TO READ
0175  00EC    0201        LI   R1,OPEN*256+READA OPEN LUNO 7
      00EE    0009
0176          00F0' DMH02A EQU  $
0177  00F0    D801        MOVB R1,@PRBOP
      00F2    0008'
0178  00F4    020A        LI   PRBA,PRB
      00F6    0006'
0179  00F8    0420        BLWP @SVCALT
      00FA    0000
0180  00FC    06C1        SWPB R1                SET THE PRB TO FUNCTION
0181  00FE    D801        MOVB R1,@PRBOP           CODE TO WRITE/READ ASCII
      0100    0008'
0182  0102    0209        LI   BFRE,BFR+48       SET UP END OF OUTPUT BUFFERS
      0104    004A'
0183  0106    C806        MOV  END,@ENDADD
      0108    0014'
0184  010A    04C6        CLR  ROMADD            SET BEGINNING ROM ADDRESS
0185  010C    04CD        CLR  R13
0186  010E    0284        CI   ACT,'C'           IF TOGGLE SET TO COMPARE
      0110    0043
0187  0112    1318        JEQ  DMH04             BRANCH TO COMPARE SEQUENCE
0188                *1= MEM=STRT
0189                * **REGISTER REASSIGNMENT:OLD R5=STRT, NW R5=MEM
0190                *1= DO WHILE(MEM.LT.END)
```

```
0191                    *2  CALL BLNK(BFRA,BFRE)
0192            0114'   DMH03   EQU   $
0193    0114    0208            LI    BFRA,BFR          SET UP START OF OUTPUT BUFFER
        0116    001A'
0194    0118    06A0            BL    @BLNK             BLANK-FILL THE BUFFER
        011A    02B4'
0195                    *
0196                    *1- CALL FILL(BFRA,E,MEM,END)
0197    011C    06A0            BL    @FILL               FILL BFR W/BNPF DATA
        011E    016C'
0198                    *2  CALL WRTRD(PRB)
0199    0120    06A0            BL    @WRTRD
        0122    02C4'
0200    0124    16D1            JNE   ABORT
0201                    *1  CONTINUE
0202    0126    0226            AI    ROMADD,8
        0128    0008
0203    012A    0286            CI    ROMADD,>FF
        012C    00FF
0204    012E    12F2            JLE   DMH03
0205    0130    0208            LI    BFRA,BFR
        0132    001A'
0206                    *1- CALL WRTRD(PRBA)
0207    0134    0200            LI    R0,WEOF*256       WRITE AN END-OF-FILE
        0136    0D00
0208    0138    D800            MOVB R0,@PRBOP            SET OPCODE
        013A    0008'
0209                    *1 CALL WRTRD(WEOF)
0210    013C    06A0            BL    @WRTRD
        013E    02C4'
0211    0140    16C3            JNE   ABORT               ABORT IF WEOF FAILED
0212    0142    100F            JMP   DMHTRM
0213            0144'   DMH04   EQU   $
0214                    * **REGISTER REASSIGNMENT: OLD R4=ACT, NEW R4=FREE
0215            0004    R4      EQU   4
0216    0144    0208            LI    BFRA,BFR          SET UP STRT OF OUTPUT BUFFER
        0146    001A'
0217                    *1-CALL WRTRD(PRB)
0218    0148    06A0            BL    @WRTRD            READ A RECORD
        014A    02C4'
0219    014C    C040            MOV   R0,R1             SYSTEM FLAGS
0220    014E    0240            ANDI  R0,>4000
        0150    4000
0221    0152    16BA            JNE   ABORT             ABORT IF IO ERROR
0222    0154    0241            ANDI  R1,>2000
        0156    2000
0223    0158    1604            JNE   DMHTRM            IF EOF THEN EXIT
0224                    *2 CALL UNFILL(BFRA,E,MEM,END)
0225    015A    06A0            BL    @UNFILL           COMPARE BUFFER AND MEMORY
        015C    01DE'
0226    015E    1001            JMP   DMHTRM            ESC RETURN
0227                    *1 CONTINUE
0228    0160    10F1            JMP   DMH04
0229                    *1  END DMHL
0230            0162'   DMHTRM EQU   $
```

```
0231  0162  C360          MOV  @SAVERG,R13      RETURN WKSP POINTER
      0164  0012'
0232  0166  0420          BLWP @RR
      0168  0000
0233  016A  045B          RT
```

```
0236                    * TITLE    :FILL
0237                    *           FILL THE OUTPUT BUFFER WITH HIGH/LOW FORMAT
0238                    *
0239                    * REVISION: 03/15/76
0240                    *           ORIGINAL
0241                    * ABSTRACT:
0242                    *           FILL BUFFER WITH TI 4 X 256 HIGH/LOW FORMAT.
0243                    * COMPUTER: 990,ASSEMBLY
0244                    * CALLING SEQUENCE:
0245                    *           ENTRY:
0246                    *             R5=MEM =ADDRESS OF MEMORY DATA
0247                    *             R8=BFRA=ADDRESS OF OUTPUT BUFFER
0248                    *             R9=BFRE=ADDR. OF END OF OUTPUT BUFFER
0249                    *             R6=END =ENDING ADDR. OF MEMORY DATA
0250                    *             R6=ROMADD=ADDRESS OF ROM WORD
0251                    *             R7=BIT =BIT DISPLACEMENT IN MEMORY WORD
0252                    *
0253                    *           EXIT:
0254                    *             R0=>R4
0255                    *             MEM =ADDRESS OF NEXT BYTE TO BE CONVERTED
0256                    *             BFRA=ADDR+1 OF LAST BYTE IN OUTPUT BUFFER
0257                    *             THE BNPF RECORD IS FORMATTED IN 'BFR'.
0258          016C'     FILL   EQU  $              ****ENTRY POINT****
0259                    *
0260                    *1  CALL BINDEC(MEM,BFRA)
0261                    *       IN LINE EXPANSION
0262  016C  04CE               CLR  R14                FIRST OR LAST LINE ADDR COUNT
0263  016E  C046               MOV  ROMADD,R1          INIT: VALUE TO BE CONVERTED
0264  0170  0204               LI   R4,10                    CONVERSION BASE
      0172  000A
0265  0174  0203               LI   R3,100                   MAX # OF DGTS IN CNVTD
      0176  0064
0266          0178' FLL1   EQU  $
0267  0178  04C0               CLR  R0
0268  017A  3C03               DIV  R3,R0              FIND DIGIT
0269  017C  0220               AI   R0,>30             CONVERT THE DIGIT TO DECIMAL
      017E  0030
0270  0180  06C0               SWPB R0                   ASCII AND
0271  0182  DE00               MOVB R0,*BFRA+            STORE IN THE OUTPUT BFR.
0272  0184  04C2               CLR  R2                   REDUCE NUMBER OF
0273  0186  3C84               DIV  R4,R2                DIGITS REMAINING BY 1, AND
0274  0188  A0C2               A    R2,R3              WHEN # OF DIGITS REMAINING IS
0275  018A  C082               MOV  R2,R2
0276  018C  16F5               JNE  FLL1               NOT ZERO, REPEAT
0277  018E  091E               SRL  R14,1              ELSE, IF COUNT=0
0278  0190  180A               JOC  FLL2
0279  0192  0201               LI   R1,HYPHON          MOVE HYPHON TO BUFFER, SET UP
      0194  2D00
0280  0196  DE01               MOVB R1,*BFRA+          TO CONVERT END-OF-LINE ADDR
0281  0198  C046               MOV  ROMADD,R1          TO DECIMAL AND REPEAT CONVER-
0282  019A  0221               AI   R1,7               SION
      019C  0007
0283  019E  0203               LI   R3,100
      01A0  0064
```

```
0284  01A2  058E          INC   R14              INCR CT TO INDICATE LAST ADDR
0285  01A4  10E9          JMP   FLL1
0286        01A6' FLL2    EQU   $
0287  01A6  0588          INC   BFRA             PUTS A BLANK IN BUFFER
0288        01A8' FLL3    EQU   $
0289  01A8  0588          INC   BFRA             PUTS A BLANK IN BUFFER
0290  01AA  0201          LI    R1,4             BIT COUNTER IN R1
      01AC  0004
0291  01AE  8B05          C     MEM,@ENDADD
      01B0  0014'
0292  01B2  1B0D          JH    FLL6
0293  01B4  C0F5          MOV   *MEM+,R3         SHIFT WORD TO BIT
0294  01B6  C007          MOV   BIT,R0           IF BIT WITHIN WORD NOT ZERO,
0295  01B8  1301          JEQ   FLL4
0296  01BA  0A03          SLA   R3,R0
0297        01BC' FLL4    EQU   $
0298  01BC  0202          LI    R2,AHL           ADDR. OF 'HL' IN R2
      01BE  0018'
0299  01C0  0A13          SLA   R3,1             BIT TO BE STORED IN ARRY STAT
0300  01C2  1801          JOC   FLL5             JUMP FOR POSITIVE BIT
0301  01C4  0582          INC   R2                 INC TO 'L' FOR NEG. BIT
0302  01C6  DE12  FLL5    MOVB  *R2,*BFRA+       STORE 'H' OR 'L'
0303  01C8  0601          DEC   R1               DECREMENT BIT COUNT, AND
0304  01CA  16F8          JNE   FLL4               LOOP BACK IF BYTE NOT DONE
0305  01CC  1005          JMP   FLL7
0306        01CE' FLL6    EQU   $
0307  01CE  0202          LI    R2,AHL           GET 'HL'
      01D0  0018'
0308  01D2  DE12          MOVB  *R2,*BFRA+       FILL BYTE WITH H'S=MEM > END
0309  01D4  0601          DEC   R1               DEC COUNTER UNTIL END OF BYTE
0310  01D6  16FB          JNE   FLL6
0311        01D8' FLL7    EQU   $
0312  01D8  8248          C     BFRA,BFRE        BYTE DONE, CHECK END OF BFR,
0313  01DA  1AE6          JL    FLL3             AND JUMP BACK IF NOT
0314  01DC  045B          RT
```

```
0317                    * TITLE:    UNFILL
0318                    *           COMPARE BUFFER AND MEMORY
0319                    * REVISION: 03/15/76
0320                    *           ORIGINAL
0321                    * ABSTRACT:
0322                    *           GENERATE THE SAME FORMAT AS IN SUBROUTINE FILL.
0323                    *           INSTEAD OF STORING CONVERTED VALUES IN THE
0324                    *           BUFFER, COMPARE THEM TO THE VALUES ALREADY
0325                    *           THERE. IF THE ADDRESSES DO NOT MATCH RETURN TO
0326                    *           READ ANOTHER RECORD. IF STRINGS DO NOT MATCH
0327                    *           BRANCH TO A DISPLAY ROUTINE WHICH DISPLAYS EACH
0328                    *           STRING.
0329                    * COMPUTER: 990,ASSEMBLY
0330                    * CALLING SEQUENCE:
0331                    *           ENTRY:
0332                    *             R5=MEM =ADDRESS OF MEMORY DATA
0333                    *             R8=BFRA=ADDRESS OF OUTPUT BUFFER
0334                    *             R9=BFRE=ADDR. OF END OF OUTPUT BUFFER
0335                    *             R6=ROMADD=ADDRESS OF ROM WORD
0336                    *             R7=BIT =BIT DISPLACEMENT IN MEMORY WORD
0337                    *
0338                    *           EXIT:
0339                    *             R0=>R4
0340                    *             MEM =ADDRESS OF NEXT BYTE TO BE CONVERTED
0341                    *             BFRA=ADDR+1 OF LAST BYTE IN OUTPUT BUFFER
0342                    *             THE BNPF RECORD IS FORMATTED IN 'BFR'.
0343          01DE' UNFILL EQU  $                 ****ENTRY POINT****
0344                    *
0345                    *1  CALL BINDEC(MEM,BFRA)
0346                    *
0347                    *  SAVE RETURN ADDRESS
0348    01DE  C3CB            MOV  R11,R15
0349    01E0  0228            AI   BFRA,8              SKIP ADDRESSES
        01E2  0008
0350          01E4' UNFLL4 EQU  $
0351    01E4  8805            C    MEM,@ENDADD
        01E6  0014'
0352    01E8  1B3D            JH   UNFL13              IF MEM > END ADDR THEN EXIT
0353    01EA  0588            INC  BFRA                SKIP A BLANK IN BUFFER
0354    01EC  0201            LI   R1,4                BIT COUNTER IN R1
        01EE  0004
0355    01F0  C0F5            MOV  *MEM+,R3            SHIFT WORD TO BIT
0356    01F2  C007            MOV  BIT,R0              IF BIT WITHIN WORD NOT ZERO,
0357    01F4  1301            JEQ  UNFLL5
0358    01F6  0A03            SLA  R3,R0
0359          01F8' UNFLL5 EQU  $
0360    01F8  0202            LI   R2,AHL              ADDR. OF 'HL' IN R2
        01FA  0018'
0361    01FC  0A13            SLA  R3,1                BIT TO BE STORED IN ARRY STAT
0362    01FE  1801            JOC  UNFLL6
0363    0200  0582            INC  R2                    INC TO 'H' FOR NEG. BIT
0364          0202' UNFLL6 EQU  $
0365    0202  9E12            CB   *R2,*BFRA+          COMPARE 'H' OR 'L'
0366    0204  132A            JEQ  UNFL11              IF EQUAL CONTINUE COMPARISON
```

```
0367  0206  0645          DECT  MEM              GET PREVIOUS WORD ADDRESS
0368  0208  C045          MOV   MEM,R1
0369  020A  C115          MOV   *MEM,R4          GET MEMORY WORD
0370  020C  C0C7          MOV   BIT,R3
0371  020E  0200          LI    R0,12
      0210  000C
0372  0212  6003          S     R3,R0            GET COUNT TO SHIFT BIT STRING
0373  0214  0B04          SRC   R4,R0            TO RIGHT OF WORD THEN TO
0374  0216  0AC4          SLA   R4,12            LEFT--CLEARS REST OF WORD
0375  0218  06A0          BL    @DISPLY          AND DISPLAY
      021A  0268'
0376  021C  014D          DATA  MPRT
0377  021E  1023          JMP   UNEXIT           ESC RETURN
0378  0220  0201          LI    R1,'  '          R1 CONTAINS BLANK
      0222  2020
0379        0224' UNFLL7  EQU   $
0380  0224  0608          DEC   BFRA             DECREASE BUFFER ADDRESS
0381  0226  9058          CB    *BFRA,R1         UNTIL ADDRESS OF PREVIOUS
0382  0228  16FD          JNE   UNFLL7           ' ' IS ENCOUNTERED
0383  022A  0588          INC   BFRA             SKIP BLANK
0384  022C  0704          SETO  R4               BUFFER BYTE VALUE TO BE IN R4
0385  022E  0202          LI    R2,AHL           ADDR OF 'HL' IN R2
      0230  0018'
0386        0232' UNFLL8  EQU   $
0387  0232  9488          CB    *BFRA+,*R2       IF BUFFER CONTAINS 'H',
0388  0234  1302          JEQ   UNFLL9           JUMP TO STORE A 1
0389  0236  0A14          SLA   R4,1             SHIFT 0 INTO RIGHTMOST BIT R4
0390  0238  1001          JMP   UNFL10
0391        023A' UNFLL9  EQU   $
0392  023A  0BF4          SRC   R4,15            SHIFT 1 INTO RIGHTMOST BIT R4
0393        023C' UNFL10  EQU   $
0394  023C  9058          CB    *BFRA,R1         IF NOT END OF BYTE,
0395  023E  16F9          JNE   UNFLL8           CONTINUE CONVERSION
0396  0240  0AC4          SLA   R4,12            GET BYTE VALUE
0397  0242  C046          MOV   ROMADD,R1
0398  0244  04C3          CLR   R3
0399  0246  06A0          BL    @DISPLY          AND DISPLAY
      0248  0268'
0400  024A  0154          DATA  TPRT
0401  024C  100C          JMP   UNEXIT           ESC RETURN
0402  024E  05C5          INCT  MEM              GET NEXT WORD ADDRESS
0403  0250  028D          CI    R13,4
      0252  0004
0404  0254  1601          JNE   UNF10A
0405  0256  04CD          CLR   R13
0406        0258' UNF10A  EQU   $
0407  0258  1002          JMP   UNFL12
0408        025A' UNFL11  EQU   $
0409  025A  0601          DEC   R1               DECREMENT BIT COUNT, AND
0410  025C  16CD          JNE   UNFLL5
0411        025E' UNFL12  EQU   $
0412  025E  0586          INC   ROMADD           GET NEXT ROM BYTE ADDRESS
0413  0260  8248          C     BFRA,BFRE        BYTE DONE, CHECK END OF BFR,
0414  0262  1AC0          JL    UNFLL4           AND JUMP BACK IF NOT
0415        0264' UNFL13  EQU   $
```

```
0416  0264  05CF          INCT R15                  NORMAL RETURN TO CALLER
0417        0266! UNEXIT EQU  $
0418  0266  045F          B    *R15
```

```
0421                    * TITLE:    DISPLY
0422                    *           DISPLAY STRING
0423                    * REVISION: 03/15/76
0424                    *           ORIGINAL
0425                    * ABSTRACT:
0426                    *           DISPLAY STRING AS BYTE ADDRESS,BIT-STRING
0427                    * COMPUTER: 990,ASSEMBLY
0428                    * CALLING SEQUENCE:
0429                    *           ENTRY:
0430                    *             R1 CONTAINS ADDRESS
0431                    *             R14 CONTAINS STRING
0432                    *           EXIT:
0433                    *             R1,R10,R0 DESTROYED
0434                    *             BFRE SAVED
0435          0268'     DISPLY EQU  $               ****ENTRY POINT****
0436  026P    C08B              MOV   R11,R2            SAVE RETURN ADDRESS
0437  026A    C389              MOV   BFRE,R14          SAVE BFRE
0438  026C    C34D              MOV   R13,R13
0439  026E    1603              JNE   DISP1
0440                    *  CARRIAGE RETURN, LINE FEED
0441  0270    06A0              BL    @PRCRLF
      0272    0000
0442  0274    101D              JMP   DSEXIT            ESC RETURN
0443          0276'     DISP1   EQU   $
0444  0276    058D              INC   R13
0445  0278    C282              MOV   R2,R10            GET TYPE
0446  027A    05C2              INCT  R2
0447                    *  CALL PRINT(TYPE)
0448  027C    06A0              BL    @PRNTC
      027E    0000
0449  0280    1017              JMP   DSEXIT            ESC RETURN
0450  0282    C281              MOV   R1,R10            GET ADDRESS
0451                    *  CALL PRINT(ADDR)
0452  0284    06A0              BL    @PRNTHN
      0286    0000
0453  0288    1013              JMP   DSEXIT            ESC RETURN
0454  028A    020A              LI    R10,PRTPRD
      028C    0000
0455                    *   CALL PRINT(.)
0456  028E    06A0              BL    @PRNTC
      0290    027E'
0457  0292    100E              JMP   DSEXIT            ESC RETURN
0458  0294    C283              MOV   R3,R10            GET BIT VALUE
0459                    *   CALL PRINT(BIT)
0460  0296    06A0              BL    @PRNTHN
      0298    0286'
0461  029A    100A              JMP   DSEXIT            ESC RETURN
0462  029C    020A              LI    R10,EQSIGN
      029E    0000
0463                    *  CALL PRINT(=)
0464  02A0    06A0              BL    @PRNTC
      02A2    0290'
0465  02A4    1005              JMP   DSEXIT            ESC RETURN
0466  02A6    C284              MOV   R4,R10            GET FOUR-BIT STRING VALUE
```

```
0467                    *  CALL PRINT(STRING)
0468  02A8  06A0               BL    @PRNTHX
      02AA  0000
0469  02AC  1001               JMP   DSEXIT
0470  02AE  05C2               INCT  R2                  NORMAL RETURN TO CALLER
0471        02B0' DSEXIT        EQU   $
0472  02B0  C24E               MOV   R14,BFRE            RESTORE BFRE
0473  02B2  0452               B     *R2
```

```
0476                    * TITLE     :BLNK
0477                    *            BLANK BUFFER
0478                    * REVISION: 03/15/76
0479                    *            ORIGINAL
0480                    * ABSTRACT:
0481                    *            SIMPLE ROUTNE TO BLANK FILL THE BUFFER
0482                    * COMPUTER: 990,ASSEMBLY
0483                    * CALLING SEQUENCE:
0484                    *            ENTRY:
0485                    *                 R8=BFRA=ADDRESS OF BFR
0486                    *                 R9=BFRE=ADDRESS OF END OF BUFFER
0487                    *            EXIT :
0488                    *                 R0 CLOBBERED
0489                    *                 ALL OTHER REGS INTACT
0490                    *                 BUFFER IS BLANK FILLED
0491                    *
0492          02B4'     BLNK      EQU   $              ****ENTRY POINT****
0493                    *
0494    02B4  0200              LI    R0,'  '              INITIALIZATION VALUE
        02B6  2020
0495          02B8'     BLNK1     EQU   $
0496    02B8  CE00              MOV   R0,*BFRA+            STORE INIT VALUE IN BFR
0497    02BA  8248              C     BFRA,BFRE            CHECK FOR END OF BUFFER AND
0498    02BC  12FD              JLE   BLNK1                  REPEAT IF NOT
0499    02BE  0208              LI    BFRA,BFR             RESET BFRA TO START OF BUFFER
        02C0  001A'
0500    02C2  045B              RT
```

```
 0503                    * TITLE     :WRTRD
 0504                    *            WRITE/READ
 0505                    * REVISION: 03/15/76
 0506                    *            ORIGINAL
 0507                    * ABSTRACT:
 0508                    *            FLUSH THE BUFFER AS DESCRIBED BY THE PRB
 0509                    *            SET AU SATAUS .EQ. FOR GOOD COMPLETION
 0510                    *                         .NE. FOR I/O ABORT
 0511                    *            R10 WILL CONTAIN THEAPPROPRIATE ERROR MESSAGE
 0512                    *            ID IN CASE OF AN ABORT.
 0513                    *
 0514                    * COMPUTER: 990,ASSEMBLY
 0515                    *CALLING SEQUENCE:
 0516                    *            ENTRY:
 0517                    *                 R10=PRBA=ADDRESS OF PRB
 0518                    *            EXIT:
 0519                    *                 R0,R10 CLOBBERED
 0520                    *                 ALL OTHER REGS INTACT
 0521                    *1= SUBROUTINE WRTRD(PRBA,AU=TAT,ERCD)
 0522          02C4'     WRTRD  EQU  $              ****ENTRY POINT****
 0523   02C4   020A             LI   PRBA,PRB
        02C6   0006'
 0524   02C8   0420             BLWP @SVCALT
        02CA   00FA'
 0525   02CC   020A             LI   R10,MX01          GET I/O ABRT MSG(JUST IN CASE
        02CE   0001
 0526   02D0   D020             MOVB @PRBSF,R0         SET AU STATUS WITH SYSTEM FLG
        02D2   000A'
 0527   02D4   045B             RT
 0528                            END
0000 ERS
```

960 = 980 CONCORDANCE

$		0046	0060	0062	0067	0104	0123	0144	0146	0150
		0170	0176	0192	0213	0230	0258	0266	0286	0288
		0297	0306	0311	0343	0350	0359	0364	0379	0386
		0391	0393	0406	0408	0411	0415	0417	0435	0443
		0471	0492	0495	0522					
ABORT	0146	0122	0129	0135	0142	0158	0200	0211	0221	
ACL		0017	0106							
ACT	0089	0117	0118	0119	0121	0173	0186			
AHL	0067	0298	0307	0360	0385					
BFR	0069	0052	0182	0193	0205	0216	0499			
BFRA	0095	0193	0205	0216	0271	0280	0287	0289	0302	0308
		0312	0349	0353	0365	0380	0381	0383	0387	0394
		0413	0496	0497	0499					
BFRE	0096	0182	0312	0413	0437	0472	0497			
BIT	0094	0137	0140	0141	0294	0356	0370			
BLANK	0066									
BLNK	0492	0194								
BLNK1	0495	0498								
CHARCT	0062	0172								
CPRM	0087	0105	0108	0114	0117	0124	0127	0133	0140	
DISP1	0443	0439								
DISPLY	0435	0375	0399							
DMH00	0123	0120								
DMH01	0144	0112	0126	0132						
DMH01A	0150	0139	0143							
DMH02	0170									
DMH02A	0176	0174								
DMH03	0192	0204								
DMH04	0213	0187	0228							
DMHL	0104	0014	0027							
DMHTRM	0230	0148	0212	0223	0226					
DP03	0034	0153								
DSEXIT	0471	0442	0449	0453	0457	0461	0465	0469		
END	0092	0133	0134	0136	0152	0183				
ENDADD	0060	0183	0291	0351						
EQSIGN		0021	0462							
ERROR		0015	0147							
FILL	0258	0197								
FLL1	0266	0276	0285							
FLL2	0286	0278								
FLL3	0288	0313								
FLL4	0297	0295	0304							
FLL5	0302	0300								
FLL6	0306	0292	0310							
FLL7	0311	0305								
HDR	0088	0108	0111	0125	0131	0138				
HYPHON	0059	0279								
MEM	0091	0291	0293	0351	0355	0367	0368	0369	0402	
MP00	0035	0113								
MPRT	0064	0376								
MS05	0032	0145								
MX01	0033	0525								
OPEN	0041	0171	0175							
PRB	0046	0178	0523							
PRBA	0097	0178	0523							
PRBBF	0052									
PRBCC	0054	0172								
PRBLN	0053									
PRBLU	0049									
PRBOP	0048	0177	0181	0208						

```
PRBSC   0047
PRBSF   0050        0526
PRBUF   0051
PRCRLF              0018  0441
PRNTC               0019  0448  0456  0464
PRNTHN              0020  0452  0460
PRNTHX              0022  0468
PRTPRO              0024  0454
R0      0085        0114  0115  0116  0124  0152  0154  0157  0207  0208
                    0219  0220  0267  0268  0269  0270  0271  0294  0296
                    0356  0358  0371  0372  0373  0494  0496  0526
R1      0086        0171  0175  0177  0180  0181  0219  0222  0263  0279
                    0280  0281  0282  0290  0303  0309  0354  0368  0378
                    0381  0394  0397  0409  0450
R10     0098        0105  0113  0145  0153  0445  0450  0454  0458  0462
                    0466  0525
R11     0099        0348  0436
R13     0100        0107  0185  0231  0403  0405  0438  0438  0444
R14     0101        0262  0277  0284  0437  0472
R15     0102        0348  0416  0418
R2      0164        0272  0273  0274  0275  0275  0298  0301  0302  0307
                    0308  0360  0363  0365  0385  0387  0436  0445  0446
                    0470  0473
R3      0165        0265  0268  0274  0283  0293  0296  0299  0355  0358
                    0361  0370  0372  0398  0458
R4      0215        0264  0273  0369  0373  0374  0384  0389  0392  0396
                    0466
READA   0040        0175
ROMADD  0093        0184  0202  0203  0263  0281  0397  0412
RR                  0016  0232
SAVERG  0058        0107  0231
STRT    0090        0127  0128  0130  0154
SVCALT              0023  0179  0524
TPRT    0065        0400
UNEXIT  0417        0377  0401
UNF10A  0406        0404
UNFILL  0343        0225
UNFL10  0393        0390
UNFL11  0408        0366
UNFL12  0411        0407
UNFL13  0415        0352
UNFLL4  0350        0414
UNFLL5  0359        0357  0410
UNFLL6  0364        0362
UNFLL7  0379        0382
UNFLL8  0386        0395
UNFLL9  0391        0388
WEOF    0042        0207
WRITA   0039        0171
WRTRO   0522        0199  0210  0218

THERE ARE   0095  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV	
SHEET	

REV STATUS OF SHEETS

REV	
SHEET	

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, RUNPGP, RUN USER PROGRAM, PROTOTYPE-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945326-9901

SCALE	REV.	SHEET 1 of 7

TI-0023-C

```
0003                *   PROCEDURE RUN(CPL);
0004                *      /* RUN CONTROLS THE EXECUTION OF
0005                *         A USER PROGRAM UNDER EITHER SIE
0006                *         OR TRACE MODE. IT PROVIDES THE
0007                *         USER'S KEYBOARD LEVEL CONTROL
0008                *         OF THE SYSTEM.
0009                *      */
0010                *      IF CPL.PARM1 .NE. NULL THEN DO;
0011                *         INSTRUCTION COUNT = CPL.PARM1;
0012                *      END;
0013                *      DO UNITL RETURN TO OPERATOR;
0014                *         CALL SCAN TABLE(TRACE,USER PC,FOUND);
0015                *         IF .NOT. FOUND THEN DO;
0016                *            CALL SIE(USER PC,USER WP,USER ST);
0017                *         END;ELSE DO;
0018                *            IF TRACE NE RESIDENT THEN ERROR EXIT;
0019                *            CALL TRACER(USER PC,USER WP,USER ST);
0020                *         END;
0021                *         IF WRITE PROTECT ERROR = 1 THEN ERROR EXIT
0022                *         IF USER PC < LOW LIMIT .OR. USER PC >
0023                *            HIGH LIMIT THEN DO;
0024                *            INSTRUCTION COUNT = INSTRTUCTION COUNT -1;
0025                *            CALL SCAN TABLE(BREAKPOINT,USER PC,FOUND);
0026                *            IF FOUND THEN DO;
0027                *               CALL BREAKPOINT PROCESS(USER PC,BREAK #,
0028                *                  BREAK ENTRY;TERM);
0029                *            IF TERM THEN SIGNAL RETURN TO OPERATOR
0030                *            END;
0031                *            IF INSTRUCTION COUNT .EQ. 0  THEN
0032                *               SIGNAL RETURN TO OPERATOR;
0033                *            IF INSTRUCTION COUNT .EQ. -1 THEN
0034                *               INSTRUCTION COUNT = INSTRUCTION COUNT +1;
0035                *            CALL MONITOR CHAR(CHAR,NO WAIT);
0036                *            IF CHAR .EQ. ESCAPE THEN SIGNAL
0037                *               RETURN TO OPERATOR;
0038                *         END;
0039                *      END;
0040                *      RETURN TO OPERATOR;
0041                *   END RUN;
0042                       IDT  'RUNPGP'
0043                * TITLE:    RUN
0044                *           RUN USER PGM UNDER DEBUG
0045                * REVISION:
0046                *           ORIGINAL
0047                * COMPUTER: 990,ASM
0048                * ABSTRACT: THIS ROUTINE CONTROLS THE DEBUG EXECUTION
0049                *           OF A USER'S PROGRAM.
0050                * CALLING SEQUENCE:
0051                *           BL @RUN
0052                *           R10  = PTR TO COMMOND PARAMETER LIST
0053                *
0054                *      REF'S AND DEF'S
0055                *
0056                       DEF  RUN
```

```
0057                        REF  GETBUF
0058                        REF  RETBUF
0059                        REF  LWP
0060                        REF  RWP
0061                        REF  ACL
0062                        REF  RR
0063                        REF  INSCNT              INSTRUCTION COUNT
0064                        REF  TRACE
0065                        REF  ERROR
0066                        REF  BKPT
0067                        REF  BKPTPR
0068                        REF  PCOUT
0069                        REF  LOWLIM
0070                        REF  HILIM
0071                        REF  ESC
0072                        REF  SIE
0073                        REF  TRACER
0074                        REF  SREGN
0075                        REF  MONCHR
0076                        REF  USRPC
0077                        REF  SCTB
0078                        REF  STRPRT
0079                        REF  CRUPRT
0080                  *
0081                  *WORKSPACE REGISTER DEFINITIONS
0082                  *
0083          0000    R0    EQU  0
0084          0001    R1    EQU  1
0085          0002    R2    EQU  2
0086          0003    R3    EQU  3
0087          0004    R4    EQU  4
0088          0005    R5    EQU  5
0089          0006    R6    EQU  6
0090          0007    R7    EQU  7
0091          0008    R8    EQU  8
0092          0009    R9    EQU  9
0093          000A    R10   EQU  10
0094          000B    R11   EQU  11
0095          000C    R12   EQU  12
0096          000D    R13   EQU  13
0097          000E    R14   EQU  14
0098          000F    R15   EQU  15
0099                  *
0100                  *
0101          0004    MX04  EQU  >0004               TRY TO EX TRACE WHEN NOT IN O
0102          0007    MX07  EQU  >0007               WRITE PROTECT VIOLATION
0103          0001    PRTVIO EQU 1
0104                  *
0105                  *                              1=PROCEDURE RUN(CPL);
0106          0000'   RUN   EQU  $
0107                  *     *ALLOC,COPY,LINK
0108  0000    0420          BLWP @ACL
      0002    0000
0109                  *                              2=/* RUN CONTROLS THE EXECUTIO
0110                  *                              2=   A USER PROGRAM UNDER EITH
```

```
0111                    *                                  2=   OR TRACE MODE, IT PROVIDE
0112                    *                                  2=   USER'S KEYBOARD LEVEL CON
0113                    *                                  2=   OF THE SYSTEM.
0114                    *                                  2=*/
0115                    *                                  2=IF CPL.PARM1 .NE. NULL THEN
0116  0004  C27A                MOV   *R10+,R9
0117  0006  0A99                SLA   R9,9
0118  0008  1702                JNC   RUN010
0119                    *                                  3=INSTRUCTION COUNT = CPL.PARM
0120  000A  C81A                MOV   *R10,@INSCNT
      000C  0000
0121                    *                                  2=END;
0122          000E'     RUN010  EQU   $
0123                    *                                  2=DO UNITL RETURN TO OPERATOR;
0124                    *                                  3=CALL SCAN TABLE(TRACE,USER P
0125  000E  020A                LI    R10,TRACE
      0010  0000
0126  0012  C260                MOV   @USRPC,R9
      0014  0000
0127  0016  06A0                BL    @SCTB
      0018  0000
0128                    *                                  3=IF .NOT. FOUND THEN DO;
0129  001A  C28A                MOV   R10,R10               0 => FOUND
0130  001C  1303                JEQ   RUN020
0131          001E'     RUN015  EQU   $
0132                    *                                  4=CALL SIE(USER PC,USER WP,USE
0133  001E  06A0                BL    @SIE
      0020  0000
0134                    *                                  3=END;ELSE DO;
0135  0022  1008                JMP   RUN030
0136          0024'     RUN020  EQU   $
0137                    **********CHECK THAT TRACE OVERLAY IS PRESENT
0138                    *                                  4=IF TRACE NE RESIDENT THEN ER
0139  0024  C2A0                MOV   @SREGN,R10
      0026  0000
0140  0028  1330                JEQ   RUN055
0141                    *                                  4=CALL TRACER(USER PC,USER WP,
0142  002A  06A0                BL    @TRACER
      002C  0000
0143                    *                                  3=END;
0144  002E  1031                JMP   RUN060
0145  0030  06A0                BL    @PCOUT                OUTPUT USRPC TO PANEL
      0032  0000
0146          0034'     RUN030  EQU   $
0147                    *                                  3=IF WRITE PROTECT ERROR = 1 T
0148  0034  020A                LI    R10,MX07
      0036  0007
0149  0038  020C                LI    R12,CRUPRT
      003A  0000
0150  003C  1F01                TB    PRTVIO
0151  003E  1603                JNE   RUN032
0152  0040  3020                LDCR  @STRPRT,0             RESET VIO FLAG AND RESTORE PR
      0042  0000
0153  0044  1024                JMP   RUN057                ERROR EXIT
0154          0046'     RUN032  EQU   $
```

```
0155  0046  C260          MOV   @USRPC,R9
      0048  0014'
0156                *                          3=IF USER PC < LOW LIMIT ,OR,
0157                *                          4=HIGH LIMIT THEN DO;
0158  004A  8809          C     R9,@LOWLIM
      004C  0000
0159  004E  1A03          JL    RUN035
0160  0050  8809          C     R9,@HILIM
      0052  0000
0161  0054  1ADC          JL    RUN010
0162        0056' RUN035 EQU   $
0163                *                          4=INSTRUCTION COUNT = INSTRTUC
0164  0056  0620          DEC   @INSCNT
      0058  000C'
0165                *                          4=CALL SCAN TABLE(BREAKPOINT,U
0166  005A  020A          LI    R10,BKPT
      005C  0000
0167  005E  06A0          BL    @SCTB
      0060  0018'
0168                *                          4=IF FOUND THEN DO;
0169  0062  C28A          MOV   R10,R10
0170  0064  1604          JNE   RUN040
0171                *                          5=CALL BREAKPOINT PROCESS(USER
0172                *                          6=BREAK ENTRY;TERM);
0173  0066  06A0          BL    @BKPTPR
      0068  0000
0174                *                          4=IF TERM THEN SIGNAL RETURN T
0175  006A  C28A          MOV   R10,R10
0176  006C  1612          JNE   RUN060
0177                *                          4=END;
0178        006E' RUN040 EQU   $
0179                *                          4=IF INSTRUCTION COUNT ,EQ, 0
0180                *                          5=SIGNAL RETURN TO OPERATOR;
0181  006E  C020          MOV   @INSCNT,R0
      0070  0058'
0182  0072  130F          JEQ   RUN060
0183                *                          4=IF INSTRUCTION COUNT ,EQ, -1
0184                *                          5=INSTRUCTION COUNT = INSTRUCT
0185  0074  0280          CI    R0,-1          NO INSTRUCTION LIMIT
      0076  FFFF
0186  0078  1602          JNE   RUN050
0187  007A  05A0          INC   @INSCNT
      007C  0070'
0188        007E' RUN050 EQU   $
0189                *                          4=CALL MONITOR CHAR(CHAR,NO WA
0190  007E  06A0          BL    @MONCHR
      0080  0000
0191                *                          4=IF CHAR ,EQ, ESCAPE THEN SIG
0192                *                          5=RETURN TO OPERATOR;
0193  0082  9809          CB    R9,@ESC
      0084  0000
0194  0086  16C3          JNE   RUN010
0195  0088  1004          JMP   RUN060
0196                *                          3=END;
0197                *                          2=END;
```

```
 0198          008A' RUN055 EQU  $
 0199  008A  020A           LI   R10,MX04
       008C  0004
 0200          008E' RUN057 EQU  $
 0201  008E  06A0           BL   @ERROR
       0090  0060
 0202                *                                  2=RETURN TO OPERATOR;
 0203          0092' RUN060 EQU  $
 0204                *                                  1=END RUN;
 0205                *     *LINK TO PREV WKSP, RET CURR WKSP
 0206  0092  0420           BLWP @RR
       0094  0000
 0207  0096  0458           RT
 0208                       END
0000 ERS
```

960 - 980 CONCORDANCE

$		0106	0122	0131	0136	0146	0154	0162	0178	0188
		0198	0200	0203						
ACL		0061	0108							
BKPT		0066	0166							
BKPTPR		0067	0173							
CRUPRT		0079	0149							
ERROR		0065	0201							
ESC		0071	0193							
GETBUF		0057								
HILIM		0070	0160							
INSCNT		0063	0120	0164	0181	0187				
LOWLIM		0069	0158							
LWP		0059								
MONCHR		0075	0190							
MX04	0101	0199								
MX07	0102	0148								
PCOUT		0068	0145							
PRTVID	0103	0150								
R0	0083	0181	0185							
R1	0084									
R10	0093	0116	0120	0125	0129	0129	0139	0148	0166	0169
		0169	0175	0175	0199					
R11	0094									
R12	0095	0149								
R13	0096									
R14	0097									
R15	0098									
R2	0085									
R3	0086									
R4	0087									
R5	0088									
R6	0089									
R7	0090									
R8	0091									
R9	0092	0116	0117	0126	0155	0158	0160	0193		
RETBUF		0058								
RR		0062	0206							
RUN	0106	0056								
RUN010	0122	0118	0161	0194						
RUN015	0131									
RUN020	0136	0130								
RUN030	0146	0135								
RUN032	0154	0151								
RUN035	0162	0159								
RUN040	0178	0170								
RUN050	0188	0186								
RUN055	0198	0140								
RUN057	0200	0153								
RUN060	0203	0144	0176	0182	0195					
RWP		0060								
SCTB		0077	0127	0167						
SIE		0072	0133							
SREGN		0074	0139							
STRPRT		0078	0152							
TRACE		0064	0125							
TRACER		0073	0142							
USRPC		0076	0126	0155						

THERE ARE 0055 SYMBOLS

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV
SHEET

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN — DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, CMSTPP, COMMAND STRING PROCESSOR, PROTOTYPE-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945330-9901

SCALE REV. SHEET 1 of 13

TI-0023-C

```
0003              *    PROCEDURE COMMAND STRING PROCESSOR
0004              *         DECLARE
0005              *            1 BUFFER  CONTROL  PTR,
0006              *              2 CHARCNT  FIXED(8),
0007              *              2 TERMCHAR CHAR(1),
0008              *              2 CSTRING(30) CHAR(1);
0009              *         DECLARE CHAR  CHAR(1);
0010              *       DO FOREVER;
0011              *         IF WRITEPROTECTERROR THEN CALL ERROR;
0012              *         CALL PRINT(PROMPT);
0013              *         CALL GETFIELD(PTR);
0014              *            DO I=0 UNTIL CMDFOUND OR CMDNOTFOUND
0015              *            IF PTR.CHARCNT <2 THEN SIGNAL CMDNOTFOUND;
0016              *               IF COMMANDTABLE(I)=PTR. CMD THEN
0017              *                  SIGNAL CMDFOUND ;
0018              *               I=I+1;
0019              *               IF I>MAXCOMMAND THEN SIGNAL CMDNOTFOUND;
0020              *            END;
0021              *            CMDNOTFOUND: DO;
0022              *               CALL RETBUF(PTR);
0023              *               CALL ERROR(CMDERROR);
0024              *            END;
0025              *            CMDFOUND: DO;
0026              *               CALL RETBUF(PTR);
0027              *               CALL GETBUF(CPLPTR);
0028              *               CPLPARMCOUNT = 0;
0029              *               CPLBITS = 0;
0030              *               CPLCURRPTR = CPLPTR + 2;
0031              *               PARMINDEX=0;
0032              *               DO UNTIL ENDOFRECORD OR PARMERROR;
0033              *                  IF TERMCHAR = 'EOR' THEN SIGNAL
0034              *                     ENDOFRECORD;
0035              *                  IF TERMCHAR = 'ESC' THEN SIGNAL ESCAPE
0036              *                  PARM = COMMANDTABLEPARM(PARMINDEX);
0037              *                     PARM=HAB: DO;
0038              *                     CALL GETHEXVALUE(VALUE,TERMCHAR);
0039              *                        C(CPLCURRPTR) = VALUE;
0040              *                        CPLCURRPTR = CPLCURRPTR +2;
0041              *                        CALL CPLPARMMARK(CPLPTR,PARMIN
0042              *                     END;
0043              *                     PARM=STR: DO;
0044              *                        CALL GETFIELD(PTR);
0045              *                         IF PTR.CHARCOUNT+2  + CPLCURRP
0046              *                         - CPLPTR > 30 THEN SIGNAL PARME
0047              *                        CCNT = PTR.CHARCOUNT
0048              *                        IF CCNT > 0 THEN DO;
0049              *                        P1 = PTR;
0050              *                        CPLCURRPTR.CHAR = P1.CHARCNT;
0051              *                        P1 = P1+2;           /* SKIP TERM C
0052              *                        CPLCURRPTR = CPLCURRPTR + 1;
0053              *                        DO WHILE CCNT >= 0;
0054              *                           P1.CHAR=CPLCURRPTR.CHAR;
0055              *                           P1=P1+1;
0056              *                           CPLCURRPTR = CPLCURRPTR+1;
```

```
0057                *                           CCNT = CCNT -1;
0058                *                        END
0059                *                        CPLCURRPTR = SLA(SRL(CPLCURRPT
0060                *                             ),1),1)
0061                *                        CALL CPLPARMMARK(CPLPTR,PARMIN
0062                *                        END;
0063                *                        CALL RETBUF(PTR);
0064                *                     END;
0065                *                     PARM-OTHER: DO;
0066                *                        /*DON'T PUSH INDEX PAST END OF PAR
0067                *                        PARMINDEX = PARMINDEX -1;
0068                *                        TERMCHAR = 'EOR';
0069                *                     END;
0070                *                  END CASE;
0071                *                  PARMINDEX=PARMINDEX+1;
0072                *               END
0073                *                ENDRECORD: IF PTR .NE. 0 THEN DO;
0074                *                  CALL COMMANDSERVICEROUTINE(CPL)
0075                *               END;
0076                *               ELSE MISSINGOVLY: DO;
0077                *                  CALL ERROR(MISSOV);
0078                *               END
0079                *               PARMERROR: DO;
0080                *                  CALL ERROR(PARAMETERERROR);
0081                *               END;
0082                *               ESCAPE;
0083                *               CALL RETBUF(CPL)
0084                *            END;
0085                *         END;
0086                *      END;
0087                *      PROCEDURE CPLPARMMARK(CPLPTR,PARMINDEX);
0088                *         /* CPLPARM WILL INCREMENT
0089                *            THE PARAMETER COUNT AND SET
0090                *            THE PARAMETER PRESENCE BIT
0091                *            FOR THE PARAMETER CORRESPONDING
0092                *            TO PARMINDEX.
0093                *         */
0094                *         TEMP = SRL( '80',PARM)INDEX);
0095                *         CPLPTR.BITS = CPLPTR.BITS .PR. TEMP;
0096                *         CPLPTR.PARMCOUNT = CPLPTR.PARMCOUNT+1;
0097                *      END CPLPARMMARK;
0098                *   END CMDSTRPROCESSOR;
0099                      IDT  'CMSTPP'
0100                * TITLE:    CMSTPP
0101                *           COMMAND STRING PROCESSOR
0102                * REVISION:
0103                *           ORIGINAL
0104                * COMPUTER: 990,ASM
0105                * ABSTRACT: THE COMMAND STRING PROCESSOR PROVIDES
0106                *           THE SYSTEM KEYBOARD INTERFACE. THE
0107                *           PROCESSOR ACCEPTS STRINGS FROM THE
0108                *           KEYBOARD WHICH ARE PARSED IN ACCORDANCE
0109                *           WITH THE COMMAND DEFINITION TABLE. WHEN
0110                *           A VALID COMMAND STRING IS INPUT, A COMMAND
0111                *           PARAMETER TABLE IS BUILT WHICH CONTAINS
```

```
0112                *          THE INPUT PARAMETERS SUITABLY TRANSLATED.
0113                * CALLING SEQUENCE:
0114                *           BL   @ CSP
0115                *
0116                *WORKSPACE REGISTER DEFINITIONS
0117                *
0118         0000   R0     EQU  0
0119         0001   R1     EQU  1
0120         0002   R2     EQU  2
0121         0003   R3     EQU  3
0122         0004   R4     EQU  4
0123         0005   R5     EQU  5
0124         0006   R6     EQU  6
0125         0007   R7     EQU  7
0126         0008   R8     EQU  8
0127         0009   R9     EQU  9
0128         000A   R10    EQU  10
0129         000B   R11    EQU  11
0130         000C   R12    EQU  12
0131         000D   R13    EQU  13
0132         000E   R14    EQU  14
0133         000F   R15    EQU  15
0134                *
0135                *      REF'S AND DEF'S
0136                       REF  PROMPT       MONITOR PROMPT STRING
0137                       REF  CRUOFF
0138                       REF  USRPC
0139                       REF  PRINT
0140                       REF  GETFLD
0141                       REF  PRNTC
0142                       REF  CMDTBL       COMMAND PARAMETER LIST
0143                       REF  NOCMDS       NUMBER COMMANDS
0144                       REF  PCOUT
0145                       REF  GETHEX
0146                       REF  ERROR        PRINT ERROR SUBROUTINE
0147                       REF  TERMCR
0148                       REF  EOR          CHARS
0149                       REF  BLANK          .
0150                       REF  COMMA          .
0151                       REF  PLUS           .
0152                       REF  MINUS          .
0153                       REF  ESC
0154                       REF  GETCHR
0155                       REF  CPLSAV
0156                       REF  CRUPRT
0157                       REF  STRPRT
0158                       DEF  CSP
0159                       REF  GETBUF
0160                       REF  RETBUF
0161                       REF  LWP
0162                       REF  RWP
0163                       REF  ACL
0164                       REF  RR
0165                *                              3-DECLARE
0166                *                              4-1 BUFFER  CONTROL  PTR.
```

```
                                                    5-2 CHARCNT  FIXED(8),
                                                    5-2 TERMCHAR CHAR(1),
                                                    5-2 CSTRING(30) CHAR(1);
                                                    3-DECLARE CHAR  CHAR(1);
                *
                *
                *
                *
                *
                *
                *
0000   CMDCDE EQU  0
0004   CMDRTN EQU  4
0006   CMDLEN EQU  6
0201   CMDERR EQU  >0201
0100   PRMERR EQU  >0100
0003   MISSOV EQU  >0003
0007   MX07   EQU  >0007                    WRITE PROTECT ERROR
0001   PRTVIO EQU  1                        PROTECT VIOLATION FLAG
0001   TRMCHR EQU  1
0002   STRNG  EQU  2
0001   SCERR  EQU  1
0000   PDSTR  EQU  0
0001   PHSTR  EQU  1
0002   PCSTR  EQU  2
0003   PNULL  EQU  3
000D   HAB    EQU  >D
000B   DAB    EQU >B
       *      REGISTER ASSIGNMENTS
       *      R2       PARAMETER TYPE
       *      R5       COMMANDTABLE - POINTER TO CURRENT
       *                 ENTRY IN COMMAND TABLE
       *      R6       PARMINDEX  INDEX IN PARAMETER LIST FOR
       *                 CURRENT COMMAND
       *      R8       COMMAND PARAMETER LIST POINTER
       *      R10      PTR      POINTER TO STRING RETURNED
       *                 BY GET FIELD
       *      R12      COMMAND PARM LIST CURRENT LOC
```

```
0202          0000' CSP     EQU  $                    NO PROLOGUE REQ'
0203                *                                 2-DO FOREVER;
0204  0000    06A0          BL   @PCOUT                OUTPUT USRPC TO PANEL
      0002    0000
0205                *                                 3-IF WRITEPROTECTERROR THEN
0206  0004    020A          LI   R10,MX07
      0006    0007
0207  0008    020C          LI   R12,CRUPRT
      000A    0000
0208  000C    1F01          TB   PRTVIO
0209  000E    1603          JNE  CSP005
0210  0010    3020          LDCR @STRPRT,0
      0012    0000
0211  0014    101F          JMP  CSP022
0212          0016' CSP005  EQU  $
0213                *                                 3-CALL PRINT(PROMPT);
0214  0016    020A          LI   R10,PROMPT
      0018    0000
0215  001A    06A0          BL   @PRNTC
      001C    0000
0216  001E    1000          NOP
0217                *                                 3-CALL GETFIELD(PTR);
0218  0020    06A0          BL   @GETFLD
      0022    0000
0219  0024    D82A          MOVB @TRMCHR(R10),@TERMCR
      0026    0001
      0028    0000
0220                *                                 4-DO I=0 UNTIL CMDFOUND OR CM
0221  002A    04C4          CLR  R4
0222  002C    0205          LI   R5,CMDTBL
      002E    0000
0223          0030' CSP010  EQU  $
0224                *                                 4-IF PTR.CHARCNT <2 THEN SIGN
0225  0030    D05A          MOVB *R10,R1              R10,POINTS TO STRING BUFFER
0226  0032    0981          SRL  R1,8                 RYTE 0 = @ CHARS
0227  0034    0641          DECT R1
0228  0036    110A          JLT  CSP020
0229                *                                 5-IF COMMANDTABLE(I)=PTR. CMD
0230                *                                 6-SIGNAL CMDFOUND ;
0231  0038    8A95          C    *R5,@2(R10)
      003A    0002
0232  003C    130E          JEQ  CSP025
0233                *                                 5-I=I+1;
0234  003E    0584          INC  R4
0235                *                                 5-IF I>MAXCOMMAND THEN SIGNAL
0236  0040    8804          C    R4,@NOCMDS
      0042    0000
0237  0044    1403          JHE  CSP020
0238                *                                 4-END;
0239  0046    0225          AI   R5,CMDLEN
      0048    0006
0240  004A    10F2          JMP  CSP010
0241                *                                 4-CMDNOTFOUND: DO;
0242          004C' CSP020  EQU  $
```

```
0243                    *                                  5-CALL RETBUF(PTR);
0244                    *     *RETBUF
0245  004C  0420              BLWP @RETBUF
      004E  0000
0246                    *                                  5-CALL ERROR(CMDERROR);
0247  0050  020A              LI   R10,CMDERR
      0052  0201
0248        0054' CSP022  EQU  $
0249  0054  06A0              BL   @ERROR
      0056  0000
0250                    *                                  4-END;
0251  0058  10D3              JMP  CSP
0252                    *                                  4-CMDFOUND: DO;
0253        005A' CSP025  EQU  $
0254                    *                                  5-CALL RETBUF(PTR);
0255                    *                                  5-CALL GETBUF(CPLPTR);
0256  005A  C20A              MOV  R10,R8                    RE-USE EXISTING BUFFER
0257                    *                                  5-CPLPARMCOUNT = 0;
0258                    *                                  5-CPLBITS = 0;
0259  005C  04D8              CLR  *R8
0260                    *                                  5-CPLCURRPTR = CPLPTR + 2;
0261  005E  C308              MOV  R8,R12
0262  0060  05CC              INCT R12
0263                    *                                  5-PARMINDEX=0;
0264  0062  04C6              CLR  R6
0265                    *                                  5-DO UNTIL ENDOFRECORD OR PA
0266        0064' CSP030  EQU  $
0267                    *                                  6-IF TERMCHAR = 'EOR' THEN SI
0268                    *                                  7-ENDOFRECORD;
0269  0064  9820              CB   @TERMCR,@EOR
      0066  0028'
      0068  0000
0270  006A  134E              JEQ  CSP090
0271                    *                                  6-IF TERMCHAR = 'ESC' THEN SI
0272  006C  9820              CB   @TERMCR,@ESC
      006E  0066'
      0070  0000
0273  0072  135B              JEQ  CSP105
0274                    *                                  6-PARM = COMMANDTABLEPARM(PA
0275  0074  C045              MOV  R5,R1
0276  0076  05C1              INCT R1
0277  0078  C006              MOV  R6,R0                     ISOLATE 2 BIT FIELD
0278  007A  0240              ANDI R0,7
      007C  0007
0279  007E  C091              MOV  *R1,R2                    BITS OF R2
0280  0080  0A10              SLA  R0,1
0281  0082  1301              JEQ  C2
0282  0084  0A02              SLA  R2,R0
0283        0086' C2      EQU  $
0284  0086  09E2              SRL  R2,14
0285        0088' CSP060  EQU  $
0286                    *                                  7-PARM=HAB: DO;
0287  0088  0282              CI   R2,PHSTR
      008A  0001
0288  008C  1611              JNE  CSP065
```

```
0289                    *                                  7-CALL GETHEXVALUE(VALUE,TER
0290  008E  06A0               BL    @GETHEX
      0090  0000
0291  0092  104B               JMP   CSP105                   IF INPUT ERROR
0292  0094  1001               JMP   CSP061                   IF VALUE INPUT
0293  0096  1009               JMP   CSP062                   IF NO VALUE INPUT
0294        0098' CSP061 EQU   $
0295  0098  C04C               MOV   R12,R1
0296  009A  6048               S     R8,R1
0297  009C  0221               AI    R1,-30
      009E  FFE2
0298  00A0  1540               JGT   CSP100
0299                    *                                  8-C(CPLCURRPTR) = VALUE;
0300  00A2  C70A               MOV   R10,*R12
0301                    *                                  8-CPLCURRPTR = CPLCURRPTR
0302  00A4  05CC               INCT  R12
0303                    *                                  8-CALL CPLPARMMARK(CPLPTR,P
0304  00A6  06A0               BL    @CPLPM
      00A8  0134'
0305        00AA' CSP062 EQU   $                             SAVE USER INPUT TERM CHAR
0306  00AA  D809               MOVB  R9,@TERMCR
      00AC  006E'
0307                    *                                  7-END;
0308  00AE  102A               JMP   CSP080
0309                    *                                  7-PARM=STR: DO;
0310        00B0' CSP065 EQU   $
0311  00B0  0282               CI    R2,PCSTR
      00B2  0002
0312  00B4  1623               JNE   CSP075
0313                    *                                  8-CALL GETFIELD(PTR);
0314  00B6  06A0               BL    @GETFLD
      00B8  0022'
0315  00BA  D82A               MOVB  @TRMCHR(R10),@TERMCR     SAVE TERMINATING CHAR
      00BC  0001
      00BE  00AC'
0316                    *                                  8- IF PTR.CHARCOUNT+2  + CPL
0317                    *                                  8- - CPLPTR > 30 THEN SIGNAL
0318  00C0  D05A               MOVB  *R10,R1
0319  00C2  0981               SRL   R1,8
0320  00C4  05C1               INCT  R1
0321  00C6  0911               SRL   R1,1
0322  00C8  0A11               SLA   R1,1
0323  00CA  A04C               A     R12,R1
0324  00CC  6048               S     R8,R1
0325  00CE  0221               AI    R1,-29
      00D0  FFE3
0326  00D2  1103               JLT   CSP068
0327                    *     *RETBUF
0328  00D4  0420               BLWP  @RETBUF
      00D6  004E'
0329  00D8  1024               JMP   CSP100
0330        00DA' CSP068 EQU   $
0331                    *                                  8-CCNT = PTR.CHARCOUNT
0332  00DA  D0DA               MOVB  *R10,R3
0333  00DC  0983               SRL   R3,8
```

```
0334                    *                       8-IF CCNT > 0 THEN DO;
0335  00DE  130B                JEQ   CSP073
0336                    *                       8-P1 = PTR;
0337  00E0  C08A                MOV   R10,R2
0338                    *                       8-CPLCURRPTR.CHAR = P1.CHAR
0339                    *                       8-P1 = P1+2;          /* SKIP
0340                    *                       8-CPLCURRPTR = CPLCURRPTR
0341  00E2  DF12                MOVB  *R2,*R12+
0342  00E4  05C2                INCT  R2
0343                    *                       8-DO WHILE CCNT >= 0;
0344        00E6' CSP070 EQU    $
0345                    *                       9-P1.CHAR=CPLCURRPTR.CHAR;
0346                    *                       9-P1=P1+1;
0347                    *                       9-CPLCURRPTR = CPLCURRPTR+
0348  00E6  DF32                MOVB  *R2+,*R12+
0349                    *                       9-CCNT = CCNT -1;
0350  00E8  0603                DEC   R3
0351                    *                       8-END
0352  00EA  13FD                JEQ   CSP070
0353  00EC  15FC                JGT   CSP070
0354                    *                       8-CPLCURRPTR = SLA(SRL(CPLC
0355                    *                       9-  ),1),1)
0356  00EE  091C                SRL   R12,1
0357  00F0  0A1C                SLA   R12,1
0358                    *                       8-CALL CPLPARMMARK(CPLPTR,P
0359  00F2  06A0                BL    @CPLPM
      00F4  0134'
0360                    *                       8-END;
0361                    *                       8-CALL RETBUF(PTR);
0362        00F6' CSP073 EQU    $
0363                    *     *RETBUF
0364  00F6  0420                BLWP  @RETBUF
      00F8  00D6'
0365                    *                       7-END;
0366  00FA  1004                JMP   CSP080
0367                    *                       7-PARM-OTHER: DO;
0368                    *                       8-/*DON'T PUSH INDEX PAST END
0369                    *                       8-PARMINDEX = PARMINDEX -1;
0370        00FC' CSP075 EQU    $
0371  00FC  0606                DEC   R6
0372                    *                       8-TERMCHAR = 'EOR';
0373  00FE  D820                MOVB  @EOR,@TERMCR
      0100  0068'
      0102  00BE'
0374        0104' CSP080 EQU    $
0375                    *                       7-END;
0376                    *                       6-END CASE;
0377                    *                       6-PARMINDEX=PARMINDEX+1;
0378  0104  0586                INC   R6
0379                    *                       5-END
0380  0106  10AE                JMP   CSP030
0381                    *                       5- ENDRECORD: IF PTR .NE. 0 T
0382        0108' CSP090 EQU    $
0383                    *                       6-CALL COMMANDSERVICEROUTINE
0384  0108  C1A5                MOV   @CMDRTN(R5),R6
```

```
      010A  0004
0385  010C  1307          JEQ   CSP095
0386  010E  C288          MOV   R8,R10
0387  0110  C808          MOV   R8,@CPLSAV          (REMEMBER CPL PTR)
      0112  0000
0388  0114  0696          BL    *R6
0389  0116  C2A0          MOV   @CPLSAV,R10
      0118  0112'
0390                 *                              5-END;
0391  011A  1008          JMP   CSP110
0392                 *                              5-ELSE MISSINGOVLY: DO;
0393        011C' CSP095 EQU   $
0394                 *                              6-CALL ERROR(MISSOV);
0395  011C  020A          LI    R10,MISSOV
      011E  0003
0396  0120  1002          JMP   CSP102
0397                 *                              5-END
0398                 *                              5-PARMERROR: DO;
0399        0122' CSP100 EQU   $
0400                 *                              6-CALL ERROR(PARAMETERERROR);
0401  0122  020A          LI    R10,PRMERR
      0124  0100
0402        0126' CSP102 EQU   $
0403  0126  06A0          BL    @ERROR
      0128  0056'
0404                 *                              5-END;
0405                 *                              5-ESCAPE:
0406        012A' CSP105 EQU   $
0407  012A  C288          MOV   R8,R10
0408        012C' CSP110 EQU   $
0409                 *                              5-CALL RETBUF(CPL)
0410                 *    *RETBUF
0411  012C  0420            BLWP @RETBUF
      012E  00F8'
0412                 *                              4-END;
0413                 *                              3-END;
0414                 *                              2-END;
0415  0130  0460          B     @CSP
      0132  0000'
```

```
0417                    *                                2-PROCEDURE CPLPARMMARK(CPL
0418                    *                                3-/* CPLPARM WILL INCREMENT
0419                    *                                3-   THE PARAMETER COUNT AND S
0420                    *                                3-   THE PARAMETER PRESENCE BI
0421                    *                                3-   FOR THE PARAMETER CORRESP
0422                    *                                3-   TO PARMINDEX.
0423                    *                                3-*/
0424                    *          REGISTER ASSIGNMENTS
0425                    *     R8   - POINTER TO COMMAND PARAMETER LIST
0426                    *     R6   - PARM INDEX
0427                    *     R0,R1- SCRATCH REG
0428                    *
0429          0134' CPLPM  EQU  $
0430                    *                                3-TEMP = SRL( '80',PARM)INDEX)
0431  0134  0201           LI   R1,>80
      0136  0080
0432  0138  C006           MOV  R6,R0
0433  013A  1301           JEQ  C4
0434  013C  0901           SRL  R1,R0
0435          013E' C4     EQU  $
0436                    *                                3-CPLPTR.BITS = CPLPTR.BITS
0437  013E  E601           SOC  R1,*R8
0438                    *                                3-CPLPTR.PARMCOUNT = CPLPTR
0439  0140  06D8           SWPB *R8
0440  0142  0598           INC  *R8
0441  0144  06D8           SWPB *R8
0442                    *                                2-END CPLPARMMARK;
0443  0146  045B           RT
0444                    *                                1-END CMDSTRPROCESSOR;
0445                        END
0000 ERS
```

960 - 980 CONCORDANCE

```
PNULL   0188
PRINT           0139
PRMERR  0178    0401
PRNTC           0141  0215
PROMPT          0136  0214
PRTVIO  0181    0208
R0      0118    0277  0278  0280  0282  0432  0434
R1      0119    0225  0226  0227  0275  0276  0279  0295  0296  0297
                0318  0319  0320  0321  0322  0323  0324  0325  0431
                0434  0437
R10     0128    0206  0214  0219  0225  0231  0247  0256  0300  0315
                0318  0332  0337  0386  0389  0395  0401  0407
R11     0129
R12     0130    0207  0261  0262  0295  0300  0302  0323  0341  0348
                0356  0357
R13     0131
R14     0132
R15     0133
R2      0120    0279  0282  0284  0287  0311  0337  0341  0342  0348
R3      0121    0332  0333  0350
R4      0122    0221  0234  0236
R5      0123    0222  0231  0239  0275  0384
R6      0124    0264  0277  0371  0378  0384  0388  0432
R7      0125
R8      0126    0256  0259  0261  0296  0324  0386  0387  0407  0437
                0439  0440  0441
R9      0127    0306
RETBUF          0160  0245  0328  0364  0411
RR              0164
RWP             0162
SCERR   0184
STRNG   0183
STRPRT          0157  0210
TERMCR          0147  0219  0269  0272  0306  0315  0373
TRMCHR  0182    0219  0315
USRPC           0138

THERE ARE   0087  SYMBOLS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																														
SHEET																														
REV STATUS OF SHEETS	REV																													
	SHEET																													

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DATE
DWN	
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, ASGLUN, ASSIGN LUNO-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945346-9901

SCALE	REV.	SHEET 1 of 5

TI-0023-C

```
0003                *       DO UNTIL ERROR OR DONE;
0004                *          IF CPL PARM COUNT .NE. 2 THEN
0005                *             SIGNAL ERROR;
0006                *          IF CPL PARM2(CHR COUNT) .NE. 3 THEN
0007                *             SIGNAL ERROR;
0008                *          CALL CDNDT(DEV NAME,DEV TYPE,DEV NUM);
0009                *          IF DEV TYPE<0 THEN SIGNAL ERROR;
0010                *          CALL SETLUN(LUNO,DEV TYPE,DEV NUM,ERRC);
0011                *          IF ERRC .EQ. 0 THEN SIGNAL DONE;
0012                *          SIGNAL ERROR;
0013                *       END;
0014                *       ERROR: DO;
0015                *          CALL ERROR(MX01)
0016                *       END;
0017                *       DONE: DO;END;
0018                *       RETURN;
0019                        IDT  'ASGLUN'
0020                * TITLE:     ASGLUN
0021                *            ASSIGN LUNO
0022                * REVISION:
0023                *            ORIGINAL
0024                * COMPUTER: 990,ASM
0025                * ABSTRACT: ACCEPTS AN INPUT COMMAND OF THE FORM:
0026                *             AL,<LUNO>,<DEVICE NAME>
0027                *            WHERE
0028                *               <LUNO> ::= LOG
0029                *                          CS1
0030                *                          CS2
0031                *                          DUM
0032                *            AFTER VALIDATING THE CORRECT NUMBER OF
0033                *            PARAMETERS, CALLS ARE MADE TO A SUBROUTINE TO D
0034                *            A DEVICE TYPE AND UNIT NUMBER, THE
0035                *            INPUT IS ASSIGNED TO THAT DEVICE.
0036                * CALLING SEQUENCE:
0037                *            CALLED BY THE MONITOR
0038                *
0039                *       REF'S AND DEF'S
0040                *
0041                                                  DEF  ASLUN
0042                        REF  GETBUF
0043                        REF  RETBUF
0044                        REF  LWP
0045                        REF  RWP
0046                        REF  ACL
0047                        REF  RR
0048                        REF  CDNDT              CONVERT DEV NM => DEV TYPE
0049                        REF  SETLUN             SET LOGICAL UNIT
0050                        REF  ERROR
0051                *
0052                *WORKSPACE REGISTER DEFINITIONS
0053                *
0054        0000    R0      EQU  0
0055        0001    R1      EQU  1
0056        0002    R2      EQU  2
```

```
0057          0003  R3      EQU  3
0058          0004  R4      EQU  4
0059          0005  R5      EQU  5
0060          0006  R6      EQU  6
0061          0007  R7      EQU  7
0062          0008  R8      EQU  8
0063          0009  R9      EQU  9
0064          000A  R10     EQU  10
0065          000B  R11     EQU  11
0066          000C  R12     EQU  12
0067          000D  R13     EQU  13
0068          000E  R14     EQU  14
0069          000F  R15     EQU  15
0070                *
0071                *
0072                *       COMMAND PARAMETER LIST
0073          0000  CPLPC   EQU  0                    PARM COUNT
0074          0001  CPLBIT  EQU  1                    PARAMETER PRESENCE BITS
0075          0002  CPLPRM  EQU  2                    FIRST PARAMETER
0076          0002  MX02    EQU  2                    INVALID LUNO
0077                *
0078                *       REGISTER ASIGNMENTS
0079                *       R11   SUBROUTINE LINK REG
0080                *       R10-R7 PARM REGS
0081                *       R6    COMMAND PARM LIST P
0082                *       R5    RETURN ADDRESS
0083                *
0084                *
0085                *
0086          0000' ASLUN   EQU  $
0087                *                                2-DO UNTIL ERROR OR DONE;
0088                *                                3-IF CPL PARM COUNT .NE. 2 THE
0089                *                                4-SIGNAL ERROR;
0090  0000  C14B            MOV  R11,R5               SAVE RETURN
0091  0002  C18A            MOV  R10,R6
0092  0004  D116            MOVB *R6,R4
0093  0006  0984            SRL  R4,8
0094  0008  0644            DECT R4
0095  000A  1612            JNE  ASL010
0096                *                                3-IF CPL PARM2(CHR COUNT) .NE.
0097                *                                4-SIGNAL ERROR;
0098  000C  D126            MOVB @CPLPRM+2(R6),R4
      000E  0004
0099  0010  0984            SRL  R4,8
0100  0012  0284            CI   R4,3
      0014  0003
0101  0016  160C            JNE  ASL010
0102                *                                3-CALL CONDT(DEV NAME,DEV TYPE
0103  0018  022A            AI   R10,5                 5TH BYTE OF CPL
      001A  0005
0104  001C  06A0            BL   @CONDT
      001E  0000
0105                *                                3-IF DEV TYPE<0 THEN SIGNAL ER
0106  0020  C249            MOV  R9,R9
0107  0022  1106            JLT  ASL010
```

```
 0108                    *                                  3=CALL SETLUN(LUNO,DEV TYPE,DE
 0109  0024  C2A6              MOV   @CPLPRM(R6),R10
       0026  0002
 0110  0028  06A0              BL    @SETLUN
       002A  0000
 0111                    *                                  3=IF ERRC .EQ. 0 THEN SIGNAL D
 0112  002C  C1C7              MOV   R7,R7
 0113  002E  1304              JEQ   ASL020
 0114                    *                                  3=SIGNAL ERROR;
 0115                    *                                  2=END;
 0116        0030!     ASL010 EQU   $
 0117                    *                                  2=ERROR: DO;
 0118                    *                                  3=CALL ERROR(MX01)
 0119  0030  020A              LI    R10,MX02
       0032  0002
 0120  0034  06A0              BL    @ERROR
       0036  0000
 0121                    *                                  2=END;
 0122                    *                                  2=DONE: DO;END;
 0123        0038!     ASL020 EQU   $
 0124                    *                                  2=RETURN;
 0125  0038  C286              MOV   R6,R10
 0126  003A  0455              B     *R5
 0127                          END
0000 ERS
```

960 - 980 CONCORDANCE

$		0086	0116	0123		
ACL		0046				
ASL010	0116	0095	0101	0107		
ASL020	0123	0113				
ASLUN	0086	0041				
CDNDT		0048	0104			
CPLBIT	0074					
CPLPC	0073					
CPLPRM	0075	0098	0109			
ERROR		0050	0120			
GETBUF		0042				
LWP		0044				
MX02	0076	0119				
R0	0054					
R1	0055					
R10	0064	0091	0103	0109	0119	0125
R11	0065	0090				
R12	0066					
R13	0067					
R14	0068					
R15	0069					
R2	0056					
R3	0057					
R4	0058	0092	0093	0094	0098	0099	0100
R5	0059	0090	0126			
R6	0060	0091	0092	0098	0109	0125
R7	0061	0112	0112			
R8	0062					
R9	0063	0106	0106			
RETBUF		0043				
RR		0047				
RWP		0045				
SETLUN		0049	0110			

THERE ARE 0033 SYMBOLS

A[

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																		
SHEET																																		
REV STATUS OF SHEETS	REV																																	
	SHEET																																	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, BKPPR, BREAKPOINT PROCESSOR-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945347-9901

SCALE	REV.	SHEET 1 of 5

TI-9923-C

```
0003                *    PROCEDURE BREAKPOINT PROCESSOR(BREAK FLAG,
0004                *       BREAK ENTRY #,BREAK ENTRY PTR);
0005                *       BREAK ENTRY PTR.REF COUNT =
0006                *          BREAK ENTRY PTR.REF COUNT + 1;
0007                *       CLEAR BREAKFLAG
0008                *       IF BREAK ENTRY PTR.REF COUNT =
0009                *          BREAK ENTRY PTR.LOOP COUNT THEN DO;
0010                *          BREAK ENTRY PTR.REF COUNT = -1;
0011                *          CALL PRINT(BREADK ENTRY #);
0012                *          CALL INSPECT REGISTERS;
0013                *          IF(BREAK ENTRY PTR.FLAGS .AND.
0014                *             SSDEF)  .NE. 0  THEN DO;
0015                *             CALL PRINT SNAPSHOT(BREAK ENTRY PTR.SS#)
0016                *             BREAK FLAG = 0;
0017                *          END;ELSE BREAK FLAG =-1
0018                *       END;
0019                *    END BREAKPOINT PROCESSOR;
0020                       IDT  'BKPPR'
0021                * TITLE:    BKPTPR
0022                *           BREAKPOINT PROCESSOR
0023                * REVISION:
0024                *           ORIGINAL
0025                * COMPUTER: 990,ASM
0026                * ABSTRACT: BKPTPR CHCKS A BREAKPOINT TO SEE IF
0027                *           ITS REFERENCE COUNT HAS BEEN EXHAUSTED
0028                *           AND PRINTS TH BREAKPOINT INFORMATION
0029                *           IF NECESSARY
0030                * CALLING SEQUENCE:
0031                *           BL @BKPTPR
0032                *              R9  = BREAKPOINT ENTRY NUMBER
0033                *              R8  = BREAKPOINT ENTRY POINTER
0034                *            RETURN
0035                *              R10 = 0  => DO NOT BREAK
0036                *
0037                *       REF'S AND DEF'S
0038                *
0039                       DEF  BKPTPR
0040                       REF  IRP
0041                       REF  PRNTC
0042                       REF  PRTSS
0043                       REF  CLST
0044                       REF  SNAPS
0045                       REF  SNPTAB
0046                       REF  BKSTR
0047                       REF  BKSTRB
0048                       REF  GETBUF
0049                       REF  RETBUF
0050                       REF  LWP
0051                       REF  RWP
0052                       REF  ACL
0053                       REF  RR
0054                *
0055                *WORKSPACE REGISTER DEFINITIONS
0056                *
```

```
0057          0000   R0      EQU  0
0058          0001   R1      EQU  1
0059          0002   R2      EQU  2
0060          0003   R3      EQU  3
0061          0004   R4      EQU  4
0062          0005   R5      EQU  5
0063          0006   R6      EQU  6
0064          0007   R7      EQU  7
0065          0008   R8      EQU  8
0066          0009   R9      EQU  9
0067          000A   R10     EQU  10
0068          000B   R11     EQU  11
0069          000C   R12     EQU  12
0070          000D   R13     EQU  13
0071          000E   R14     EQU  14
0072          000F   R15     EQU  15
0073                 *
0074          0100   LBYTE   EQU  256
0075                 *
0076                 *       BREAKPOINT ENTRY
0077                 *
0078          0000   BPDEF   EQU  0              1 BYTE
0079          4000   SSDEF   EQU  >40*LBYTE
0080          0001   BPSSNM  EQU  1
0081          0002   BPPC    EQU  2
0082          0006   CURREF  EQU  6
0083          0008   LOOPCT  EQU  8
0084          8000   PRESEN  EQU  >80*LBYTE
0085                 *
0086                 *                                  1=PROCEDURE BREAKPOINT PROCESS
0087                 *                                  2=BREAK ENTRY #,BREAK ENTRY PT
0088          0000'  BKPTPR  EQU  $
0089                 *     *ALLOC,COPY,LINK
0090  0000    0420           BLWP @ACL
      0002    0000
0091                 *                                  2=BREAK ENTRY PTR,REF COUNT =
0092                 *                                  3=BREAK ENTRY PTR,REF COUNT +
0093  0004    05A8           INC  @CURREF(R8)
      0006    0006
0094                 *                                  2=CLEAR BREAKFLAG
0095  0008    04C7           CLR  R7
0096                 *                                  2=IF BREAK ENTRY PTR,REF COUNT
0097                 *                                  3=BREAK ENTRY PTR,LOOP COUNT T
0098  000A    8A28           C    @CURREF(R8),@LOOPCT(R8)
      000C    0006
      000F    0008
0099  0010    1624           JNE  BKP020
0100                 *                                  3=BREAK ENTRY PTR,REF COUNT =
0101  0012    0728           SETO @CURREF(R8)
      0014    0006
0102                 *                                  3=CALL PRINT(BREADK ENTRY #);
0103  0016    0206           LI   R6,CLST
      0018    0000
0104  001A    A189           A    R9,R6
0105  001C    D816           MOVB *R6,@BKSTRB
```

```
      001E  0000
 0106 0020  020A          LI   R10,BKSTR
      0022  0000
 0107 0024  06A0          BL   @PRNTC
      0026  0000
 0108 0028  1000          NOP
 0109               *                                  3=CALL INSPECT REGISTERS;
 0110 002A  06A0          BL   @IRP
      002C  0000
 0111               *                                  3=IF(BREAK ENTRY PTR,FLAGS .AN
 0112               *                                  4=SSDEF)  .NE. 0  THEN DO;
 0113 002E  C198          MOV  *R8,R6
 0114 0030  0246          ANDI R6,SSDEF
      0032  4000
 0115 0034  1311          JEQ  BKP010
 0116               *                                  4=CALL PRINT SNAPSHOT(BREAK EN
 0117 0036  C298          MOV  *R8,R10
 0118 0038  024A          ANDI R10,>FF                 BYTE 1, ONLY
      003A  00FF
 0119 003C  C18A          MOV  R10,R6
 0120 003E  0207          LI   R7,SNAPS
      0040  0000
 0121 0042  39A7          MPY  @>4(R7),R6
      0044  0004
 0122 0046  D1E7          MOVB @SNPTAB(R7),R7
      0048  0000
 0123 004A  0247          ANDI R7,PRESEN
      004C  8000
 0124 004E  1304          JEQ  BKP010
 0125               *                                  4=BREAK FLAG = 0;
 0126 0050  04C7          CLR  R7
 0127 0052  06A0          BL   @PRTSS
      0054  0000
 0128               *                                  3=END;ELSE BREAK FLAG =-1
 0129 0056  1001          JMP  BKP020
 0130       0058' BKP010  EQU  $
 0131 0058  0707          SETO R7
 0132               *                                  2=END;
 0133       005A' BKP020  EQU  $
 0134               *                                  1=END BREAKPOINT PROCESSOR;
 0135 005A  C747          MOV  R7,*R13       RETURN BREADK FLAG IN CALLER'S R10
 0136               *     *LINK TO PREV WKSP, RET CURR WKSP
 0137 005C  0420          BLWP @RR
      005E  0000
 0138 0060  C280          MOV  R0,R10
 0139 0062  0458          RT
 0140                     END
0000 ERS
```

960 - 980 CONCORDANCE

$		0088	0130	0133				
ACL		0052	0090					
BKP010	0130	0115	0124					
BKP020	0133	0099	0129					
BKPTPR	0088	0039						
BKSTR		0046	0106					
BKSTRB		0047	0105					
BPDEF	0078							
BPPC	0081							
BPSSNM	0080							
CLST		0043	0103					
CURREF	0082	0093	0098	0101				
GETBUF		0048						
IRP		0040	0110					
LBYTE	0074	0079	0084					
LOOPCT	0083	0098						
LWP		0050						
PRESEN	0084	0123						
PRNTC		0041	0107					
PRTSS		0042	0127					
R0	0057	0138						
R1	0058							
R10	0067	0106	0117	0118	0119	0138		
R11	0068							
R12	0069							
R13	0070	0135						
R14	0071							
R15	0072							
R2	0059							
R3	0060							
R4	0061							
R5	0062							
R6	0063	0103	0104	0105	0113	0114	0119	0121
R7	0064	0095	0120	0122	0122	0123	0126	0131 0135
R8	0065	0093	0098	0098	0101	0113	0117	
R9	0066	0104						
RETBUF		0049						
RR		0053	0137					
RWP		0051						
SNAPS		0044	0120					
SNPTAB		0045	0122					
SSDEF	0079	0114						

THERE ARE 0042 SYMBOLS

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																		
SHEET																																		
REV STATUS OF SHEETS	REV																																	
	SHEET																																	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DATE
DWN	
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, CHARIN, CHARACTER INPUT-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945348-9901

SCALE	REV.	SHEET 1 of 10

TI-0023-C

```
0003          *    PROCEDURE GETCHR(CHAR)
0004          *       /*
0005          *         INPUT A CHAR FROM THE 733
0006          *       */
0007          *       CHAR = 'CR'   /* TREAT NO CHAR AS 'CR'  */
0008          *       CALL SVC(SVC CALL BLOCK,CHAR);
0009          *       IF UNRECOVERABLE ERROR THEN RETURN('ESC');
0010          *       RETURN;
0011          *    END GETCHR
0012          *    PROCEDURE PRINT(STRING,CHAR COUNT)
0013          *       /*
0014          *         OUTPUT CHARACTER STRING
0015          *       */
0016          *       PROCEDURE PRINT CRLF;
0017          *          /*PRINT <LINE FEED><CARRIAGE RETURN> */
0018          *          CALL PRINT(LOC(CRLF));
0019          *       END PRINT CRLF;
0020          *       CHAR COUNT = STRING(1)
0021          *       PRB BUFFER PTR = STRING;
0022          *       PRB CHAR COUNT = CHAR COUNT;
0023          *       CALL SVC(SVC CALL BLOCK);
0024          *       IF UNRECOVERABLE ERROR THEN DO;
0025          *          RETURN(ERROR);
0026          *       END;
0027          *    PROCEDURE ERROR(ERR CODE);
0028          *       1*
0029          *         ERROR ACCEPTS AN ERROR CODE,
0030          *         CONVERTS IT TO AN ASCII STRING
0031          *         AND PRINTS IT.
0032          *       */
0033          *       ERRPTR = LOC(ERROR OUTPUT BUFFER);
0034          *       CHAR =  CLST1(AND(SRC(ERR CODE,12),'F'));
0035          *       ERR CODE = SRC(ERR CODE,12);
0036          *       ERRPTR = ERRPTR +1;
0037          *       ERR CODE = SRC (ERR CODE,12);
0038          *       ERRPTR,CHAR = CLST2(AND(ERR CODE,'F'));
0039          *       ERRPTR = ERRPTR +1;
0040          *       ERR CODE = SRC(ERR CODE,12);
0041          *       ERRPTR,CHAR = CLST(AND(ERR CODE,'F'));
0042          *       ERRPTR = ERRPTR +1;
0043          *       ERR CODE = SRC(ERR CODE,12);
0044          *       ERRPTR,CHAR = CLST(AND(ERR CODE,'F'));
0045          *       CALL PRINT(ERBUF,#WCHARS);
0046          *       END ERROR;
0047                  IDT  'CHARIN'
0048          * TITLE:    CHARIN
0049          *           INPUT SINGLE CHAR IN CHARACTER MODE.
0050          * REVISION:
0051          *           ORIGINAL
0052          * COMPUTER: 990,ASM
0053          * ABSTRACT: CHARIN INPUTS A SINGLE CHAR FROM
0054          *           THE 733 USING CHARACTER MODE I10.
0055          * CALLING SEQUENCE:
0056          *           BL @GETCHR
```

```
0057                    *            CHARACTER RETURNED IN REGISTER 10
0058                    *
0059                    *
0060                    *WORKSPACE REGISTER DEFINITIONS
0061                    *
0062          0000      R0      EQU  0
0063          0001      R1      EQU  1
0064          0002      R2      EQU  2
0065          0003      R3      EQU  3
0066          0004      R4      EQU  4
0067          0005      R5      EQU  5
0068          0006      R6      EQU  6
0069          0007      R7      EQU  7
0070          0008      R8      EQU  8
0071          0009      R9      EQU  9
0072          000A      R10     EQU  10
0073          000B      R11     EQU  11
0074          000C      R12     EQU  12
0075          000D      R13     EQU  13
0076          000E      R14     EQU  14
0077          000F      R15     EQU  15
0078                    *
```

```
0080                    *                                  1=PROCEDURE GETCHR(CHAR)
0081                    *                                  2=/*
0082                    *                                  2=  INPUT A CHAR FROM THE 733
0083                    *                                  2=*/
0084                    *
0085                    *       REF'S AND DEF'S
0086                    *
0087                            DEF   GETCHR
0088                            REF   SVCPRB
0089                            REF   INCHAR
0090                            REF   SVCALT
0091                            REF   CR
0092                            REF   SVCSFG
0093                            REF   ESC
0094                            REF   GETBUF
0095                            REF   RETBUF
0096                            REF   LWP
0097                            REF   RWP
0098                            REF   ACL
0099                            REF   RR
0100            0000' GETCHR    EQU    $
0101                    *                                  2=CHAR = 'CR'    /* TREAT NO CH
0102    0000    D820            MOVB  @CR,@INCHAR
        0002    0000
        0004    0000
0103                    *                                  2=CALL SVC(SVC CALL BLOCK,CHAR
0104    0006    020A            LI    R10,SVCPRB
        0008    0000
0105    000A    0420            BLWP  @SVCALT
        000C    0000
0106                    *                                  2=IF UNRECOVERABLE ERROR THEN
0107    000E    D2A0            MOVB  @SVCSFG,R10
        0010    0000
0108    0012    1603            JNE   GET010
0109    0014    D2A0            MOVB  @INCHAR,R10
        0016    0004'
0110                    *                                  2=RETURN;
0111    0018    045B            RT
0112            001A' GET010    EQU   $
0113    001A    D2A0            MOVB  @ESC,R10
        001C    0000
0114    001E    045B            RT
0115                    *                                  1=END GETCHR
```

```
0118                * TITLE:     PRNTR
0119                *            OUTPUT CHARACTER STRING
0120                * REVISION:
0121                *            ORIGINAL
0122                * COMPUTER:  990,ASM
0123                * ABSTRACT:  PRNTR ACCEPTS A POINTER TO A STRING
0124                *            AND A CHARACTER COUNT AS INPUT AND
0125                *            PRINTS THE STRING TO  LOG.
0126                * CALLING SEQUENCE:
0127                *            BL  @PRINT
0128                *            R10 POINTS TO STRING
0129                *            R9  CHARACTER COUNT
0130                *            R11 = ERROR RETURN
0131                *            R11+2 = NORMAL RETURN
```

```
0133                    *                                   1*PROCEDURE PRINT(STRING,CHAR
0134                    *                                   2*/*
0135                    *                                   2*  OUTPUT CHARACTER STRING
0136                    *                                   2**/
0137                    *
0138                    *       REF'S AND DEF'S
0139                            DEF   PRNTC
0140                    *
0141                            DEF   PRINT
0142                            REF   PRBBUF
0143                            REF   PRBUER
0144                            REF   PRBCC
0145                            REF   SVCWRT
0146                            DEF   PRCRLF
0147                            REF   CRLF
0148                            EVEN
0149                    *                                   2*PROCEDURE PRINT CRLF;
0150                    *                                   3*/*PRINT <LINE FEED><CARRIAGE
0151                    *                                   3*CALL PRINT(LOC(CRLF));
0152          0020' PRCRLF  EQU   $
0153    0020  020A            LI    R10,CRLF
        0022  0000
0154                    *                                   2*END PRINT CRLF;
0155                    *                                   2*CHAR COUNT = STRING(1)
0156          0024' PRNTC   EQU   $
0157    0024  D27A            MOVB  *R10+,R9
0158    0026  0989            SRL   R9,8
0159          0028' PRINT   EQU   $
0160                    *                                   2*PRB BUFFER PTR = STRING;
0161    0028  C80A            MOV   R10,@PRBBUF
        002A  0000
0162                    *                                   2*PRB CHAR COUNT = CHAR COUNT;
0163    002C  C809            MOV   R9,@PRBCC
        002E  0000
0164                    *                                   2*CALL SVC(SVC CALL BLOCK);
0165    0030  020A            LI    R10,SVCWRT
        0032  0000
0166    0034  0420            BLWP  @SVCALT
        0036  000C'
0167                    *                                   2*IF UNRECOVERABLE ERROR THEN
0168                    *                                   3*RETURN(ERROR);
0169    0038  D260            MOVB  @PRBUER,R9
        003A  0000
0170    003C  1601            JNE   P1
0171    003E  05C0            INCT  R11
0172          0040' P1        EQU   $
0173    0040  045B            RT
0174                    *                                   2*END;
```

```
0177                    * TITLE:     PRNTHX
0178                    *            PRINT VALUE IN HEX
0179                    * REVISION:
0180                    *            ORIGINAL
0181                    * COMPUTER:  990,ASM
0182                    * ABSTRACT:  PRNTHX PRINTS THE HEX ASCII EQUIVALENT
0183                    *            OF ITS INPUT PARM FOLLOWED BY TWO BLANKS
0184                    *            PRNTHB PRINTS TWO ASCII CHARACTERS
0185                    * CALLING SEQUENCE:
0186                    *            BL  @PRNTHX
0187                    *            R10 = VALUE TO BE PRINTED
0188                           DEF   PRNTHB
0189                           DEF   PRNTHX
0190                           DEF   PRNTHN
0191                           REF   CBH
0192                           REF   CBH0
0193          0042' PRNTHB EQU   $
0194   0042   0200           LI    R0,2                # OF CHARS TO PRINT
       0044   0002
0195   0046   1005           JMP   PRN005
0196          0048' PRNTHN EQU   $
0197   0048   0200           LI    R0,4                  # CHARS TO PRINT
       004A   0004
0198   004C   1002           JMP   PRN005
0199          004E' PRNTHX EQU   $
0200   004E   0200           LI    R0,6
       0050   0006
0201          0052' PRN005 EQU   $
0202                  *     *ALLOC,COPY,LINK
0203   0052   0420           BLWP  @ACL
       0054   0000
0204   0056   C240           MOV   R0,R9
0205   0058   C00A           MOV   R10,R0              CONVERT BIN TO HEX ASCII
0206   005A   020A           LI    R10,CBH
       005C   0000
0207   005E   0420           BLWP  @SVCALT
       0060   0036'
0208   0062   020A           LI    R10,CBH0
       0064   0000
0209   0066   06A0           BL    @PRINT
       0068   0028'
0210   006A   1002           JMP   PRN010
0211   006C   05ED           INCT  @R11*2(R13)
       006E   0016
0212          0070' PRN010 EQU   $
0213                  *     *LINK TO PREV WKSP, RET CURR WKSP
0214   0070   0420           BLWP  @RR
       0072   0000
0215   0074   045B           RT
```

```
0218                    * TITLE:    ERRPR
0219                    *           ERROR CODE PRINT ROUTINE
0220                    * REVISION:
0221                    *           ORIGINAL
0222                    * COMPUTER: 990,ASM
0223                    * ABSTRACT: ERRPR ACCEPTS AS INPUT AN ERROR CODE
0224                    *           AND CONVERTS IT TO A STRING REPRESENTING
0225                    *           THE ERROR CODE. THE FIRST TWO HEX DIGITS
0226                    *           ARE USED TO INDEX INTO A TABLE OF CHARACTERS
0227                    *           THE LAST TWO ARE CONVERTED DIRECTLY AS HEX
0228                    *           CHARS
0229                    * CALLING SEQUENCE:
0230                    *     BL    @ERROR
0231                    *     R10   CONTAINS ERROR CODE
0232                    *
0233                    *     REF'S AND DEF'S
0234                    *
0235                          DEF   ERROR
0236                          REF   ERBUF
0237                          REF   ERSTR
0238                          REF   CLST
0239  0076    4D    CLST1   TEXT  'MDLP'
0240  007A    58    CLST2   TEXT  'XPSLD '
0241                    *                                   1=PROCEDURE ERROR(ERR CODE);
0242        0080'   ERROR   EQU   $
0243                    *     *ALLOC,COPY,LINK
0244  0080  0420            BLWP  @ACL
      0082  0054'
0245                    *                                   2=1*
0246                    *                                   2=  ERROR ACCEPTS AN ERROR COD
0247                    *                                   2=  CONVERTS IT TO AN ASCII ST
0248                    *                                   2=  AND PRINTS IT.
0249                    *                                   2=*/
0250                    *                                   2=ERRPTR = LOC(ERROR OUTPUT BU
0251  0084  0206            LI    R6,ERBUF
      0086  0000
0252                    *                                   2=CHAR =  CLST1(AND(SRC(ERR CO
0253                    *                                   2=ERR CODE = SRC(ERR CODE,12);
0254  0088  0BCA            SRC   R10,12
0255  008A  C24A            MOV   R10,R9
0256  008C  0249            ANDI  R9,>F
      008E  000F
0257  0090  0229            AI    R9,CLST1
      0092  0076'
0258  0094  DD99            MOVB  *R9,*R6+
0259                    *                                   2=ERRPTR = ERRPTR +1;
0260                    *                                   2=ERR CODE = SRC (ERR CODE,12)
0261                    *                                   2=ERRPTR,CHAR = CLST2(AND(ERR
0262                    *                                   2=ERRPTR = ERRPTR +1;
0263  0096  0BCA            SRC   R10,12
0264  0098  C24A            MOV   R10,R9
0265  009A  0249            ANDI  R9,>F
      009C  000F
0266  009E  0229            AI    R9,CLST2
```

```
        00A0  007A'
 0267   00A2  DD99          MOVB *R9,*R6+
 0268                 *                          2=ERR CODE = SRC(ERR CODE,12)!
 0269                 *                          2=ERRPTR,CHAR = CLST(AND(ERR C
 0270                 *                          2=ERRPTR = ERRPTR +1!
 0271   00A4  0BCA          SRC  R10,12
 0272   00A6  C24A          MOV  R10,R9
 0273   00A8  0249          ANDI R9,>F
        00AA  000F
 0274   00AC  0229          AI   R9,CLST
        00AE  0000
 0275   00B0  DD99          MOVB *R9,*R6+
 0276                 *                          2=ERR CODE = SRC(ERR CODE,12)!
 0277                 *                          2=ERRPTR,CHAR = CLST(AND(ERR C
 0278   00B2  0BCA          SRC  R10,12
 0279   00B4  C24A          MOV  R10,R9
 0280   00B6  0249          ANDI R9,>F
        00B8  000F
 0281   00BA  0229          AI   R9,CLST
        00BC  00AE'
 0282   00BE  D599          MOVB *R9,*R6
 0283                 *                          2=CALL PRINT(ERBUF,#WCHARS)!
 0284   00C0  020A          LI   R10,ERSTR
        00C2  0000
 0285   00C4  06A0          BL   @PRNTC
        00C6  0024'
 0286   00C8  1000          NOP
 0287                 *                          2=END ERROR!
 0288                 *     *LINK TO PREV WKSP, RET CURR WKSP
 0289   00CA  0420            BLWP @RR
        00CC  0072'
 0290   00CE  045B          RT
 0291                       END
0000 ERS
```

960 = 980 CONCORDANCE

$		0100	0112	0152	0156	0159	0172	0193	0196	0199
		0201	0212	0242						
ACL		0098	0203	0244						
CBH		0191	0206							
CBH0		0192	0208							
CLST		0238	0274	0281						
CLST1	0239	0257								
CLST2	0240	0266								
CR		0091	0102							
CRLF		0147	0153							
ERBUF		0236	0251							
ERROR	0242	0235								
ERSTR		0237	0284							
ESC		0093	0113							
GET010	0112	0108								
GETBUF		0094								
GETCHR	0100	0087								
INCHAR		0089	0102	0109						
LWP		0096								
P1	0172	0170								
PRBBUF		0142	0161							
PRBCC		0144	0163							
PRBUER		0143	0169							
PRCRLF	0152	0146								
PRINT	0159	0141	0209							
PRN005	0201	0195	0198							
PRN010	0212	0210								
PRNTC	0156	0139	0285							
PRNTHB	0193	0188								
PRNTHN	0196	0190								
PRNTHX	0199	0189								
R0	0062	0194	0197	0200	0204	0205				
R1	0063									
R10	0072	0104	0107	0109	0113	0153	0157	0161	0165	0205
		0206	0208	0254	0255	0263	0264	0271	0272	0278
		0279	0284							
R11	0073	0171	0211							
R12	0074									
R13	0075	0211								
R14	0076									
R15	0077									
R2	0064									
R3	0065									
R4	0066									
R5	0067									
R6	0068	0251	0258	0267	0275	0282				
R7	0069									
R8	0070									
R9	0071	0157	0158	0163	0169	0204	0255	0256	0257	0258
		0264	0265	0266	0267	0272	0273	0274	0275	0279
		0280	0281	0282						
RETBUF		0095								
RR		0099	0214	0289						
RWP		0097								
SVCALT		0090	0105	0166	0207					
SVCPRB		0088	0104							
SVCSFG		0092	0107							
SVCWRT		0145	0165							

THERE ARE 0053 SYMBOLS

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																		
SHEET																																		
REV STATUS OF SHEETS	REV																																	
	SHEET																																	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, CLEAR, CLEAR COMMAND PROCESSOR-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945349-9901

SCALE	REV.	SHEET
		1 of 6

TI-0023.C

```
0003                *   PROCEDURE CLEAR(TABLE TYPE,CPL);
0004                *      /* CLEAR WILL CLEAR A RANGE OF DEBUG
0005                *         ENTRIES (TRACE, BREAKPOINT, SNAPSHOT)
0006                *         IN A STANDARD FORMAT TABLE.
0007                *      */
0008                *      TBLPTR = TABLE TYPE(1),
0009                *      NUMENTRIES = TABLE TYPE(2);
0010                *      ENTRY SIZE = TABLE TYPE(3);
0011                *      CALL RANGE(LOW,HIGH,NUMENTRIES,ERROR);
0012                *      IF .NOT. ERROR THEN DO;
0013                *         TBLPTR = TBLPTR + ENTRY SIZE * LOW;
0014                *         DO WHILE LOW .LE. HIGH;
0015                *            TBLPTR.DEF = 0;
0016                *            TBLPTR = TBLPTR + ENTRY SIZE;
0017                *            LOW = LOW +1;
0018                *         END;
0019                *      END;
0020                *   END CLEAR;.
0021                       IDT  'CLEAR'
0022                * TITLE:    CLEAR
0023                *           CLEAR STANDARD TABLE ENTRY
0024                * REVISION:
0025                *           ORIGINAL
0026                * COMPUTER: 990,ASM
0027                * ABSTRACT: CLEAR PROCESSES THE CLEAR BREAKPOINT, SNAPSHOT,
0028                *           TRACE, PROTECTED REGION COMMANDS. IT IS ENTERED
0029                *           WITH A TABLE DEFINITION LIST AND A COMMAND
0030                *           PARAMETER LIST AND CLEARS A RANGE,OF ENTRIES
0031                * CALLING SEQUENCE:
0032                *           BL @XXXX
0033                *              WHERE XXXX IS A TABLE SPECIFIC ENTRY NAME
0034                *              R10 POINTS TO COMMAND PARAMETER LIST
0035                *
0036                *      REF'S AND DEF'S
0037                *
0038                       DEF  CRP
0039                       DEF  CSS
0040                       REF  BKPT
0041                       REF  SNAPS
0042                       DEF  CRR
0043                       REF  TRACE
0044                       REF  RANGEX
0045                       REF  GETBUF
0046                       REF  RETBUF
0047                       REF  LWP
0048                       REF  RWP
0049                       REF  ACL
0050                       REF  RR
0051                *
0052                *WORKSPACE REGISTER DEFINITIONS
0053                *
0054          0000  R0       EQU  0
0055          0001  R1       EQU  1
0056          0002  R2       EQU  2
```

```
0057          0003  R3      EQU  3
0058          0004  R4      EQU  4
0059          0005  R5      EQU  5
0060          0006  R6      EQU  6
0061          0007  R7      EQU  7
0062          0008  R8      EQU  8
0063          0009  R9      EQU  9
0064          000A  R10     EQU  10
0065          000B  R11     EQU  11
0066          000C  R12     EQU  12
0067          000D  R13     EQU  13
0068          000E  R14     EQU  14
0069          000F  R15     EQU  15
0070                *
```

```
0072                    *                                  1=PROCEDURE CLEAR(TABLE TYPE,C
0073                    *                                  2=/* CLEAR WILL CLEAR A RANGE
0074                    *                                  2=   ENTRIES (TRACE, BREAKPOIN
0075                    *                                  2=   IN A STANDARD FORMAT TABL
0076                    *                                  2=*/
0077          0000' CBP     EQU   $
0078  0000    0209          LI    R9,BKPT
      0002    0000
0079  0004    1005          JMP   CLEAR
0080          0006' CSS     EQU   $
0081  0006    0209          LI    R9,SNAPS
      0008    0000
0082  000A    1002          JMP   CLEAR
0083          000C' CRR     EQU   $
0084  000C    0209          LI    R9,TRACE
      000E    0000
0085          0010' CLEAR   EQU   $
0086                  *     *ALLOC,COPY,LINK
0087  0010    0420          BLWP  @ACL
      0012    0000
0088                    *                                  2=TBLPTR = TABLE TYPE(1),
0089  0014    C1B9          MOV   *R9+,R6
0090                    *                                  2=NUMENTRIES = TABLE TYPE(2);
0091  0016    C239          MOV   *R9+,R8
0092                    *                                  2=ENTRY SIZE = TABLE TYPE(3);
0093  0018    C1D9          MOV   *R9,R7
0094                    *                                  2=CALL RANGE(LOW,HIGH,NUMENTRI
0095  001A    06A0          BL    @RANGEX                     R9= LOW  R10 = HIGH
      001C    0000
0096                    *                                  2=IF .NOT. ERROR THEN DO;
0097  001E    100A          JMP   CLR020
0098                    *                                  3=TBLPTR = TBLPTR + ENTRY SIZE
0099  0020    C107          MOV   R7,R4
0100  0022    04C5          CLR   R5
0101  0024    3909          MPY   R9,R4
0102  0026    A185          A     R5,R6
0103                    *                                  3=DO WHILE LOW .LE. HIGH;
0104          0028' CLR010  EQU   $
0105  0028    8289          C     R9,R10
0106  002A    1304          JH    CLR020
0107                    *                                  4=TBLPTR,DEF = 0;
0108  002C    04D6          CLR   *R6
0109                    *                                  4=TBLPTR = TBLPTR + ENTRY SIZE
0110  002E    A187          A     R7,R6
0111                    *                                  4=LOW = LOW +1;
0112  0030    0589          INC   R9
0113                    *                                  3=END;
0114  0032    10FA          JMP   CLR010
0115                    *                                  2=END;
0116          0034' CLR020  EQU   $
0117                    *                                  1=END CLEAR;
0118                    *     *LINK TO PREV WKSP, RET CURR WKSP
0119  0034    0420          BLWP  @RR
      0036    0000
```

```
 0120  003B  045B          RT
 0121                      END
0000 ERS
```

```
960 - 980  CONCORDANCE
$                       0076  0079  0082  0084  0103  0115
ACL                     0048  0086
BKPT                    0039  0077
CBP         0076        0037
CLEAR       0084        0078  0081
CLR010      0103        0113
CLR020      0115        0096  0105
CRN         0082        0041
CSS         0079        0038
GETBUF                  0044
LWP                     0046
R0          0053
R1          0054
R10         0063        0104
R11         0064
R12         0065
R13         0066
R14         0067
R15         0068
R2          0055
R3          0056
R4          0057        0098  0100
R5          0058        0099  0101
R6          0059        0088  0101  0107  0109
R7          0060        0092  0098  0109
R8          0061        0090
R9          0062        0077  0080  0083  0088  0090  0092  0100  0104  0111
RANGEX                  0043  0094
RETBUF                  0045
RR                      0049  0118
RWP                     0047
SNAPS                   0040  0080
TRACE                   0042  0083

THERE ARE    0033  SYMBOLS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV	
SHEET	
REV STATUS OF SHEETS	REV
	SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, CMDDEF, COMMAND DEFINITIONS-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945350-9901

SCALE	REV.	SHEET
		1 of 9

TI-9923-C

```
0003                    IDT  'CMDDEF'
0004              * TITLE:    CMDDEF
0005              *           COMMAND DEFINITION TABLE
0006              * REVISION: 2/1/76
0007              *           ORIGINAL
0008              * COMPUTER: 990,ASM
0009              * ABSTRACT: CONTAINS A LIST OF THE COMMANDS,THEIR
0010              *           ALLOWABLE PARAMETERS, AND THE ENTRY POINT
0011              *           FOR THE COMMAND PROCESSER ROUTINE.
0012              * CALLING SEQUENCE:
0013              *           NON EXECUTABLE
0014              *
0015              *      COMMAND STRUCTURE
0016              *
0017              *           **********************
0018              *           *      COMMAND       *
0019              *           **********************
0020              *           *   P1 - P8          *
0021              *           **********************
0022              *           *  SERVICE ROUTINE   *
0023              *           **********************
0024              *
0025              *      PARAMETER DEFINITIONS
0026              *
0027        0001  PH    EQU    1          HEX  STRING
0028        0002  PC    EQU    2          CHARACTER STRING
0029        0003  PN    EQU    3          NULL
0030              *
```

```
0032                    *
0033                    *       DEF'S AND REF'S
0034                    *
0035                            DEF   CMDTBL
0036                            DEF   NOCMDS
0037                            DEF   SREGN
0038                    *
0039                    *
0040          0000'     CMDTBL EQU   $
0041                    *
0042                    *       ASSIGN LUNO
0043                    *
0044                            REF   ASLUN
0045  0000      41              TEXT 'AL'
0046  0002    6FFF              DATA PH*4+PC*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0047  0004    0000              DATA ASLUN
0048                    *
0049                    *       DUMP IN ABSOLUTE FORMAT
0050                    *
0051  0006      44              TEXT 'DP'
0052  0008    56BF              DATA PH*4+PH*4+PH*4+PC*4+PC*4+PN*4+PN*4+PN
0053  000A    0000              DATA 0
0054                    *
0055                    *       DUMP IN BNPF FORMAT
0056                    *
0057  000C      44              TEXT 'DB'
0058  000E    97FF              DATA PC*4+PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN
0059  0010    0000              DATA 0
0060                    *
0061                    *       DUMP IN HIGH/LOW FORMAT
0062                    *
0063  0012      48              TEXT 'HL'
0064  0014    95FF              DATA PC*4+PH*4+PH*4+PH*4+PN*4+PN*4+PN*4+PN
0065  0016    0000              DATA 0
0066                    *
0067                    *       EXECUTE USER PROGRAM
0068                    *
0069                            REF   EXCT
0070  0018      45              TEXT 'EX'
0071  001A    FFFF              DATA PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0072  001C    0000              DATA EXCT
0073                    *
0074                    *       LINK AND LOAD
0075                    *
0076  001E      4C              TEXT 'LL'
0077  0020    FFFF              DATA PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0078  0022    0000              DATA 0
0079                    *
0080                    *       STANDARD LOADER
0081                    *
0082                            REF   LOAD
0083  0024      4C              TEXT 'LP'
0084  0026    5FFF              DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0085  0028    0000              DATA LOAD
```

```
0086                    *
0087                    *       ABSOLUTE LOADER
0088                    *
0089  002A     4C               TEXT 'LA'
0090  002C   7FFF               DATA PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0091  002E   0000               DATA 0
0092                    *
0093                    *
0094                    *       LOAD UP FRONT LOADER AND PROGRAM
0095                    *
0096                            REF  LDUFL
0097  0030     4C               TEXT 'LU'
0098  0032   5FFF               DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0099  0034   0000               DATA LDUFL
0100                    *
0101                    *            LOAD PROM PROGRAMMER
0102                    *
0103                            REF  PL
0104  0036     50               TEXT 'PL'
0105  0038   5FFF               DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0106  003A   0000               DATA PL
0107                    *       OVERLAY SUPERVISOR
0108                    *
0109                            REF  OVLY
0110  003C     4F               TEXT 'OV'
0111  003E   7FFF               DATA PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0112  0040   0000               DATA OVLY
0113                    *
0114                    *       PROM PROGRAMMER
0115                    *
0116  0042     50               TEXT 'PP'
0117  0044   9555               DATA PC*4+PH*4+PH*4+PH*4+PH*4+PH*4+PH*4+PH
0118  0046   0000               DATA 0
0119                    *
0120                    *       STANDARD PROM PROGRAMMER
0121                    *
0122  0048     50               TEXT 'PS'
0123  004A   AFFF               DATA PC*4+PC*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0124  004C   0000               DATA 0
0125                    *
0126                    *
0127                    *
0128                            REF  RUN
0129  004E     52               TEXT 'RU'
0130  0050   7FFF               DATA PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0131  0052   0000               DATA RUN
0132                    *
0133                    *       INSPECT CRU
0134                    *
0135                            REF  ICP
0136  0054     49               TEXT 'IC'
0137  0056   5FFF               DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0138  0058   0000               DATA ICP
0139                    *
0140                    *       INSPECT MEMORY
```

```
0141                    *
0142                            REF  IMP
0143  005A    49                TEXT 'IM'
0144  005C  5FFF               DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0145  005F  0000               DATA IMP
0146                    *
0147                    *       INSPECT REGISTERS
0148                    *
0149                            REF  IRP
0150  0060    49                TEXT 'IR'
0151  0062  FFFF               DATA PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0152  0064  0000               DATA IRP
0153                    *
0154                    *       INSPECT SNAPSHOT
0155                    *
0156                            REF  ISS
0157  0066    49                TEXT 'IS'
0158  0068  5FFF               DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0159  006A  0000               DATA ISS
0160                    *
0161                    *       INSPECT WORKSPACE
0162                    *
0163                            REF  IWP
0164  006C    49                TEXT 'IW'
0165  006F  5FFF               DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0166  0070  0000               DATA IWP
0167                    *
0168                    *       MODIFY CRU
0169                    *
0170                            REF  MODCRU
0171  0072    4D                TEXT 'MC'
0172  0074  5FFF               DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0173  0076  0000               DATA MODCRU
0174                    *
0175                    *       MODIFY MEMORY
0176                    *
0177                            REF  MODMEM
0178  0078    4D                TEXT 'MM'
0179  007A  7FFF               DATA PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0180  007C  0000               DATA MODMEM
0181                    *
0182                    *       MODIFY REGISTERS
0183                    *
0184                            REF  MRP
0185  007E    4D                TEXT 'MR'
0186  0080  FFFF               DATA PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0187  0082  0000               DATA MRP
0188                    *
0189                    *       MODIFY WORKSPACE REGISTERS
0190                    *
0191                            REF  MODWSP
0192  0084    4D                TEXT 'MW'
0193  0086  7FFF               DATA PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0194  0088  0000               DATA MODWSP
0195                    *
```

```
0196                    *
0197                    *       SET BREAKPOINT
0198                    *
0199                            REF  SBP
0200  008A    53                TEXT 'SB'
0201  008C  55FF                DATA PH*4+PH*4+PH*4+PH*4+PN*4+PN*4+PN*4+PN
0202  008E  0000                DATA SBP
0203                    *
0204                    *       SET  SNAPSHOT
0205                    *
0206                            REF  SSS
0207  0090    53                TEXT 'SS'
0208  0092  557F                DATA PH*4+PH*4+PH*4+PH*4+PH*4+PN*4+PN*4+PN
0209  0094  0000                DATA SSS
0210                    *
0211                    *       SET REGION
0212                    *
0213  0096    53                TEXT 'SR'
0214  0098  5595                DATA PH*4+PH*4+PH*4+PH*4+PC*4+PH*4+PH*4+PH
0215  009A  0000    SREGN       DATA 0
0216                    *
0217                    *       SET TRACE
0218                    *
0219  009C    53                TEXT 'ST'
0220  009E  6FFF                DATA PH*4+PC*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0221  00A0  0000                DATA 0
0222                    *
0223                    *       SET PROTECT REGION
0224                    *
0225                            REF  LDPRT
0226  00A2    53                TEXT 'SP'
0227  00A4  5FFF                DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0228  00A6  0000                DATA LDPRT
0229                    *
0230                    *       CLEAR BREAKPOINT
0231                    *
0232                            REF  CBP
0233  00A8    43                TEXT 'CB'
0234  00AA  5FFF                DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0235  00AC  0000                DATA CBP
0236                    *
0237                    *       CLEAR SNAPSHOTS
0238                    *
0239                            REF  CSS
0240  00AE    43                TEXT 'CS'
0241  00B0  5FFF                DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0242  00B2  0000                DATA CSS
0243                    *
0244                    *       CLEAR REGION
0245                    *
0246                            REF  CRR
0247  00B4    43                TEXT 'CR'
0248  00B6  5FFF                DATA PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0249  00B8  0000                DATA CRR
0250                    *
```

```
0251                    *       CLEAR PROTECT REGION
0252                    *
0253                            REF   CLEARP
0254  00BA     43               TEXT  'CP'
0255  00BC   FFFF               DATA  PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0256  00BF   0000               DATA  CLEARP
0257                    *
0258                    *       FIND WORD
0259                    *
0260                            REF   FWD
0261  00C0     46               TEXT  'FW'
0262  00C2   55FF               DATA  PH*4+PH*4+PH*4+PH*4+PN*4+PN*4+PN*4+PN
0263  00C4   0000               DATA  FWD
0264                    *
0265                    *       FIND BYTE
0266                    *
0267                            REF   FBT
0268  00C6     42               TEXT  'FB'
0269  00C8   55FF               DATA  PH*4+PH*4+PH*4+PH*4+PN*4+PN*4+PN*4+PN
0270  00CA   0000               DATA  FBT
0271                    *
0272                    *       HEX ARITHMETIC
0273                    *
0274                            REF   HXAR
0275  00CC     48               TEXT  'HA'
0276  00CE   5FFF               DATA  PH*4+PH*4+PN*4+PN*4+PN*4+PN*4+PN*4+PN
0277  00D0   0000               DATA  HXAR
0278                    *
0279                    *
0280                    *
0281          00D2'   NOCMDS    EQU   $
0282  00D2   0023               DATA  35
0283                            END
0000 ERS
```

960 - 980 CONCORDANCE

```
SREGN   0215        0037
SSS                 0206   0209

THERE ARE    0033   SYMBOLS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV	
SHEET	

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS INCORPORATED
Equipment Group *Dallas, Texas*

AL, CONVRT, ASCII/BINARY CONVERSION-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945352-9901
SCALE	REV.	SHEET 1 of 13

TI-0023-C

```
0003          *  PROCEDURE CNVRT(SCB,VALUE);
0004          *     /* CNVRT USES GENERALIZED BASE CONVERSION
0005          *        ROUTINES TO PERFORM ASCII TO/FROM BINARY
0006          *        CONVERSIONS. CONVERSION IS PERFORMED
0007          *        BY USING A TABLE TO PROVIDE THE ORDERED
0008          *        SET OF NUMERIC  CHARACTERS WITH MODULUS
0009          *        ARITHMETIC IN THE SPECIFIED BASE ON THE
0010          *        ASCII STRING TO DO THE CONVERSION. */
0011          *     DECLARE
0012          *        ( 1 SCB,                 /* SUPV CALL BLOCK */
0013          *             2 SCCMD     FIXED(8),
0014          *             2 SCERR     FIXED(8),
0015          *             2 SCDATA,
0016          *                3 STRING(6) CHAR(1));
0017          *     DECLARE
0018          *        ( 1 VAL,
0019          *             2 VALUE   FIXED(16),
0020          *             2 DUM     FIXED(16));
0021          *     DECLARE
0022          *        (DAB,BDA,HAB,BHA) FIXED(8);
0023          *     DECLARE          /* CONVERSION TABLE */
0024          *        CLST(16)  CHAR(1)
0025          *           INIT ('0','1',...,'9','A',...'F');
0026          *     CALL GET WKSP(VALUE,R0)
0027          *     ERR=0
0028          *     SCBSV=LOC(SCB)
0029          *     DO CASE  UNTIL ERROR OCCURS;
0030          *        C1: SCB(SCCMD)=DAB;
0031          *             /* 1ST CHAR IN STRING IS SIGN */
0032          *           CALL CNVF(LOC(STRING(2),5,VAL,10,ERR);
0033          *            IF VALUE>32678 THEN SIGNAL
0034          *              ERROR OCCURS;
0035          *           IF STRING(1) NE '-' OR '+'
0036          *              OR ' ' OR '0' THEN
0037          *              ERROR OCCURS
0038          *           IF STRING(1)='-' THEN
0039          *              VALUE = -VALUE;
0040          *        C2: SCB(SCCMD)=HAB;
0041          *           CALL CNVF(LOC(STRING(1),4,VAL,16,ERR);
0042          *        C3: SCB(SCCMD)=BDA;
0043          *           IF VALUE>=0 THEN DO;
0044          *              STRING(1)='+';
0045          *           END;ELSE
0046          *              VALUE=-VALUE;
0047          *              STRING(1)='-';
0048          *           CALL CNVR(LOC(STRING(2)),5,VAL,10);
0049          *        C4: SCB(SCCMD)=BHA;
0050          *           CALL CNVR(LOC(STRING(1),4,VAL,16);
0051          *        C5:  SIGNAL ERROR OCCURS;
0052          *     END CASE;
0053          *     ERROR OCCURS:  ERR= ERR+1;
0054          *     CALL SET WKSP(VALUE,R0);
0055          *     SCB(SCERR) = ERR;
0056          *     RETURN
```

```
0057                *       PROCEDURE CNVF(STR,NUM,VAL,RADIX,ERR)
0058                *          /* CNVF CONVERTS AN ASCII STRING WHICH
0059                *            REPRESENTS A BINARY VALUE TO BINARY
0060                *            CONVERSION TAKES PLACE IN ANY BASE
0061                *            FROM 2-16. IF ANY CHARACTER IS INVALID
0062                *            OR THE RESULT CANNOT BE REPRESENTED IN
0063                *            16 BITS, AN ERROR IS RETURNED. */
0064                *          DECLARE
0065                *             STR POINTER
0066                *             NUM FIXED(16),
0067                *             VAL FIXED(32),  /*VALUE IN MOST SIGNIFICANT
0068                *                               16 BITS */
0069                *             RADIX FIXED(16),
0070                *             ERR FIXED(16),
0071                *             I   FIXED(16),
0072                *             J   FIXED(16),
0073                *          I=0;
0074                *          VAL=0;
0075                *          DO UNTIL NO LEADING BLANKS .OR. ERROR;
0076                *             IF (C(STR) .NE. ' ') THEN
0077                *                SIGNAL NO LEADING BLANKS;
0078                *             I=I+1
0079                *             IF I>=NUM THEN SIGNAL ERROR,
0080                *             STR=STR+1
0081                *          END
0082                *          NO LEADING BLANKS:
0083                *             DO UNTIL I=NUM .OR. ERROR;
0084                *                VALUE=VALUE*BASE;
0085                *                IF VALUE>2**16 THEN SIGNAL ERROR;
0086                *                DO FOR J=BASE TO 0 BY -1 UNTIL MATCH,
0087                *                OR ERROR
0088                *                   IF (C(STR)=CLST(J-1) THEN SIGNAL MATCH;
0089                *                   J=J-1;
0090                *                   IF J=0 THEN SIGNAL ERROR;
0091                *                END;
0092                *                MATCH:
0093                *                   VALUE=VALUE+J-1
0094                *                STR=STR+1
0095                *                I=I+1
0096                *             END;
0097                *          END;
0098                *          ERROR:
0099                *             ERR=ERR+1
0100                *          END;
0101                *       END CNVF;
0102                *       PROCEDURE CNVR(STR,NUM,VAL,RADIX)
0103                *          /* CNVR CONVERTS A BINARY VALUE
0104                *            TO A CORRESPONDING ASCII STRING
0105                *            IN THE SPECIFIED RADIX. THE BINARY
0106                *            VALUE IS A POSITIVE VALUE REPRESENTED
0107                *            IN 16 BITS  */
0108                *          DECLARE
0109                *             STR POINTER,
0110                *             NUM FIXED(16),  /* # CHARS IN TARGET */
0111                *             VAL FIXED(32)
```

```
0112              *             RADIX FIXED(16),
0113              *          STR=STR+NUM-1;
0114              *          DO WHILE NUM>0;
0115              *             REM=MOD(VAL,RADIX);
0116              *             VAL=VAL/RADIX;
0117              *             C(STR)=CLST(REM);
0118              *             STR=STR-1;
0119              *             NUM=NUM-1;
0120              *          END;
0121              *       END CNVR;
0122              *    END CNVRT
0123                       IDT  'CONVRT'
0124              * TITLE:    CNVRT
0125              *           CONVERT
0126              * REVISION:
0127              *           ORIGINAL
0128              * COMPUTER: 990,ASM
0129              * ABSTRACT: USES GENERALIZED BASE CONVERSION ROUTINES
0130              *           TO PERFORM ASCII TO/FROM BINARY
0131              *           CONVERSIONS. CONVERSION IS PERFORMED
0132              *           BY USING A TABLE TO PROVIDE THE ORDERED
0133              *           SET OF NUMERIC CHARACTERS WITH MODULUS
0134              *           ARITHMETIC IN THE SPECIFIED BASE ON THE
0135              *           ASCII STRING TO DO THE CONVERSION.
0136              * CALLING SEQUENCE:
0137              *           BL   @CNVRT
0138              *           R10 = ADDRESS OF PRB BLOCK
0139              *                          1-PROCEDURE CNVRT(SCB,VALUE);
0140              *                          2-/* CNVRT USES GENERALIZED BA
0141              *                          2-   ROUTINES TO PERFORM ASCII
0142              *                          2-   CONVERSIONS. CONVERSION I
0143              *                          2-   BY USING A TABLE TO PROVI
0144              *                          2-   SET OF NUMERIC  CHARACTER
0145              *                          2-   ARITHMETIC IN THE SPECIFI
0146              *                          2-   ASCII STRING TO DO THE CO
0147              *                          2-DECLARE
0148              *                          3-( 1 SCB,                /* SUP
0149              *                          4-  2 SCCMD     FIXED(8),
0150              *                          4-  2 SCERR     FIXED(8),
0151              *                          4-  2 SCDATA,
0152              *                          5-  3 STRING(6) CHAR(1));
0153        0000  SCCMD  EQU  0             COMMAND
0154        0001  SCERR  EQU  1             ERROR RETURN
0155        0002  SCSTR  EQU  2             CONVERSION STRING
0156              *                          2-DECLARE
0157              *                          3-( 1 VAL,
0158              *                          4-  2 VALUE   FIXED(16),
0159              *                          4-  2 DUM     FIXED(16));
0160              *                          2-DECLARE
0161              *                          3-(DAB,BDA,HAB,BHA) FIXED(8);
0162        0B00  DAB    EQU  >B00          DECIMAL ASCII TO BINARY
0163        0A00  BDA    EQU  >A00          BINARY TO DECIMAL ASCII
0164        0C00  BHA    EQU  >C00          BINARY TO HEX ASCII
0165        0D00  HAB    EQU  >D00          HEX ASCII TO BINARY
0166              *      REF'S AND DEF'S
```

```
0167                    *
0168                            DEF   CNVRT
0169                            REF   ZERO
0170                            REF   PLUS
0171                            REF   MINUS
0172                            REF   BLANK
0173                            REF   CLST
0174                            REF   GETBUF
0175                            REF   RETBUF
0176                            REF   LWP
0177                            REF   RWP
0178                            REF   ACL
0179                            REF   RR
0180                    *                                    2=DECLARE         /* CONVERSION
0181                    *                                    3=CLST(16)  CHAR(1)
0182                    *                                    4=INIT ('0','1',...,'9','A',...
0183  0000     30       CLST   TEXT '0123456789ABCDEF'
0184                    *
0185                    *WORKSPACE REGISTER DEFINITIONS
0186                    *
0187            0000    R0      EQU  0
0188            0001    R1      EQU  1
0189            0002    R2      EQU  2
0190            0003    R3      EQU  3
0191            0004    R4      EQU  4
0192            0005    R5      EQU  5
0193            0006    R6      EQU  6
0194            0007    R7      EQU  7
0195            0008    R8      EQU  8
0196            0009    R9      EQU  9
0197            000A    R10     EQU  10
0198            000B    R11     EQU  11
0199            000C    R12     EQU  12
0200            000D    R13     EQU  13
0201            000E    R14     EQU  14
0202            000F    R15     EQU  15
0203                    *
0204                    *
0205                    *
0206                    *       REGISTER USAGE = LOCAL WORKSPACE
0207                    *
0208                    *       R14 = CALLER'S WP
0209                    *       R10 = SCB @
0210                    *        9  = # CHARS TO CONVERT
0211                    *        8  = BASE
0212                    *        7
0213                    *        6   =CONVERSION VALUE
0214                    *        5   =ERR
0215                    *        1   =SCB @
```

```
0217          0010' CNVRT  EQU  $
0218  0010  0420         BLWP @ACL
      0012  0000
0219  0014  C3AD         MOV  @R13*2(R13),R14
      0016  001A
0220                *                                  2-CALL GET WKSP(VALUE,R0)
0221  0018  C19E         MOV  *R14,R6
0222                *                                  2-ERR=0
0223  001A  04C5         CLR  R5
0224                *                                  2-SCBSV=LOC(SCB)
0225  001C  C04A         MOV  R10,R1
0226                *                                  2-DO CASE  UNTIL ERROR OCCURS;
0227                *                                  3-C1; SCB(SCCMD)=DAB;
0228  001E  04C4         CLR  R4
0229  0020  D11A         MOVB *R10,R4       COMMAND FIELD
0230  0022  0284         CI   R4,DAB
      0024  0B00
0231  0026  1619         JNE  CNV010        IF NOT DAB
0232                *                                  4-  /* 1ST CHAR IN STRING IS S
0233                *                                  4-CALL CNVF(LOC(STRING(2),5,VA
0234  0028  022A         AI   R10,SCSTR+1   START OF DEC STRING
      002A  0003
0235  002C  0209         LI   R9,5          # CHARS TO CONVERT
      002E  0005
0236  0030  0208         LI   R8,10         ARITHMETIC BASE
      0032  000A
0237  0034  06A0         BL   @CNVF
      0036  00C4'
0238                *                                  4- IF VALUE>32678 THEN SIGNAL
0239                *                                  5-ERROR OCCURS;
0240  0038  C186         MOV  R6,R6
0241  003A  113C         JLT  CNV050
0242  003C  060A         DEC  R10
0243                *                                  4-IF STRING(1) NE '-' OR '+'
0244                *                                  5-OR ' ' OR '0' THEN
0245                *                                  5-ERROR OCCURS
0246  003E  981A         CB   *R10,@ZERO
      0040  0000
0247  0042  1330         JEQ  CNV060
0248  0044  981A         CB   *R10,@PLUS
      0046  0000
0249  0048  1336         JEQ  CNV060
0250  004A  981A         CB   *R10,@BLANK
      004C  0000
0251  004E  1333         JEQ  CNV060
0252  0050  981A         CB   *R10,@MINUS
      0052  0000
0253  0054  162F         JNE  CNV050
0254                *                                  4-IF STRING(1)='-' THEN
0255                *                                  5-VALUE = -VALUE;
0256  0056  0580         NEG  R0
0257  0058  102F         JMP  CNV060
0258                *                                  3-C2; SCB(SCCMD)=HAB;
0259          005A' CNV010 EQU  $
```

```
0260  005A  0284          CI    R4,HAB        IF NOT HEX TO BINARY;
      005C  0D00
0261  005E  1608          JNE   CNV020
0262                *                              4=CALL CNVF(LOC(STRING(1),4,VA
0263  0060  05CA          INCT  R10
0264  0062  0209          LI    R9,4          # CHARS
      0064  0004
0265  0066  0208          LI    R8,16         RADIX
      0068  0010
0266  006A  06A0          BL    @CNVF
      006C  00C4'
0267  006E  1023          JMP   CNV060
0268                *                              3=C3: SCB(SCCMD)=BDA;
0269        0070' CNV020 EQU   $
0270  0070  0284          CI    R4,BDA        IF NOT BIN TO DECIMAL ASCII
      0072  0A00
0271  0074  1614          JNE   CNV045
0272                *                              4=IF VALUE>=0 THEN DO;
0273  0076  C186          MOV   R6,R6
0274  0078  1104          JLT   CNV030
0275                *                              5=STRING(1)=' ';
0276  007A  DAA0          MOVB  @BLANK,@SCSTR(R10)
      007C  004C'
      007E  0002
0277                *                              4=END;ELSE
0278  0080  1004          JMP   CNV040
0279        0082' CNV030 EQU   $
0280                *                              5=VALUE=-VALUE;
0281  0082  0746          ABS   R6
0282                *                              5=STRING(1)='-';
0283  0084  DAA0          MOVB  @MINUS,@SCSTR(R10)
      0086  0052'
      0088  0002
0284        008A' CNV040 EQU   $
0285                *                              4=CALL CNVR(LOC(STRING(2)),5,V
0286  008A  022A          AI    R10,SCSTR+1   STRING(2)
      008C  0003
0287  008E  0209          LI    R9,5          # CHARS
      0090  0005
0288  0092  0208          LI    R8,10         BASE
      0094  000A
0289  0096  06A0          BL    @CNVR
      0098  0102'
0290  009A  060A          DEC   R10
0291  009C  100C          JMP   CNV060
0292        009E' CNV045 EQU   $
0293                *                              3=C4: SCB(SCCMD)=BHA;
0294  009E  0284          CI    R4,BHA
      00A0  0C00
0295  00A2  1608          JNE   CNV050        IF INVALID COMMAND
0296                *                              4=CALL CNVR(LOC(STRING(1),4,VA
0297  00A4  05CA          INCT  R10           STRING(1)
0298  00A6  0209          LI    R9,4          #CHARS
      00A8  0004
0299  00AA  0208          LI    R8,16         BASE
```

```
      00AC  0010
0300  00AE  06A0          BL    @CNVR
      00B0  0102'
0301  00B2  1001          JMP   CNV060
0302                *                                  3=CS;  SIGNAL ERROR OCCURS;
0303                *                                  2=END CASE;
0304                *                                  2=ERROR OCCURS;  ERR= ERR+1;
0305        00B4' CNV050  EQU   $
0306  00B4  0585          INC   R5
0307        00B6' CNV060  EQU   $
0308                *                                  2=CALL SET WKSP(VALUE,R0);
0309  00B6  C786          MOV   R6,*R14
0310                *                                  2=SCB(SCERR) = ERR;
0311  00B8  06C5          SWPB  R5
0312  00BA  0845          MOVB  R5,@SCERR(R1)
      00BC  0001
0313                *                                  2=RETURN
0314  00BE  0420          BLWP  @RR
      00C0  0000
0315  00C2  045B          RT
```

```
0317                    * TITLE:    CNVF
0318                    *           CONVERT ASCII TO BINARY
0319                    * REVISION:
0320                    *           ORIGINAL
0321                    * COMPUTER: 990,ASM
0322                    * ABSTRACT: CNVF CONVERTS AN ASCII STRING WHICH
0323                    *           REPRESENTS A BINARY VALUE TO BINARY.
0324                    *           CONVERSION TAKES PLACE IN ANY BASE
0325                    *           FROM 2-16. IF ANY CHARACTER IS ILLEGAL
0326                    *           OR THE RESULT CANNOT BE REPRESENTED IN
0327                    *           16 BITS, AN ERROR IS RETURNED.
0328                    * CALLING SEQUENCE:
0329                    *           BL   @CNVF
0330                    *                                2-PROCEDURE CNVF(STR,NUM,VAL,R
0331                    *                                3-/* CNVF CONVERTS AN ASCII ST
0332                    *                                3-  REPRESENTS A BINARY VALUE
0333                    *                                3-  CONVERSION TAKES PLACE IN
0334                    *                                3-  FROM 2-16, IF ANY CHARACTE
0335                    *                                3-  OR THE RESULT CANNOT BE RE
0336                    *                                3-  16 BITS, AN ERROR IS RETUR
0337                    *                                3-DECLARE
0338                    *                                4-STR POINTER
0339                    *                                4-NUM FIXED(15),
0340                    *                                4-VAL FIXED(32),  /*VALUE IN M
0341                    *                                4-                  16 BITS */
0342                    *                                4-RADIX FIXED(15),
0343                    *                                4-ERR FIXED(15),
0344                    *                                4-I   FIXED(15),
0345                    *                                4-J   FIXED(15),
0346                    *
0347                    *           REGISTER ASSIGNMENTS
0348                    *           R11 - RETURN
0349                    *           R10 - STRING @
0350                    *           R9  - # CHARS
0351                    *           R8  - RADIX
0352                    *           R6,R7 - VALUE
0353                    *           R5  - ERR
0354                    *           R4  - I
0355                    *           R3  - J
0356                    *           R2 - WORKING STRING @
0357                    *
0358          00C4' CNVF    EQU  $
0359    00C4  C08A          MOV  R10,R2
0360                    *                                3-I=0;
0361    00C6  04C4          CLR  R4
0362                    *                                3-VAL=0;
0363    00C8  3984          MPY  R4,R6          CLEAR R6,R7
0364                    *                                3-DO UNTIL NO LEADING BLANKS .
0365          00CA' CNVF10  EQU  $
0366                    *                                4-IF (C(STR) ,NE, ' ') THEN
0367                    *                                5-SIGNAL NO LEADING BLANKS;
0368    00CA  9812          CB   *R2,@BLANK
      00CC  007C'
0369    00CE  1605          JNE  CNVF20
```

```
0370                    *                                  4=I=I+1
0371  00D0  0584               INC   R4
0372                    *                                  4=IF I>=NUM THEN SIGNAL ERROR,
0373  00D2  8244               C     R4,R9
0374  00D4  1414               JHE   CNVF50
0375                    *                                  4=STR=STR+1
0376  00D6  0582               INC   R2
0377                    *                                  3=END
0378  00D8  10F8               JMP   CNVF10
0379                    *                                  3=NO LEADING BLANKS;
0380        00DA' CNVF20 EQU   $
0381                    *                                  4=DO UNTIL I=NUM ,OR, ERROR;
0382                    *                                  5=VALUE=VALUE*BASE;
0383  00DA  3988               MPY   R8,R6
0384                    *                                  5=IF VALUE>2**16 THEN SIGNAL E
0385  00DC  C186               MOV   R6,R6                    ERROR= RESULT MUST BE ENTIREL
0386  00DE  160F               JNE   CNVF50                   IN RIGHT HALFWORD
0387  00E0  C187               MOV   R7,R6
0388                    *                                  5=DO FOR J=BASE TO 0 BY -1 UNT
0389                    *                                  5=OR ERROR
0390  00E2  C0C8               MOV   R8,R3          RADIX=> J
0391        00E4' CNVF30 EQU   $
0392                    *                                  6=IF (C(STR)=CLST(J=1) THEN SI
0393  00E4  98D2               CB    *R2,@CLST-1(R3)
      00E6  FFFF'
0394  00E8  1303               JEQ   CNVF40
0395                    *                                  6=J=J-1;
0396                    *                                  6=IF J=0 THEN SIGNAL ERROR;
0397  00EA  0603               DEC   R3
0398  00EC  15F8               JGT   CNVF30
0399  00EE  1007               JMP   CNVF50
0400                    *                                  5=END;
0401                    *                                  5=MATCH;
0402                    *                                  6=VALUE=VALUE+J-1
0403        00F0' CNVF40 EQU   $
0404  00F0  0603               DEC   R3             J
0405  00F2  A183               A     R3,R6          VALUE
0406                    *                                  5=STR=STR+1
0407  00F4  0582               INC   R2
0408                    *                                  5=I=I+1
0409  00F6  0584               INC   R4
0410                    *                                  4=END;
0411                    *                                  3=END;
0412  00F8  8244               C     R4,R9
0413  00FA  11EF               JLT   CNVF20
0414  00FC  1001               JMP   CNVF60
0415                    *                                  3=ERROR;
0416        00FE' CNVF50 EQU   $
0417                    *                                  4=ERR=ERR+1
0418  00FE  0585               INC   R5
0419                    *                                  3=END;
0420        0100' CNVF60 EQU   $
0421                    *                                  2=END CNVF;
0422  0100  045B               RT
```

```
 0424                    *                                  2*PROCEDURE CNVR(STR,NUM,VAL,R
 0425                    *                                  3*/* CNVR CONVERTS A BINARY VA
 0426                    *                                  3*  TO A CORRESPONDING ASCII S
 0427                    *                                  3*  IN THE SPECIFIED RADIX. TH
 0428                    *                                  3*  VALUE IS A POSITIVE VALUE
 0429                    *                                  3*  IN 16 BITS  */
 0430                    *                                  3*DECLARE
 0431                    *                                  4*STR POINTER,
 0432                    *                                  4*NUM FIXED(16),  /* # CHARS I
 0433                    *                                  4*VAL FIXED(32)
 0434                    *                                  4*RADIX FIXED(16),
 0435                    *           REGISTER ASSIGNMENTS
 0436                    *           R11    * RETURN
 0437                    *           R10    * STR (STRING #)
 0438                    *           R9     * # CHARS
 0439                    *           R8     * RADIX
 0440                    *           R6     * VALUE
 0441                    *           R5     * RESERVED
 0442                    *           R7     * REMAINDER
 0443                    *           R2    * WORKING STRING#
 0444         0102' CNVR   EQU  $
 0445  0102   C08A         MOV  R10,R2
 0446                    *                                  3*STR*STR+NUM*1;
 0447  0104   A089         A    R9,R2
 0448  0106   0602         DEC  R2
 0449                    *                                  3*DO WHILE NUM>0;
 0450         0108' CNVR10 EQU  $
 0451  0108   C249         MOV  R9,R9
 0452  010A   1208         JLE  CNVR20
 0453                    *                                  4*REM*MOD(VAL,RADIX);
 0454                    *                                  4*VAL*VAL/RADIX;
 0455  010C   C1C6         MOV  R6,R7
 0456  010E   04C6         CLR  R6
 0457  0110   3D88         DIV  R8,R6
 0458                    *                                  4*C(STR)*CLST(REM);
 0459  0112   D4A7         MOVB @CLST(R7),*R2
       0114   0000'
 0460                    *                                  4*STR*STR*1;
 0461  0116   0602         DEC  R2
 0462                    *                                  4*NUM*NUM*1;
 0463  0118   0609         DEC  R9
 0464                    *                                  3*END;
 0465  011A   10F6         JMP  CNVR10
 0466                    *                                  2*END CNVR;
 0467         011C' CNVR20 EQU  $
 0468  011C   0458         RT
 0469                    *                                  1*END CNVRT
 0470                         END
0000 ERS
```

AL

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																
SHEET																															
REV STATUS OF SHEETS	REV																															
	SHEET																															

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, DSR733, 733ASR DEVICE DRIVER-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945353-9901

SCALE	REV.	SHEET
		1 of 77

TI-0023-C

```
0003                    IDT  'DSR733'
0004              *   PROCEDURE DSR733;
```

```
0006                *       DECLARE (1 PHYSICALRECORDBLOCK,
0007                *                  3 PRB CONTROL (PRBPTR),
0008                *                     5 IOOP BIT (8),
0009                *                     5 LUNO BIT (8),
0010                *                     5 SYSTEMFLAGS,
0011                *                        7 FILLER1 BIT (1),
0012                *                        7 UNRECOVERABLEERROR BIT (1),
0013                *                        7 EOFFLAG BIT (1),
0014                *                        7 FILLER2 BIT (5),
0015                *                     5 USERFLAGS,
0016                *                        7 FILLER3 BIT (3),
0017                *                        7 CHARACTERIO BIT (1),
0018                *                        7 NOWAITFLAG BIT(1),
0019                *                        7 FILLER 4 BIT(3),
0020                *                     5 BUFFERADDRESS POINTER,
0021                *                     5 BUFFERLENGTH FIXED (16),
0022                *                     5 CHARCOUNT FIXED (16),
0023                *                     5 PRBRESERVED BIT (32));
0024                *                  3 BSFLAG FIXED (16) ;
```

```
0026                *       DECLARE PRBPTR POINTER;
0027                *       DECLARE DCTPTR POINTER;
0028                *       DECLARE (1 DEVICECONTROLTABLE733ASR,
0029                *                  3 DCT733 CONTROL (DCTPTR),
0030                *                     5 CRUBASE FIXED (16),
0031                *                     5 CASSETTECONTROL (2),
0032                *                        7 PLAYBACK BIT (1),
0033                *                        7 RECORD BIT (1),
0034                *                        7 FILLER10 BIT (6));
```

```
0036                *       DECLARE BUFFER (1) CHARACTER
0037                *           CONTROL (BUFFERADDRESS);
0038                *       DECLARE PARMLIST LITERALLY
0039                *           'PRBPTR, DEVTYPE,UNITNO, DCTPTR';
0040                *       DECLARE (1 ASR733INPUT,
0041                *                   3 ASR733IN CRUCONTROL (CRUBASE),
0042                *                      5 INCHAR BIT (8),
0043                *                      5 XMTING  BIT (1),
0044                *                      5 ASRID   BIT (1),
0045                *                      5 WRQ     BIT (1),
0046                *                      5 RRQ     BIT (1),
0047                *                      5 DCD     BIT (1),
0048                *                      5 DSR     BIT (1),
0049                *                      5 INT     BIT (1));
0050                *       DECLARE (1 ASR733OUTPUT,
0051                *                   3 ASR733OUT CRUCONTROL (CRUBASE),
0052                *                      5 OUTCHAR BIT (8),
0053                *                      5 FILLER7331 BIT (1),
0054                *                      5 DTR     BIT (1),
0055                *                      5 RTS     BIT (1),
0056                *                      5 CLRWRQ  BIT (1),
0057                *                      5 CLRRRQ  BIT (1),
0058                *                      5 CLRNSF  BIT (1));
0059                *       DECLARE MODRCD LITERALLY '"4080"';
0060                *       DECLARE MODPBK LITERALLY '"8040"';
0061                *       DECLARE MODBTS BIT (16) INITIAL ("C0C0");
0062                *       DECLARE (1 STATUS,
0063                *                   3 FILLERSTT1 BIT (2),
0064                *                   3 PNTRDY BIT (1),
0065                *                   3 RCDRDY BIT (1),
0066                *                   3 BOE (1) BIT (1),
0067                *                   3 PBKERR BIT (1),
0068                *                   3 PBKRDY BIT (1));
```

```
0071                *       DECLARE DLE LITERALLY '"10"',
0072                *               DC2 LITERALLY '"12"',
0073                *               DC3 LITERALLY '"13"',
0074                *               DC4 LITERALLY '"14"',
0075                *               CR  LITERALLY '"0D"',
0076                *               LF  LITERALLY '"0A"',
0077                *               DEL LITERALLY '"7F"',
0078                *               SP  LITERALLY '"20"',
0079                *               HT  LITERALLY '"09"',
0080                *               ESC LITERALLY '"1B"',
0081                *               BEL LITERALLY '"07"',
0082                *               FF  LITERALLY '"0C"',
0083                *               ETB LITERALLY '"17"',
0084                *               VT  LITERALLY '"0B"',
0085                *               BS  LITERALLY '"08"';
0086                *       DECLARE MDID LITERALLY '"80"';
0087                *       DECLARE REWIND LITERALLY '"31"+MDID',
0088                *               LOAD LITERALLY '"33"+MDID',
0089                *               RECORD LITERALLY '"35"+MDID',
0090                *               PLYBK LITERALLY '"36"+MDID',
0091                *               FWDBLK LITERALLY '"37"',
0092                *               BCKBLK LITERALLY '"38"',
0093                *               STATUS LITERALLY '"3C"',
0094                *               ENDGRP LITERALLY '"FF"',
0095                *               DLYGRP LITERALLY '"FE"';
```

```
0098          *     DECLARE RECORDMODE (7) BIT (8) INITIAL
0099          *        (DLYGRP, 60, DLE, DLYGRP, 3, RECORD, ENDGRP);
0100          *     DECLARE PLYBACKMODE (7) BIT (8) INITIAL
0101          *        (DLYGRP, 60, DLE, DLYGRP, 3, PLYBK, ENDGRP);
0102          *     DECLARE READTAPE (5) BIT (8) INITIAL
0103          *        (DLE, DLYGRP, 3, FWDBLK, ENDGRP);
0104          *     DECLARE WTTAPESTART (4) BIT (8) INITIAL
0105          *        (DC2, DLYGRP, 3, ENDGRP);
0106          *     DECLARE WTTAPEEND (9) BIT (8) INITIAL
0107          *        (CR, LF, DC4, DLYGRP, 3, DEL, DLYGRP, 3,
0108          *        ENDGRP);
0109          *     DECLARE EOFSEQ (4) BIT (8) INITIAL
0110          *        (DC3, ENDGRP);
0111          *     DECLARE REWINDSEQ (9) BIT (8) INITIAL
0112          *        (DLYGRP, 60, DLE, DLYGRP, 3, REWIND,
0113          *        DLYGRP, 180,ENDGRP);
0114          *     DECLARE LOADSEQ (5) BIT (8) INITIAL
0115          *        (DLE, DLYGRP, 3, LOAD, ENDGRP);
0116          *     DECLARE BACKSPC (7) BIT (8) INITIAL
0117          *        (DLE, DLYGRP, 3, BCKBLK, DLYGRP, 48, ENDGRP);
0118          *     DECLARE ECHOFF  ( 4)  BIT (8) INITIAL
0119          *        (LF, DLYGRP, 3, ENDGRP);
0120          *     DECLARE STATUSSEQ (5) BIT (8) INITIAL
0121          *        (DLE, DLYGRP, 3, STATUS, ENDGRP);
0122          *     DECLARE ECHOBS (7) BIT (8) INITIAL
0123          *        (BS, DLYGRP, 3, ENDGRP);
0124          *     DECLARE ECHODEL (7) BIT (8) INITIAL
0125          *        (LF, DLYGRP, 3, CR, DLYGRP, 27, ENDGRP);
0126          *     DECLARE BYTMOD (6) BIT (8) INITIAL
0127          *        (1, 0, 1, 0, 1, -1);
```

```
0130          *   PROCEDURE CSOPEN (PARMLIST);
0131          *      CALL RSTMOD;
0132          *      RETURN;
0133          *      END CSOPEN;
```

```
0136                *    PROCEDURE CSFWD (PARMLIST);
0137                *       DECLARE (1 DUMMYSVCBLOCK,
0138                *                  3 DMYSVCCODE BIT (8),
0139                *                  3 FILLERDMY1 BIT (8),
0140                *                  3 DMYIOOP BIT (8),
0141                *                  3 DMYLUNO BIT (8),
0142                *                  3 DMYSYSFLAGS,
0143                *                     5 FILLERDMY2 BIT (2),
0144                *                     5 DMYEOFFLAG BIT (1),
0145                *                     5 FILLERDMY3 BIT (5),
0146                *                  3 DMYUSRFLAGS BIT (8),
0147                *                  3 DMYBFRADDR POINTER,
0148                *                  3 DMYBFRLNTH FIXED (16),
0149                *                  3 DMYCHRCOUNT FIXED (16));
0150                *       DECLARE DMYBUFFER BIT (8);
0151                *       DECLARE READASCII LITERALLY '9';
0152                *       DECLARE IOREQUEST LITERALLY '0';
0153                *       DECLARE I FIXED;
0154                *       DECLARE WKSP POINTER;
0155                *       DMYSVCCODE = IOREQUEST;
0156                *       DMYIOOP = READASCII;
0157                *       DMYLUNO = LUNO;
0158                *       DMYSYSFLAGS = 0;
0159                *       DMYUSRFLAGS = 0;
0160                *       DMYBFRADDR = ADDR (DMYBUFFER);
0161                *       DMYBFRLNTH = 1;
0162                *       I = 0;
0163                *       CALL GETBUF (WKSP);
0164                *       DO WHILE I < CHARCOUNT & DMYEOFFLAG = 0;
0165                *          CALL IO (WKSP);
0166                *          SYSTEMFLAGS = DMYSYSFLAGS;
0167                *          I = I + 1;
0168                *          END;
0169                *       CALL RETBUF (WKSP);
0170                *       RETURN;
0171                *       END CSFWD;
```

```
0174              *   PROCEDURE CSBACK (PARMLIST);
0175              *      DECLARE I FIXED;
0176              *      CALL CHGMOD (PRBPTR, DCTPTR, MODPBK);
0177              *      DO I FROM 1 TO CHARCOUNT;
0178              *         CALL CMND (ADDR (BKSPC));
0179              *         END;
0180              *      RETURN;
0181              *      END CSBACK;
```

```
0184          *   PROCEDURE CSRASC (PARMLIST);
0185          *      DECLARE CHAR BIT (8);
0186          *      DECLARE I FIXED;  /* CHARACTER CURSOR */
0187          *      CALL CHGMOD (PRBPTR, DCTPTR, MODPBK);
0188          *      CALL STTCHR (CHAR);  /*  GET STATUS  */
0189          *      IF PBKRDY = 0 THEN DO;
0190          *         UNRECOVERABLEERROR = 1;
0191          *         RETURN;
0192          *         END;
0193          *      I = 0;
0194          *      CALL CMND (ADDR (RTAPE));
0195          *      CALL RCVCHR (CHAR, WAIT);
0196          *      IF CHAR = LF THEN CALL RCVCHR (CHAR, WAIT);
0197          *      IF CHAR = DEL THEN CALL RCVCHR (CHAR, WAIT);
0198          *      DO WHILE CHAR .NE. CR & I < BUFFERLENGTH
0199          *               & EOFFLAG = 0;
0200          *         IF CHAR = ETB THEN CHAR = CR;
0201          *         IF  CHAR >= SP .OR.
0202          *            (CHAR >= BEL & CHAR <= HT) .OR.
0203          *             CHAR = FF  .OR.  CHAR = CR
0204          *            THEN DO;
0205          *               CALL PUTCHR (CHAR, I);
0206          *               I = I + 1;
0207          *               END;
0208          *         IF CHAR = DC3
0209          *            THEN DO;
0210          *               IF I = 0 THEN EOFFLAG = 1;
0211          *               CALL RCVCHR (CHAR, WAIT);
0212          *               IF CHAR .NE. CR
0213          *                     THEN CALL CMND (ADDR (RTAPE));
0214          *               END;
0215          *            ELSE CALL RCVCHR (CHAR, WAIT);
0216          *         END;
0217          *      DO WHILE CHAR .NE. CR;
0218          *         CALL RCVCHR (CHAR, WAIT);
0219          *         END;
0220          *      CHARCOUNT = I;
0221          *      RETURN;
0222          *      END CSRASC
```

```
0225              *   PROCEDURE CSWASC (PARMLIST);
0226              *      DECLARE I FIXED;
0227              *      DECLARE CHAR BIT (8);
0228              *      CALL CHGMOD (PRBPTR, DCTPTR, MODRCD);
0229              *      CALL STTCHR (CHAR) /*GET STATUS
0230              *      IF RCDRDY = 0 THEN DO;
0231              *         UNRECOVERABLE ERROR = 1;
0232              *         RETURN;
0233              *         END;
0234              *      CALL CMND (ADDR (WTAPES));
0235              *      DO I FROM 0 TO MIN (CHARCOUNT-1, 79);
0236              *         CALL GETCHR (CHAR, I)
0237              *         IF CHAR = CR THEN CHAR = ETB;
0238              *         IF CHAR >= SP .OR. (CHAR >= BEL & CHAR <= LF)
0239              *            .OR. CHAR = FF .OR. CHAR = ETB
0240              *            THEN CALL SNDCHR (CHAR, 0);
0241              *         END;
0242              *      CALL DELAY (CCOUNT)
0243              *      CALL CMND (ADDR (WTAPEE));
0244              *      RETURN;
0245              *      END CSWASC;
```

```
0248                *    PROCEDURE CSWEOF (PARMLIST);
0249                *       CALL CHGMOD (MODRCD);
0250                *       CALL CMND (ADDR (WTAPES));
0251                *       CALL CMND (ADDR (EOFSEQ));
0252                *       CALL CMND (ADDR (WTAPEE));
0253                *       RETURN;
0254                *       END CSWEOF;
```

```
0257              *   PROCEDURE CSRWND (PARMLIST);
0258              *      DECLARE I BIT (1);
0259              *      CALL RSTMOD;
0260              *      CALL CMND (ADDR (RWDSEQ));
0261              *      I = 0;
0262              *      DO WHILE I = 0;
0263              *         CALL STTCHR (STATUS);
0264              *         I = BOE (ABS (MOD (DRIVEID, 2) - 1));
0265              *         END;
0266              *      CALL CHGMOD (PRBPTR, DCTPTR, MODPBK);
0267              *      CALL CMND (ADDR (LOADSQ));
0268              *      I = 0;
0269              *      DO WHILE I = 0;
0270              *         CALL STTCHR (STATUS);
0271              *         I = PBKRDY
0272              *         END;
0273              *      RETURN;
0274              *      END CSRWND;
```

```
0277          *   PROCEDURE CSUNLD (PARMLIST);
0278          *      DECLARE I BIT (1);
0279          *      CALL RSTMOD;
0280          *      CALL CMND (ADDR (RWDSEQ));
0281          *      I = 0;
0282          *      DO WHILE I = 0;
0283          *         CALL STTCHR (STATUS);
0284          *         I = BOE (ABS (MOD (DRIVEID, 2) - 1));
0285          *         END;
0286          *      RETURN;
0287          *      END CSUNLD;
```

```
0290              *   PROCEDURE LGRASC (PARMLIST);
0291              *      DECLARE I FIXED (16);
0292              *      DECLARE CHAR BIT (8);
0293              *      I = 0;
0294              *      DO WHILE EOFFLAG = 0 & I < BUFFERLENGTH
0295              *            & UNRECOVERABLEERROR = 0;
0296              *         IF NOWAITFLAG=0 THEN
0297              *            CALL RCVCHR(CHAR,WAIT);
0298              *         ELSE CALL RCVCHR(CHAR,NOWAIT);
0299              *         IF NOWAITFLAG=1 & CHAR=0 THEN RETURN;
0300              *         IF CHAR NE BS THEN BSFLAG = 0;
0301              *         IF CHAR >= SP & CHAR < DEL THEN DO;
0302              *            CALL SNDCHR (CHAR,DELAY3);
0303              *            CALL PUTCHR (CHAR, I);
0304              *            I = I + 1;
0305              *            END;
0306              *         ELSE IF CHAR = DEL THEN DO;
0307              *            IF CHARACTERIO = 1
0308              *               THEN DO;
0309              *                  CALL PUTCHR (CHAR, I);
0310              *                  I = I + 1;
0311              *                  END;
0312              *            CALL CMND (ADDR (ECODEL));
0313              *            END;
0314              *         ELSE IF CHAR = BS THEN DO;
0315              *            IF CHARACTERIO = 1 THEN DO;
0316              *               CALL PUTCHR (CHAR, I);
0317              *               I = I + 2;
0318              *               END;
0319              *            I = MAX (I - 1, 0);
0320              *            CALL CMND (ADDR (ECOBS));
0321              *            IF BSFLAG EQ 0 THEN DO;
0322              *               BSFLAG = -1;
0323              *               CALL SNDCHR (LF,DELAY3);
0324              *               END
0325              *            END;
0326              *         ELSE IF CHAR = HT THEN DO;
0327              *            CALL PUTCHR (CHAR, I);
0328              *            CALL SNDCHR (' ', DELAY3);
0329              *            I = I + 1;
0330              *            END;
0331              *         ELSE IF CHAR = DC3 & I = 0 THEN EOFFLAG=1
0332              *         ELSE IF CHAR = CR THEN DO;
0333              *            CALL SNDCHR (CHAR, DLY27);
0334              *            CHARCOUNT = I;
0335              *            RETURN;
0336              *            END;
0337              *         ELSE IF CHAR = LF
0338              *            THEN CALL SNDCHR (CHAR, DELAY3)
0339              *         ELSE IF CHAR = ESC
0340              *            THEN UNRECOVERABLEERROR = 1;
0341              *         END;
0342              *      CHARCOUNT = I;
0343              *      RETURN;
```

```
0344                *       END LGRASC;
```

```
0347              *   PROCEDURE LGWASC (PARMLIST);
0348              *      DECLARE (I, J) FIXED (16);
0349              *      DECLARE CHAR BIT (8);
0350              *      DECLARE FOUND FIXED (16);
0351              *      I = CHARCOUNT;
0352              *      IF I = 0 THEN RETURN;
0353              *      I = I - 1;
0354              *      DO J FROM 0 TO I BY 1;
0355              *         CALL RCVCHR (CHAR, NOWAIT);
0356              *         IF CHAR = ESC THEN DO;
0357              *            UNRECOVERABLEERROR = 1;
0358              *            RETURN;
0359              *            END;
0360              *         CALL GETCHR (CHAR, J);
0361              *         IF CHAR = HT THEN CHAR = SP;
0362              *         IF CHAR >= SP .OR. (CHAR >= BEL & CHAR <= LF)
0363              *            THEN CALL SNDCHR (CHAR, DELAY3);
0364              *         IF CHAR = CR THEN CALL SNDCHR (CHAR, DLY27);
0365              *         IF CHAR = FF THEN DO K FROM 1 TO 8;
0366              *            CALL CMND (ADDR (ECOFF));
0367              *            END;
0368              *         END;
0369              *      RETURN;
0370              *      END LGWASC;
```

```
0373                *     PROCEDURE MONITORCHAR (CHAR);
0374                *       DECLARE CHAR CHARACTER (1);
0375                *
0376                *       CALL RCVCHR (CHAR, NOWAIT);
0377                *       RETRN;
0378                *       END MONITORCHAR;
```

```
0381                *   PROCEDURE CHGMOD (PRBPTR, DCTPTR, MODE);
0382                *      DECLARE MODE BIT (16);
0383                *      /* NOTE PRBPTR AND DCTPTR GLOBALLY
0384                *         DEFINED */
0385                *      DECLARE (INUSE, OTHER) FIXED;
0386                *      INUSE = MOD (UNITNO, 2);
0387                *      OTHER = ABS (INUSE-1);
0388                *      IF (MODE = MODPBK &
0389                *          CASSETTECONTROL (INUSE).PLAYBACK = 1)
0390                *         (MODE = MODRCD &
0391                *          CASSETTECONTROL (INUSE).RECORD = 1)
0392                *         THEN RETURN;
0393                *      CASSETTECONTROL (INUSE) = 0;
0394                *      CASSETTECONTROL (OTHER) = 0;
0395                *      IF MODE = MODPBK THEN DO;
0396                *            CASSETTECONTROL (INUSE).PLAYBACK = 1;
0397                *            CASSETTECONTROL (OTHER).RECORD = 1;
0398                *            CALL CMND (ADDR (PMODE));
0399                *            END;
0400                *         ELSE DO;
0401                *            CASSETTECONTROL (INUSE).RECORD = 1;
0402                *            CASSETTECONTROL (OTHER).PLAYBACK = 1;
0403                *            CALL CMND (ADDR (RMODE));
0404                *            END;
0405                *      RETURN;
0406                *      END CHGMOD;
```

```
0409          *   PROCEDURE STTCHR (STATUSCHAR);
0410          *      DECLARE STATUSCHAR BIT (8);
0411          *
0412          *      CALL CMND (ADDR (STTSEQ));
0413          *      CALL RCVCHR (STATUSCHAR, WAIT);
0414          *      RETURN;
0415          *      END STTCHR;
```

```
0418                *    PROCEDURE CMND (CMNDPTR, DRIVEID)
0419                *       DECLARE (CMNDPTR, BYTEPTR) POINTER;
0420                *       DECLARE COMMANDBYTE FIXED (8)
0421                *           CONTROL (BYTEPTR);
0422                *       DECLARE DRIVE-ID FIXED (8);
0423                *       DECLARE BYTE FIXED;
0424                *       BYTEPTR = COMNDPTR
0425                *       DO FOREVER;
0426                *          BYTE = COMMANDBYTE;
0427                *          BYTEPTR = BYTEPTR + 1;
0428                *          IF BYTE = ENDGROUP THEN EXIT DO;
0429                *          IF BYTE = DELAYGROUP THEN DO;
0430                *                CALL DLYCHR (COMMANDBYTE);
0431                *                BYTEPTR = BYTEPTR + 1;
0432                *                END;
0433                *             ELSE DO;
0434                *                IF BYTE < 0
0435                *                    THEN /* THIS IS A COMMAND BYTE
0436                *                            TO BE MODIFIED BY DRIVEID */ D
0437                *                   J = MOD (DRIVEID, 2);
0438                *                   IF J .NE. 0
0439                *                      THEN BYTE = BYTE + BYTMOD (BYTE);
0440                *                CALL SNDCHR (BYTE, 0);
0441                *                END;
0442                *          END;
0443                *       RETURN;
0444                *       END CMND;
```

```
0447                *   PROCEDURE SNDCHR (DELAYTIME, CHAR);
0448                *      DECLARE DELAYTIME FIXED;
0449                *      DECLARE CHAR BIT (8);
0450                *      IF DSR = 1 THEN DO;
0451                *            RTS = 1;
0452                *            DTR = 1;
0453                *            OUTCHR = CHAR;
0454                *            DO WHILE WRQ = 0;
0455                *               END;
0456                *            CLRWRQ = 1;
0457                *            CALL DLYCHR (DELAYTIME);
0458                *            END;
0459                *         ELSE UNRECOVERABLEERROR = 1;
0460                *      RETURN;
0461                *      END SNDCHR;
```

```
0464              *   PROCEDURE DLYCHR (DELAYCOUNT);
0465              *      DECLARE (DELAYCOUNT, I) FIXED;
0466              *
0467              *      I = DELAYCOUNT;
0468              *      DO WHILE I > 0;
0469              *         RTS = 0;
0470              *         OUTCHR = ANYTHING;
0471              *         DO WHILE WRQ = 0;
0472              *            END;
0473              *         CLRWRQ = 1;
0474              *         END;
0475              *      RETURN;
0476              *      END DLYCHR;
```

```
0479          *   PROCEDURE RCVCHR (WAITFLAG, CHAR);
0480          *      DECLARE WAITFLAG FIXED;
0481          *      DECLARE CHAR BIT (8);
0482          *      IF TIMERR = 1 THEN CLRRRQ = 1;
0483          *      DO WHILE RRQ = 0;
0484          *          IF WAITFLAG .NE. 0 THEN DO;
0485          *            CHAR = 0;
0486          *            RETURN;
0487          *            END;
0488          *         END;
0489          *      CHAR = INCHAR;
0490          *      CLRRRQ = 1;
0491          *      RETURN;
0492          *      END RCVCHR;
```

```
0495              *   PROCEDURE GETCHR (DISPLACEMENT, CHAR);
0496              *      DECLARE DISPLACEMENT FIXED (16);
0497              *      DECLARE CHAR BIT (8);
0498              *      CHAR = BUFFER (DISPLACEMENT);
0499              *      RETURN;
0500              *      END GETCHR;
```

```
0503              *   PROCEDURE PUTCHR (DISPLACEMENT, CHAR);
0504              *      DECLARE DISPLACEMENT FIXED;
0505              *      DECLARE CHAR BIT (8);
0506              *      BUFFER (DISPLACEMENT) = CHAR;
0507              *      RETURN;
0508              *      END PUTCHR;
```

```
0511                *   PROCEDURE RSTMOD;
0512                *
0513                *      CASSETTECONTROL (1).PLAYBACK = 0;
0514                *      CASSETTECONTROL (2).PLAYBACK = 0;
0515                *      CASSETTECONTROL (1).RECORD = 0;
0516                *      CASSETTECONTROL (2).RECORD = 0;
0517                *      RETURN;
0518                *      END RSTMOD;
```

```
0521                *                          1-PROCEDURE DSR733;
0522                * TITLE:     DSR733
0523                *            DEVICE SERVICE ROUTINES - 733 ASR
0524                * REVISION:
0525                *            ORIGINAL
0526                * COMPUTER:  990
0527                * ABSTRACT:
0528                *            A GROUPING OF SUBROUTINES WHICH CONSTITUTE
0529                *            THE SILENT 733 ASR (1200 BAUD INTERFACE)
0530                *            DEVICE SERVICE ROUTINES.
0531                *
0532                *            TABLE OF CONTENTS:
0533                *                I. LEVEL ONE SUBROUTINES
0534                *                   A. CASSETTE COMMAND PROCESSORS
0535                *                      1. OPEN (CSOPEN)
0536                *                      2. FORWARD SPACE  (CSFWD)
0537                *                      3. BACKSPACE  (CSBACK)
0538                *                      4. READ ASCII  (CSRASC)
0539                *                      5. WRITE ASCII  (CSWASC)
0540                *                      6. WRITE END OF FILE  (CSWEOF)
0541                *                      7. REWIND  (CSRWND)
0542                *                      8. UNLOAD  (CSUNLD)
0543                *                   B. LOG (KEYBOARD/PRINTER) COMMANDS
0544                *                      1. READ ASCII  (LGRASC)
0545                *                      2. WRITE ASCII  (LGWASC)
0546                *               II. LEVEL TWO SUBROUTINES
0547                *                   A. CHANGE RECORD/PLAYBACK MODE (CHGMOD)
0548                *                   B. GET STATUS CHARACTER (STTCHR)
0549                *              III. LEVEL THREE SUBROUTINES
0550                *                   A. SEND COMPLEX COMMAND (CMND)
0551                *               IV. LEVEL FOUR SUBROUTINES
0552                *                   A. SEND CHARACTER TO 733ASR (SNDCHR)
0553                *                V. LEVEL FIVE SUBROUTINES
0554                *                   A. SEND DELAY CHARACTERS (DLYCHR)
0555                *                   B. RECEIVE CHARACTER FROM 733ASR (RCVCHR
0556                *                   C. FETCH CHARACTER FROM BUFFER (GETCHR)
0557                *                   D. PUT CHARACTER INTO BUFFER (PUTCHR)
0558                *                   E. RESET CASSETTE RECORD/PLAYBACK
0559                *                      MODE (RSTMOD)
0560                * CALLING SEQUENCE:
0561                *            THE STANDARD CALLING SEQUENCE FOR ALL LEVEL
0562                *            ONE SUBROUTINES IS MADE BY THE IO SERVICE
0563                *            DISPATCHER.  UPON ENTRY THE DATA STRUCTURE
0564                *            IS AS FOLLOWS:
0565                *                CURRENT WORKSPACE:
0566                *                    R13 - ADDR OF DISPATCHER WORKSPACE
0567                *                    R12 - CRU BASE ADDRESS
0568                *                    R11 - RETURN
0569                *                    R10 - ADDRESS OF PRB
0570                *                DISPATCHER WORKSPACE:  (READ ONLY)
0571                *                    R10 - ADDR OF CURRENT WORKSPACE
0572                *                    R9  - DEVICE TYPE
0573                *                    R8  - UNIT NUMBER
0574                *                    R7  - IO OPERATION CODE
```

```
0575                *                   R6  - DEVICE CONTROL TABLE (DCT) ADDR
0576                *                   R3  - PRB ADDRESS
0577                *
0578                *             RULES FOR SUBROUTINE REGISTER USAGE ARE:
0579                *                 1. R0, R1, R14 AND R15 ARE VOLATILE AND
0580                *                    MAY BE USED BY ANY SUBROUTINE
0581                *                 2. LEVEL 1 SAVES RETURN ADDR IN R2
0582                *                    LEVEL 2 SAVES RETURN ADDR IN R3
0583                *                    LEVEL 3 SAVES RETURN ADDR IN R4
0584                *                    LEVEL 4 SAVES RETURN ADDR IN R5
0585                *                    LEVEL 5 SAVES RETURN ADDR IN R11
0586                *                    THUS LEVEL 1 MAY USE R3-R5 AS VOLATILE
0587                *                    STORAGE (LEVEL 2 R4-R5 AND LEVEL 3 R5).
0588                *                 3. SUBROUTINES MAY MODIFY ONLY THOSE
0589                *                    REGISTERS DESIGNATED AS OUTPUT PARAMETER
0590                *                    OR VOLATILE FOR THE APPROPRIATE LEVEL.
0591                *                 4. SUBROUTINE PARAMETERS ARE PASSED
0592                *                    ACCORDING TO PX990 STANDARD (R10 TOWARD
0593                *                    R0) EXCEPT FOR ASSEMBLY-TIME CONSTANT
0594                *                    PARAMETERS, WHICH ARE CODED AFTER THE
0595                *                    BL IN THE CALLING ROUTINE.
```

```
0597                    *                       2-DECLARE (1 PHYSICALRECORDBLOCK,
0598                    *                       3-          3 PRB CONTROL (PRBPTR),
0599                    *                       4-            5 IOOP BIT (8),
0600                    *                       4-            5 LUNO BIT (8),
0601                    *                       4-            5 SYSTEMFLAGS,
0602                    *                       5-              7 FILLER1 BIT (1),
0603                    *                       5-              7 UNRECOVERABLEERROR BIT
0604                    *                       5-              7 EOFFLAG BIT (1),
0605                    *                       5-              7 FILLER2 BIT (5),
0606                    *                       4-            5 USERFLAGS,
0607                    *                       5-              7 FILLER3 BIT (3),
0608                    *                       5-              7 CHARACTERIO BIT (1),
0609                    *                       5-              7 NOWAITFLAG BIT(1),
0610                    *                       5-              7 FILLER 4 BIT(3),
0611                    *                       4-            5 BUFFERADDRESS POINTER,
0612                    *                       4-            5 BUFFERLENGTH FIXED (16
0613                    *                       4-            5 CHARCOUNT FIXED (16),
0614                    *                       4-            5 PRBRESERVED BIT (32));
0615                    *                       3-          3 BSFLAG FIXED (16) ;
0616                    *
0617                    *                               PRB EQUATES
0618                    *                               DISPLACEMENTS:
0619          0000  IOOP    EQU  0                          IO OP CODE
0620          0001  LUNO    EQU  1                          LOGICAL UNIT NUMBER
0621          0002  SYSFLG  EQU  2                          SYSTEM FLAGS
0622          0003  USEFLG  EQU  3                          USER FLAGS
0623          0004  BFADDR  EQU  4                          BUFFER ADDRESS
0624          0006  BFLNTH  EQU  6                          BUFFER LENGTH
0625          0008  CCOUNT  EQU  8                          CHARACTER COUNT
0626                *                               FLAG SHIFT COUNTS
0627          000F  UE      EQU  16-1                       UNRECOVERABLE ERROR
0628          000E  EOFLG   EQU  16-2                       END.OF.FILE FLAG
0629          000D  CHRMOD  EQU  16-3                       CHARACTER MODE
0630          000C  NWFLG   EQU  16-4                       NO WAIT FLAG
0631                        REF  BSFLG                BACKSPACE FLAG
0632                *                               FLAG SET MASKS
0633                        REF  UEMASK                     UNRECOVERABLE ERROR
0634                        REF  EFMASK                     END.OF.FILE
0635                        REF  CMDMSK                     CHARACTER MODE
0636                        REF  UEFMSK                     UN.ERROR AND EOF FLAG
```

```
0638                    *                       2-DECLARE PRBPTR POINTER;
0639                    *                       2-DECLARE DCTPTR POINTER;
0640                    *                       2-DECLARE (1 DEVICECONTROLTABLE73
0641                    *                       3-          3 DCT733 CONTROL (DCTPTR
0642                    *                       4-          5 CRUBASE FIXED (16),
0643                    *                       4-          5 CASSETTECONTROL (2),
0644                    *                       5-          7 PLAYBACK BIT (1),
0645                    *                       5-          7 RECORD BIT (1),
0646                    *                       5-          7 FILLER10 BIT (6));
0647                    *
0648                    *                               DEVICE CONTROL TABLES 733ASR
0649                    *
0650          0000      DCTCRU EQU  0                   CRU BASE
0651          0002      CSTCNT EQU  2                   CASSETTE CONTROL ARRAY
0652                    *
0653                           EVEN
0654                    *
0655                           DEF  DC7331             733ASR #1
0656    0000  0000      DC7331 DATA >0                   (CRU BASE ADDR)
0657    0002    00             BYTE 0                   (CASSETTE CONTROL LEFT)
0658    0003    00             BYTE 0                   (CASSETTE CONTROL RIGHT)
0659                           EVEN
```

```
0661                    *                       2-DECLARE BUFFER (1) CHARACTER
0662                    *                       2-    CONTROL (BUFFERADDRESS);
0663                    *                       2-DECLARE PARMLIST LITERALLY
0664                    *                       2-    'PRBPTR, DEVTYPE,UNITNO, DC
0665                    *                       2-DECLARE (1 ASR733INPUT,
0666                    *                       3-          3 ASR733IN CRUCONTROL (C
0667                    *                       4-          5 INCHAR BIT (8),
0668                    *                       4-          5 XMTING  BIT (1),
0669                    *                       4-          5 ASRID   BIT (1),
0670                    *                       4-          5 WRQ     BIT (1),
0671                    *                       4-          5 RRQ     BIT (1),
0672                    *                       4-          5 DCD     BIT (1),
0673                    *                       4-          5 DSR     BIT (1),
0674                    *                       4-          5 INT     BIT (1));
0675                    *
0676                    *                       2-DECLARE (1 ASR733OUTPUT,
0677                    *                       3-          3 ASR733OUT CRUCONTROL (
0678                    *                       4-          5 OUTCHAR BIT (8),
0679                    *                       4-          5 FILLER7331 BIT (1),
0680                    *                       4-          5 DTR     BIT (1),
0681                    *                       4-          5 RTS     BIT (1),
0682                    *                       4-          5 CLRWRQ  BIT (1),
0683                    *                       4-          5 CLRRRQ  BIT (1),
0684                    *                       4-          5 CLRNSF  BIT (1));
0685                    *
0686                    *                             CRU INTERFACE DEFINITIONS
0687                    *
0688          0000  INCHAR EQU  0                     INPUT CHARACTER
0689          0008  XMTING EQU  8                     TRANSMIT IN PROGRESS
0690          0009  TIMERR EQU  9                     TIMING ERROR
0691          000A  ASRID  EQU  10                    ASR IDENTIFIER
0692          000B  WRQ    EQU  11                    WRITE REQUEST
0693          000C  RRQ    EQU  12                    READ REQUEST
0694          000D  DCD    EQU  13                    DATA CARRIER DETECT
0695          000E  DSR    EQU  14                    DATA SET READY
0696          000F  INT    EQU  15                    INTERRUPT
0697                *
0698          0000  OUTCHR EQU  0                     OUTPUT CHARACTER
0699          0009  DTR    EQU  9                     DATA TERMINAL READY
0700          000A  RTS    EQU  10                    REQUEST TO SEND
0701          000B  CLRWRQ EQU  11                    CLEAR WRITE REQUEST
0702          000C  CLRRRQ EQU  12                    CLEAR READ REQUEST
0703          000D  CLRNSF EQU  13                    CLEAR NEW STATUS
0704                *
0705                *
0706                *                       2-DECLARE MODRCD LITERALLY '"4080"';
0707          4080  MODRCD EQU  >4080                 SET RECORD ON CASSETTE
0708                *                       2-DECLARE MODPBK LITERALLY '"8040"';
0709          8040  MODPBK EQU  >8040                 SET PLAYBACK ON CASETTE
0710                *                       2-DECLARE MODBTS BIT (16) INITIAL ("
0711  0004    C0C0  MODBTS DATA MODRCD+MODPBK
0712                *                       2-DECLARE (1 STATUS,
0713                *                       3-          3 FILLERSTT1 BIT (2),
0714                *                       3-          3 PNTRDY BIT (1),
```

```
0715                *                          3-          3 RCDRDY BIT (1),
0716                *                          3-          3 BOE (1) BIT (1),
0717                *                          3-          3 PBKERR BIT (1),
0718                *                          3-          3 PBKRDY BIT (1));
0719         000D   RCDRDY EQU  16-3
0720         000B   BOE    EQU  16-5
0721         0009   PBKRDY EQU  16-7
0722         0100   LBYTE  EQU  >100                MOVE LITERAL TO LEFT BYTE
0723         0003   DELAY3 EQU  3                   DELAY 3 CHARACTERS
0724         001B   DLY27  EQU  27                  DELAY 27 CHARACTERS
0725                *
0726         0050   MAXCHR EQU  80                  MAX CHARS PER RECORD
0727                *
0728                *       WORKSPACE REGISTER DEFINITIONS
0729                *
0730         0000   R0     EQU  0
0731         0001   R1     EQU  1
0732         0002   R2     EQU  2
0733         0003   R3     EQU  3
0734         0004   R4     EQU  4
0735         0005   R5     EQU  5
0736         0006   R6     EQU  6
0737         0007   R7     EQU  7
0738         0008   R8     EQU  8
0739         0009   R9     EQU  9
0740         000A   R10    EQU  10
0741         000B   R11    EQU  11
0742         000C   R12    EQU  12
0743         000D   R13    EQU  13
0744         000E   R14    EQU  14
0745         000F   R15    EQU  15
0746                *
```

```
0749                    *                                  2-DECLARE DLE LITERALLY '"10"',
0750          0010      DLE     EQU   >10
0751                    *                                  4-DC2 LITERALLY '"12"',
0752          0012      DC2     EQU   >12
0753                    *                                  4-DC3 LITERALLY '"13"',
0754          0013      DC3     EQU   >13
0755                    *                                  4-DC4 LITERALLY '"14"',
0756          0014      DC4     EQU   >14
0757                    *                                  4-CR  LITERALLY '"0D"',
0758          000D      CR      EQU   >0D
0759                    *                                  4-LF  LITERALLY '"0A"',
0760          000A      LF      EQU   >0A
0761                    *                                  4-DEL LITERALLY '"7F"',
0762          007F      DEL     EQU   >7F
0763                    *                                  4-SP  LITERALLY '"20"',
0764          0020      SP      EQU   >20
0765                    *                                  4-HT  LITERALLY '"09"',
0766          0009      HT      EQU   >09
0767                    *                                  4-ESC LITERALLY '"1B"',
0768          001B      ESC     EQU   >1B
0769                    *                                  4-BEL LITERALLY '"07"',
0770          0007      BEL     EQU   >07
0771                    *                                  4-FF  LITERALLY '"0C"',
0772          000C      FF      EQU   >0C
0773                    *                                  4-ETB LITERALLY '"17"',
0774          0017      ETB     EQU   >17
0775                    *                                  4-VT  LITERALLY '"0B"',
0776          000B      VT      EQU   >0B
0777                    *                                  4-BS  LITERALLY '"08"';
0778          0008      BS      EQU   >08
0779                    *
0780                    *                                  2-DECLARE MDID LITERALLY '"80"';
0781          0080      MDID    EQU   >80
0782                    *                                  2-DECLARE REWIND LITERALLY '"31"+MDI
0783          00B1      REWIND  EQU   '1'+MDID
0784                    *                                  4-LOAD LITERALLY '"33"+MDID',
0785          00B3      LOAD    EQU   '3'+MDID
0786                    *                                  4-RECORD LITERALLY '"35"+MDID',
0787          00B5      RECORD  EQU   '5'+MDID
0788                    *                                  4-PLYBK LITERALLY '"36"+MDID',
0789          00B6      PLYBK   EQU   '6'+MDID
0790                    *                                  4-FWDBLK LITERALLY '"37"',
0791          0037      FWDBLK  EQU   '7'
0792                    *                                  4-BCKBLK LITERALLY '"38"',
0793          0038      BCKBLK  EQU   '8'
0794                    *                                  4-STATUS LITERALLY '"3C"',
0795          003C      STATUS  EQU   '<'
0796                    *                                  4-ENDGRP LITERALLY '"FF"',
0797          00FF      ENDGRP  EQU   >FF
0798                    *                                  4-DLYGRP LITERALLY '"FE"';
0799          00FE      DLYGRP  EQU   >FE
```

```
0802                    *                       2-DECLARE RECORDMODE (7) BIT (8) IN
0803                    *                       3-(DLYGRP, 60, DLE, DLYGRP, 3, RECOR
0804          0006' RMODE   EQU  $                      RECORD MODE
0805  0006      FE          BYTE DLYGRP,60
      0007      3C
0806  0008      10          BYTE DLE
0807  0009      FE          BYTE DLYGRP,3
      000A      03
0808  000B      B5          BYTE RECORD
0809  000C      FF          BYTE ENDGRP
0810                    *                       2-DECLARE PLYBACKMODE (7) BIT (8) I
0811                    *                       3-(DLYGRP, 60, DLE, DLYGRP, 3, PLYBK
0812          000D' PMODE   EQU  $                      PLAYBACK MODE
0813  000D      FE          BYTE DLYGRP,60
      000E      3C
0814  000F      10          BYTE DLE
0815  0010      FE          BYTE DLYGRP,3
      0011      03
0816  0012      B6          BYTE PLYBK
0817  0013      FF          BYTE ENDGRP
0818                    *                       2-DECLARE READTAPE (5) BIT (8) INIT
0819                    *                       3-(DLE, DLYGRP, 3, FWDBLK, ENDGRP);
0820          0014' RTAPE   EQU  $                      BLOCK FORWARD
0821  0014      10          BYTE DLE
0822  0015      FE          BYTE DLYGRP,3
      0016      03
0823  0017      37          BYTE FWDBLK
0824  0018      FF          BYTE ENDGRP
0825                    *                       2-DECLARE WTTAPESTART (4) BIT (8)
0826                    *                       3-(DC2, DLYGRP, 3, ENDGRP);
0827          0019' WTAPES  EQU  $                      BEGIN WRITE TO CASSETTE
0828  0019      12          BYTE DC2
0829  001A      FE          BYTE DLYGRP,3
      001B      03
0830  001C      FF          BYTE ENDGRP
0831                    *                       2-DECLARE WTTAPEEND (9) BIT (8) IN
0832                    *                       3-(CR, LF, DC4, DLYGRP, 3, DEL, DLYG
0833                    *                       3-ENDGRP);
0834          001D' WTAPEE  EQU  $                      END WRITE TO CASSETTE
0835  001D      0D          BYTE CR
0836  001E      0A          BYTE LF
0837  001F      14          BYTE DC4
0838  0020      FE          BYTE DLYGRP,3
      0021      03
0839  0022      7F          BYTE DEL
0840  0023      FE          BYTE DLYGRP,3
      0024      03
0841  0025      FF          BYTE ENDGRP
0842                    *                       2-DECLARE EOFSEQ (4) BIT (8) INITIAL
0843                    *                       3-(DC3, ENDGRP);
0844          0026' EOFSEQ  EQU  $
0845  0026      13          BYTE DC3
0846  0027      FF          BYTE ENDGRP
0847                        EVEN
```

```
0848                    *                       2-DECLARE REWINDSEQ (9) BIT (8) INI
0849                    *                       3-(DLYGRP, 60, DLE, DLYGRP, 3, REWIN
0850                    *                       3-DLYGRP, 180,ENDGRP);
0851  0000                      DORG 0
0852          0000      RWDSEQ  EQU  $                  REWIND CASSETTE
0853  0000      FE              BYTE DLYGRP,60
      0001      3C
0854  0002      10              BYTE DLE
0855  0003      FE              BYTE DLYGRP,3
      0004      03
0856  0005      B1              BYTE REWIND
0857  0006      FE              BYTE DLYGRP,180
      0007      B4
0858  0008      FF              BYTE ENDGRP
0859                    *                       2-DECLARE LOADSEQ (5) BIT (8) INITI
0860                    *                       3-(DLE, DLYGRP, 3, LOAD, ENDGRP);
0861          0009      LOADSQ  EQU  $                  LOAD TAPE
0862  0009      10              BYTE DLE
0863  000A      FE              BYTE DLYGRP,3
      000B      03
0864  000C      B3              BYTE LOAD
0865  000D      FF              BYTE ENDGRP
0866                    *                       2-DECLARE BACKSPC (7) BIT (8) INITIA
0867                    *                       3-(DLE, DLYGRP, 3, BCKBLK, DLYGRP, 4
0868          000E      BKSPC   EQU  $                  BLOCK REVERSE
0869  000E      10              BYTE DLE
0870  000F      FE              BYTE DLYGRP,3
      0010      03
0871  0011      38              BYTE BCKBLK
0872  0012      FE              BYTE DLYGRP,48
      0013      30
0873  0014      FF              BYTE ENDGRP
0874  0028                      RORG
0875                    *                       2-DECLARE ECHOFF  ( 4)  BIT (8) INI
0876                    *                       3-(LF, DLYGRP, 3, ENDGRP);
0877          0028'     ECOFF   EQU  $                  ECHO FF
0878  0028      0A              BYTE LF
0879  0029      FE              BYTE DLYGRP,3
      002A      03
0880  002B      FF              BYTE ENDGRP
0881                    *                       2-DECLARE STATUSSEQ (5) BIT (8) INI
0882                    *                       3-(DLE, DLYGRP, 3, STATUS, ENDGRP);
0883          002C'     STTSEQ  EQU  $                  ASK STATUS
0884  002C      10              BYTE DLE
0885  002D      FE              BYTE DLYGRP,3
      002E      03
0886  002F      3C              BYTE STATUS
0887  0030      FF              BYTE ENDGRP
0888                    *                       2-DECLARE ECHOBS (7) BIT (8) INITIA
0889                    *                       3-(BS, DLYGRP, 3, ENDGRP);
0890          0031'     ECOBS   EQU  $                  ECHO BS
0891  0031      08              BYTE BS
0892  0032      FE              BYTE DLYGRP,3
      0033      03
0893  0034      FF              BYTE ENDGRP
```

```
0894                    *                       2-DECLARE ECHODEL (7) BIT (8) INITI
0895                    *                       3-(LF, DLYGRP, 3, CR, DLYGRP, 27, EN
0896          0035' ECODEL EQU  $                       ECHO DEL
0897  0035     0A           BYTE LF
0898  0036     FE           BYTE DLYGRP,3
      0037     03
0899  0038     0D           BYTE CR
0900  0039     FE           BYTE DLYGRP,27
      003A     1B
0901  003B     FF           BYTE ENDGRP
0902                    *                       2-DECLARE BYTMOD (6) BIT (8) INITIAL
0903                    *                       3-(1, 0, 1, 0, 1, -1);
0904          003C' BYTMOD EQU  $
0905  003C     01           BYTE 1,0,1,0,1,-1
      003D     00
      003E     01
      003F     00
      0040     01
      0041     FF
0906                         EVEN
```

```
0909                *  TITLE:     CSOPEN
0910                *             CASSETTE - OPEN
0911                *  REVISION:
0912                *             ORIGINAL
0913                *  ABSTRACT:
0914                *             CASSETTE PLAYBACK/RECORD MODE BITS RESET.
0915                *  CALLING SEQUENCE:
0916                *             STANDARD LEVEL ONE LINKAGE.
0917                *                          1-PROCEDURE CSOPEN (PARMLIST);
0918                       DEF   CSOPEN
0919          0042' CSOPEN EQU   $
0920   0042   C08B         MOV   R11,R2              SAVE RETURN LEVEL ONE
0921                *                          2-CALL RSTMOD;
0922   0044   06A0         BL    @RSTMOD
       0046   0448'
0923                *                          2-RETURN;
0924   0048   0452         B     *R2
0925                *                          2-END CSOPEN;
```

```
0928                    * TITLE:     CSFWD
0929                    *            CASSETTE - FORWARD SPACE RECORD
0930                    * REVISION:
0931                    *            ORIGINAL
0932                    * ABSTRACT:
0933                    *            THE INDICATED CASSETTE IS FORWARD SPACED BY
0934                    *            ISSUING READS UNTIL THE INDICATED NUMBER OF
0935                    *            BLOCKS ARE SKIPPED OR UNTIL AN END OF FILE
0936                    *            IS REACHED.
0937                    * CALLING SEQUENCE:
0938                    *            STANDARD LEVEL ONE ROUTINE.
0939                    *
0940                    *                        1-PROCEDURE CSFWD (PARMLIST);
0941                    *                        2-DECLARE (1 DUMMYSVCBLOCK,
0942                    *                        3-          3 DMYSVCCODE BIT (8),
0943                    *                        3-          3 FILLERDMY1 BIT (8),
0944                    *                        3-          3 DMYIOOP BIT (8),
0945                    *                        3-          3 DMYLUNO BIT (8),
0946                    *                        3-          3 DMYSYSFLAGS,
0947                    *                        4-          5 FILLERDMY2 BIT (2),
0948                    *                        4-          5 DMYEOFFLAG BIT (1),
0949                    *                        4-          5 FILLERDMY3 BIT (5),
0950                    *                        3-          3 DMYUSRFLAGS BIT (8),
0951                    *                        3-          3 DMYBFRADDR POINTER,
0952                    *                        3-          3 DMYBFRLNTH FIXED (16)
0953                    *                        3-          3 DMYCHRCOUNT FIXED (16
0954                    *                        2-DECLARE DMYBUFFER BIT (8);
0955                    *                        2-DECLARE READASCII LITERALLY '9';
0956                    *                        2-DECLARE IOREQUEST LITERALLY '0';
0957          0009      RDASCI EQU  9
0958                    *                        2-DECLARE I FIXED;
0959                    *                        2-DECLARE WKSP POINTER;
0960  0000                     DORG 0
0961                     DEF  CSFWD
0962          0000      CSFWD  EQU  $                    CASSETTE - FORWARD SPACE
0963  0000    C08B             MOV  R11,R2
0964                    *                        2-DMYSVCCODE = IOREQUEST;
0965  0002    04C3             CLR  R3                      BUILD DUMMY PRB/SVC
0966                    *                        2-DMYIOOP = READASCII;
0967  0004    0204             LI   R4,RDASCI
      0006    0009
0968                    *                        2-DMYLUNO = LUNO;
0969  0008    D12A             MOVB @LUNO(R10),R4
      000A    0001
0970  000C    0B84             SRC  R4,8
0971                    *                        2-DMYSYSFLAGS = 0;
0972                    *                        2-DMYUSRFLAGS = 0;
0973  000E    04C5             CLR  R5
0974                    *                        2-DMYBFRADDR = ADDR (DMYBUFFER);
0975  0010    02A6             STWP R6                      USE R15 AS DUMMY BUFFER
0976  0012    0226             AI   R6,R15*2
      0014    001E
0977                    *                        2-DMYBFRLNTH = 1;
0978  0016    0207             LI   R7,1
```

```
      0018  0001
0979                *                           2-I = 0;
0980  001A  04C1          CLR   R1
0981                *                           2-CALL GETBUF (WKSP);
0982  001C  C24A          MOV   R10,R9                PRESERVE REAL PRB
0983                      REF   GETBUF
0984  001E  0420          BLWP  @GETBUF
      0020  0000
0985                *                           2-DO WHILE I < CHARCOUNT & DMYEOF
0986        0022  CF010   EQU   $
0987  0022  8A41          C     R1,@CCOUNT(R9)      IF I >= CHARCOUNT
      0024  0008
0988  0026  140F          JHE   CF020
0989                *
0990  0028  0BE5          SRC   R5,EOFLG            IF EOF DETECTED
0991  002A  180D          JOC   CF020
0992                *                           3-CALL IO (WKSP);
0993                      REF   LWP                   MOVE TO NEW WKSP FOR IO
0994  002C  0420          BLWP  @LWP
      002E  0000
0995  0030  C28D          MOV   R13,R10               SET DMY SVC ADDRESS
0996  0032  022A          AI    R10,R3*2
      0034  0006
0997                      REF   IO                    CALL IO TO READ
0998  0036  06A0          BL    @IO
      0038  0000
0999                *
1000                      REF   RWP                   BACK TO ORIGINAL WKSP
1001  003A  0420          BLWP  @RWP
      003C  0000
1002                *                           3-SYSTEMFLAGS = DMYSYSFLAGS;
1003  003E  DA45          MOVB  R5,@SYSFLG(R9)
      0040  0002
1004                *                           3-I = I + 1;
1005  0042  0581          INC   R1
1006                *                           3-END;
1007  0044  10EE          JMP   CF010
1008                *
1009        0046  CF020   EQU   $                     END.OF.LOOP
1010                *                           2-CALL RETBUF (WKSP);
1011                      REF   RETBUF
1012  0046  0420          BLWP  @RETBUF
      0048  0000
1013                *                           2-RETURN;
1014  004A  0452          B     *R2
1015                *                           2-END CSFWD;
1016  004A                RORG
```

```
1019                    * TITLE:    CSBACK
1020                    *           CASSETTE - BACKSPACE RECORD
1021                    * REVISION:
1022                    *           ORIGINAL
1023                    * ABSTRACT:
1024                    *           THE INDICATED CASSETTE IS BLOCK REVERSED THE
1025                    *           NUMBER OF RECORDS INDICATED IN CHARCOUNT.
1026                    * CALLING SEQUENCE:
1027                    *           STANDARD LEVEL ONE LINKAGE.
1028                    *                       1-PROCEDURE CSBACK (PARMLIST);
1029                    *                       2-DECLARE I FIXED;
1030  0000                      DORG 0
1031                      DEF  CSBACK
1032        0000  CSBACK EQU   $
1033  0000  C08B        MOV   R11,R2            RETURN SAVE LEVEL ONE
1034                    *                       2-CALL CHGMOD (PRBPTR, DCTPTR, MOD
1035  0002  06A0        BL    @CHGMOD
      0004  0342'
1036  0006  8040        DATA  MODPBK
1037                    *                       2-DO I FROM 1 TO CHARCOUNT;
1038  0008  04C6        CLR   R6                INITIALIZE I
1039  000A  C1EA        MOV   @CCOUNT(R10),R7   LOOP MAX
      000C  0008
1040        000E  CSB010 EQU   $
1041  000E  0586        INC   R6
1042  0010  81C6        C     R6,R7             IF END.OF.LOOP
1043  0012  1504        JGT   CSB020
1044                    *                       3-CALL CMND (ADDR (BKSPC));
1045  0014  06A0        BL    @CMND
      0016  0396'
1046  0018  000E        DATA  BKSPC
1047                    *                       3-END;
1048  001A  10F9        JMP   CSB010
1049                    *
1050        001C  CSB020 EQU   $
1051                    *                       2-RETURN;
1052  001C  0452        B     *R2
1053                    *                       2-END CSBACK;
1054  004A              RORG
```

```
1057                * TITLE:    CSRASC
1058                *           CASSETTE - READ ASCII
1059                * REVISION:
1060                *           ORIGINAL
1061                * ABSTRACT:
1062                *           CHARACTERS ARE TRANSFERRED FROM A CASSETTE
1063                *           RECORD TO THE BUFFER UNTIL AN END OF RECORD
1064                *           SITUATION IS ENCOUNTERED, IE:
1065                *               - BUFFER LENGTH IS EXHAUSTED. ADDITIONAL
1066                *                 CHARACTERS ON THE PHYSICAL RECORD ARE
1067                *                 IGNORED.
1068                *               - A CR IS DETECTED
1069                *               - A DC3 AS THE FIRST VALID CHARACTER IS
1070                *                 DETECTED.
1071                * CALLING SEQUENCE:
1072                *           STANDARD LEVEL ONE ROUTINE
1073                *                       1-PROCEDURE CSRASC (PARMLIST);
1074                *                       2-DECLARE CHAR BIT (8);
1075                *                       2-DECLARE I FIXED;  /* CHARACTER CUR
1076                          DEF  CSRASC
1077          004A' CSRASC EQU  $                   CASSETTE - READ ASCII
1078  004A    C08B         MOV  R11,R2              SAVE RETURN LEVEL ONE
1079                *                       2-CALL CHGMOD (PRBPTR, DCTPTR, MOD
1080  004C    06A0         BL   @CHGMOD             ASSURE PLAYBACK MODE
      004E    0342'
1081  0050    8040         DATA MODPBK
1082                *                       2-CALL STTCHR (CHAR);  /*  GET STATU
1083  0052    06A0         BL   @STTCHR
      0054    0386'
1084                *                       2-IF PBKRDY = 0 THEN DO;
1085  0056    0B99         SRC  R9,PBKRDY           IF PBKRDY READY
1086  0058    1804         JOC  CSR005
1087                *                       3-UNRECOVERABLEERROR = 1;
1088  005A    FAA0         SOCB @UEMASK,@SYSFLG(R10)
      005C    0000
      005E    0002
1089                *                       3-RETURN;
1090  0060    0452         B    *R2
1091                *                       3-END;
1092          0062' CSR005 EQU  $
1093                *                       2-I = 0;
1094  0062    04C6         CLR  R6                  INITIALIZE CURSOR
1095                *                       2-CALL CMND (ADDR (RTAPE));
1096  0064    06A0         BL   @CMND               BLOCK FORWARD
      0066    0396'
1097  0068    0014'        DATA RTAPE
1098                *                       2-CALL RCVCHR (CHAR, WAIT);
1099  006A    04C9         CLR  R9                  WAIT FOR CHARACTER
1100  006C    06A0         BL   @RCVCHR
      006E    0412'
1101                *                       2-IF CHAR = LF THEN CALL RCVCHR (CHA
1102  0070    0289         CI   R9,LF*LBYTE         IF CHAR .NE. LF
      0072    0A00
1103  0074    1603         JNE  CSR010
```

```
1104  0076  04C9          CLR  R9                    IGNORE LF, GET NEXT CHAR
1105  0078  06A0          BL   @RCVCHR
      007A  0412'
1106        007C' CSR010 EQU  $
1107              *                          2-IF CHAR = DEL THEN CALL RCVCHR (CH
1108  007C  0289          CI   R9,DEL*LBYTE       IF CHAR .NE. DEL
      007E  7F00
1109  0080  1603          JNE  CSR020
1110  0082  04C9          CLR  R9                    IGNORE DEL, GET NEXT CHAR
1111  0084  06A0          BL   @RCVCHR
      0086  0412'
1112        0088' CSR020 EQU  $
1113              *                          2-DO WHILE CHAR .NE. CR & I < BUFFER
1114              *                          2-         & EOFFLAG = 0;
1115        0088' CSR100 EQU  $
1116  0088  0289          CI   R9,CR*LBYTE        IF CHAR = CR
      008A  0D00
1117  008C  1336          JEQ  CSR500
1118  008E  8A86          C    R6,@BFLNTH(R10)    IF I >= BUFFER LENGTH
      0090  0006
1119  0092  1433          JHE  CSR500
1120  0094  D06A          MOVB @SYSFLG(R10),R1    IF EOF FLAG .NE. 0
      0096  0002
1121  0098  0BE1          SRC  R1,EOFLG
1122  009A  182F          JOC  CSR500
1123              *                          3-IF CHAR = ETB THEN CHAR = CR;
1124  009C  0289          CI   R9,ETB*LBYTE       IF CHAR .NE. ETB
      009E  1700
1125  00A0  1602          JNE  CSR110
1126  00A2  0209          LI   R9,CR*LBYTE        TRANSFORM ETB TO CR
      00A4  0D00
1127        00A6' CSR110 EQU  $
1128              *                          3-IF  CHAR >= SP .OR.
1129              *                          3-   (CHAR >= BEL & CHAR <= HT) .OR.
1130              *                          3-    CHAR = FF  .OR.  CHAR = CR
1131  00A6  0289          CI   R9,SP*LBYTE        IF CHAR >= SP
      00A8  2000
1132  00AA  140D          JHE  CSR130
1133  00AC  0289          CI   R9,BEL*LBYTE       IF BEL <= CHAR <= HT
      00AE  0700
1134  00B0  1109          JLT  CSR120
1135  00B2  0289          CI   R9,HT*LBYTE
      00B4  0900
1136  00B6  1207          JLE  CSR130
1137  00B8  0289          CI   R9,FF*LBYTE        IF CHAR = FF
      00BA  0C00
1138  00BC  1304          JEQ  CSR130
1139  00BE  0289          CI   R9,CR*LBYTE        IF CHAR = CR
      00C0  0D00
1140  00C2  1301          JEQ  CSR130
1141        00C4' CSR120 EQU  $
1142  00C4  1004          JMP  CSR140
1143              *                          4-THEN DO;
1144        00C6' CSR130 EQU  $
1145              *                          5-CALL PUTCHR (CHAR, I);
```

```
1146  00C6  C206          MOV   R6,R8
1147  00C8  06A0          BL    @PUTCHR
      00CA  043A'
1148                *                             5-I = I + 1;
1149  00CC  0586          INC   R6
1150                *                             5-END;
1151        00CE' CSR140 EQU   $
1152                *                             3-IF CHAR = DC3
1153  00CE  0289          CI    R9,DC3*LBYTE            IF CHAR .NE. DC3
      00D0  1300
1154  00D2  160F          JNE   CSR170
1155                *                             4-THEN DO;
1156                *                             5-IF I = 0 THEN EOFFLAG = 1;
1157  00D4  C186          MOV   R6,R6                   IF I .NE. 0
1158  00D6  1603          JNE   CSR150
1159  00D8  FAA0          SOCB  @EFMASK,@SYSFLG(R10)  SET EOF FLAG
      00DA  0000
      00DC  0002
1160        00DE' CSR150 EQU   $
1161                *                             5-CALL RCVCHR (CHAR, WAIT);
1162  00DE  04C9          CLR   R9                      GET CHAR AFTER DC3
1163  00E0  06A0          BL    @RCVCHR
      00E2  0412'
1164                *                             5-IF CHAR .NE. CR
1165                *                             7-THEN CALL CMND (ADDR (RTAPE));
1166  00E4  0289          CI    R9,CR*LBYTE             IF CHAR = CR
      00E6  0D00
1167  00E8  1303          JEQ   CSR160
1168  00EA  06A0          BL    @CMND                   GET REST OF BLOCK
      00EC  0396'
1169  00EE  0014'         DATA  RTAPE
1170        00F0' CSR160 EQU   $
1171                *                             5-END;
1172  00F0  1003          JMP   CSR180
1173                *                             4-ELSE CALL RCVCHR (CHAR, WAIT);
1174        00F2' CSR170 EQU   $
1175  00F2  04C9          CLR   R9
1176  00F4  06A0          BL    @RCVCHR
      00F6  0412'
1177        00F8' CSR180 EQU   $
1178                *                             3-END;
1179  00F8  10C7          JMP   CSR100
1180                *
1181        00FA' CSR500 EQU   $
1182                *                             2-DO WHILE CHAR .NE. CR;
1183  00FA  0289          CI    R9,CR*LBYTE             IF CHAR = CR
      00FC  0D00
1184  00FE  1304          JEQ   CSR600
1185                *                             3-CALL RCVCHR (CHAR, WAIT);
1186  0100  04C9          CLR   R9
1187  0102  06A0          BL    @RCVCHR
      0104  0412'
1188                *                             3-END;
1189  0106  10F9          JMP   CSR500
1190                *
```

```
 1191          0108' CSR600 EQU  $
 1192                *                          2-CHARCOUNT = I;
 1193  0108    CA86          MOV  R6,@CCOUNT(R10)
       010A    0008
 1194                *                          2-RETURN;
 1195  010C    0452          B    *R2
 1196                *                          2-END CSRASC
```

```
1199                * TITLE:     CSWASC
1200                *            CASSETTE - WRITE ASCII
1201                * REVISION:
1202                *            ORIGINAL
1203                * ABSTRACT:
1204                *            A RECORD OF 80 OR LESS CHARACTERS IS
1205                *            TRANSFERRED TO THE INDICATED CASSETTE.
1206                * CALLING SEQUENCE:
1207                *            STANDARD LEVEL ONE LINKAGE
1208                *                        1-PROCEDURE CSWASC (PARMLIST);
1209                *                        2-DECLARE I FIXED;
1210                *                        2-DECLARE CHAR BIT (8);
1211                          DEF  CSWASC
1212          010E' CSWASC    EQU  $                  CASSETTE - WRITE ASCII
1213  010E    C08B            MOV  R11,R2             SAVE RETURN - LEVEL ONE
1214                *                        2-CALL CHGMOD (PRBPTR, DCTPTR, MOD
1215  0110    06A0            BL   @CHGMOD            ASSURE RECORD MODE
      0112    0342'
1216  0114    4080            DATA MODRCD
1217                *                        2-CALL STTCHR (CHAR) /*GET STATUS
1218  0116    06A0            BL   @STTCHR
      0118    0386'
1219  011A    0BD9            SRC  R9,RCDRDY
1220                *                        2-IF RCDRDY = 0 THEN DO;
1221  011C    1804            JOC  CW005
1222                *                        3-UNRECOVERABLE ERROR = 1;
1223  011E    FAA0            SOCB @UEMASK,@SYSFLG(R10)
      0120    005C'
      0122    0002
1224                *                        3-RETURN;
1225  0124    0452            B    *R2
1226                *                        3-END;
1227          0126' CW005     EQU  $
1228                *                        2-CALL CMND (ADDR (WTAPES));
1229  0126    06A0            BL   @CMND
      0128    0396'
1230  012A    0019'           DATA WTAPES
1231                *                        2-DO I FROM 0 TO MIN (CHARCOUNT-1,
1232  012C    04C6            CLR  R6                 INITIALIZE I
1233  012E    C1EA            MOV  @CCOUNT(R10),R7    COMPUTE LIMIT VALUE
      0130    0008
1234  0132    0287            CI   R7,MAXCHR
      0134    0050
1235  0136    1202            JLE  CW010
1236  0138    0207            LI   R7,MAXCHR
      013A    0050
1237          013C' CW010     EQU  $
1238  013C    81C6            C    R6,R7              IF END.OF.LOOP
1239  013E    141D            JHE  CW100
1240                *                        3-CALL GETCHR (CHAR, I)
1241  0140    C206            MOV  R6,R8
1242  0142    06A0            BL   @GETCHR
      0144    042C'
1243                *                        3-IF CHAR = CR THEN CHAR = ETB;
```

```
1244  0146  0289          CI    R9,CR*LBYTE
      0148  0D00
1245  014A  1602          JNE   CW020
1246  014C  0209          LI    R9,ETB*LBYTE
      014E  1700
1247        0150' CW020    EQU   $
1248              *                          3-IF CHAR >= SP .OR. (CHAR >= BEL &
1249              *                          4-.OR. CHAR = FF .OR. CHAR = ETB
1250  0150  0289          CI    R9,SP*LBYTE        IF CHAR NOT IN SET
      0152  2000
1251  0154  140D          JHE   LL20
1252  0156  0289          CI    R9,FF*LBYTE
      0158  0C00
1253  015A  130A          JEQ   LL20
1254  015C  0289          CI    R9,ETB*LBYTE
      015E  1700
1255  0160  1307          JEQ   LL20
1256  0162  0289          CI    R9,LF*LBYTE
      0164  0A00
1257  0166  1503          JGT   LL18
1258  0168  0289          CI    R9,BEL*LBYTE
      016A  0700
1259  016C  1401          JHE   LL20
1260        016E' LL18     EQU   $
1261  016E  1003          JMP   CW030
1262        0170' LL20     EQU   $
1263              *                          4-THEN CALL SNDCHR (CHAR, 0);
1264  0170  06A0          BL    @SNDCHR
      0172  03DA'
1265  0174  0000          DATA  0
1266              *                          3-END;
1267        0176' CW030    EQU   $
1268  0176  0586          INC   R6
1269  0178  10E1          JMP   CW010
1270        017A' CW100    EQU   $
1271              *                          2-CALL DELAY (CCOUNT)
1272  017A  0208          LI    R8,32-11
      017C  0015
1273  017E  622A          S     @CCOUNT(R10),R8
      0180  0008
1274  0182  06A0          BL    @DLYCHR
      0184  03FC'
1275              *                          2-CALL CMND (ADDR (WTAPEE));
1276  0186  06A0          BL    @CMND              WRITE END OF RECORD SEQUENCE
      0188  0396'
1277  018A  001D'         DATA  WTAPEE
1278              *                          2-RETURN;
1279  018C  0452          B     *R2
1280              *                          2-END CSWASC;
```

```
1283                    * TITLE:     CSWEOF
1284                    *            CASSETTE - WRITE END OF FILE
1285                    * REVISION:
1286                    *            ORIGINAL
1287                    * ABSTRACT:
1288                    *
1289                    *
1290                    * CALLING SEQUENCE:
1291                    *              STANDARD LEVEL ONE ROUTINE
1292                    *
1293                    *                                  1-PROCEDURE CSWEOF (PARMLIST);
1294                           DEF  CSWEOF
1295          018E'     CSWEOF EQU  $                              CASSETTE - WRITE EOF
1296  018E    C08B           MOV  R11,R2                         LEVEL ONE RETURN SAVE
1297                    *                                  2-CALL CHGMOD (MODRCD);
1298  0190    06A0           BL   @CHGMOD                        ASSURE RECORD MODE
        0192    0342'
1299  0194    4080           DATA MODRCD
1300                    *                                  2-CALL CMND (ADDR (WTAPES));
1301  0196    06A0           BL   @CMND
        0198    0396'
1302  019A    0019'          DATA WTAPES
1303                    *                                  2-CALL CMND (ADDR (EOFSEQ));
1304  019C    06A0           BL   @CMND
        019E    0396'
1305  01A0    0026'          DATA EOFSEQ
1306                    *                                  2-CALL CMND (ADDR (WTAPEE));
1307  01A2    06A0           BL   @CMND
        01A4    0396'
1308  01A6    001D'          DATA WTAPEE
1309                    *                                  2-RETURN;
1310  01A8    0452           B    *R2
1311                    *                                  2-END CSWEOF;
```

```
1314                * TITLE:     CSRWND
1315                *            CASSETTE - REWIND TO LOAD POINT
1316                * REVISION:
1317                *            ORIGINAL
1318                * ABSTRACT:
1319                *            THE INDICATED CASSETTE IS REWOUND AND THEN
1320                *            LOADED.
1321                * CALLING SEQUENCE:
1322                *            STANDARD LEVEL ONE LINKAGE.
1323                *
1324                *                         1-PROCEDURE CSRWND (PARMLIST);
1325                *                         2-DECLARE I BIT (1);
1326                          DEF  CSRWND
1327  0000                    DORG 0
1328          0000  CSRWND    EQU  $                  CASSETTE - REWIND
1329  0000  C08B              MOV  R11,R2
1330                *                         2-CALL RSTMOD;
1331  0002  06A0              BL   @RSTMOD            FORCE MODE CHANGE NEXT TIME
      0004  0448'
1332                *                         2-CALL CMND (ADDR (RWDSEQ));
1333  0006  06A0              BL   @CMND
      0008  0396'
1334  000A  0000              DATA RWDSEQ
1335                *                         2-I = 0;
1336                *                         2-DO WHILE I = 0;
1337          000C  CRWD10    EQU  $
1338                *                         3-CALL STTCHR (STATUS);
1339  000C  06A0              BL   @STTCHR
      000E  0386'
1340                *                         3-I = BOE (ABS (MOD (DRIVEID, 2) -
1341  0010  C02D              MOV  @R8*2(R13),R0      DRIVE ID MODULO 2
      0012  0010
1342  0014  0240              ANDI R0,1
      0016  0001
1343  0018  0220              AI   R0,BOE             SHIFT COUNT TO BOE FOR DRIVE
      001A  000B
1344  001C  0B09              SRC  R9,0
1345                *                         3-END;
1346  001E  17F6              JNC  CRWD10             IF NOT BOE
1347                *
1348                *                         2-CALL CHGMOD (PRBPTR, DCTPTR, MOD
1349  0020  06A0              BL   @CHGMOD
      0022  0342'
1350  0024  8040              DATA MODPBK
1351                *                         2-CALL CMND (ADDR (LOADSQ));
1352  0026  06A0              BL   @CMND
      0028  0396'
1353  002A  0009              DATA LOADSQ
1354                *                         2-I = 0;
1355                *                         2-DO WHILE I = 0;
1356          002C  CRWD20    EQU  $
1357                *                         3-CALL STTCHR (STATUS);
1358  002C  06A0              BL   @STTCHR
      002E  0386'
```

```
1359                    *                               3-I = PBKRDY
1360  0030  0B99               SRC   R9,PBKRDY
1361                    *                               3-END;
1362  0032  17FC               JNC   CRWD20
1363                    *
1364                    *                               2-RETURN;
1365  0034  0452               B     *R2
1366                    *                               2-END CSRWND;
1367  01AA                     RORG
```

```
1370                    * TITLE:     CSUNLD
1371                    *            CASSETTE - UNLOAD
1372                    * REVISION:
1373                    *            ORIGINAL
1374                    * ABSTRACT:
1375                    *            THE INDICATED CASSETTE IS REWOUND TO CLEAR
1376                    *            LEADER.
1377                    * CALLING SEQUENCE:
1378                    *            STANDARD LEVEL ONE SEQUENCE.
1379                    *
1380                    *                        1-PROCEDURE CSUNLD (PARMLIST);
1381                    *                        2-DECLARE I BIT (1);
1382                           DEF  CSUNLD
1383  0000                     DORG 0
1384          0000      CSUNLD EQU  $                  CASSETTE - UNLOAD
1385  0000    C08B             MOV  R11,R2
1386                    *                        2-CALL RSTMOD;
1387  0002    06A0             BL   @RSTMOD           FORCE MODE CHANGE
      0004    0448'
1388                    *                        2-CALL CMND (ADDR (RWDSEQ));
1389  0006    06A0             BL   @CMND
      0008    0396'
1390  000A    0000             DATA RWDSEQ
1391                    *                        2-I = 0;
1392                    *                        2-DO WHILE I = 0;
1393          000C      CUNL10 EQU  $
1394                    *                        3-CALL STTCHR (STATUS);
1395  000C    06A0             BL   @STTCHR
      000E    0386'
1396                    *                        3-I = BOE (ABS (MOD (DRIVEID, 2) -
1397  0010    C02D             MOV  @R8*2(R13),R0     DRIVE ID MODULO 2
      0012    0010
1398  0014    0240             ANDI R0,1
      0016    0001
1399  0018    0220             AI   R0,BOE            SHIFT COUNT TO BOE FOR DRIVE
      001A    000B
1400  001C    0B09             SRC  R9,0
1401  001E    17F6             JNC  CUNL10
1402                    *
1403                    *                        2-RETURN;
1404  0020    0452             B    *R2
1405                    *                        2-END CSUNLD;
1406  01AA                     RORG
```

```
1409                    * TITLE:     LGRASC
1410                    *            LOG - READ ASCII
1411                    * REVISION:
1412                    *            ORIGINAL
1413                    * ABSTRACT:
1414                    *            CHARACTERS ARE TRANSFERRED FROM THE 733ASR
1415                    *            KEYBOARD TO THE USER'S BUFFER UNTIL:
1416                    *                - THE DESIRED NUMBER OF CHARACTERS HAVE
1417                    *                  BEEN TRANSFERRED,
1418                    *                - A CARRIAGE RETURN (END OF RECORD) IS
1419                    *                  DETECTED, OR
1420                    *                - A DC3 IS THE FIRST CHARACTER DETECTED
1421                    *                  (END OF FILE CONDITION).
1422                    * CALLING SEQUENCE:
1423                    *            STANDARD LEVEL ONE SEQUENCE.
1424                    *
1425                    *                          1-PROCEDURE LGRASC (PARMLIST);
1426                    *                          2-DECLARE I FIXED (16);
1427                    *                          2-DECLARE CHAR BIT (8);
1428                            DEF   LGRASC
1429          01AA'     LGRASC EQU   $
1430   01AA   C08B             MOV   R11,R2              SAVE RETURN-LEVEL ONE
1431                    *                          2-I = 0;
1432   01AC   04C6             CLR   R6
1433                    *                          2-DO WHILE EOFFLAG = 0 & I < BUFFER
1434                    *                          2-       & UNRECOVERABLEERROR = 0;
1435                    *
1436          01AE'     LR010  EQU   $
1437   01AE   D06A             MOVB  @SYSFLG(R10),R1     ISOLATE SYSTEM FLAGS
       01B0   0002
1438   01B2   0BE1             SRC   R1,EOFLG            IF END.OF.FILE
1439   01B4   1839             JOC   LR028
1440          0001     EOF2UE EQU   UE-EOFLG
1441   01B6   0B11             SRC   R1,EOF2UE           IF UNRECOVERABLE ERROR
1442   01B8   1837             JOC   LR028
1443   01BA   8A86             C     R6,@BFLNTH(R10)     IF I >= BUFFER LENGTH
       01BC   0006
1444   01BE   1434             JHE   LR028
1445                    *                          3-IF NOWAITFLAG=0 THEN
1446   01C0   D06A             MOVB  @USEFLG(R10),R1
       01C2   0003
1447   01C4   0BC1             SRC   R1,NWFLG
1448   01C6   1804             JOC   LR013
1449                    *                          4-CALL RCVCHR(CHAR,WAIT);
1450   01C8   04C9             CLR   R9                  WAIT FOR CHARACTER.
1451   01CA   06A0             BL    @RCVCHR
       01CC   0412'
1452   01CE   1003             JMP   LR014
1453                    *                          3-ELSE CALL RCVCHR(CHAR,NOWAIT);
1454          01D0'     LR013  EQU   $
1455   01D0   0709             SETO  R9
1456   01D2   06A0             BL    @RCVCHR
       01D4   0412'
1457          01D6'     LR014  EQU   $
```

```
1458                    *                       3-IF NOWAITFLAG=1 & CHAR=0 THEN RE
1459  01D6  D06A        MOVB @USEFLG(R10),R1
      01D8  0003
1460  01DA  0BC1        SRC  R1,NWFLG
1461  01DC  1703        JNC  LR017
1462  01DE  C249        MOV  R9,R9                IF NO CHAR RETURNED
1463  01E0  1601        JNE  LR017
1464  01E2  0452        B    *R2                  RETURN
1465        01E4' LR017  EQU  $
1466                    *                       3-IF CHAR NE BS THEN BSFLAG = 0;
1467  01E4  0289        CI   R9,BS*LBYTE
      01E6  0800
1468  01E8  1302        JEQ  LR018                4
1469  01EA  04E0        CLR  @BSFLG
      01EC  0000
1470        01EE' LR018  EQU  $
1471                    *                       3-IF CHAR >= SP & CHAR < DEL THEN DO
1472  01EE  0289        CI   R9,SP*LBYTE          IF CHAR < SP
      01F0  2000
1473  01F2  110B        JLT  LR020
1474  01F4  0289        CI   R9,DEL*LBYTE         IF CHAR = DEL
      01F6  7F00
1475  01F8  1408        JHE  LR020
1476                    *                       4-CALL SNDCHR (CHAR,DELAY3);
1477  01FA  06A0        BL   @SNDCHR              ECHO CHARACTER
      01FC  03DA'
1478  01FE  0003        DATA DELAY3
1479                    *                       4-CALL PUTCHR (CHAR, I);
1480  0200  C206        MOV  R6,R8                PLACE IN USERS BUFFER
1481  0202  06A0        BL   @PUTCHR
      0204  043A'
1482                    *                       4-I = I + 1;
1483  0206  0586        INC  R6
1484                    *                       4-END;
1485  0208  1058        JMP  LR200
1486                    *
1487                    *                       3-ELSE IF CHAR = DEL THEN DO;
1488        020A' LR020  EQU  $
1489  020A  160F        JNE  LR030                IF CHAR .NE. DEL
1490                    *                       4-IF CHARACTERIO = 1
1491                    *                       5-THEN DO;
1492  020C  D06A        MOVB @USEFLG(R10),R1      IF NOT CHARACTER MODE
      020E  0003
1493  0210  0BD1        SRC  R1,CHRMOD
1494  0212  1705        JNC  LR023
1495                    *                       6-CALL PUTCHR (CHAR, I);
1496  0214  C206        MOV  R6,R8
1497  0216  06A0        BL   @PUTCHR              PLACE DEL IN BUFFER
      0218  043A'
1498                    *                       6-I = I + 1;
1499  021A  0586        INC  R6
1500                    *                       6-END;
1501  021C  1001        JMP  LR026
1502                    *
1503        021E' LR023  EQU  $
```

```
1504                    *           ELSE I = 0;
1505  021E  04C6               CLR   R6
1506                    *
1507        0220'       LR026 EQU   $
1508                    *                           4-CALL CMND (ADDR (ECODEL));
1509  0220  06A0               BL    @CMND
      0222  0396'
1510  0224  0035'               DATA  ECODEL
1511                    *                           4-END;
1512  0226  1049               JMP   LR200
1513        0228'       LR028 EQU   $
1514  0228  104A               JMP   LR900                 INTERMEDIATE JUMP
1515                    *
1516                    *                           3-ELSE IF CHAR = BS THEN DO;
1517        022A'       LR030 EQU   $
1518  022A  0289               CI    R9,BS*LBYTE           IF CHAR .NE. BS
      022C  0800
1519  022E  1619               JNE   LR040
1520                    *                           4-IF CHARACTERIO = 1 THEN DO;
1521  0230  D06A               MOVB  @USEFLG(R10),R1       IF NOT CHARACTER MODE
      0232  0003
1522  0234  0BD1               SRC   R1,CHRMOD
1523  0236  1704               JNC   LR035
1524                    *                           5-CALL PUTCHR (CHAR, I);
1525  0238  C206               MOV   R6,R8                  PUT BS IN BUFFER
1526  023A  06A0               BL    @PUTCHR
      023C  043A'
1527                    *                           5-I = I + 2;
1528  023E  05C6               INCT  R6
1529                    *                           5-END;
1530        0240'       LR035 EQU   $
1531                    *                           4-I = MAX (I - 1, 0);
1532  0240  0606               DEC   R6
1533  0242  1501               JGT   LL4
1534  0244  04C6               CLR   R6
1535        0246'       LL4   EQU   $
1536                    *                           4-CALL CMND (ADDR (ECOBS));
1537  0246  06A0               BL    @CMND
      0248  0396'
1538  024A  0031'               DATA  ECOBS
1539                    *                           4-IF BSFLAG EQ 0 THEN DO;
1540  024C  C220               MOV   @BSFLG,R8
      024E  01EC'
1541  0250  1607               JNE   LR038
1542                    *                           5- BSFLAG = -1;
1543  0252  0720               SETO  @BSFLG
      0254  024E'
1544                    *                           5-CALL SNDCHR (LF,DELAY3);
1545  0256  0209               LI    R9,LF*LBYTE                    4
      0258  0A00
1546  025A  06A0               BL    @SNDCHR
      025C  03DA'
1547  025E  0003               DATA  DELAY3
1548                    *                           5-END
1549        0260'       LR038 EQU   $
```

```
1550                    *                               4-END;
1551  0260  102C         JMP   LR200
1552                    *
1553                    *                               3-ELSE IF CHAR = HT THEN DO;
1554        0262' LR040 EQU   $
1555  0262  0289         CI    R9,HT*LBYTE               IF CHAR .NE. HT
      0264  0900
1556  0266  160A         JNE   LR050
1557                    *                               4-CALL PUTCHR (CHAR, I);
1558  0268  C206         MOV   R6,R8                     HT IN BUFFER
1559  026A  06A0         BL    @PUTCHR
      026C  043A'
1560                    *                               4-CALL SNDCHR (' ', DELAY3);
1561  026E  0209         LI    R9,SP*LBYTE               SP TO PRINTER
      0270  2000
1562  0272  06A0         BL    @SNDCHR
      0274  03DA'
1563  0276  0003         DATA  DELAY3
1564                    *                               4-I = I + 1;
1565  0278  0586         INC   R6
1566                    *                               4-END;
1567  027A  101F         JMP   LR200
1568                    *
1569                    *                               3-ELSE IF CHAR = DC3 & I = 0 THEN EO
1570        027C' LR050 EQU   $
1571  027C  0289         CI    R9,DC3*LBYTE              IF CHAR .NE. DC3
      027E  1300
1572  0280  1606         JNE   LR060
1573  0282  C186         MOV   R6,R6                     IF I .NE. 0
1574  0284  1604         JNE   LR060
1575                    *
1576  0286  FAA0         SOCB  @EFMASK,@SYSFLG(R10)
      0288  00DA'
      028A  0002
1577  028C  1016         JMP   LR200
1578                    *
1579                    *                               3-ELSE IF CHAR = CR THEN DO;
1580        028E' LR060 EQU   $
1581  028E  0289         CI    R9,CR*LBYTE               IF CHAR .NE. CR
      0290  0D00
1582  0292  1606         JNE   LR070
1583                    *                               4-CALL SNDCHR (CHAR, DLY27);
1584  0294  06A0         BL    @SNDCHR
      0296  03DA'
1585  0298  001B         DATA  DLY27
1586                    *                               4-CHARCOUNT = I;
1587  029A  CA86         MOV   R6,@CCOUNT(R10)
      029C  0008
1588                    *                               4-RETURN;
1589                    *                               4-END;
1590  029E  0452         B     *R2
1591                    *
1592                    *                               3-ELSE IF CHAR = LF
1593        02A0' LR070 EQU   $
1594  02A0  0289         CI    R9,LF*LBYTE               IF CHAR .NE. LF
```

```
       02A2  0A00
1595   02A4  1604          JNE  LR080
1596               *                          4-THEN CALL SNDCHR (CHAR, DELAY3)
1597   02A6  06A0          BL   @SNDCHR
       02A8  03DA'
1598   02AA  0003          DATA DELAY3
1599   02AC  1006          JMP  LR200
1600               *
1601               *                          3-ELSE IF CHAR = ESC
1602         02AE' LR080   EQU  $
1603   02AE  0289          CI   R9,ESC*LBYTE       IF CHAR .NE. ESC
       02B0  1B00
1604   02B2  1603          JNE  LR090
1605               *                          4-THEN UNRECOVERABLEERROR = 1;
1606   02B4  FAA0          SOCB @UEMASK,@SYSFLG(R10)
       02B6  0120'
       02B8  0002
1607               *       JMP  LR200
1608               *
1609         02BA' LR090   EQU  $
1610               *                          3-END;
1611         02BA' LR200   EQU  $
1612   02BA  0460          B    @LR010
       02BC  01AE'
1613               *
1614         02BE' LR900   EQU  $                    END.OF.LOOP
1615               *                          2-CHARCOUNT = I;
1616   02BE  CA86          MOV  R6,@CCOUNT(R10)
       02C0  0008
1617               *                          2-RETURN;
1618   02C2  0452          B    *R2
1619               *                          2-END LGRASC;
```

```
1622                  * TITLE:    LGWASC
1623                  *           LOG - WRITE ASCII
1624                  * REVISION:
1625                  *           ORIGINAL
1626                  * ABSTRACT:
1627                  *           CHARACTER'S ARE TRANSFERRED FROM THE USER'S
1628                  *           BUFFER TO THE 733 ASR PRINTER UNTIL:
1629                  *               - THE DESIRED NUMBER OF CHARACTERS HAVE
1630                  *                 BEEN TRANSFERRED, OR
1631                  *               - UNTIL ALL REMAINING CHARACTERS IN THE
1632                  *                 BUFFER ARE EQUAL TO SPACES.
1633                  * CALLING SEQUENCE:
1634                  *           STANDARD LEVEL ONE SEQUENCE.
1635                  *
1636                  *                       1-PROCEDURE LGWASC (PARMLIST);
1637                  *                       2-DECLARE (I, J) FIXED (16);
1638                  *                       2-DECLARE CHAR BIT (8);
1639                  *                       2-DECLARE FOUND FIXED (16);
1640                        DEF   LGWASC
1641          02C4' LGWASC EQU   $
1642  02C4    C08B          MOV   R11,R2            SAVE RETURN - LEVEL ONE
1643                  *                       2-I = CHARCOUNT;
1644  02C6    C1AA          MOV   @CCOUNT(R10),R6
      02C8    0008
1645                  *                       2-IF I = 0 THEN RETURN;
1646  02CA    1601          JNE   LL8
1647  02CC    0452          B     *R2
1648          02CE' LL8     EQU   $
1649                  *                       2-I = I - 1;
1650  02CE    0606          DEC   R6
1651                  *                       2-DO J FROM 0 TO I BY 1;
1652  02D0    04C7          CLR   R7                INITIALIZE J
1653          02D2' LW030   EQU   $
1654  02D2    8187          C     R7,R6             IF J > I
1655  02D4    1530          JGT   LW900
1656                  *                       3-CALL RCVCHR (CHAR, NOWAIT);
1657  02D6    0709          SETO  R9
1658  02D8    06A0          BL    @RCVCHR
      02DA    0412'
1659                  *                       3-IF CHAR = ESC THEN DO;
1660  02DC    0289          CI    R9,ESC*LBYTE
      02DE    1B00
1661  02E0    1604          JNE   LW035
1662                  *                       4-UNRECOVERABLEERROR = 1;
1663  02E2    FAA0          SOCB  @UEMASK,@SYSFLG(R10)
      02E4    02B6'
      02E6    0002
1664                  *                       4-RETURN;
1665  02E8    0452          B     *R2
1666                  *                       4-END;
1667          02EA' LW035   EQU   $
1668                  *                       3-CALL GETCHR (CHAR, J);
1669  02EA    C207          MOV   R7,R8
1670  02EC    06A0          BL    @GETCHR
```

```
       02EE  042C'
1671                 *                          3-IF CHAR = HT THEN CHAR = SP;
1672   02F0  0289           CI    R9,HT*LBYTE
       02F2  0900
1673   02F4  1602           JNE   LL12
1674   02F6  0209           LI    R9,SP*LBYTE
       02F8  2000
1675         02FA' LL12      EQU   $
1676                 *                          3-IF CHAR >= SP .OR. (CHAR >= BEL &
1677   02FA  0289           CI    R9,SP*LBYTE        IF CHAR >= SP
       02FC  2000
1678   02FE  1406           JHE   LL14
1679   0300  0289           CI    R9,BEL*LBYTE       IF CHAR < BEL .OR. CHAR > LF
       0302  0700
1680   0304  1106           JLT   LW040
1681   0306  0289           CI    R9,LF*LBYTE
       0308  0A00
1682   030A  1503           JGT   LW040
1683         030C' LL14      EQU   $
1684                 *                          4-THEN CALL SNDCHR (CHAR, DELAY3);
1685   030C  06A0           BL    @SNDCHR
       030E  03DA'
1686   0310  0003           DATA  DELAY3
1687                 *                          3-IF CHAR = CR THEN CALL SNDCHR (CHA
1688         0312' LW040     EQU   $
1689   0312  0289           CI    R9,CR*LBYTE        IF CHAR .NE. CR
       0314  0D00
1690   0316  1603           JNE   LW045
1691   0318  06A0           BL    @SNDCHR            CALL SNDCHR (CHAR, DLY27)
       031A  03DA'
1692   031C  001B           DATA  DLY27
1693                 *
1694         031E' LW045     EQU   $
1695                 *                          3-IF CHAR = FF THEN DO K FROM 1 TO 8
1696   031E  0289           CI    R9,FF*LBYTE        IF CHAR .NE. FF
       0320  0C00
1697   0322  1607           JNE   LW050
1698   0324  0203           LI    R3,8
       0326  0008
1699         0328' LL16      EQU   $
1700                 *                          4-CALL CMND (ADDR (ECOFF));
1701   0328  06A0           BL    @CMND
       032A  0396'
1702   032C  0028'          DATA  ECOFF
1703                 *                          4-END;
1704   032E  0603           DEC   R3
1705   0330  15FB           JGT   LL16
1706                 *                          3-END;
1707         0332' LW050     EQU   $
1708   0332  0587           INC   R7                 J = J + 1;
1709   0334  10CE           JMP   LW030
1710                 *
1711                 *                          2-RETURN;
1712         0336' LW900     EQU   $
1713   0336  0452           B     *R2
```

```
1714                *                    2-END LGWASC;
```

```
1717                    * TITLE:     MONCHR
1718                    *            SINGLE CHARACTER INPUT FOR MONITOR
1719                    * REVISION:
1720                    *            ORIGINAL
1721                    * ABSTRACT:
1722                    *            NON-STANDARD LEVEL ONE SUBROUTINE.
1723                    *
1724                    *            THIS SUBROUTINE PROVDES SINGLE CHARACTER
1725                    *            INPUT FOR THE MONITOR.  IF A CHARACTER IS
1726                    *            PRESENT ON THE CRU INTERFACE, IT IS RETURNED
1727                    *            IN R9, LEFT JUSTIFIED WITH ZERO FILL.  IF NO
1728                    *            CHARACTER IS PRESENT ON THE INTERFACE, A
1729                    *            ZERO IS RETURNED IN R9.
1730                    * CALLING SEQUENCE:
1731                    *            BL   @MONCHR
1732                    *            UPON RETURN CHARACTER IS IN R9
1733                    *            R12 DESTROYED
1734                    *
1735                    *                         1- PROCEDURE MONITORCHAR (CHAR);
1736                    *                         2-DECLARE CHAR CHARACTER (1);
1737                    *                         2-
1738                           DEF  MONCHR
1739          0338'    MONCHR EQU  $
1740                    *                         2-CALL RCVCHR (CHAR, NOWAIT);
1741    0338  0709           SETO R9                SET NO WAIT FLAG
1742    033A  C320           MOV  @DC7331+DCTCRU,R12
        033C  0000'
1743    033E  0460           B    @RCVCHR
        0340  0412'
1744                    *                         2-RETRN;
1745                    *                         2-END MONITORCHAR;
```

```
1748                * TITLE:     CHGMOD
1749                *            CHANGE RECORD/PLAYBACK MODE
1750                * REVISION:
1751                *            ORIGINAL
1752                * ABSTRACT:
1753                *            THIS ROUTINE IS CALLED BY THOSE CASSETTE
1754                *            FUNCTIONS WHICH MUST INSURE PROPER PLAYBACK/
1755                *            RECORD MODE.  THE DEVICE CONTROL TABLE
1756                *            CONTAINS CURRENT MODE.  IF A CHANGE OF MODE
1757                *            IS REQUIRED, THEN THE APPROPRIATE COMMAND IS
1758                *            ISSUED TO THE 733ASR.
1759                * CALLING SEQUENCE:
1760                *            LEVEL TWO ROUTINE.
1761                *
1762                *            BL    @CHGMOD
1763                *            DATA (DESIRED MODE INDICATOR)
1764                *
1765                *                          1-PROCEDURE CHGMOD (PRBPTR, DCTPTR
1766                *                          2-DECLARE MODE BIT (16);
1767                *                          2-/* NOTE PRBPTR AND DCTPTR GLOBAL
1768                *                          2-   DEFINED */
1769                *                          2-DECLARE (INUSE, OTHER) FIXED;
1770          0342' CHGMOD EQU   $
1771  0342    C3FB        MOV   *R11+,R15          FETCH MODE INDICATOR
1772  0344    C0CB        MOV   R11,R3             LEVEL TWO RETURN
1773                *                          2-INUSE = MOD (UNITNO, 2);
1774                *                          2-OTHER = ABS (INUSE-1);
1775  0346    C3AD        MOV   @R8*2(R13),R14     UNIT NUMBER
      0348    0010
1776  034A    024E        ANDI  R14,1              MODULO 2
      034C    0001
1777                *                          2-IF (MODE = MODPBK &
1778                *                          2-    CASSETTECONTROL (INUSE).PLAY
1779                *                          2-   (MODE = MODRCD &
1780                *                          2-    CASSETTECONTROL (INUSE).RECO
1781                *                          2-   THEN RETURN;
1782  034E    C16D        MOV   @R6*2(R13),R5      DCT ADDRESS
      0350    000C
1783  0352    A14E        A     R14,R5
1784  0354    53E5        SZCB  @CSTCNT(R5),R15
      0356    0002
1785  0358    1601        JNE   CM010              IF PLAYBACK/RECORD BIT
1786                *                                NOT ON IN DCT
1787  035A    0453        B     *R3
1788                *                          2-CASSETTECONTROL (INUSE) = 0;
1789                *                          2-CASSETTECONTROL (OTHER) = 0;
1790          035C' CM010 EQU   $
1791  035C    614E        S     R14,R5
1792  035E    4960        SZC   @MODBTS,@CSTCNT(R5)
      0360    0004'
      0362    0002
1793  0364    C04F        MOV   R15,R1
1794                *                          2-IF MODE = MODPBK THEN DO;
1795                *                          4-CASSETTECONTROL (INUSE).PLAYBACK
```

```
1796                    *                       4-CASSETTECONTROL (OTHER).RECORD =
1797                    *                       4-CALL CMND (ADDR (PMODE));
1798                    *                       4-END;
1799                    *                       3-ELSE DO;
1800                    *                       4-CASSETTECONTROL (INUSE).RECORD =
1801                    *                       4-CASSETTECONTROL (OTHER).PLAYBACK
1802                    *                       4-CALL CMND (ADDR (RMODE));
1803                    *                       4-END;
1804  0366  C38E              MOV   R14,R14             IF RIGHT CASSETTE, SWAP
1805  0368  1301              JEQ   CM020                 BYTES IN MODE MASK
1806  036A  06CF              SWPB  R15
1807        036C' CM020       EQU   $
1808  036C  E94F              SOC   R15,@CSTCNT(R5)     SET MODE BITS FOR BOTH
      036E  0002
1809  0370  0281              CI    R1,MODRCD           IF NOT RECORD MODE
      0372  4080
1810  0374  1604              JNE   CM030
1811                    *
1812  0376  06A0              BL    @CMND               SEND RECORD MODE
      0378  0396'
1813  037A  0006'             DATA  RMODE
1814  037C  1003              JMP   CM040
1815        037E' CM030       EQU   $
1816  037E  06A0              BL    @CMND               SEND PLAYBACK MODE
      0380  0396'
1817  0382  000D'             DATA  PMODE
1818                    *                       2-RETURN;
1819        0384' CM040       EQU   $
1820  0384  0453              B     *R3                 RETURN
1821                    *
1822                    *                       2-END CHGMOD;
```

```
1825                    * TITLE:    STTCHR
1826                    *           GET STATUS CHARACTER
1827                    * REVISION:
1828                    *           ORIGINAL
1829                    * ABSTRACT:
1830                    *           THE STATUS COMMAND SEQUENCE IS SENT AND THE
1831                    *           FIRST CHARACTER RECEIVED IS RETURNED TO THE
1832                    *           CALLER AS THE STATUS CHARACTER.
1833                    * CALLING SEQUENCE:
1834                    *           LEVEL TWO ROUTINE
1835                    *
1836                    *           BL   @STTCHR
1837                    *           STATUS CHARACTER IS RETURNED IN R9 LEFT BYTE.
1838                    *
1839                    *                      1-PROCEDURE STTCHR (STATUSCHAR);
1840                    *                      2-DECLARE STATUSCHAR BIT (8);
1841                    *                      2-
1842          0386'     STTCHR EQU  $
1843   0386   C0CB              MOV  R11,R3          LEVEL TWO RETURN SAVE
1844                    *                      2-CALL CMND (ADDR (STTSEQ));
1845   0388   06A0              BL   @CMND
       038A   0396'
1846   038C   002C'            DATA STTSEQ
1847                    *                      2-CALL RCVCHR (STATUSCHAR, WAIT);
1848   038E   04C9              CLR  R9              SET WAIT INDICATOR
1849   0390   06A0              BL   @RCVCHR
       0392   0412'
1850                    *                      2-RETURN;
1851   0394   0453              B    *R3
1852                    *                      2-END STTCHR;
```

```
1855                    * TITLE:     CMND
1856                    *            SEND COMMAND SEQUENCE
1857                    * REVISION:
1858                    *            ORIGINAL
1859                    * ABSTRACT:
1860                    *            A SERIES OF CHARACTERS IS TRANSFERRED TO THE
1861                    *            733ASR UNTIL THE END GROUP CHARACTER IS
1862                    *            DETECTED.  THIS SERIES OF CHARACTERS BEGINS
1863                    *            AT THE ADDRESS PASSED BY THE CALLER AND CONSIST
1864                    *            OF THREE TYPES OF ENTRIES:
1865                    *                1. A CHARACTER TO BE SEND WITHOUT CHANGE.
1866                    *                   HIGH ORDER BIT = 1 ==> MOD BY DRIVE ID
1867                    *                2. DELAY GROUP INDICATOR. BYTE = >FE.
1868                    *                   NEXT BYTE IS COUNT OF DELAY CHARACTERS
1869                    *                   TO ISSUE.
1870                    *                3. END GROUP CHARACTER. BYTE = >FF.
1871                    * CALLING SEQUENCE:
1872                    *            LEVEL THREE ROUTINE
1873                    *            BL   @CMND
1874                    *            DATA <ADDR OF COMMAND SEQUENCE>
1875                    *            R10 - PRB ADDRESS
1876                    *            R8, R9 RESERVED
1877                    *                        1-PROCEDURE CMND (CMNDPTR, DRIVEID
1878                    *                        2-DECLARE (CMNDPTR, BYTEPTR) POINT
1879                    *                        2-DECLARE COMMANDBYTE FIXED (8)
1880                    *                        2-    CONTROL (BYTEPTR);
1881                    *                        2-DECLARE DRIVE-ID FIXED (8);
1882                    *                        2-DECLARE BYTE FIXED;
1883                    *
1884          0396' CMND    EQU  $
1885  0396    C10B          MOV  R11,R4              LEVEL TWO RETURN SAVE
1886                  *                        2-BYTEPTR = COMNDPTR
1887  0398    C3B4          MOV  *R4+,R14
1888                  *                        2-DO FOREVER;
1889  039A    04C9          CLR  R9
1890          039C' CMND10  EQU  $
1891                  *                        3-BYTE = COMMANDBYTE;
1892                  *                        3-BYTEPTR = BYTEPTR + 1;
1893  039C    D27E          MOVB *R14+,R9
1894                  *                        3-IF BYTE = ENDGROUP THEN EXIT DO;
1895  039E    0289          CI   R9,ENDGRP*LBYTE
      03A0    FF00
1896  03A2    131A          JEQ  CMND90
1897                  *                        3-IF BYTE = DELAYGROUP THEN DO;
1898  03A4    0289          CI   R9,DLYGRP*LBYTE
      03A6    FE00
1899  03A8    1605          JNE  CMND20
1900                  *                        5-CALL DLYCHR (COMMANDBYTE);
1901                  *                        5-BYTEPTR = BYTEPTR + 1;
1902  03AA    D23E          MOVB *R14+,R8
1903  03AC    0988          SRL  R8,8
1904  03AE    06A0          BL   @DLYCHR
      03B0    03FC'
1905                  *                        5-END;
```

```
1906  03B2  1011          JMP   CMND40
1907                *                              4-ELSE DO;
1908        03B4' CMND20 EQU   $
1909                *                              5-IF BYTE < 0
1910  03B4  C249          MOV   R9,R9                    IF BYTE >= 0
1911  03B6  150C          JGT   CMND30
1912  03B8  130B          JEQ   CMND30
1913                *                              5-    THEN /* THIS IS A COMMAND BYTE
1914                *                              5-           TO BE MODIFIED BY DRIV
1915                *                              6-J = MOD (DRIVEID, 2);
1916  03BA  C3ED          MOV   @R8*2(R13),R15
      03BC  0010
1917  03BE  024F          ANDI  R15,1
      03C0  0001
1918                *                              6-IF J .NE. 0
1919  03C2  1306          JEQ   CMND30
1920                *                              7-THEN BYTE = BYTE + BYTMOD (BYTE);
1921  03C4  0A19          SLA   R9,1
1922  03C6  0999          SRL   R9,9
1923  03C8  D3E9          MOVB  @BYTMOD-'1'(R9),R15
      03CA  000B'
1924  03CC  0B89          SRC   R9,8
1925  03CE  B24F          AB    R15,R9
1926                *
1927        03D0' CMND30 EQU   $
1928                *                              5-CALL SNDCHR (BYTE, 0);
1929  03D0  06A0          BL    @SNDCHR
      03D2  03DA'
1930  03D4  0000          DATA  0
1931                *                              5-END;
1932                *                              3-END;
1933        03D6' CMND40 EQU   $
1934  03D6  10E2          JMP   CMND10
1935                *
1936                *                              2-RETURN;
1937        03D8' CMND90 EQU   $
1938  03D8  0454          B     *R4
1939                *                              2-END CMND;
```

```
1942                    * TITLE:     SNDCHR
1943                    *            SEND CHARACTER TO 733 ASR
1944                    * REVISION:
1945                    *            ORIGINAL
1946                    * ABSTRACT:
1947                    *            ONE CHARACTER IS SENT TO THE 733 ASR THROUGH
1948                    *            THE CRU INTERFACE, AFTER WHICH A CALLER
1949                    *            SPECIFIED NUMBER OF DELAY CHARACTERS ARE SENT.
1950                    * CALLING SEQUENCE:
1951                    *            LEVEL FOUR ROUTINE
1952                    *            BL    @SNDCHR
1953                    *            DATA <NUMBER OF DELAY CHARACTERS TO SEND>
1954                    *            R10 - PRB POINTER
1955                    *            R9 - CHARACTER TO SEND IN LEFT BYTE
1956                    *            R8 - RESERVED
1957                    *                         1-PROCEDURE SNDCHR (DELAYTIME, CHAR
1958                    *                         2-DECLARE DELAYTIME FIXED;
1959                    *                         2-DECLARE CHAR BIT (8);
1960                    *
1961          03DA'     SNDCHR EQU   $
1962    03DA  C14B              MOV   R11,R5
1963    03DC  C235              MOV   *R5+,R8              GET DELAYTIME
1964                    *                         2-IF DSR = 1 THEN DO;
1965    03DE  1F0E              TB    DSR                  IF DSR = 0
1966    03E0  1609              JNE   SND015
1967                    *                         4-RTS = 1;
1968    03E2  1D0A              SBO   RTS
1969                    *                         4-DTR = 1;
1970    03E4  1D09              SBO   DTR
1971                    *                         4-OUTCHR = CHAR;
1972    03E6  3209              LDCR  R9,8
1973                    *                         4-DO WHILE WRQ = 0;
1974                    *                         5-END;
1975          03E8'     SND010 EQU   $
1976    03E8  1F0B              TB    WRQ
1977    03EA  16FE              JNE   SND010
1978                    *                         4-CLRWRQ = 1;
1979    03EC  1D0B              SBO   CLRWRQ
1980                    *                         4-CALL DLYCHR (DELAYTIME);
1981    03EE  06A0              BL    @DLYCHR
        03F0  03FC'
1982                    *                         4-END;
1983    03F2  1003              JMP   SND020
1984          03F4'     SND015 EQU   $
1985                    *                         3-ELSE UNRECOVERABLEERROR = 1;
1986    03F4  FAA0              SOCB  @UEMASK,@SYSFLG(R10)
        03F6  02E4'
        03F8  0002
1987                    *                         2-RETURN;
1988          03FA'     SND020 EQU   $
1989    03FA  0455              B     *R5
1990                    *                         2-END SNDCHR;
```

```
1993                 * TITLE:     DLYCHR
1994                 *            SEND DELAY CHARACTERS TO 733 ASR
1995                 * REVISION:
1996                 *            ORIGINAL
1997                 * ABSTRACT:
1998                 *            THE INDICATED NUMBER OF DELAY CHARACTERS ARE
1999                 *            SENT TO THE 733ASR INTERFACE WITH RTS = 0.
2000                 * CALLING SEQUENCE:
2001                 *            LEVEL FIVE SUBROUTINE
2002                 *            MOVE  <NUMBER OF DELAY CHARACTERS>,R8
2003                 *            BL    @DLYCHR
2004                 *            R10 - PRB ADDRESS
2005                 *            DOES NOT MODIFY R14 OR R15
2006                 *
2007                 *                         1-PROCEDURE DLYCHR (DELAYCOUNT);
2008                 *                         2-DECLARE (DELAYCOUNT, I) FIXED;
2009                 *                         2-
2010         03FC' DLYCHR EQU   $
2011                 *                         2-I = DELAYCOUNT;
2012  03FC   C048          MOV   R8,R1
2013  03FE   1108          JLT   DLY090
2014                 *                         2-DO WHILE I > 0;
2015         0400' DLY010 EQU   $
2016  0400   0601          DEC   R1
2017  0402   1106          JLT   DLY090
2018                 *                         3-RTS = 0;
2019  0404   1E0A          SBZ   RTS
2020                 *                         3-OUTCHR = ANYTHING;
2021  0406   3209          LDCR  R9,8
2022                 *                         3-DO WHILE WRQ = 0;
2023                 *                         4-END;
2024         0408' DLY020 EQU   $
2025  0408   1F0B          TB    WRQ
2026  040A   16FE          JNE   DLY020
2027                 *                         3-CLRWRQ = 1;
2028  040C   1D0B          SBO   CLRWRQ
2029                 *                         3-END;
2030  040E   10F8          JMP   DLY010
2031                 *                         2-RETURN;
2032         0410' DLY090 EQU   $
2033  0410   045B          B     *R11
2034                 *                         2-END DLYCHR;
```

```
                    * TITLE:     RCVCHR
                    *            RECEIVE CHARACTER FRM 733ASR
                    * REVISION:
                    *            ORIGINAL
                    * ABSTRACT:
                    *            ONE CHARACTER IS RECEIVED FROM THE 733ASR
                    *            THROUGH THE CRU INTERFACE.  IF A CHARACTER
                    *            IS NOT PRESENT, THE SUBROUTINE WILL CONDITION-
                    *            ALLY WAIT UNTIL SUCH TIME AS CHARACTER IS
                    *            TRANSMITTED.
                    * CALLING SEQUENCE:
                    *            LEVEL FIVE ROUTINE.
                    *            MOV  <WAIT CODE>,R9  WAIT=0/NO WAIT =0
                    *            BL   @RCVCHR
                    *            UPON EXIT, CHARACTER IS IN LEFT BYTE OF R9.
                    *            IF NO WAIT WAS CHOSEN AND A CHARACTER WAS
                    *            NOT TRANSMITTED, R9 = 0.
                    *
                    *                                 CLEARED AND IGNORED.
                    *                               1-PROCEDURE RCVCHR (WAITFLAG, CHAR)
                    *                               2-DECLARE WAITFLAG FIXED;
                    *                               2-DECLARE CHAR BIT (8);
          0412'     RCVCHR EQU  $
                    *                               2-IF TIMERR = 1 THEN CLRRRQ = 1;
0412  1F09                TB   TIMERR                    I=   TIMING ERROR
0414  1601                JNE  RCV010
0416  1D0C                SBO  CLRRRQ
          0418'     RCV010 EQU  $
                    *                               2-DO WHILE RRQ = 0;
0418  1F0C                TB   RRQ
041A  1304                JEQ  RCV020
                    *                               3- IF WAITFLAG .NE. 0 THEN DO;
041C  C249                MOV  R9,R9
041E  13FC                JEQ  RCV010
                    *                               4-CHAR = 0;
0420  04C9                CLR  R9
                    *                               4-RETURN;
                    *                               4-END;
0422  045B                B    *R11
                    *                               3-END;
          0424'     RCV020 EQU  $
                    *                               2-CHAR = INCHAR;
0424  04C9                CLR  R9
0426  35C9                STCR R9,7
                    *                               2-CLRRRQ = 1;
0428  1D0C                SBO  CLRRRQ
                    *                               2-RETURN;
042A  045B                B    *R11
                    *                               2-END RCVCHR;
```

```
2088                    * TITLE:    GETCHR
2089                    *           GET CHARACTER FROM BUFFER
2090                    * REVISION:
2091                    *           ORIGINAL
2092                    * ABSTRACT:
2093                    *           A CHARACTER IS FETCHED FROM THE BUFFER
2094                    * CALLING SEQUENCE:
2095                    *           LEVEL FIVE SUBROUTINE
2096                    *           MOV  <ZERO RELATIVE DISPLACEMENT>,R8
2097                    *           BL   @GETCHR
2098                    *           UPON RETURN, R9 CONTAINS CHARACTER IN LEFT BYTE
2099                    *                        1-PROCEDURE GETCHR (DISPLACEMENT, CH
2100                    *                        2-DECLARE DISPLACEMENT FIXED (16);
2101                    *                        2-DECLARE CHAR BIT (8);
2102          042C'     GETCHR EQU  $
2103                    *                        2-CHAR = BUFFER (DISPLACEMENT);
2104    042C  C06A              MOV  @BFADDR(R10),R1
        042E  0004
2105    0430  A048              A    R8,R1
2106    0432  D251              MOVB *R1,R9
2107    0434  0249              ANDI R9,DEL*LBYTE
        0436  7F00
2108                    *                        2-RETURN;
2109    0438  045B              B    *R11
2110                    *                        2-END GETCHR;
```

```
2113                    * TITLE:     PUTCHR
2114                    *            PUT CHARACTER INTO BUFFER
2115                    * REVISION:
2116                    *            ORIGINAL
2117                    * ABSTRACT:
2118                    *            A CHARACTER IS PLACED INTO THE BUFFER.
2119                    * CALLING SEQUENCE:
2120                    *            LEVEL FIVE SUBROUTINE
2121                    *            MOV  <ZERO RELATIVE DISPLACEMENT>,R8
2122                    *            MOVB <CHARACTER>,R9
2123                    *            BL   @PUTCHR
2124                    *
2125                    *                         1-PROCEDURE PUTCHR (DISPLACEMENT, CH
2126                    *                         2-DECLARE DISPLACEMENT FIXED;
2127                    *                         2-DECLARE CHAR BIT (8);
2128           043A'  PUTCHR EQU  $
2129                    *                         2-BUFFER (DISPLACEMENT) = CHAR;
2130    043A   0249           ANDI R9,DEL*LBYTE
       043C   7F00
2131    043E   C06A           MOV  @BFADDR(R10),R1
       0440   0004
2132    0442   A048           A    R8,R1
2133    0444   D449           MOVB R9,*R1
2134                    *                         2-RETURN;
2135    0446   045B           B    *R11
2136                    *                         2-END PUTCHR;
```

```
 2139                 * TITLE:    RSTMOD
 2140                 *           RESET CASSETTE RECORD/PLAYBACK MODE
 2141                 * REVISION:
 2142                 *           ORIGINAL
 2143                 * ABSTRACT:
 2144                 *           THE RECORD/PLAYBACK BITS IN THE DCT
 2145                 *           CASSETTE CONTROL FIELDS ARE RESET.  THIS
 2146                 *           FORCES A FUTURE MODE CHANGE.
 2147                 * CALLING SEQUENCE:
 2148                 *           LEVEL FIVE SUBROUTINE
 2149                 *           BL   @RSTMOD
 2150                 *                      1-PROCEDURE RSTMOD;
 2151                 *                      2-
 2152          0448'  RSTMOD EQU  $
 2153                 *                      2-CASSETTECONTROL (1).PLAYBACK = 0;
 2154                 *                      2-CASSETTECONTROL (2).PLAYBACK = 0;
 2155                 *                      2-CASSETTECONTROL (1).RECORD = 0;
 2156                 *                      2-CASSETTECONTROL (2).RECORD = 0;
 2157  0448  C06D            MOV  @R6*2(R13),R1     DCT POINTER
       044A  000C
 2158  044C  4860            SZC  @MODBTS,@CSTCNT(R1)  RESET BITS
       044E  0004'
       0450  0002
 2159                 *                      2-RETURN;
 2160  0452  045B            B    *R11
 2161                 *                      2-END RSTMOD;
 2162                        END
0000 ERS
```

```
SP        0764        1131  1250  1472  1561  1674  1677
STATUS    0795        0886
STTCHR    1842        1083  1218  1339  1358  1395
STTSEQ    0883        1846
SYSFLG    0621        1003  1088  1120  1159  1223  1437  1576  1606  1663
                      1986
TIMERR    0690        2061
UE        0627        1440
UEFMSK                0636
UEMASK                0633  1088  1223  1606  1663  1986
USEFLG    0622        1446  1459  1492  1521
VT        0776
WRQ       0692        1976  2025
WTAPEE    0834        1277  1308
WTAPES    0827        1230  1302
XMTING    0689

THERE ARE   0203  SYMBOLS
```

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV	
SHEET	
REV STATUS OF SHEETS	REV
	SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, ERRINT, ERROR INTERRUPT PROCESSOR-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945354-9901

SCALE	REV.	SHEET
		1 of 3

TI-0023-C

```
0003                        IDT  'ERRINT'
0004                  *  TITLE:   ERROR INTERRUPT
0005                  *  REVISION: 11/25/75
0006                  *            ORIGINAL
0007                  *   COMPUTER: 990,ASM
0008                  *   ABSTRACT: WHEN AN ERROR OCCURS AND A TRAP TO LOCATION
0009                  *               8 OCCURS,TRAP WILL BE DIRECTED HERE.IF PROGRA
0010                  *                WHICH CAUSED ERROR IS IN SIE, SIE WILL BE
0011                  *               TURNED OFF. PROGRAM WILL BRANCH TO MONITOR.
0012                  *
0013                  *            REFS AND DEFS
0014                        DEF  EI
0015                        REF  FREMEM
0016                        REF  ERROR
0017                        REF  INIT
0018                  *
0019          000A    R10   EQU  10
0020          0006    MX06  EQU  >0006              EXECUTION ERROR
0021                  *
0022          0000'   EI    EQU  $
0023    0000  02E0          LWPI FREMEM
        0002  0000
0024    0004  0360          RSET
0025    0006  020A          LI   R10,EIA
        0008  000E'
0026    000A  C80A          MOV  R10,@>80
        000C  0080
0027                  *     IF IN SIE PROGRAM WILL BRANCH BACK TO HERE
0028          000E'   EIA   EQU  $
0029    000E  020A          LI   R10,MX06
        0010  0006
0030    0012  06A0          BL   @ERROR
        0014  0000
0031    0016  0460          B    @INIT
        0018  0000
0032                        END
0000 ERS
```

```
960 - 980  CONCORDANCE
$                   0022  0028
EI        0022      0014
EIA       0028      0025
ERROR               0016  0030
FREMEM              0015  0023
INIT                0017  0031
MX06      0020      0029
R10       0019      0025  0026  0029

THERE ARE   0008  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV	
SHEET	

REV STATUS OF SHEETS	
REV	
SHEET	

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, FIND, SCAN MEMORY FOR VALUES-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945355-9901

SCALE	REV.	SHEET 1 of 11

TI-0023-C

```
0003  *    PROCEDURE FWD (CPL POINTER);
0004  *       DECLARE (MASK, VAL) FIXED (16);
0005  *       DECLARE WORD TESTED FIXED (16)
0006  *             CONTROL (TEST POINTER);
0007  *       DECLARE (TEST POINTER, LIMIT POINTER)
0008  *             POINTER;
0009  *       DECLARE OPTION BIT (1);
0010  *       DECLARE PARMS LITERALLY
0011  *          'TEST POINTER, LIMIT POINTER, VAL, MASK';
0012  *       CALL FWBPP (CPL POINTER, PARMS);
0013  *       TEST POINTER = TEST POINTER -
0014  *             MOD (TEST POINTER, 2);
0015  *       LIMIT POINTER = LIMIT POINTER -
0016  *             MOD (LIMIT POINTER, 2);
0017  *       DO TEST POINTER = TEST POINTER
0018  *             TO LIMIT POINTER BY 2;
0019  *          IF (WORD TESTED & MASK) = VAL THEN DO;
0020  *             CALL FWBPRT (TEST POINTER, OPTION);
0021  *             IF OPTION = 1 THEN RETURN;
0022  *             END;
0023  *    PROCEDURE FBT (CPL POINTER);
0024  *       DECLARE (MASK, VAL) BIT (8);
0025  *       DECLARE BYTE TESTED BIT (8) CONTROL
0026  *             (TEST POINTER);
0027  *       DECLARE (TEST POINTER, LIMIT POINTER)
0028  *             POINTER;
0029  *       DECLARE OPTION BIT (1);
0030  *       DECLARE PARMS LITERALLY
0031  *          'TEST POINTER, LIMIT POINTER, VAL, MASK';
0032  *       DECLARE CPL POINTER POINTER;
0033  *       CALL FWBPP (CPL POINTER, PARMS);
0034  *       DO TEST POINTER = TEST POINTER
0035  *             TO LIMIT POINTER BY 1;
0036  *          IF (BYTE TESTED & MASK) = VAL THEN DO;
0037  *             CALL FWBPRT (TEST POINTER, OPTION);
0038  *             IF OPTION = 1 THEN RETURN;
0039  *             END;
0040  *    END FBT;
0041  *    PROCEDURE FWBPP  (CPL POINTER, BEGIN PTR,
0042  *          END PTR, VAL, MASK)  /* PARSE PARAMETERS */
0043  *       DECLARE (VAL, MASK) BIT (16);
0044  *       DECLARE (BEGIN PTR, END PTR) POINTER;
0045  *       DECLARE (1 COMMAND PARAMETER LIST,
0046  *                   3 CPL CONTROL (CPL POINTER),
0047  *                      5 NUMBER PARMS FIXED (8),
0048  *                      5 PARM PRESENT (8) BIT (1),
0049  *                      5 FIRST PARM FIXED (8));
0050  *       DECLARE NUMERIC PARAMETER FIXED (16)
0051  *            CONTROL (PARM POINTER);
0052  *       DECLARE (1 CHARACTER PARAMETER,
0053  *                   3 CHAR PARM CONTROL (PARM POINTER),
0054  *                      5 STRING LENGTH FIXED (8),
0055  *                      5 PARM STRING (1) CHARACTER (1));
0056  *       DECLARE (CPL POINTER, PARM POINTER) POINTER;
```

```
0057                *       PARM POINTER = ADDR (FIRST PARM);
0058                *       /* SET DEFAULT PARAMETER VALUES */
0059                *       BEGIN PTR = 0;
0060                *       END PTR = "FFFF";
0061                *       MASK = 0;
0062                *       IF PARM PRESENT (1) = 1 THEN DO;
0063                *          BEGIN PTR = NUMERIC PARAMETER;
0064                *          PARM POINTER = PARM POINTER + 2;
0065                *          END;
0066                *       IF PARM PRESENT (2) = 1 THEN DO;
0067                *          END PTR = NUMERIC PARAMETER;
0068                *          PARM POINTER = PARM POINTER + 2;
0069                *          END;
0070                *       IF PARM PRESENT (3) = 0
0071                *          THEN CALL ERROR ("0205"); /* NO RETURN */
0072                *       VAL = NUMERIC PARAMETER;
0073                *       PARM POINTER = PARM POINTER + 2;
0074                *       IF PARM PRESENT (4) = 1 THEN DO;
0075                *          MASK = .NOT. NUMERIC PARAMETER
0076                *          END;
0077                *       IF END PTR < BEGIN PTR
0078                *          THEN CALL ERROR ("1101"); /* NO RETURN */
0079                *       RETURN; END FWBPP;
0080                *    PROCEDURE FWBPRT (PTR, OPTION);
0081                *       DECLARE PTR POINTER;
0082                *       DECLARE OPTION BIT (1);
0083                *       CALL PRTMEM (PTR, PTR);
0084                *       CALL GET HEX VLAUE(NO LEADING BLANKS,,
0085                *          ERROR,TERM CHAR,);
0086                *       IF TERM CHAR .NE. BLANK THEN OPTION=1;
0087                *       RETURN
0088                *       END FWBPRT
0089                        IDT  'FIND'
```

```
0091                * TITLE:     FIND
0092                *            FIND WORD AND FIND BYTE COMMAND PROCESSOR
0093                * REVISION:
0094                *            ORIGINAL
0095                * COMPUTER: 990
0096                * ABSTRACT:
0097                *            THE FIND BYTE AND FIND WORD COMMANDS ARE
0098                *            DECODED.  MEMORY IS SEARCHED WITHIN THE
0099                *            BOUNDS INDICATED UNDER MASK FOR EQUALITY.
0100                *            WHEN A MATCH IS FOUND, THE MEMORY ADDRESS
0101                *            AND CONTENTS ARE PRINTED, AND A USER RESPONSE
0102                *            IS REQUESTED.  A CARRIAGE RETURN TERMINATES
0103                *            THE SEARCH, WHILE ENTRY OF A VALID HEX NUMBER
0104                *            OR A SPACE INDICATES CONTINUATION OF THE
0105                *            SEARCH.  MEMORY MODIFICATION IS NOT SUPPORTED
0106                *            BY THIS COMMAND.
0107                * CALLING SEQUENCE:
0108                *            CALLED FROM COMMAND STRING PROCESSOR
0109                * CALLING SEQUENCE:
0110                *                                 1*PROCEDURE FWD (CPL POINTER);
0111                *                                 2*DECLARE (MASK, VAL) FIXED (1
0112                *                                 2*DECLARE WORD TESTED FIXED (1
0113                *                                 2*       CONTROL (TEST POINTER)
0114                *                                 2*DECLARE (TEST POINTER, LIMIT
0115                *                                 2*       POINTER);
0116                *                                 2*DECLARE OPTION BIT (1);
0117                *                                 2*DECLARE PARMS LITERALLY
0118                *                                 3*'TEST POINTER, LIMIT POINTER
0119                *
0120                *WORKSPACE REGISTER DEFINITIONS
0121                *
0122     0000  R0      EQU  0
0123     0001  R1      EQU  1
0124     0002  R2      EQU  2
0125     0003  R3      EQU  3
0126     0004  R4      EQU  4
0127     0005  R5      EQU  5
0128     0006  R6      EQU  6
0129     0007  R7      EQU  7
0130     0008  R8      EQU  8
0131     0009  R9      EQU  9
0132     000A  R10     EQU  10
0133     000B  R11     EQU  11
0134     000C  R12     EQU  12                                              4
0135     000D  R13     EQU  13
0136     000E  R14     EQU  14
0137     000F  R15     EQU  15
0138                *
0139                   REF  BLANK
0140                   REF  GETHXN
0141                   REF  PRTMEM
0142                   REF  ERROR
0143                   REF  ACL,RR
0144                   DEF  FWD
```

```
0145          0000' FWD     EQU   $
0146  0000  C08B            MOV   R11,R2            SAVE RETURN
0147                *                               2=CALL FWBPP (CPL POINTER, PAR
0148  0002  06A0            BL    @FWBPP            GET PARAMETERS INTO REGISTERS
      0004  0044'
0149                *                               2=TEST POINTER = TEST POINTER
0150                *                               5=MOD (TEST POINTER, 2);
0151  0006  0818            SRA   R8,1              ADJUST TO WORD BOUNDARY
0152  0008  0A18            SLA   R8,1
0153                *                               2=LIMIT POINTER = LIMIT POINTE
0154                *                               5=MOD (LIMIT POINTER, 2);
0155  000A  0817            SRA   R7,1
0156  000C  0A17            SLA   R7,1
0157                *                               2=DO TEST POINTER = TEST POINT
0158                *                               5=TO LIMIT POINTER BY 2;
0159          000E' FWD10   EQU   $
0160                *                               3=IF (WORD TESTED & MASK) = VA
0161  000E  C118            MOV   *R8,R4            WORD UNDER TEST
0162  0010  4105            SZC   R5,R4             AND=ED WITH MASK
0163  0012  8184            C     R4,R6
0164  0014  1603            JNE   FWD20             IF NO MATCH
0165                *                               4=CALL FWBPRT (TEST POINTER, 0
0166  0016  06A0            BL    @FWBPRT
      0018  0070'
0167                *                               4=IF OPTION = 1 THEN RETURN;
0168  001A  1044            JMP   FWBEND
0169                *                               4=END;
0170          001C' FWD20   EQU   $
0171  001C  81C8            C     R8,R7
0172  001E  1442            JHE   FWBEND
0173  0020  05C8            INCT  R8
0174  0022  10F5            JMP   FWD10
```

```
0176                    *                                  1=PROCEDURE FBT (CPL POINTER);
0177                    *                                  2=DECLARE (MASK, VAL) BIT (8);
0178                    *                                  2=DECLARE BYTE TESTED BIT (8)
0179                    *                                  2=       (TEST POINTER);
0180                    *                                  2=DECLARE (TEST POINTER, LIMIT
0181                    *                                  2=       POINTER;
0182                    *                                  2=DECLARE OPTION BIT (1);
0183                    *                                  2=DECLARE PARMS LITERALLY
0184                    *                                  3='TEST POINTER, LIMIT POINTER
0185                    *                                  2=DECLARE CPL POINTER POINTER;
0186                            DEF   FBT
0187          0024'     FBT     EQU   $
0188    0024  C08B              MOV   R11,R2                SAVE RETURN
0189                    *                                  2=CALL FWBPP (CPL POINTER, PAR
0190    0026  06A0              BL    @FWBPP
        0028  0044'
0191    002A  0A85              SLA   R6,8                  USE LEAST SIGNIFICANT BYTE
0192    002C  0A85              SLA   R5,8                    OF MASK AND VAL
0193                    *                                  2=DO TEST POINTER = TEST POINT
0194                    *                                  5=TO LIMIT POINTER BY 1;
0195          002E'     FBT10   EQU   $
0196                    *                                  3=IF (BYTE TESTED & MASK) = VA
0197    002E  D118              MOVB  *R8,R4
0198    0030  5105              SZCB  R5,R4                 BYTE TESTED & MASK
0199    0032  9184              CB    R4,R6
0200    0034  1603              JNE   FBT20                 IF NO MATCH
0201                    *                                  4=CALL FWBPRT (TEST POINTER, 0
0202    0036  06A0              BL    @FWBPRT
        0038  0070'
0203                    *                                  4=IF OPTION = 1 THEN RETURN;
0204    003A  1034              JMP   FWBEND
0205                    *                                  4=END;
0206          003C'     FBT20   EQU   $
0207    003C  81C8              C     R8,R7
0208    003E  1432              JHE   FWBEND
0209    0040  0588              INC   R8
0210    0042  10F5              JMP   FBT10
0211                    *                                  1=END FBT;
```

```
0213                    *                                  1-PROCEDURE FWBPP  (CPL POINTE
0214                    *                                  3-END PTR, VAL, MASK)  /* PARS
0215                    *                                  2-DECLARE (VAL, MASK) BIT (16)
0216                    *                                  2-DECLARE (BEGIN PTR, END PTR)
0217                    *
0218                    *                                  2-DECLARE (1 COMMAND PARAMETER
0219                    *                                  3-          3 CPL CONTROL (CPL
0220                    *                                  4-          5 NUMBER PARMS FIXE
0221                    *                                  4-          5 PARM PRESENT (8)
0222                    *                                  4-          5 FIRST PARM FIXED
0223                    *                                  2-DECLARE NUMERIC PARAMETER FI
0224                    *                                  2-     CONTROL (PARM POINTER);
0225                    *                                  2-DECLARE (1 CHARACTER PARAMET
0226                    *                                  3-          3 CHAR PARM CONTROL
0227                    *                                  4-          5 STRING LENGTH FIX
0228                    *                                  4-          5 PARM STRING (1) C
0229                    *                                  2-DECLARE (CPL POINTER, PARM P
0230          0001  PRMPST EQU  1
0231                      DEF  FWBPP
0232          0044' FWBPP  EQU  $
0233  0044  D06A          MOVB @PRMPST(R10),R1    ISOLATE PRESENCE BITS
      0046  0001
0234                    *                                  2-PARM POINTER = ADDR (FIRST P
0235  0048  05CA          INCT R10
0236                    *                                  2-/* SET DEFAULT PARAMETER VAL
0237                    *                                  2-BEGIN PTR = 0;
0238  004A  04C8          CLR  R8
0239                    *                                  2-END PTR = "FFFF";
0240  004C  0707          SETO R7
0241                    *                                  2-MASK = 0;        3
0242  004E  04C5          CLR  R5
0243                    *                                  2-IF PARM PRESENT (1) = 1 THEN
0244  0050  0A11          SLA  R1,1
0245  0052  1701          JNC  FWBPP2              IF NOT PRESENT
0246                    *                                  3-BEGIN PTR = NUMERIC PARAMETE
0247                    *                                  3-PARM POINTER = PARM POINTER
0248                    *                                  3-END;
0249  0054  C23A          MOV  *R10+,R8
0250          0056' FWBPP2 EQU  $
0251                    *                                  2-IF PARM PRESENT (2) = 1 THEN
0252  0056  0A11          SLA  R1,1
0253  0058  1701          JNC  FWBPP4              IF NOT PRESENT
0254                    *                                  3-END PTR = NUMERIC PARAMETER;
0255                    *                                  3-PARM POINTER = PARM POINTER
0256                    *                                  3-END;
0257  005A  C1FA          MOV  *R10+,R7
0258          005C' FWBPP4 EQU  $
0259                    *                                  2-IF PARM PRESENT (3) = 0
0260  005C  0A11          SLA  R1,1
0261                    *                                  3-THEN CALL ERROR ("0205"); /*
0262  005E  1715          JNC  FWBE1               IF RQD PARM MISSING
0263                    *                                  2-VAL = NUMERIC PARAMETER;
0264                    *                                  2-PARM POINTER = PARM POINTER
0265  0060  C18A          MOV  *R10+,R6
```

```
0266                    *                       2=IF PARM PRESENT (4) = 1 THEN
0267  0062  0A11               SLA   R1,1
0268  0064  1702               JNC   FWBPP6       IF NOT PRESENT
0269                    *                       3=MASK = .NOT. NUMERIC PARAMET
0270  0066  C15A               MOV   *R10,R5
0271  0068  0545               INV   R5
0272                    *                       3=END;
0273        006A'   FWBPP6 EQU   $
0274                    *                       2=IF END PTR < BEGIN PTR
0275                    *                       3=THEN CALL ERROR ("1101"); /*
0276  006A  8207               C     R7,R8
0277  006C  1A17               JL    FWBE2
0278                    *                       2=RETURN; END FWBPP;
0279  006E  0458               B     *R11
```

```
0281                    *                          1-PROCEDURE FWBPRT (PTR, OPTIO
0282                    *                          2-DECLARE PTR POINTER;
0283                    *                          2-DECLARE OPTION BIT (1);
0284                    *                          2-CALL PRTMEM (PTR, PTR);
0285          0070' FWBPRT EQU  $
0286  0070    0420          BLWP @ACL
      0072    0000
0287  0074    C248          MOV  R8,R9
0288  0076    C288          MOV  R8,R10
0289  0078    06A0          BL   @PRTMEM
      007A    0000
0290  007C    1000          JMP  FWBPT2                 ESC ENTERED
0291                    *                          2-CALL GET HEX VLAUE(NO LEADIN
0292                    *                          3-ERROR,TERM CHAR,);
0293  007E    06A0          BL   @GETHXN
      0080    0000
0294  0082    1005          JMP  FWBPT2                 ERROR OR EXC
0295  0084    1000          NOP
0296                    *                          2-IF TERM CHAR ,NE, BLANK THEN
0297  0086    9809          CB   R9,@BLANK
      0088    0000
0298  008A    1602          JNE  FWBPT2
0299  008C    05ED          INCT @R11*2(R13)
      008E    0016
0300          0090' FWBPT2 EQU  $
0301  0090    0420          BLWP @RR
      0092    0000
0302                    *                          2-RETURN
0303  0094    045B          B    *R11
0304                    *                          2-END FWBPRT
```

```
 0306                    *                                ERROR PROCESSING
 0307           0096'    FWBE1   EQU   $
 0308   0096    020A             LI    R10,>0205          "MS05"  ROD PARM MISSING
        0098    0205
 0309   009A    1002             JMP   FWBERR
 0310           009C'    FWBE2   EQU   $
 0311   009C    020A             LI    R10,>1113          "DP13" END<BEGIN
        009E    1113
 0312           00A0'    FWBERR  EQU   $
 0313   00A0    06A0             BL    @ERROR
        00A2    0000
 0314           00A4'    FWBEND  EQU   $                  RETURN TO CMND STRING PROC
 0315   00A4    0452             B     *R2
 0316                            END
0000 ERS
```

```
$                       0145  0159  0170  0187  0195  0206  0232  0250  0258
                        0273  0285  0300  0307  0310  0312  0314
ACL                     0143  0286
BLANK                   0139  0297
ERROR                   0142  0313
FBT         0187        0186
FBT10       0195        0210
FBT20       0206        0200
FWBE1       0307        0262
FWBE2       0310        0277
FWBEND      0314        0168  0172  0204  0208
FWBERR      0312        0309
FWBPP       0232        0148  0190  0231
FWBPP2      0250        0245
FWBPP4      0258        0253
FWBPP6      0273        0268
FWBPRT      0285        0166  0202
FWBPT2      0300        0290  0294  0298
FWD         0145        0144
FWD10       0159        0174
FWD20       0170        0164
GETHXN                  0140  0293
PRMPST      0230        0233
PRTMEM                  0141  0289
R0          0122
R1          0123        0233  0244  0252  0260  0267
R10         0132        0233  0235  0249  0257  0265  0270  0288  0308  0311
R11         0133        0146  0188  0279  0299  0303
R12         0134
R13         0135        0299
R14         0136
R15         0137
R2          0124        0146  0188  0315
R3          0125
R4          0126        0161  0162  0163  0197  0198  0199
R5          0127        0162  0192  0198  0242  0270  0271
R6          0128        0163  0191  0199  0265
R7          0129        0155  0156  0171  0207  0240  0257  0276
R8          0130        0151  0152  0161  0171  0173  0197  0207  0209  0238
                        0249  0276  0287  0288
R9          0131        0287  0297
RR                      0143  0301

THERE ARE   0040  SYMBOLS
```

A[

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																															
SHEET																															
REV STATUS OF SHEETS	REV																														
	SHEET																														

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DATE
DWN	
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, GTFELD, INPUT CHARACTER STRING-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945356-9901

SCALE	REV.	SHEET
		1 of 6

TI-9923-C

```
0003                *     PROCEDURE GET FIELD
0004                *          CALL GETBUF(PTR);
0005                *          CHAR CNT = 0;
0006                *          TERM CHAR= 0;
0007                *          IF NO LEADING BLANKS=-1 THEN DO;
0008                *             DO UNTIL CHAR .NE. BLANK
0009                *             CALL GETCHR (CHAR);
0010                *             END;
0011                *          END;
0012                *          DO UNTIL CHAR .EQ. BLANK
0013                *             .OR. CHAR .EQ. 'CR' .OR.  CHAR .EQ. ',';
0014                *             IF CHAR .EQ. 'BS' THEN
0015                *                CHAR CNT = MAX(0,CHAR CNT-1);
0016                *             ELSE IF CHAR .EQ. 'RUBOUT' THEN
0017                *                CHAR CNT = 0;
0018                *             ELSE DO;
0019                *                IF CHAR CNT < 30 THEN DO;
0020                *                   CHAR CNT=CHAR CNT+1;
0021                *                   CSTRING(CHAR CNT) = CHAR;
0022                *                END;
0023                *             END;
0024                *             CALL GETCHR(CHAR);
0025                *          END;
0026                *          TERM CHR =CHAR;
0027                *       END GET FIELD
0028                        IDT  'GTFELD'
0029                * TITLE:    GTFELD
0030                *           INPUT CHARACTER FIELD
0031                * REVISION:
0032                *           ORIGINAL
0033                * COMPUTER: 990,ASM
0034                * ABSTRACT: GET FIELD INPUTS A CHARACTER STRING ONE
0035                *           CHARACTER AT A TIME AND BUILDS A BUFFER
0036                *           CONTAINING THE STRING, SEPERATE ENTRY POINTS AL
0037                *           LEADING BLANKS, THE STRING IS TERMINATED
0038                *           ON A COMMA OR 'CR', A STRING GREATER THAN
0039                *           30 CHARACTERS IS TRUNCATED ON THE RIGHT
0040                *           WITHOUT ANY WARNING.
0041                * CALLING SEQUENCE:
0042                *           BL   @GETFLN
0043                *              DOES NOT SKIP LEADING BLANKS
0044                *           BL   @GETFLD
0045                *              SKIPS LEADING BLANKS
0046                *
0047                *       REF'S AND DEF'S
0048                *
0049                          DEF  GETFLD
0050                          DEF  GETFLN
0051                          REF  COMMA
0052                          REF  BLANK
0053                          REF  ESC
0054                          REF  EOR
0055                          REF  PERIOD
0056                          REF  BS
```

```
0057                        REF  RUBOUT
0058                        REF  GETCHR
0059                        REF  GETBUF
0060                        REF  RETBUF
0061                        REF  LWP
0062                        REF  RWP
0063                        REF  ACL
0064                        REF  RR
0065                  *
0066                  *WORKSPACE REGISTER DEFINITIONS
0067                  *
0068          0000    R0    EQU  0
0069          0001    R1    EQU  1
0070          0002    R2    EQU  2
0071          0003    R3    EQU  3
0072          0004    R4    EQU  4
0073          0005    R5    EQU  5
0074          0006    R6    EQU  6
0075          0007    R7    EQU  7
0076          0008    R8    EQU  8
0077          0009    R9    EQU  9
0078          000A    R10   EQU  10
0079          000B    R11   EQU  11
0080          000C    R12   EQU  12
0081          000D    R13   EQU  13
0082          000E    R14   EQU  14          44
0083          000F    R15   EQU  15
0084                  *
0085                  *
0086                  *     REGISTER ASSIGNMENTS
0087                  *     R10  =   CHAR
0088                  *     R8   =   POINTER TO BUFFER
0089                  *     R7   =   TEMP
0090          0000'   GETFLN EQU  $
0091    0000  04C0          CLR  R0
0092    0002  1001          JMP  GET010
0093          0004'   GETFLD EQU  $
0094    0004  0700          SETO R0
0095          0006'   GET010 EQU  $
0096                  *     *ALLOC,COPY,LINK
0097    0006  0420          BLWP @ACL
        0008  0000
0098                  *                                  3=CALL GETBUF(PTR);
0099                  *     *GETBUF
0100    000A  0420          BLWP @GETBUF
        000C  0000
0101    000E  C74A          MOV  R10,*R13               PLACE @ IN CALLER WP
0102                  *                                  3=CHAR CNT = 0;
0103                  *                                  3=TERM CHAR= 0;
0104    0010  04DA          CLR  *R10
0105    0012  C20A          MOV  R10,R8
0106                  *                                  3=IF NO LEADING BLANKS=='1 THEN
0107    0014  C000          MOV  R0,R0
0108    0016  132F          JEQ  GETF30                 ***NEED TO FIX COMMENTS****
0109                  *                                  4=DO UNTIL CHAR .NE. BLANK
```

```
0110          0018' GETF10 EQU  $
0111                *                              4=CALL GETCHR (CHAR);
0112  0018  06A0           BL   @GETCHR
      001A  0000
0113                *                              4=END;
0114  001C  980A           CB   R10,@BLANK
      001E  0000
0115  0020  13FB           JEQ  GETF10
0116          0022' GETF20 EQU  $
0117  0022  980A           CB   R10,@PERIOD
      0024  0000
0118  0026  13F8           JEQ  GETF10
0119                *                              3=END;
0120                *                              3=DO UNTIL CHAR .EQ. BLANK
0121                *                              4=.OR. CHAR .EQ. 'CR' .OR.  CH
0122  0028  980A           CB   R10,@BLANK
      002A  001E'
0123  002C  1327           JEQ  GETF40
0124  002E  980A           CB   R10,@EOR
      0030  0000
0125  0032  1324           JEQ  GETF40
0126  0034  980A           CB   R10,@ESC
      0036  0000
0127  0038  1321           JEQ  GETF40
0128  003A  980A           CB   R10,@COMMA
      003C  0000
0129  003E  131E           JEQ  GETF40
0130                *                              4=IF CHAR .EQ. '8S' THEN
0131  0040  980A           CB   R10,@BS
      0042  0000
0132  0044  1607           JNE  GETF25
0133                *                              5=CHAR CNT = MAX(0,CHAR CNT-1)
0134  0046  06D8           SWPB *R8
0135  0048  0618           DEC  *R8
0136  004A  1502           JGT  G1
0137  004C  1301           JEQ  G1
0138  004E  0598           INC  *R8
0139          0050' G1     EQU  $
0140  0050  06D8           SWPB *R8
0141  0052  1011           JMP GETF30
0142                *                              4=ELSE IF CHAR .EQ. 'RUBOUT' T
0143          0054' GETF25 EQU  $
0144  0054  980A           CB   R10,@RUBOUT
      0056  0000
0145  0058  1602           JNE  GETF27
0146                *                              5=CHAR CNT = 0;
0147  005A  04D8           CLR  *R8                    2ND BYTE IS DON'T CARE
0148  005C  100C           JMP  GETF30
0149                *                              4=ELSE DO;
0150          005E' GETF27 EQU  $                                  3
0151                *                              5=IF CHAR CNT < 30 THEN DO;
0152  005E  0207           LI   R7,29*256           30 CHARS = LEFT BYTE
      0060  1D00
0153  0062  9607           CB   R7,*R8
0154  0064  1108           JLT  GETF30
```

```
0155                    *                                   6=CHAR CNT=CHAR CNT+1;
0156  0066  06D8               SWPB *R8
0157  0068  0598               INC  *R8
0158  006A  06D8               SWPB *R8
0159                    *                                   6=CSTRING(CHAR CNT) = CHAR;
0160  006C  D1D8               MOVB *R8,R7
0161  006E  0987               SRL  R7,8
0162  0070  A1C8               A    R8,R7
0163  0072  0587               INC  R7                        (SKIP TERM CHAR)
0164  0074  D5CA               MOVB R10,*R7
0165                    *                                   5=END;
0166                    *                                   4=END;
0167          0076' GETF30 EQU  $
0168                    *                                   4=CALL GETCHR(CHAR);
0169  0076  06A0               BL   @GETCHR
      0078  001A'
0170                    *                                   3=END;
0171  007A  10D3               JMP  GETF20
0172          007C' GETF40 EQU  $
0173                    *                                   3=TERM CHR =CHAR;
0174  007C  DA0A               MOVB R10,@1(R8)
      007E  0001
0175                    *                                   2=END GET FIELD
0176  0080  0420               BLWP @RR
      0082  0000
0177  0084  C280               MOV  R0,R10
0178  0086  045B               RT
0179                      END
0000 ERS
```

```
$                0090  0093  0095  0110  0116  0139  0143  0150  0167
                 0172
ACL              0063  0097
BLANK            0052  0114  0122
BS               0056  0131
COMMA            0051  0128
EOR              0054  0124
ESC              0053  0126
G1       0139    0136  0137
GET010   0095    0092
GETBUF           0059  0100
GETCHR           0058  0112  0169
GETF10   0110    0115  0118
GETF20   0116    0171
GETF25   0143    0132
GETF27   0150    0145
GETF30   0167    0108  0141  0148  0154
GETF40   0172    0123  0125  0127  0129
GETFLD   0093    0049
GETFLN   0090    0050
LWP              0061
PERIOD           0055  0117
R0       0068    0091  0094  0107  0107  0177
R1       0069
R10      0078    0101  0104  0105  0114  0117  0122  0124  0126  0128
                 0131  0144  0164  0174  0177
R11      0079
R12      0080
R13      0081    0101
R14      0082
R15      0083
R2       0070
R3       0071
R4       0072
R5       0073
R6       0074
R7       0075    0152  0153  0160  0161  0162  0163  0164
R8       0076    0105  0134  0135  0138  0140  0147  0153  0156  0157
                 0158  0160  0162  0174
R9       0077
RETBUF           0060
RR               0064  0176
RUBOUT           0057  0144
RWP              0062

THERE ARE   0041  SYMBOLS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV

SHEET

REV STATUS OF SHEETS

REV

SHEET

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, GTHEX, INPUT HEXADECIMAL NUMBER-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945357-9901

SCALE | REV. | SHEET 1 of 6

TI-0023-C

```
0003                *    PROCEDURE GET HEX VALUE(LD BLNK,VALUE,ERROR,
0004                *       TERM CHAR,NO VALUE INPUT);
0005                *       IF LD BLNK THEN CALL GET FIELD(PTR,
0006                *          ALLOW LEADING BLANKS);
0007                *       ELSE CALL GET FIELD(PTR,NO LEADING BLANKS)
0008                *       PTR SAVE =PTR;
0009                *       CHAR COUNT = PTR.CHR CNT;
0010                *       PTR=PTR+1;
0011                *       TERM CHAR = PTR.TERM CHAR;
0012                *       PTR = PTR+1;
0013                *       IF TERM CHAR='ESC' THEN RETURN(ERROR);
0014                *       IF CHAR COUNT=0 THEN RETURN(NO VALUE INPUT);
0015                *       VALUE = 0;
0016                *       DO WHILE CHAR COUNT .GT. 0 ;
0017                *          CHAR = PTR.CHAR
0018                *          PTR=PTR+1;          3
0019                *          I=0;
0020                *          DO WHILE I<16 AND CHAR.NE.CLIST(I);
0021                *             I=I+1
0022                *          END;
0023                *          IF  I .EQ.16 THEN DO;
0024                *             CALL ERROR(HEX CONV);
0025                *             RETURN(ERROR)
0026                *          IF VLAUE > 'FFF' THEN RETURN(ERROR);
0027                *          VALUE = VALUE*16 + I;
0028                *          CHAR COUNT = CHAR COUNT -1;
0029                *       END
0030                *       PTR = PTR SAVE;
0031                *       CALL RETBUF(PTR);
0032                *    END GET HEX VALUE;
0033                         IDT  'GTHEX'
0034                * TITLE:    GETHEX
0035                *           INPUT HEX VALUE
0036                * REVISION:
0037                *           ORIGINAL
0038                * COMPUTER: 990,ASM
0039                * ABSTRACT: GETHEX WILL INPUT A HEXADECIMAL VALUE
0040                *           FROM THE KEYBOARD AND CONVERT IT TO
0041                *           BINARY. SEPARATE ENTRY POINTS ALLOW/EXCLUDE
0042                *           LEADING BLANKS. AN ERROR EXIT IS TAKEN IF
0043                *           AN INVALID VALUE IS INPUT OR NO VALUE IS
0044                *           INPUT.  THE TERMINATING CHAR IS RETURNED.
0045                * CALLING SEQUENCE:
0046                *           BL @GETHEX  = SKIPS LEADING BLANKS
0047                *                       =
0048                *           BL @GETHXN  = DOES NOT ALLOW LEADING BLANKS
0049                *             RETURN =
0050                *                R10 = VALUE
0051                *                R900 = TERMINATOR
0052                *                CALL+2 = ERROR OR ESC
0053                *                CALL+6 = NO VALUE RETURNED (IMMEDIATE CR
0054                *                         OR BLANK OR COMMA)
0055                *                CALL+4 = VALUE RETURNED
0056                *
```

```
0057                    *       REF'S AND DEF'S
0058                    *
0059                            DEF   GETHEX
0060                            DEF   GETHXN
0061                            REF   GETFLD
0062                            REF   GETFLN
0063                            REF   ERROR
0064                            REF   ESC
0065                            REF   CLST
0066                            REF   GETBUF
0067                            REF   RETBUF
0068                            REF   LWP
0069                            REF   RWP
0070                            REF   ACL
0071                            REF   RR
0072                    *
0073                    *WORKSPACE REGISTER DEFINITIONS
0074                    *
0075          0000      R0      EQU   0
0076          0001      R1      EQU   1
0077          0002      R2      EQU   2
0078          0003      R3      EQU   3
0079          0004      R4      EQU   4
0080          0005      R5      EQU   5     3
0081          0006      R6      EQU   6
0082          0007      R7      EQU   7
0083          0008      R8      EQU   8
0084          0009      R9      EQU   9
0085          000A      R10     EQU   10
0086          000B      R11     EQU   11
0087          000C      R12     EQU   12
0088          000D      R13     EQU   13
0089          000E      R14     EQU   14
0090          000F      R15     EQU   15
0091                    *
0092          0100      DP00    EQU   >0100
0093                    *
0094                    *       REGISTER ASSIGNMENTS
0095                    *
0096                    *       R10   RUNNING PTR IN CHAR STRING BUFFER
0097                    *       R9    TERM CHAR
0098                    *       R8    SAVE FOR PTR TO CHAR STRING BUFFER
0099                    *       R7    CHAR COUNT
0100                    *       R6    VALUE TO BE RETURNED
0101                    *       R5    CHAR
0102                    *       R4    HEX ARRAY COUNT
0103                    *       R3    TEMP
0104                    *                                  1=PROCEDURE GET HEX VALUE(LD B
0105                    *                                  2=TERM CHAR,NO VALUE INPUT)!
0106          0000'     GETHXN  EQU   $
0107  0000    04C9              CLR   R9                    LD BLNK = FALSE
0108  0002    1001              JMP   GET010
0109          0004'     GETHEX  EQU   $
0110  0004    0709              SETO  R9                    LD BLNK = TRUE
0111          0006'     GET010  EQU   $
```

```
0112                    *     *ALLOC,COPY,LINK
0113  0006  0420               BLWP  @ACL
      0008  0000
0114                    *                                  2=IF LD BLNK THEN CALL GET FIE
0115                    *                                  3=ALLOW LEADING BLANKS);
0116  000A  C249               MOV   R9,R9
0117  000C  1303               JEQ   GET020
0118  000E  06A0               BL    @GETFLD
      0010  0000
0119  0012  1002               JMP   GET030
0120                    *                                  2=ELSE CALL GET FIELD(PTR,NO L
0121        0014' GET020  EQU   $
0122  0014  06A0               BL    @GETFLN
      0016  0000
0123        0018' GET030  EQU   $
0124                    *                                  2=PTR SAVE =PTR;
0125  0018  C20A               MOV   R10,R8
0126                    *                                  2=CHAR COUNT = PTR.CHR CNT;
0127                    *                                  2=PTR=PTR+1;
0128  001A  D1FA               MOVB  *R10+,R7
0129  001C  0987               SRL   R7,8
0130                    *                                  2=TERM CHAR = PTR.TERM CHAR;
0131                    *                                  2=PTR = PTR+1;
0132  001E  D27A               MOVB  *R10+,R9
0133                    *                                  2=IF TERM CHAR='ESC' THEN RETU
0134  0020  9809               CB    R9,@ESC
      0022  0000
0135  0024  1603               JNE   GET035
0136  0026  066D               DECT  @R11*2(R13)
      0028  0016
0137  002A  101F               JMP   GET070
0138        002C' GET035  EQU   $
0139                    *                                  2=IF CHAR COUNT=0 THEN RETURN(
0140  002C  C1C7               MOV   R7,R7
0141  002E  1603               JNE   GET038
0142  0030  05ED               INCT  @R11*2(R13)
      0032  0016
0143  0034  101A               JMP   GET070
0144                    *                                  2=VALUE = 0;
0145        0036' GET038  EQU   $
0146  0036  04C6               CLR   R6
0147                    *                                  2=DO WHILE CHAR COUNT .GT. 0 ;
0148        0038' GET040  EQU   $
0149                    *                                  3=CHAR = PTR.CHAR
0150                    *                                  3=PTR=PTR+1;
0151  0038  D17A               MOVB  *R10+,R5
0152                    *                                  3=I=0;
0153  003A  04C4               CLR   R4
0154                    *                                  3=DO WHILE I<16 AND CHAR.NE.CL
0155        003C' GET050  EQU   $
0156  003C  0203               LI    R3,CLST
      003E  0000
0157  0040  A0C4               A     R4,R3
0158  0042  94C5               CB    R5,*R3
0159  0044  130B               JEQ   GET060
```

```
0160                    *                    3          4=I=I+1
0161  0046  0584               INC   R4
0162                    *                               3=END;
0163  0048  0284               CI    R4,16
      004A  0010
0164  004C  16F7               JNE   GET050
0165                    *                               3=IF  I .EQ.16 THEN DO;
0166                    *                               4=CALL ERROR(HEX CONV);
0167        004E' GET055 EQU   $
0168  004E  020A               LI    R10,DP00
      0050  0100
0169  0052  06A0               BL    @ERROR
      0054  0000
0170                    *                               4=RETURN(ERROR)
0171  0056  066D               DECT  @R11*2(R13)        FORCE RETURN TO CALL+2
      0058  0016
0172  005A  1007               JMP   GET070
0173        005C' GET060 EQU   $
0174                    *                               3=IF VLAUE > 'FFF' THEN RETURN
0175  005C  0286               CI    R6,>FFF
      005E  0FFF
0176  0060  1BF6               JH    GET055
0177                    *                               3=VALUE = VALUE*16 + I;
0178  0062  0A46               SLA   R6,4
0179  0064  A184               A     R4,R6
0180                    *                               3=CHAR COUNT = CHAR COUNT -1;
0181  0066  0607               DEC   R7
0182                    *                               2=END
0183  0068  16E7               JNE   GET040
0184        006A' GET070 EQU   $
0185                    *                               2=PTR = PTR SAVE;
0186  006A  C288               MOV   R8,R10
0187                    *                               2=CALL RETBUF(PTR);
0188                    *     *RETBUF
0189  006C  0420               BLWP  @RETBUF
      006E  0000
0190                    *                               1=END GET HEX VALUE;
0191  0070  CB49               MOV   R9,@R9*2(R13)
      0072  0012
0192  0074  CB46               MOV   R6,@R10*2(R13)
      0076  0014
0193  0078  0420               BLWP  @RR
      007A  0000
0194  007C  05CB               INCT  R11
0195  007E  045B               RT
0196                          END
0000 ERS
```

960 - 980 CONCORDANCE

$		0106	0109	0111	0121	0123	0138	0145	0148	0155
		0167	0173	0184						
ACL		0070	0113							
CLST		0065	0156							
DP00	0092	0168								
ERROR		0063	0169							
ESC		0064	0134							
GET010	0111	0108								
GET020	0121	0117								
GET030	0123	0119								
GET035	0138	0135								
GET038	0145	0141								
GET040	0148	0183								
GET050	0155	0164								
GET055	0167	0176								
GET060	0173	0159								
GET070	0184	0137	0143	0172						
GETBUF		0066								
GETFLD		0061	0118							
GETFLN		0062	0122							
GETHEX	0109	0059								
GETHXN	0106	0060								
LWP		0068								
R0	0075									
R1	0076									
R10	0085	0125	0128	0132	0151	0168	0186	0192		
R11	0086	0136	0142	0171	0194					
R12	0087									
R13	0088	0136	0142	0171	0191	0192				
R14	0089									
R15	0090									
R2	0077									
R3	0078	0156	0157	0158						
R4	0079	0153	0157	0161	0163	0179				
R5	0080	0151	0158							
R6	0081	0146	0175	0178	0179	0192				
R7	0082	0128	0129	0140	0140	0181				
R8	0083	0125	0186							
R9	0084	0107	0110	0116	0116	0132	0134	0191	0191	
RETBUF		0067	0189							
RR		0071	0193							
RWP		0069								

THERE ARE 0041 SYMBOLS

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV	
SHEET	
REV STATUS OF SHEETS	REV
	SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DATE
DWN	
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, HXARTH, HEXADECIMAL ARITHMETIC-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945358-9901

SCALE	REV.	SHEET 1 of 7

TI-0923-C

```
0003          *   PROCEDURE HXAR (CPL POINTER);
0004          *      DECLARE (SUM, DIFF, OP1, OP2) FIXED (16);
0005          *      DECLARE (1 COMMAND PARAMETER LIST,
0006          *                  3 CPL CONTROL (CPL POINTER),
0007          *                     5 NUMBER PARMS FIXED (8),
0008          *                     5 PARM PRESENT (8) BIT (1),
0009          *                     5 FIRST PARM FIXED (8));
0010          *      DECLARE NUMERIC PARAMETER FIXED (16)
0011          *           CONTROL (PARM POINTER);
0012          *      DECLARE (1 CHARACTER PARAMETER,
0013          *                  3 CHAR PARM CONTROL (PARM POINTER),
0014          *                     5 STRING LENGTH FIXED (8),
0015          *                     5 PARM STRING (1) CHARACTER (1));
0016          *      DECLARE (CPL POINTER, PARM POINTER) POINTER;
0017          *      OP1 = 0;
0018          *      OP2 = 0;
0019          *      PARM POINTER = ADDR (FIRST PARM);
0020          *      IF PARM PRESENT (1) = 1 THEN DO;
0021          *         OP1 = NUMERIC PARAMETER;
0022          *         PARM POINTER = PARM POINTER + 2;
0023          *         END;
0024          *      IF PARM PRESENT (2) = 1
0025          *         THEN OP2 = NUMERIC PARAMETER;
0026          *      SUM = OP1 + OP2;
0027          *      DIFF = OP1 - OP2;
0028          *      CALL PRNTC (ADDR (PTSUM));
0029          *            /* PRINT HEADING FOR SUM */
0030          *      CALL PRNTHX (SUM);
0031          *            /* PRINT VALUE OF SUM IN HEX */
0032          *      CALL HXARPD (SUM);
0033          *            /* PRINT VALUE OF SUM IN DECIMAL */
0034          *      CALL PRNTC (ADDR (PTDIFF));
0035          *            /* PRINT HEADING FOR DIFF */
0036          *      CALL PRNTHX (DIFF);
0037          *            /* PRINT VALUE FOR DIFF IN HEX */
0038          *      CALL HXARPD (DIFF);
0039          *            /* PRINT VALUE FOR DIFF IN DECIMAL */
0040          *      RETURN;
0041          *
0042          *      PROCEDURE HXARPD (VAL); /* PRINT DECIMAL */
0043          *         DECLARE VAL FIXED (16);
0044          *         DECLARE DEC STRING CHARACTER (6);
0045          *         CALL CBD (VAL, DEC STRING);
0046          *               /* CONVERT TO ASCII STRING */
0047          *         CALL PRINT (DEC STRING, 6);
0048          *               /* PRINT THE ASCII STRING */
0049          *         RETURN;
0050          *      END HXARPD;
0051          *   END HXAR;
0052               IDT  'HXARTH'
```

```
0054                    * TITLE:    HXARTH
0055                    *           HEXADECIMAL ARITHMETIC
0056                    * REVISION:
0057                    *           ORIGINAL
0058                    * COMPUTER: 990
0059                    * ABSTRACT:
0060                    *           THE SUM AND DIFFERENCE OF THE FIRST TWO
0061                    *           PARAMETERS ARE COMPUTED AND PRINTED IN BOTH
0062                    *           HEXADECIMAL AND DECIMAL FORMATS.
0063                    * CALLING SEQUENCE:
0064                    *           CALLED FROM COMMAND STRING PROCESSOR.
0065                    *
0066                    *
0067                    *WORKSPACE REGISTER DEFINITIONS
0068                    *
0069          0000      R0      EQU  0
0070          0001      R1      EQU  1
0071          0002      R2      EQU  2
0072          0003      R3      EQU  3
0073          0004      R4      EQU  4
0074          0005      R5      EQU  5
0075          0006      R6      EQU  6
0076          0007      R7      EQU  7
0077          0008      R8      EQU  8
0078          0009      R9      EQU  9
0079          000A      R10     EQU  10
0080          000B      R11     EQU  11
0081          000C      R12     EQU  12
0082          000D      R13     EQU  13
0083          000E      R14     EQU  14
0084          000F      R15     EQU  15
0085                    *
0086          0001      PRMPST  EQU  1
0087                    *                                  1*PROCEDURE HXAR (CPL POINTER)
0088                    *                                  2*DECLARE (SUM, DIFF, OP1, OP2
0089                    *                                  2*DECLARE (1 COMMAND PARAMETER
0090                    *                                  3*          3 CPL CONTROL (CPL
0091                    *                                  4*          5 NUMBER PARMS FIXE
0092                    *                                  4*          5 PARM PRESENT (8)
0093                    *                                  4*          5 FIRST PARM FIXED
0094                    *                                  2*DECLARE NUMERIC PARAMETER FI
0095                    *                                  2*      CONTROL (PARM POINTER);
0096                    *                                  2*DECLARE (1 CHARACTER PARAMET
0097                    *                                  3*          3 CHAR PARM CONTROL
0098                    *                                  4*          5 STRING LENGTH FIX
0099                    *                                  4*          5 PARM STRING (1) C
0100                    *                                  2*DECLARE (CPL POINTER, PARM P
0101                            DEF  HXAR
0102                            REF  PTSUM,PTDIFF
0103                            REF  PRNTC,PRNTHX,PRINT
0104                            REF  SVCALT
0105                            REF  CBD,CBDSTG
0106          0000!     HXAR    EQU  $
0107    0000  C08B              MOV  R11,2              SAVE RETURN, CPL POINTER
```

```
0108                    *                       2=OP1 = 0;
0109  0002  04C7              CLR   R7           DEFAULT OPERANDS TO ZERO.
0110                    *                       2=OP2 = 0;
0111  0004  04C8              CLR   R8
0112                    *                       2=PARM POINTER = ADDR (FIRST P
0113  0006  D1AA              MOVB  @PRMPST(R10),R6   ISOLATE PARM PRESENCE BITS
      0008  0001
0114  000A  05CA              INCT  R10
0115                    *                       2=IF PARM PRESENT (1) = 1 THEN
0116  000C  0A16              SLA   R6,1          IF FIRST PARAMETER NOT PRESEN
0117  000E  1701              JNC   HXAR10
0118                    *                       3=OP1 = NUMERIC PARAMETER;
0119                    *                       3=PARM POINTER = PARM POINTER
0120  0010  C1FA              MOV   *R10+,R7
0121                    *                       3=END;
0122        0012' HXAR10  EQU   $
0123                    *                       2=IF PARM PRESENT (2) = 1
0124  0012  0A16              SLA   R6,1          IF PARM 2 NOT PRESENT
0125  0014  1701              JNC   HXAR20
0126                    *                       3=THEN OP2 = NUMERIC PARAMETER
0127  0016  C21A              MOV   *R10,R8
0128        0018' HXAR20  EQU   $
0129                    *                       2=SUM = OP1 + OP2;
0130  0018  C187              MOV   R7,R6         SUM IN R6
0131  001A  A188              A     R8,R6
0132                    *                       2=DIFF = OP1 - OP2;
0133  001C  61C8              S     R8,R7         DIFF IN R7
0134                    *                       2=CALL PRNTC (ADDR (PTSUM));
0135                    *                       4=/* PRINT HEADING FOR SUM */
0136  001E  020A              LI    R10,PTSUM
      0020  0000
0137  0022  06A0              BL    @PRNTC
      0024  0000
0138  0026  1013              JMP   HXAR90        **** ERROR EXIT
0139                    *                       2=CALL PRNTHX (SUM);
0140                    *                       4=/* PRINT VALUE OF SUM IN HEX
0141  0028  C286              MOV   R6,R10
0142  002A  06A0              BL    @PRNTHX
      002C  0000
0143  002E  100F              JMP   HXAR90        **** ERROR EXIT
0144                    *                       2=CALL HXARPD (SUM);
0145                    *                       4=/* PRINT VALUE OF SUM IN DEC
0146  0030  C006              MOV   R6,R0
0147  0032  06A0              BL    @HXARPD
      0034  0050'
0148                    *                       2=CALL PRNTC (ADDR (PTDIFF));
0149                    *                       4=/* PRINT HEADING FOR DIFF */
0150  0036  020A              LI    R10,PTDIFF
      0038  0000
0151  003A  06A0              BL    @PRNTC
      003C  0024'
0152  003E  1007              JMP   HXAR90        **** ERROR EXIT
0153                    *                       2=CALL PRNTHX (DIFF);
0154                    *                       4=/* PRINT VALUE FOR DIFF IN H
0155  0040  C287              MOV   R7,R10
```

```
0156  0042  06A0          BL    @PRNTHX
      0044  002C'
0157  0046  1003          JMP   HXAR90          ****  ERROR EXIT
0158                 *                          2=CALL HXARPD (DIFF)
0159                 *                          4=/* PRINT VALUE FOR DIFF IN D
0160  0048  C007          MOV   R7,R0
0161  004A  06A0          BL    @HXARPD
      004C  0050'
0162                 *                          2=RETURN
0163        004E' HXAR90  EQU   $
0164  004E  0452          B     *R2
```

```
 0166                    *                                  2=
 0167                    *                                  2=PROCEDURE HXARPD (VAL); /* P
 0168                    *                                  3=DECLARE VAL FIXED (16);
 0169                    *                                  3=DECLARE DEC STRING CHARACTER
 0170          0050'  HXARPD EQU   $
 0171  0050    C100          MOV   R11,R4                   SAVE RETURN
 0172                    *                                  3=CALL CBD (VAL, DEC STRING);
 0173                    *                                  5=/* CONVERT TO ASCII STRING *
 0174  0052    020A          LI    R10,CBD
       0054    0000
 0175  0056    0420          BLWP  @SVCALT
       0058    0000
 0176                    *                                  3=CALL PRINT (DEC STRING, 6);
 0177                    *                                  5=/* PRINT THE ASCII STRING */
 0178  005A    020A          LI    R10,CBDSTG
       005C    0000
 0179  005E    0209          LI    R9,6
       0060    0006
 0180  0062    06A0          BL    @PRINT
       0064    0000
 0181  0066    10F3          JMP   HXAR90                   **** ERROR EXIT
 0182                    *                                  3=RETURN;
 0183  0068    0454          B     *R4
 0184                    *                                  2=END HXARPD;
 0185                    *                                  1=END HXAR;
 0186                          END
0000 ERS
```

```
960 = 980  CONCORDANCE
$                       0106  0122  0128  0163  0170
CBD                     0105  0174
CBDSTG                  0105  0178
HXAR        0106        0101
HXAR10      0122        0117
HXAR20      0128        0125
HXAR90      0163        0138  0143  0152  0157  0181
HXARPD      0170        0147  0161
PRINT                   0103  0180
PRMPST      0086        0113
PRNTC                   0103  0137  0151
PRNTHX                  0103  0142  0156
PTDIFF                  0102  0150
PTSUM                   0102  0136
R0          0069        0146  0160
R1          0070
R10         0079        0113  0114  0120  0127  0136  0141  0150  0155  0174
                        0178
R11         0080        0107  0171
R12         0081
R13         0082
R14         0083
R15         0084
R2          0071        0164
R3          0072
R4          0073        0171  0183
R5          0074
R6          0075        0113  0116  0124  0130  0131  0141  0146
R7          0076        0109  0120  0130  0133  0155  0160
R8          0077        0111  0127  0131  0133
R9          0078        0179
SVCALT                  0104  0175

THERE ARE    0031  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV
SHEET

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, IC, INSPECT CRU PROCESSOR-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945359-9901

SCALE	REV.	SHEET 1 of 4

TI-0023-C

```
0003                    IDT  'IC'
0004                * TITLE:    ICP
0005                *           INSPECT CRU PROCESSOR
0006                * REVISION:
0007                *           ORIGINAL
0008                * COMPUTER: 990,ASM
0009                * ABSTRACT: THIS ROUTINE PROCESSES THE IC CMD.  IT FIRST
0010                *           CHECKS TO MAKE SURE THAT LOWER LIM < UPPER
0011                *           LIM AND THAT THE CRU ADDRESSES ARE LEGITIMATE.
0012                *           IT THEN CALLS THE PRINT CRU RANGE ROUTINE TO
0013                *           PRINT THE CRU INPUT VALUES.
0014                *
0015                * CALLING SEQUENCE:
0016                *           R10=PTR TO CMD PARM LIST
0017                *           CMD PARM LIST = #PARMS/PARM PRES LIST
0018                *                           PARM VALUES
0019                *           BL   @ICP
0020                *           DESTROYS R0,R1,R2,R3,R5,R9,R8,R14,R12,R15
0021                *
0022                *  REFS,DEFS,EQU'S
0023                *
0024                      DEF  ICP
0025                      REF  PRTCRU - PRINT CRU RANGE ROUTINE
0026                      REF  RANGEX - RANGE EXTRACTER
0027        0008  PRBITS  EQU  8
0028        2000  CRNUM   EQU  >2000
0029        0004  ON16    EQU  4
0030        0010  CRWIDT  EQU  >10
0031        0000  R0      EQU  0
0032        0005  R5      EQU  5
0033        0008  R8      EQU  8
0034        0009  R9      EQU  9
0035        000A  R10     EQU  10
0036        000B  R11     EQU  11
0037        000F  R15     EQU  15
```

```
0039                    ICP
0040  0000  C3CB               MOV  R11,R15          SAVE RT = 1ST LEVEL
0041                    *
0042                    *  GO GET LIMITS
0043                    *
0044  0002  0208               LI   R8,CRNUM
      0004  2000
0045  0006  06A0               BL   @RANGEX
      0008  0000
0046  000A  045F               B    *R15
0047  000C  0849               SRA  R9,ON16       TRUNCATE
0048  000E  0A49               SLA  R9,ON16       BACK
0049                    *
0050                    *  PRINT CRU RANGE AND RETURN
0051                    *
0052  0010  0208   ICP050 LI   R8,CRWIDT          R8=CRU BIT WIDTH
      0012  0010
0053  0014  06A0               BL   @PRTCRU
      0016  0000
0054  0018  1000               NOP                FALL THRU' ON ERROR
0055  001A  045F               B    *R15
0056                          END
0000 ERS
```

```
960 - 980  CONCORDANCE
CRNUM    0028       0044
CRWIDT   0030       0052
ICP      0039       0024
ICP050   0052
ON16     0029       0047  0048
PRBITS   0027
PRTCRU              0025  0053
R0       0031
R10      0035
R11      0036       0040
R15      0037       0040  0046  0055
R5       0032
R8       0033       0044  0052
R9       0034       0047  004A
RANGEX              0026  0045

THERE ARE   0015  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV
SHEET

REV STATUS OF SHEETS
REV
SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ANGLES ±1° 3 PLACE DECIMAL ±.010 2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS INCORPORATED
Equipment Group Dallas, Texas

SDP, IM, INSPECT MEMORY-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945360-9901

SCALE REV. SHEET 1 of 4

TI-0023-C

```
0003                    IDT  'IM'
0004                * TITLE:    IM
0005                *           INSPECT MEMORY COMMAND PROCESSOR
0006                * REVISION:
0007                *           ORIGINAL
0008                * COMPUTER: 990,ASM
0009                * ABTRACT:  THIS ROUTINE PROCESSES THE INSPECT MEMORY(IM)
0010                *           COMMAND BY SETTING UP AND CALLING THE PRINT
0011                *           MEMORY RANGE ROUTINE (PRTMEM).  IT VERIFIES THA
0012                *           LOWER LIMIT IS < UPPER LIMIT.
0013                * CALLING SEQUENCE:
0014                *           BL   @IMP
0015                *           R10=PTR TO CMD PARM LIST
0016                *           CMD PARM LIST = #PARMS/PARM PRES BITS
0017                *                           PARM VALUES
0018                *           DESTROYS R0,R1,R2,R4,R5,R8,R9,R12,R14,R15
0019                *
0020                *
0021                *  REFS,DEFS,EQUS
0022                      DEF  IMP
0023                      REF  PRTMEM - PRINT MEMORY RANGE ROUTINE
0024                      REF  ERROR  - ERROR PROCESS
0025                      REF  RANGEX - RANGE EXTRACTER ROUTINE
0026      1101  LIMERR EQU  >1101
0027      0000  R0     EQU  0
0028      0005  R5     EQU  5
0029      0008  R8     EQU  8
0030      0009  R9     EQU  9
0031      000A  R10    EQU  10
0032      000B  R11     EQU  11
0033      000F  R15    EQU  15
```

```
 0035  0000  C3CB  IMP     MOV   R11,R15          SAVE RETURN - LEVEL 1
 0036                *
 0037                *  GO GET LIMITS
 0038                *
 0039  0002  04C8          CLR   R8               NO MAX ADDRESS CHECK
 0040  0004  06A0          BL    @RANGEX
       0006  0000
 0041  0008  1003          JMP   IMP050
 0042                *
 0043                *  PRINT MEMORY AND EXIT
 0044                *
 0045  000A  06A0  IMP040  BL    @PRTMEM
       000C  0000
 0046  000E  1000          NOP
 0047  0010  045F  IMP050  B     *R15
 0048                      END
0000 ERS
```

```
960 = 980  CONCORDANCE
ERROR               0024
IMP       0035      0022
IMP040    0045
IMP050    0047      0041
LIMERR    0026
PRTMEM              0023  0045
R0        0027
R10       0031
R11       0032      0035
R15       0033      0035  0047
R5        0028
R8        0029      0039
R9        0030
RANGEX              0025  0040

THERE ARE    0014  SYMBOLS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																													
SHEET																													

REV STATUS OF SHEETS	REV		
	SHEET		

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, INSPSS, INSPECT SNAPSHOT-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945362-9901
SCALE	REV.	SHEET 1 of 4

TI-0023-C

```
0003                *   PROCEDURE INSPECT SNAPSHOT( CPL);
0004                *      /* INSPECT SNAPSHOT PRINTS THE DISPLAY
0005                *         FOR A RANGE OF DEFINED SNAPSHOTS.
0006                *      */
0007                *      CALL RANGE(LOW,HIGH,4,ERROR);
0008                *      IF .NOT. ERROR THEN DO;
0009                *         DO UNTIL ALL SNAPS COMPLETE;
0010                *            CALL PRINT SNAPSHOT(LOW);
0011                *            LOW = LOW + 1;
0012                *            IF LOW .GT. HIGH  THEN SIGNAL
0013                *               ALL SNAPS COMPLETE;
0014                *         END;
0015                *         ALL SNAPS COMPLETE:
0016                *      END;
0017                *   END INSPECT SNAPSHOT;
0018                       IDT  'INSPSS'
0019                * TITLE:    ISS
0020                *           INSPECT SNAPSHOT
0021                * REVISION:
0022                *           ORIGINAL
0023                * COMPUTER: 990,ASM
0024                * ABSTRACT: THIS ROUTINE PRINTS THE SNAPSHOT DISPLAY
0025                *           ASSOCIATED WITH ONE OR MORE SNAPSHOTS
0026                * CALLING SEQUENCE:
0027                *           BL #ISS
0028                *           R10 = COMMAND PARAMETER LIST
0029                *
0030                *      REF'S AND DEF'S
0031                *
0032                       DEF  ISS
0033                       REF  NUMSNP
0034                       REF  SNPENT
0035                       REF  SNPTAB
0036                       REF  PRTSS
0037                       REF  RANGEX
0038                       REF  GETBUF
0039                       REF  RETBUF
0040                       REF  LWP
0041                       REF  RWP
0042                       REF  ACL
0043                       REF  RR
0044                *
0045                *WORKSPACE REGISTER DEFINITIONS
0046                *
0047          0000  R0       EQU  0
0048          0001  R1       EQU  1
0049          0002  R2       EQU  2
0050          0003  R3       EQU  3
0051          0004  R4       EQU  4
0052          0005  R5       EQU  5
0053          0006  R6       EQU  6
0054          0007  R7       EQU  7
0055          0008  R8       EQU  8
0056          0009  R9       EQU  9
```

```
 0057          000A  R10     EQU  10
 0058          000B  R11     EQU  11
 0059          000C  R12     EQU  12
 0060          000D  R13     EQU  13
 0061          000E  R14     EQU  14
 0062          000F  R15     EQU  15
 0063                *
 0064          0000' ISS     EQU  $                                  
 0065                *                                              1*PROCEDURE INSPECT SNAPSHOT(
 0066                *     *ALLOC,COPY,LINK
 0067  0000  0420           BLWP @ACL
       0002  0000
 0068                *                                              2*/* INSPECT SNAPSHOT PRINTS T
 0069                *                                              2*   FOR A RANGE OF DEFINED SN
 0070                *                                              2**/
 0071                *                                              2*CALL RANGE(LOW,HIGH,4,ERROR)
 0072                *                                              2*IF .NOT. ERROR THEN DO;
 0073  0004  0208           LI   R8,NUMSNP
       0006  0000
 0074  0008  06A0           BL   @RANGEX
       000A  0000
 0075  000C  1008           JMP  ISS020
 0076                *                                              3*DO UNTIL ALL SNAPS COMPLETE;
 0077  000E  C1CA           MOV  R10,R7                             HIGH LIMIT
 0078  0010  C189           MOV  R9,R6                              LOW  LIMIT
 0079          0012' ISS010 EQU  $
 0080                *                                              4*CALL PRINT SNAPSHOT(LOW);
 0081  0012  C286           MOV  R6,R10
 0082  0014  06A0           BL   @PRTSS
       0016  0000
 0083                *                                              4*LOW = LOW + 1;
 0084  0018  0586           INC  R6
 0085                *                                              4*IF LOW .GT. HIGH  THEN SIGNA
 0086                *                                              5*ALL SNAPS COMPLETE;
 0087  001A  81C6           C    R6,R7
 0088  001C  12FA           JLE  ISS010
 0089                *                                              3*END;
 0090                *                                              3*ALL SNAPS COMPLETE;
 0091                *                                              2*END;
 0092          001E' ISS020 EQU  $
 0093                *                                              1*END INSPECT SNAPSHOT;
 0094                *     *LINK TO PREV WKSP, RET CURR WKSP
 0095  001E  0420           BLWP @RR
       0020  0000
 0096  0022  045B           RT
 0097                       END
0000 ERS
```

```
990 - 990  CONCORDANCE
$                       0063   0078   0091
ACL                     0041   0066
GETBUF                  0037
ISS         0063        0031
ISS010      0078        0087
ISS020      0091        0074
LWP                     0039
NUMSNP                  0032   0072
PRTSS                   0035   0081
R0          0046
R1          0047
R10         0056        0076   0080
R11         0057
R12         0058
R13         0059
R14         0060
R15         0061
R2          0048
R3          0049
R4          0050
R5          0051
R6          0052        0077   0080   0083   0086
R7          0053        0076   0086
R8          0054        0072
R9          0055        0077
RANGEX                  0036   0073
RETBUF                  0038
RR                      0042   0094
RWP                     0040
SNPENT                  0033
SNPTAB                  0034

THERE ARE    0031   SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV STATUS OF SHEETS

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, INWKSP, INSPECT WORKSPACE -PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945363-9901
SCALE	REV.	SHEET 1 of 5

TI-0023-C

```
0003                *    PROCEDURE INSPECT WORKSPACE(CPL);
0004                *       /* INSPECT WORKSPACE DISPLAYS THE
0005                *          CONTENTS OF ONE OR MORE OF THE
0006                *          USERS WORKSPACE REGISTERS. INPUT
0007                *          IS A REGISTER OR RANGE OF REGISTERS
0008                *          TO BE DISPLAYED (NULL DEFAULTS TO
0009                *          ALL REGISTERS.
0010                *       */
0011                *       CALL RANGE(CPL,LOW,HIGH,'F',ERROR);
0012                *       IF .NOT. ERROR THEN DO;
0013                *          RCNT = 8;  MAX # REGS ON 1ST LINE
0014                *          DO I = LOW TO HIGH  /* REGISTER RANGE */
0015                *          CALL PRINT('CRLF');
0016                *             REG = CHAR LIST(I);
0017                *             CALL PRINT(REG NO STR);
0018                *             VALUE = USER WORKSPACE(I);
0019                *             CALL PRINT HEX(VALUE);
0020                *             RCNT = RCNT-1;
0021                *             IF RCNT .EQ. 0 THEN CALL PRINT('CRLF');
0022                *             I=I+1  /* NEXT REGISTER */
0023                *          END
0024                *       END
0025                *    END INSPECT WORKSPACE
0026                         IDT  'INWKSP'
0027                * TITLE:INWKSP
0028                *        INSPECT WORKSPACE
0029                * REVISION:
0030                *             ORIGINAL
0031                * COMPUTER: 990,ASM
0032                * ABSTRACT: INSPECT WORKSPACE DISPLAYS ONE OR MORE
0033                *             OF THE USER'S WORKSPACE REGISTERS.
0034                * CALLING SEQUENCE:
0035                *             BL @IWP
0036                *             R10 POINTS TO COMMAND PARAMETER LIST
0037                *
0038                *                                  1=PROCEDURE INSPECT WORKSPACE(
0039                *                                  2=/* INSPECT WORKSPACE DISPLAY
0040                *                                  2=    CONTENTS OF ONE OR MORE O
0041                *                                  2=    USERS WORKSPACE REGISTERS
0042                *                                  2=    IS A REGISTER OR RANGE OF
0043                *                                  2=    TO BE DISPLAYED (NULL DEF
0044                *                                  2=    ALL REGISTERS.
0045                *                                  2=*/
0046                *       REF'S AND DEF'S
0047                *
0048                         DEF  IWP
0049                         REF  PRCRLF
0050                         REF  PRNTC
0051                         REF  PRNTHX
0052                         REF  RANGEX
0053                         REF  RGSTR
0054                         REF  RGSTRB
0055                         REF  GETBUF
0056                         REF  RETBUF
```

```
0057                        REF  LWP
0058                        REF  RWP
0059                        REF  ACL
0060                        REF  RR
0061                        REF  CLST
0062                        REF  USRWP
0063                  *
0064                  *WORKSPACE REGISTER DEFINITIONS
0065                  *
0066        0000  R0    EQU  0
0067        0001  R1    EQU  1
0068        0002  R2    EQU  2
0069        0003  R3    EQU  3
0070        0004  R4    EQU  4
0071        0005  R5    EQU  5
0072        0006  R6    EQU  6
0073        0007  R7    EQU  7
0074        0008  R8    EQU  8
0075        0009  R9    EQU  9
0076        000A  R10   EQU  10
0077        000B  R11   EQU  11
0078        000C  R12   EQU  12
0079        000D  R13   EQU  13
0080        000E  R14   EQU  14
0081        000F  R15   EQU  15
0082              *
0083        0000' IWP   EQU  $
0084              *     *ALLOC,COPY,LINK
0085  0000  0420       BLWP @ACL
      0002  0000
0086              *                                  2=CALL RANGE(CPL,LOW,HIGH,'F',
0087  0004  0208       LI   R8,>10
      0006  0010
0088  0008  06A0       BL   @RANGEX
      000A  0000
0089  000C  1022       JMP  IWP030
0090              *                                  2=IF .NOT. ERROR THEN DO;
0091              *                                  3=RCNT = 8;  MAX # REGS ON 1ST
0092  000E  0205       LI   R5,8
      0010  0008
0093              *                                  3=DO I = LOW TO HIGH  /* REGIS
0094  0012  C1CA       MOV  R10,R7                     UPPER REGISTER LIMIT
0095  0014  C189       MOV  R9,R6                      LOWER REGISTER LIMIT
0096              *                                  3=CALL PRINT('CRLF');
0097  0016  06A0       BL   @PRCRLF
      0018  0000
0098  001A  1000       NOP
0099        001C' IWP010 EQU  $
0100              *                                  4=REG = CHAR LIST(I);
0101  001C  0204       LI   R4,CLST       3
      001E  0000
0102  0020  A106       A    R6,R4
0103  0022  D814       MOVB *R4,@RGSTRB
      0024  0000
0104              *                                  4=CALL PRINT(REG NO STR);
```

```
0105  0026  020A          LI    R10,RGSTR
      0028  0000
0106  002A  06A0          BL    @PRNTC
      002C  0000
0107  002E  1011          JMP   IWP030
0108                *                                 4=VALUE = USER WORKSPACE(I);
0109  0030  C2A0          MOV   @USRWP,R10
      0032  0000
0110  0034  A285          A     R5,R10                 WORD OFFSET
0111  0036  A285          A     R5,R10
0112  0038  C29A          MOV   *R10,R10
0113                *                                 4=CALL PRINT HEX(VALUE);
0114  003A  06A0          BL    @PRNTHX
      003C  0000
0115  003E  1009          JMP   IWP030
0116                *                                 4=RCNT = RCNT-1;
0117  0040  0605          DEC   R5
0118                *                                 4=IF RCNT .EQ. 0 THEN CALL PRI
0119  0042  1603          JNE   IWP020
0120  0044  06A0          BL    @PRCRLF
      0046  0018'
0121  0048  1004          JMP   IWP030
0122                IWP020
0123                *                                 4=I=I+1  /* NEXT REGISTER */
0124  004A  0585          INC   R5
0125                *                                 3=END
0126  004C  81C5          C     R5,R7
0127  004E  11E5          JLT   IWP010
0128  0050  13E5          JEQ   IWP010
0129                *                                 2=END
0130        0052' IWP030 EQU   $                      1=END INSPECT WORKSPACE
0131                *
0132                *     *LINK TO PREV WKSP, RET CURR WKSP
0133  0052  0420            BLWP @RR
      0054  0000
0134  0056  045B            RT
0135                        END
0000 ERS
```

```
990 - 990  CONCORDANCE
$                       0082  0098  0129
ACL                     0058  0084
CLST                    0060  0100
GETBUF                  0054
IWP         0082        0047
IWP010      0098        0126  0127
IWP020      0121        0118
IWP030      0129        0088  0106  0114  0120
LWP                     0056
PRCRLF                  0048  0096  0119
PRNTC                   0049  0105
PRNTHX                  0050  0113
R0          0065
R1          0066
R10         0075        0093  0104  0108  0109  0110  0111  0111
R11         0076
R12         0077
R13         0078
R14         0079
R15         0080
R2          0067
R3          0068
R4          0069        0100  0101  0102
R5          0070        0091  0116
R6          0071        0094  0101  0109  0110  0123  0125
R7          0072        0093  0125
R8          0073        0086
R9          0074        0094
RANGEX                  0051  0087
RETBUF                  0055
RGSTR                   0052  0104
RGSTRR                  0053  0102
RR                      0059  0132
RWP                     0057
USRWP                   0061  0108

THERE ARE    0035  SYMBOLS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																
SHEET																															
REV STATUS OF SHEETS	REV																															
	SHEET																															

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DWN	DATE
	CHK	
	ENGR	
	QA	
	APVD	
	CONTR NO	
	DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, MODCR, MODIFY CRU-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945364-9901

SCALE	REV.	SHEET
		1 of 6

TI-0023-C

```
0003          *   PROCEDURE MODCRU (CPL POINTER);
0004          *      DECLARE CRU ADDRESS BIT (16);
0005          *      DECLARE BIT WIDTH BIT (16);
0006          *      DECLARE DEFAULT BIT WIDTH LITERALLY '16';
0007          *      DECLARE FLAG BIT (1);
0008          *      DECLARE (1 COMMAND PARAMETER LIST,
0009          *                  3 CPL CONTROL (CPL POINTER),
0010          *                     5 NUMBER PARMS FIXED (8),
0011          *                     5 PARM PRESENT (8) BIT (1),
0012          *                     5 FIRST PARM FIXED (8));
0013          *      DECLARE NUMERIC PARAMETER FIXED (16)
0014          *           CONTROL (PARM POINTER);
0015          *      DECLARE (1 CHARACTER PARAMETER,
0016          *                  3 CHAR PARM CONTROL (PARM POINTER),
0017          *                     5 STRING LENGTH FIXED (8),
0018          *                     5 PARM STRING (1) CHARACTER (1));
0019          *      DECLARE (CPL POINTER, PARM POINTER) POINTER;
0020          *      DECLARE TERM CHARACTER (1);
0021          *      DECLARE VALUE BIT (16);
0022          *      DECLARE CR LITERALLY '"0D"';
0023          *      CRU ADDRESS = 0;
0024          *      BIT WIDTH = DEFAULT BIT ADDRESS;
0025          *      PARM POINTER = ADDR (FIRST PARM);
0026          *      IF PARM PRESENT (1) THEN DO;
0027          *         CRU ADDRESS = NUMERIC PARAMETER;
0028          *         PARM POINTER = PARM POINTER + 2;
0029          *         END;
0030          *      IF PARM PRESENT (2) THEN DO;
0031          *         BIT WIDTH = NUMERIC PARAMETER;
0032          *         IF BIT WIDTH = 0 ,OR, BIT WIDTH > 16
0033          *               CALL ERROR ("0P12");  /* NO RETURN */
0034          *         END;
0035          *      FLAG = 0;
0036          *      DO WHILE FLAG=0 AND LT 2000;
0037          *         CALL PRTCRU (CRU ADDRESS, CRU ADDRESS,
0038          *               BIT WIDTH);
0039          *         CALL GETHXN (VALUE, TERM, FLAG);
0040          *         IF FLAG = 0 THEN CRU (CRU ADDRESS) = VALUE;
0041          *         CRU ADDRESS = CRU ADDRESS + BIT WIDTH * 2;
0042          *         IF TERM = CR THEN FLAG = -1 ELSE FLAG = 0;
0043          *         END;
0044          *      RETURN;
0045          *      END MODCR;
0046                IDT  'MODCR'
0047          * TITLE:    MODCR
0048          *           MODIFY CRU COMMAND PROCESSOR
0049          * REVISION:
0050          *           ORIGINAL
0051          * COMPUTER: 990
0052          * ABSTRACT:
0053          *           THIS SUBROUTINE PROCESSES THE MODIFY CRU (MC)
0054          *           COMMAND.  PARAMETERS ARE STARTING ADDRESS
0055          *           AND BIT WIDTH, WHICH DEFAULT TO ZERO AND
0056          *           SIXTEEN RESPECTIVELY.  CRU INPUT LINES ARE
```

```
0057                    *           DISPLAYED AND CRU OUTPUT LINES ARE MODIFIED
0058                    *           IN A MANNER SIMILAR TO MODIFY MEMORY.  NOTE
0059                    *           THAT BIT WIDTH IS ALSO USED AS THE ADDRESS
0060                    *           INCREMENT WHEN SUCCESSIVE VALUES ARE DISPLAYED
0061                    *           AND/OR MODIFIED.
0062                    * CALLING SEQUENCE:
0063                    *           CALLED BY COMMAND STRING PROCESSOR
0064                    *
0065                    *                                   1*PROCEDURE MODCRU (CPL POINTE
0066                    *                                   2*DECLARE CRU ADDRESS BIT (16)
0067                    *                                   2*DECLARE BIT WIDTH BIT (16);
0068                    *                                   2*DECLARE DEFAULT BIT WIDTH LI
0069          0010      CRWIDT EQU  >10                      DEFAULT BIT WIDTH
0070  0000    300A      LDCR   LDCR R10,0
0071                    *                                   2*DECLARE FLAG BIT (1);
0072                    *                                   2*DECLARE (1 COMMAND PARAMETER
0073                    *                                   3*          3 CPL CONTROL (CPL
0074                    *                                   4*          5 NUMBER PARMS FIXE
0075                    *                                   4*          5 PARM PRESENT (8)
0076                    *                                   4*          5 FIRST PARM FIXED
0077                    *                                   2*DECLARE NUMERIC PARAMETER FI
0078                    *                                   2*     CONTROL (PARM POINTER);
0079                    *                                   2*DECLARE (1 CHARACTER PARAMET
0080                    *                                   3*          3 CHAR PARM CONTROL
0081                    *                                   4*          5 STRING LENGTH FIX
0082                    *                                   4*          5 PARM STRING (1) C
0083                    *                                   2*DECLARE (CPL POINTER, PARM P
0084                            REF   CR
0085                    *                                   2*DECLARE TERM CHARACTER (1);
0086                    *                                   2*DECLARE VALUE BIT (16);
0087                    *                                   2*DECLARE CR LITERALLY '"0D"';
0088                            DEF   MODCRU
0089                            REF   GETHXN
0090                            REF   PRTCRU
0091                            REF   ERROR
0092                    *
0093                    *WORKSPACE REGISTER DEFINITIONS
0094                    *
0095          0000      R0      EQU   0
0096          0001      R1      EQU   1
0097          0002      R2      EQU   2
0098          0003      R3      EQU   3
0099          0004      R4      EQU   4
0100          0005      R5      EQU   5
0101          0006      R6      EQU   6
0102          0007      R7      EQU   7
0103          0008      R8      EQU   8
0104          0009      R9      EQU   9
0105          000A      R10     EQU   10
0106          000B      R11     EQU   11
0107          000C      R12     EQU   12
0108          000D      R13     EQU   13
0109          000E      R14     EQU   14
0110          000F      R15     EQU   15
0111                    *
```

```
0112          0002' MODCRU EQU  $
0113  0002  C08B        MOV  R11,R2          SAVE RETURN
0114              *                        2=CRU ADDRESS = 0;
0115  0004  04C4        CLR  R4              R4 = CRU ADDRESS
0116              *                        2=BIT WIDTH = DEFAULT BIT ADDR
0117  0006  0205        LI   R5,CRWIDT       R5 = BIT WIDTH
      0008  0010
0118              *                        2=PARM POINTER = ADDR (FIRST P
0119  000A  C01A        MOV  *R10,R0
0120  000C  05CA        INCT R10
0121              *                        2=IF PARM PRESENT (1) THEN DO;
0122  000E  0A90        SLA  R0,9
0123  0010  1701        JNC  MOD010
0124              *                        3=CRU ADDRESS = NUMERIC PARAME
0125              *                        3=PARM POINTER = PARM POINTER
0126  0012  C13A        MOV  *R10+,R4
0127              *                        3=END;
0128          0014' MOD010 EQU  $
0129              *                        2=IF PARM PRESENT (2) THEN DO;
0130  0014  0A10        SLA  R0,1
0131  0016  1705        JNC  MOD020
0132              *                        3=BIT WIDTH = NUMERIC PARAMETE
0133  0018  C15A        MOV  *R10,R5
0134              *                        3=IF BIT WIDTH = 0 .OR. BIT WI
0135              *                        5=CALL ERROR ("DP12");  /* NO
0136  001A  132E        JEQ  MODE1
0137  001C  0285        CI   R5,CRWIDT
      001E  0010
0138  0020  1B2B        JH   MODE1
0139              *                        3=END;
0140          0022' MOD020 EQU  $
0141  0022  C185        MOV  R5,R6           GENERATE LDCR INSTRUCTION
0142  0024  0AC6        SLA  R6,12
0143  0026  0966        SRL  R6,6
0144  0028  A1A0        A    @LDCR,R6
      002A  0000'
0145              *                        2=FLAG = 0;
0146  002C  04C7        CLR  R7
0147              *                        2=DO WHILE FLAG=0 AND LT 2000;
0148          002E' MOD030 EQU  $
0149  002E  C1C7        MOV  R7,R7
0150  0030  1622        JNE  MOD100
0151  0032  C284        MOV  R4,R10
0152  0034  A285        A    R5,R10
0153  0036  A285        A    R5,R10
0154  0038  028A        CI   R10,>2000
      003A  2000
0155  003C  141C        JHE  MOD100
0156              *                        3=CALL PRTCRU (CRU ADDRESS, CR
0157              *                        5=BIT WIDTH);
0158  003E  C284  MOD040 MOV  R4,R10
0159  0040  C244        MOV  R4,R9
0160  0042  C205        MOV  R5,R8
0161  0044  06A0        BL   @PRTCRU
      0046  0000
```

```
0162  0048  1016          JMP   MOD200            **** ERROR EXIT
0163                *                             3=CALL GETHXN (VALUE, TERM, FL
0164  004A  06A0          BL    @GETHXN
      004C  0000
0165  004E  10F7          JMP   MOD040
0166  0050  1001          JMP   MOD050            **** RECYCLE CONVERSION ERROR
0167  0052  0707          SETO  R7
0168        0054' MOD050  EQU   $                 FLAG NO INPUT
0169                *                             3=IF FLAG = 0 THEN CRU (CRU AD
0170  0054  C209          MOV   R9,R8             SAVE TERM CHARACTER
0171  0056  C1C7          MOV   R7,R7
0172  0058  1606          JNE   MOD070
0173  005A  0285          CI    R5,8
      005C  0008
0174  005E  1B01          JH    MOD060
0175  0060  06CA          SWPB  R10
0176        0062' MOD060  EQU   $
0177  0062  C304          MOV   R4,R12            SET CRU BASE
0178  0064  0486          X     R6                LDCR R10,<BIT WIDTH>
0179                *                             3=CRU ADDRESS = CRU ADDRESS +
0180  0066  A105  MOD070  A     R5,R4
0181  0068  A105          A     R5,R4
0182                *                             3=IF TERM = CR THEN FLAG = -1
0183  006A  04C7          CLR   R7
0184  006C  9220          CB    @CR,R8
      006E  0000
0185  0070  16DE          JNE   MOD030
0186  0072  0707          SETO  R7
0187  0074  10DC          JMP   MOD030
0188                *                             3=END;
0189        0076' MOD100  EQU   $
0190        0076' MOD200  EQU   $
0191                *                             2=RETURN;
0192  0076  0452          B     *R2
0193        0078' MODE1   EQU   $
0194  0078  020A          LI    R10,>1112         "DP12"  BIT WIDTH ERROR
      007A  1112
0195  007C  06A0          BL    @ERROR
      007E  0000
0196  0080  10FA          JMP   MOD200
0197                *                             2=END MODCR;
0198                      END
0000 ERS
```

```
960 - 980  CONCORDANCE
$                 0112  0128  0140  0148  0168  0176  0189  0190  0193
CR                0084  0184
CRWIDT  0069      0117  0137
ERROR             0091  0195
GETHXN            0089  0164
LDCR    0070      0144
MOD010  0128      0123
MOD020  0140      0131
MOD030  0148      0185  0187
MOD040  0158      0165
MOD050  0168      0166
MOD060  0176      0174
MOD070  0180      0172
MOD100  0189      0150  0155
MOD200  0190      0162  0196
MODCRU  0112      0088
MODE1   0193      0136  0138
PRTCRU            0090  0161
R0      0095      0119  0122  0130
R1      0096
R10     0105      0070  0119  0120  0126  0133  0151  0152  0153  0154
                  0158  0175  0194
R11     0106      0113
R12     0107      0177
R13     0108
R14     0109
R15     0110
R2      0097      0113  0192
R3      0098
R4      0099      0115  0126  0151  0158  0159  0177  0180  0181
R5      0100      0117  0133  0137  0141  0152  0153  0160  0173  0180
                  0181
R6      0101      0141  0142  0143  0144  0178
R7      0102      0146  0149  0149  0167  0171  0171  0183  0186
R8      0103      0160  0170  0184
R9      0104      0159  0170

THERE ARE   0034  SYMBOLS
```

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																															
SHEET																															
REV STATUS OF SHEETS	REV																														
	SHEET																														

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DATE
DWN	
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, MODME, MODIFY MEMORY-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945365-9901

SCALE	REV.	SHEET 1 of 4

TI-0023-C

```
0003                      IDT  'MODME'
0004                *  TITLE:    MODME
0005                *            MODIFY MEMORY PROCESSOR
0006                *  REVISION:
0007                *            ORIGINAL
0008                *  COMPUTER: 990,ASM
0009                *  ABSTRACT:
0010                *            DISPLAYS AND ALLOWS THE USER TO
0011                *            MODIFY ONE OR MORE CONSEQUTIVE WORDS OF MEMORY
0012                *
0013                *
0014                *  CALLING SEQUENCE:
0015                *            BL   @MODMEM
0016                *            R10= POINTER TO COMMAND PARM LIST
0017                *            CMD PARM LIST =  # PARMS/PARM PRES BITS
0018                *                             LOCATION
0019                *            DESTROYS R0,R1,R2,R3,R4,R5,R9,R12,R14,R15
0020                *
0021                *
0022                *  REFS, DEFS
0023                *
0024                       DEF  MODMEM
0025                       REF  GETHXN
0026                       REF  BLANK
0027                       REF  PRTMEM - PRINT MEMORY RANGE ROUTINE
0028                       REF  RANGEX
0029                *
0030                *  EQU'S
0031                *
0032          0000  R0     EQU  0
0033          0003  R3     EQU  3
0034          0005  R5     EQU  5
0035          0008  R8     EQU  8
0036          0009  R9     EQU  9
0037          000A  R10    EQU  10
0038          000B  R11    EQU  11
0039          000F  R15    EQU  15
0040          0008  BYTE   EQU  8
```

```
0042                        EVEN
0043  0000  C3CB  MODMEM  MOV   R11,R15          SAVE RETURN
0044                      *
0045                      * CALCULATE LOCATION
0046                      *
0047  0002  04C8           CLR   R8
0048  0004  D0DA           MOVB  *R10,R3
0049  0006  1601           JNE   MOD005
0050  0008  0588           INC   R8
0051        000A' MOD005  EQU   $
0052  000A  06A0           BL    @RANGEX
      000C  0000
0053  000E  100F           JMP   MOD030            IF ERROR
0054                      *
0055                      * PRINT LOC AND VALUE
0056                      *
0057  0010  C24A  MOD010  MOV   R10,R9            R9,R10=RANGE
0058  0012  C0C9           MOV   R9,R3            R3=SAVED LOC
0059  0014  06A0           BL    @PRTMEM
      0016  0000
0060  0018  100A           JMP   MOD030            ESC ERROR EXIT
0061                      *
0062                      *  GET NEW VALUE
0063                      *
0064  001A  06A0           BL    @GETHXN
      001C  0000
0065  001E  1007           JMP   MOD030
0066  0020  C4CA           MOV   R10,*R3
0067  0022  9809           CB    R9,@BLANK
      0024  0000
0068  0026  1603           JNE   MOD030              IF TERMINATOR NOT A BLANK
0069                      *
0070                      *  SET FOR NEXT LOCATION
0071                      *
0072  0028  C283  MOD020  MOV   R3,R10
0073  002A  05CA           INCT  R10              R10=NEXT LOC
0074  002C  10F1           JMP   MOD010           CONTINUE
0075                      *
0076                      *  RETURN
0077                      *
0078        002E' MOD030  EQU   $
0079  002E  045F           B     *R15
0080                        END
0000 ERS
```

```
960 - 980  CONCORDANCE
$                   0051  0078
BLANK               0026  0067
BYTE      0040
GETHXN              0025  0064
MOD005    0051      0049
MOD010    0057      0074
MOD020    0072
MOD030    0078      0053  0060  0065  0068
MODMEM    0043      0024
PRTMEM              0027  0059
R0        0032
R10       0037      0048  0057  0066  0072  0073
R11       0038      0043
R15       0039      0043  0079
R3        0033      0048  0058  0066  0072
R5        0034
R8        0035      0047  0050
R9        0036      0057  0058  0067
RANGEX              0028  0052

THERE ARE    0019  SYMBOLS
```

AL

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																		
SHEET																																		
REV STATUS OF SHEETS	REV																																	
	SHEET																																	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, MODREG, MODIFY REGISTERS-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945366-9901

SCALE	REV.	SHEET
		1 of 6

TI-9923-C

```
0003                *   PROCEDURE MODIFY REGISTERS(CPL,FLAG);
0004                *      /* MODIFY REGISTERS PRINTS THE
0005                *         USER PC, WP, ST AND ALLOWS THE
0006                *         USER TO MODIFY ANY OR ALL OF THEM
0007                *      */
0008                *      STRING = LOC('WP=');
0009                *      STRING = LOC('PC=');
0010                *      P1 = LOC(USER PC)
0011                *      CALL MRSUB(STRING,P1)
0012                *      P1 = LOC(USER WP);
0013                *      CALL MRSUB(STRING,P1);
0014                *      STRING = LOC('ST=');
0015                *      P1 = LOC(USER ST)
0016                *      CALL MRSUB(STRING,P1)
0017                *      RETURN;
0018                *      END MODIFY REGISTERS
0019                *      PROCEDURE MRSUB(STR,PTR);
0020                *         IF .NOT. NO USER INPUT THEN CALL PRINT('CRLF');
0021                *         CALL PRINT(STR);
0022                *         CALL PRINT HEX(PTR.VALUE);
0023                *         IF FLAG=INSPECT ONLY THEN RETURN;
0024                *         CALL GET HEX VALUE(NO LEADING BLANKS,
0025                *            HEX VALUE,ERROR,TERM CHAR,NO USER INPUT);
0026                *         IF ERROR THEN RETURN(ERROR);
0027                *         IF .NOT. NO USER INPUT THEN DO;
0028                *            PTR.VALUE = HEX VALUE;
0029                *            IF TERM CHAR .NE. ' ' THEN RETURN('CR');
0030                *         END;
0031                *         RETURN
0032                *      END MRSUB
0033                *   END MODIFY REGISTERS;
0034                      IDT  'MODREG'
0035                * TITLE:    MR
0036                *           MODIFY REGISTERS COMMAND PROCESSOR
0037                *           IR
0038                *           INSPECT REGISTERS COMMAND
0039                * REVISION:
0040                *           ORIGINAL
0041                * COMPUTER: 990,ASM
0042                * ABSRACT: MR PRINTS THE PROCESSOR REGISTERS (WP,PC,ST)
0043                *           AND ALLOWS THE USER TO MODIFY THEM.
0044                * CALLING SEQUENCE
0045                *           BL @MRP
0046                *           R10 = POINTER TO COMMAND PARAMETER LIST
0047                *           BL @IRP
0048                *
0049                *      REF'S AND DEF'S
0050                      REF  USRPC
0051                      REF  USRWP
0052                      REF  USRST
0053                      DEF  MRP
0054                      DEF  IRP
0055                      REF  PRINT
0056                      REF  PRNTHX
```

```
0057                        REF  PRNTC
0058                        REF  PRCRLF
0059                        REF  GETHXN
0060                        REF  BLANK
0061                        REF  GETBUF
0062                        REF  RETBUF
0063                        REF  LWP
0064                        REF  RWP
0065                        REF  ACL
0066                        REF  RR
0067                  *
0068                  *WORKSPACE REGISTER DEFINITIONS
0069                  *
0070          0000    R0    EQU  0
0071          0001    R1    EQU  1
0072          0002    R2    EQU  2
0073          0003    R3    EQU  3
0074          0004    R4    EQU  4
0075          0005    R5    EQU  5
0076          0006    R6    EQU  6
0077          0007    R7    EQU  7
0078          0008    R8    EQU  8
0079          0009    R9    EQU  9
0080          000A    R10   EQU  10
0081          000B    R11   EQU  11
0082          000C    R12   EQU  12
0083          000D    R13   EQU  13
0084          000E    R14   EQU  14
0085          000F    R15   EQU  15
0086                  *
0087  0000      03    WPSTR BYTE 3
0088  0001      57          TEXT 'WP='
0089  0004      05    PCSTR BYTE 5,>0D,>0A
      0005      0D
      0006      0A
0090  0007      50          TEXT 'PC='
0091  000A      03    STSTR BYTE 3
0092  000B      53          TEXT 'ST='
0093          000E' IRP   EQU  $
0094                  *     *ALLOC,COPY,LINK
0095  000E    0420          BLWP @ACL
      0010    0000
0096  0012    04C4          CLR  R4
0097  0014    1003          JMP  MRP005
0098          0016' MRP   EQU  $
0099                  *                                  1=PROCEDURE MODIFY REGISTERS(C
0100                  *                                  2=/* MODIFY REGISTERS PRINTS T
0101                  *                                  2=    USER PC, WP, ST AND ALLOW
0102                  *                                  2=    USER TO MODIFY ANY OR ALL
0103                  *                                  2=*/
0104                  *     *ALLOC,COPY,LINK            LINK TO NEW WORKSPACE
0105  0016    0420          BLWP @ACL
      0018    0010'
0106  001A    0704          SETO R4
0107                  *                                  2=STRING = LOC('WP='));
```

```
0108          001C' MRP005 EQU  $
0109                *                                  2=STRING = LOC('PC='); 
0110  001C  020A          LI   R10,PCSTR
      001E  0004'
0111                *                                  2=P1 = LOC(USER PC)
0112  0020  0206          LI   R6,USRPC
      0022  0000
0113                *                                  2=CALL MRSUB(STRING,P1)
0114  0024  06A0          BL   @MRSUB
      0026  004C'
0115  0028  100E          JMP  MRP010
0116  002A  020A          LI   R10,WPSTR
      002C  0000'
0117                *                                  2=P1 = LOC(USER WP);
0118  002E  0206          LI   R6,USRWP
      0030  0000
0119                *                                  2=CALL MRSUB(STRING,P1);
0120  0032  06A0          BL   @MRSUB
      0034  004C'
0121  0036  1007          JMP  MRP010                   IF ERROR OR TERM FROM USER
0122                *                                  2=STRING = LOC('ST=');
0123  0038  020A          LI   R10,STSTR
      003A  000A'
0124                *                                  2=P1 = LOC(USER ST)
0125  003C  0206          LI   R6,USRST
      003E  0000
0126                *                                  2=CALL MRSUB(STRING,P1)
0127  0040  06A0          BL   @MRSUB
      0042  004C'
0128  0044  1000          NOP
0129          0046' MRP010 EQU  $
0130                *                                  2=RETURN;
0131                *                                  2=END MODIFY REGISTERS
0132                *     *LINK TO PREV WKSP, RET CURR WKSP
0133  0046  0420            BLWP @RR
      0048  0000
0134  004A  0458            RT
0135          004C' MRSUB  EQU  $
0136                *                                  2=PROCEDURE MRSUB(STR,PTR);
0137  004C  C14B          MOV  R11,R5
0138  004E  C0CA          MOV  R10,R3
0139                *                                  3=IF .NOT. NO USER INPUT THEN
0140  0050  C104          MOV  R4,R4
0141  0052  1303          JEQ  MRS005
0142  0054  06A0          BL   @PRCRLF
      0056  0000
0143  0058  1000          NOP
0144          005A' MRS005 EQU  $
0145  005A  C283          MOV  R3,R10
0146                *                                  3=CALL PRINT(STR);
0147  005C  06A0          BL   @PRNTC
      005E  0000
0148  0060  0455          B    *R5
0149                *                                  3=CALL PRINT HEX(PTR,VALUE);
0150  0062  C296          MOV  *R6,R10
```

```
 0151  0064  06A0          BL    @PRNTHX
       0066  0000
 0152  0068  0455          B     *R5
 0153                *                          3=IF FLAG=INSPECT ONLY THEN RE
 0154  006A  C104          MOV   R4,R4
 0155  006C  1308          JEQ   MRS010
 0156                *                          3=CALL GET HEX VALUE(NO LEADIN
 0157                *                          4=HEX VALUE,ERROR,TERM CHAR,NO
 0158  006E  06A0          BL    @GETHXN
       0070  0000
 0159                *                          3=IF ERROR THEN RETURN(ERROR);
 0160  0072  0455          B     *R5
 0161                *                          3=IF .NOT. NO USER INPUT THEN
 0162                *                          4=PTR.VALUE = HEX VALUE;
 0163  0074  C58A          MOV   R10,*R6
 0164                *                          4=IF TERM CHAR .NE. ' ' THEN R
 0165  0076  9809          CB    R9,@BLANK
       0078  0000
 0166  007A  1301          JEQ   MRS010
 0167  007C  0645          DECT  R5                  FORCE 'CR' RETURN
 0168                *                          3=END;
 0169                *                          3=RETURN
 0170        007E' MRS010  EQU   $
 0171  007E  05C5          INCT  R5
 0172  0080  0455          B     *R5
 0173                *                          2=END MRSUB
 0174                *                          1=END MODIFY REGISTERS;
 0175                      END
0000 ERS
```

960 • 980 CONCORDANCE

THERE ARE 0000 SYMBOLS

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																
SHEET																																
REV STATUS OF SHEETS	REV																															
	SHEET																															

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, MODUP, MODIFY WORKSPACE-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945367-9901

SCALE	REV.	SHEET 1 of 5

TI-9923-C

```
0003                *   PROCEDURE MODIFY WORKSPACE(CPL);
0004                *      /* MODIFY WORKSPACE ACCEPTS AS INPUT
0005                *         A USER REGISTER AND PRINTS THE
0006                *         CONTENTS OF EACH REGISTER AND ALLOWS
0007                *         THE USER TO MODIFY IT.
0008                *      */
0009                *      CALL RANGE(CPL,LOW,HIGH,'F',ERROR);
0010                *      /* RANGE IS FROM USER SPECIFIED REG
0011                *         TO MAX POSSIBLE REG */
0012                *      IF  .NOT. ERROR THEN DO UNTIL REGS EXHAUSTED
0013                *      OR USER CR;
0014                *         REG = CLST(LOW)
0015                *         CALL PRINT(REG STR)   /* CR,LF,'RX=' */
0016                *         RGPTR = LOC(USER WORKSPACE(I));
0017                *         CALL PRINT HEX(RGPTR,VALUE)
0018                *         CALL GET HEX VALUE(HEX VALUE,
0019                *            TERM CHAR,ERROR);
0020                *         IF ERROR THE SIGNAL USER CR;
0021                *            USER WORKSPACE(I) = HEX VALUE;
0022                *            IF TERM CHAR .NE. BLANK
0023                *               THEN SIGNAL USER CR;
0024                *         END;
0025                *         LOW = LOW+1;
0026                *         IF LOW .GT. HIGH THEN SIGNAL
0027                *            REGS EXHAUSTED,
0028                *      END;
0029                *      REGS EXHAUSTED:
0030                *      USER CR:
0031                *      END MODIFY WORKSPACE
0032                       IDT  'MODWP'
0033                * TITLE:    MODWP
0034                *           MODIFY WORKSPACE
0035                * REVISION:
0036                *           ORIGINAL
0037                * COMPUTER: 990,ASM
0038                * ABSTRACT: MODIFY WORKSPACE PRINTS THE CONTENTS OF
0039                *           A USERS WORKSPACE AND ALLOWS THE USER
0040                *           TO MODIFY THEM.
0041                * CALLING SEQUENCE:
0042                *           BL @MODWSP
0043                *           R10 = POINTER TO COMMAND PARAMETER LIST
0044                *
0045                *      REF'S AND DEF'S
0046                       DEF  MODWSP
0047                       REF  USRWP
0048                       REF  CLST
0049                       REF  PRNTC
0050                       REF  PRNTHX
0051                       REF  PRCRLF
0052                       REF  BLANK
0053                       REF  GETHXN
0054                       REF  RANGEX
0055                       REF  RGSTR
0056                       REF  RGSTRB
```

```
0057                        REF  GETBUF
0058                        REF  RETBUF
0059                        REF  LWP
0060                        REF  RWP
0061                        REF  ACL
0062                        REF  RR
0063                  *
0064                  *WORKSPACE REGISTER DEFINITIONS
0065                  *
0066        0000      R0        EQU  0
0067        0001      R1        EQU  1
0068        0002      R2        EQU  2
0069        0003      R3        EQU  3
0070        0004      R4        EQU  4
0071        0005      R5        EQU  5
0072        0006      R6        EQU  6
0073        0007      R7        EQU  7
0074        0008      R8        EQU  8
0075        0009      R9        EQU  9
0076        000A      R10       EQU  10
0077        000B      R11       EQU  11
0078        000C      R12       EQU  12
0079        000D      R13       EQU  13
0080        000E      R14       EQU  14
0081        000F      R15       EQU  15
0082                  *
0083                  *                                  1=PROCEDURE MODIFY WORKSPACE(C
0084        0000'     MODWSP    EQU  $
0085                  *     *ALLOC,COPY,LINK
0086  0000  0420               BLWP @ACL
      0002  0000
0087                  *                                  2=/* MODIFY WORKSPACE ACCEPTS
0088                  *                                  2=    A USER REGISTER AND PRINT
0089                  *                                  2=    CONTENTS OF EACH REGISTER
0090                  *                                  2=    THE USER TO MODIFY IT.
0091                  *                                  2=*/
0092                  *                                  2=CALL RANGE(CPL,LOW,HIGH,'F',
0093                  *                                  2=/* RANGE IS FROM USER SPECIF
0094                  *                                  2=    TO MAX POSSIBLE REG */
0095  0004  04C8               CLR  R8
0096  0006  06A0               BL   @RANGEX
      0008  0000
0097  000A  1025               JMP  MOD030
0098  000C  0207               LI   R7,>F                  FORCE HIGH LIMIT
      000E  000F
0099                  *                                  2=IF  ,NOT, ERROR THEN DO UNTI
0100                  *                                  2=OR USER CR)
0101  0010  C189               MOV  R9,R6                  LOW LIM
0102  0012  8187               C    R7,R6
0103  0014  1120               JLT  MOD030
0104        0016'     MOD010    EQU  $
0105                  *                                  3=REG = CLST(LOW)
0106  0016  0204               LI   R4,CLST
      0018  0000
0107  001A  A106               A    R6,R4
```

```
 0108  001C  DB14          MOVB *R4,@RGSTRB
       001E  0000
 0109                *                                  3=CALL PRINT(REG STR)   /* CR,
 0110  0020  06A0          BL   @PRCRLF                    CR / LF
       0022  0000
 0111  0024  1000          NOP
 0112  0026  020A          LI   R10,RGSTR
       0028  0000
 0113  002A  06A0          BL   @PRNTC
       002C  0000
 0114  002E  1013          JMP  MOD030
 0115                *                                  3=RGPTR = LOC(USER WORKSPACE(I
 0116  0030  C120          MOV  @USRWP,R4
       0032  0000
 0117  0034  A106          A    R6,R4
 0118  0036  A106          A    R6,R4
 0119                *                                  3=CALL PRINT HEX(RGPTR,VALUE)
 0120  0038  C294          MOV  *R4,R10
 0121  003A  06A0          BL   @PRNTHX
       003C  0000
 0122  003E  100B          JMP  MOD030
 0123                *                                  3=CALL GET HEX VALUE(HEX VALUE
 0124                *                                  4=TERM CHAR,ERROR);
 0125                *                                  3=IF ERROR THEN SIGNAL USER CR;
 0126  0040  06A0          BL   @GETHXN                      R9B0 = TERM CHAR, R10 = RSL
       0042  0000
 0127  0044  1008          JMP  MOD030
 0128                *                                  4=USER WORKSPACE(I) = HEX VALU
 0129  0046  C50A          MOV  R10,*R4
 0130                *                                  4=IF TERM CHAR .NE. BLANK
 0131                *                                  5=THEN SIGNAL USER CR;
 0132  0048  9809          CB   R9,@BLANK
       004A  0000
 0133  004C  1604          JNE  MOD030
 0134                *                                  3=END;
 0135        004E' MOD020 EQU  $
 0136                *                                  3=LOW = LOW+1;
 0137  004E  0586          INC  R6
 0138                *                                  3=IF LOW .GT. HIGH THEN SIGNAL
 0139                *                                  4=REGS EXHAUSTED,
 0140  0050  81C6          C    R6,R7
 0141  0052  1501          JGT  MOD030
 0142  0054  10E0          JMP  MOD010
 0143                *                                  2=END;
 0144                *                                  2=REGS EXHAUSTED;
 0145                *                                  2=USER CR;
 0146                *                                  2=END MODIFY WORKSPACE
 0147        0056' MOD030 EQU  $
 0148                *    *LINK TO PREV WKSP, RET CURR WKSP
 0149  0056  0420          BLWP @RR
       0058  0000
 0150  005A  0458          RT
 0151                      END
0000 ERS
```

```
960 = 980  CONCORDANCE
$                   0084  0104  0135  0147
ACL                 0061  0086
BLANK               0052  0132
CLST                0048  0106
GETBUF              0057
GETHXN              0053  0126
LWP                 0059
MOD010   0104       0142
MOD020   0135
MOD030   0147       0097  0103  0114  0122  0127  0133  0141
MODWSP   0084       0046
PRCRLF              0051  0110
PRNTC               0049  0113
PRNTHX              0050  0121
R0       0066
R1       0067
R10      0076       0112  0120  0129
R11      0077
R12      0078
R13      0079
R14      0080
R15      0081
R2       0068
R3       0069
R4       0070       0106  0107  0108  0116  0117  0118  0120  0129
R5       0071
R6       0072       0101  0102  0107  0117  0118  0137  0140
R7       0073       0098  0102  0140
R8       0074       0095
R9       0075       0101  0132
RANGEX              0054  0096
RETBUF              0058
RGSTR               0055  0112
RGSTRB              0056  0108
RR                  0062  0149
RWP                 0060
USRWP               0047  0116

THERE ARE   0037  SYMBOLS
```

A[

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																		
SHEET																																		
REV STATUS OF SHEETS	REV																																	
	SHEET																																	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, OVERLAY, OVERLAY PROCESSOR-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945368-9901

SCALE	REV.	SHEET 1 of 7

TI-0023-C

```
0003          *   PROCEDURE OVERLAY;
0004          *      /*
0005          *        OVERLAY IS RESPONSIBLE FOR
0006          *        MANAGING THE PX990 TRANSIENT
0007          *        AREA. THIS INCLUDES;
0008          *        1)  DISABLING COMMANDS FOR THE
0009          *            CURRENT AREA.
0010          *        2)  CALLING THE LOADER TO LOAD
0011          *            THE NEW OVERLAY.
0012          *        3)  ENABLING THE COMMANDS IN THE
0013          *            NEW OVERLAY.
0014          *        4)  BUILDING A TRANSFER VECTOR FOR
0015          *            USE BY THE OVERLAY IN CALLING
0016          *            THE SYSTEM.
0017          *      */
0018          *      DECLARE
0019          *         TRANSIENT AREA(700) FIXED(16) CONTROL TPTR,
0020          *         VECTOR(2) PTR  CONTROL TPTR;   /* SVC CALL */
0021          *      DECLARE
0022          *         1 COMMAND ENTRY  CONTROL CPTR;
0023          *            2 COMMAND STRING   CHAR(2),
0024          *            2 COMMAND ROUT     POINTER;
0025          *      CPTR = LOC(TRANSIENT AREA) + LEN(VECTOR);
0026          *      DO WHILE COMMAND ENTRY .NE. 0;
0027          *         CMDPTR = LOC(COMMAND TABLE);
0028          *         CCNT = NO CMDS;
0029          *         DO WHILE CCNT>0;
0030          *            IF COMMAND STRING .NE. CMDPTR.COMMAND
0031          *            THEN DO;
0032          *            CMDPTR = CMDPTR + LEN(COMMAND)
0033          *            CCNT = CCNT -1;
0034          *         END;
0035          *         CPTR = CPTR + LEN(COMMAND ENTRY);
0036          *      END
0037          *      BIAS= LOC(INIT)
0038          *      CALL LOAD(FLAG,BIAS,ERROR)
0039          *      IF LOAD SUCCESSFUL THEN DO;
0040          *         CPTR = LOC(TRANSIENT AREA)
0041          *         DO WHILE COMMAND ENTRY .NE. 0;
0042          *            CMDPTR = LOC(COMMAND TABLE);
0043          *            CCNT = NO CMDS
0044          *            DO WHILE CCNT>0;
0045          *               IF COMMAND STRING .EQ. CMDPTR.COMMAND
0046          *               THEN DO;
0047          *               CMDPTR = CMDPTR + LEN(COMMAND);
0048          *               CCNT = CCNT -1;
0049          *            END;
0050          *                  CMDPTR.COMMAND SVC ROUTINE = COMMAND ROU
0051          *            CPTR = CPTR + LEN(COMMAND ENTRY);
0052          *         END;
0053          *      END;ELSE DO;
0054          *         /* INDICATE NO COMMANDS */
0055          *         LOC(TRANSIENT AREA) = 0;
0056          *      END;
```

```
0057                    IDT  'OVERLAY'
0058                *  TITLE:    OVERLAY
0059                *            OVERLAY SUPERVISOR
0060                *  REVISION:
0061                *            ORIGINAL
0062                *  COMPUTER: 990,ASM
0063                *  ABSTRACT: OVLY IS RESPONSIBLE FOR MANAGING THE
0064                *            PX990 TRANSIENT AREA. SPECIFICALLY:
0065                *             1) DISABLE COMMANDS IN CURRENT OVERLAY.
0066                *             2) CALL LOADER TO LOAD OVERLAY.
0067                *             3) IF LOAD SUCCESSFUL, ENABLE COMMANDS
0068                *                IN NEW OVERLAY.
0069                *  CALLING SEQUENCE:
0070                *            BL @OVLY
0071                *            R10 CONTAINS POINTER TO COMMAND PARAMETER LIST
0072                *       REF'S AND DEF'S
0073                *
0074                         REF  GETBUF
0075                         REF  RETBUF
0076                         REF  LWP
0077                         REF  RWP
0078                         REF  ACL
0079                         REF  RR
0080                         DEF  OVLY
0081                         DEF  OVLRET
0082                         REF  LOADOV
0083                         REF  TRNARA            START OF TRANSIENT AREA
0084                         REF  INIT
0085                         REF  CMDTBL
0086                         REF  NOCMDS
0087                         REF  PRCRLF
0088                         REF  PRINT
0089                *
0090                *WORKSPACE REGISTER DEFINITIONS
0091                *
0092          0000   R0      EQU  0
0093          0001   R1      EQU  1
0094          0002   R2      EQU  2
0095          0003   R3      EQU  3
0096          0004   R4      EQU  4
0097          0005   R5      EQU  5
0098          0006   R6      EQU  6
0099          0007   R7      EQU  7
0100          0008   R8      EQU  8
0101          0009   R9      EQU  9
0102          000A   R10     EQU  10
0103          000B   R11     EQU  11
0104          000C   R12     EQU  12
0105          000D   R13     EQU  13
0106          000E   R14     EQU  14
0107          000F   R15     EQU  15
0108                *
0109                *
0110                *
0111                *                              1=PROCEDURE OVERLAY;
```

```
0112          0000' OVLY   EQU  $
0113                *     *ALLOC,COPY,LINK
0114  0000    0420        BLWP @ACL
      0002    0000
0115                *                                  2-/*
0116                *                                  2-  OVERLAY IS RESPONSIBLE FOR
0117                *                                  2-  MANAGING THE PX990 TRANSIE
0118                *                                  2-  AREA. THIS INCLUDES:
0119                *                                  2-  1)  DISABLING COMMANDS FOR
0120                *                                  2-      CURRENT AREA.
0121                *                                  2-  2)  CALLING THE LOADER TO
0122                *                                  2-      THE NEW OVERLAY.
0123                *                                  2-  3)  ENABLING THE COMMANDS
0124                *                                  2-      NEW OVERLAY.
0125                *                                  2-  4)  BUILDING A TRANSFER VE
0126                *                                  2-      USE BY THE OVERLAY IN
0127                *                                  2-      THE SYSTEM.
0128                *                                  2-*/
0129                *                                  2-DECLARE
0130                *                                  3-TRANSIENT AREA(700) FIXED(16
0131                *                                  3-VECTOR(2) PTR  CONTROL TPTR;
0132                *                                  2-DECLARE
0133                *                                  3-1 COMMAND ENTRY  CONTROL CPT
0134                *                                  4-2 COMMAND STRING   CHAR(2),
0135                *                                  4-2 COMMAND ROUT     POINTER;
0136          0004  CMDRTN EQU  4
0137          0006  CMDLEN EQU  6
0138          0000  CDSTR  EQU  0                       COMMAND TEXT STRING
0139          0002  CDRTN  EQU  2                       COMMAND SERVICE ROUTINE.
0140                *
0141                *     REGISTER ASSIGNMENTS
0142                *
0143                *     R10
0144                *     R9
0145                *     R8
0146                *     R7
0147                *     R6    - COMMAND ENTRY POINTER
0148                *     R5    - COMMAND TABLE POINTER
0149                *     R4    - TEMP
0150                *                                  2-CPTR = LOC(TRANSIENT AREA) +
0151  0004  0206        LI   R6,TRNARA
      0006  0000
0152                *                                  2-DO WHILE COMMAND ENTRY .NE.
0153          0008' OVLY10 EQU  $
0154  0008  C596        MOV  *R6,*R6
0155  000A  1310        JEQ  OVLY50
0156                *                                  3-CMDPTR = LOC(COMMAND TABLE);
0157  000C  0205        LI   R5,CMDTBL
      000E  0000
0158                *                                  3-CCNT = NO CMDS;
0159  0010  C120        MOV  @NOCMDS,R4
      0012  0000
0160                *                                  3-DO WHILE CCNT>0;
0161          0014' OVLY20 EQU  $
0162  0014  1308        JEQ  OVLY50
```

```
0163                    *                                  4=IF COMMAND STRING .NE. CMDPT
0164                    *                                  4=THEN DO;
0165  0016  8556               C     *R6,*R5
0166  0018  1304               JEQ   OVLY40
0167                    *                                  4=CMDPTR = CMDPTR + LEN(COMMAN
0168  001A  0225               AI    R5,CMDLEN
      001C  0006
0169                    *                                  4=CCNT = CCNT -1;
0170  001E  0604               DEC   R4
0171                    *                                  3=END;
0172  0020  10F9               JMP   OVLY20
0173        0022' OVLY40  EQU   $
0174  0022  04E5               CLR   @CMDRTN(R5)
      0024  0004
0175                    *                                  3=CPTR = CPTR + LEN(COMMAND EN
0176  0026  0226               AI    R6,CDRTN+2
      0028  0004
0177                    *                                  2=END
0178  002A  10EE               JMP   OVLY10
0179        002C' OVLY50  EQU   $
0180                    **      LOAD  OVERLAY
0181                    *                                  2=BIAS= LOC(INIT)
0182  002C  0201               LI    R1,INIT
      002E  0000
0183                    *                                  2=CALL LOAD(FLAG,BIAS,ERROR)
0184  0030  06A0               BL    @LOADOV
      0032  0000
0185        0034' OVLRET  EQU   $
0186                    *                                  2=IF LOAD SUCCESSFUL THEN DO;
0187  0034  C000               MOV   R0,R0
0188  0036  161E               JNE   OVLZ50
0189                    *
0190                    *                                  3=CPTR = LOC(TRANSIENT AREA)
0191  0038  0206               LI    R6,TRNARA
      003A  0006'
0192                    *                                  3=DO WHILE COMMAND ENTRY .NE.
0193        003C' OVLZ10  EQU   $
0194  003C  C596               MOV   *R6,*R6
0195  003E  131C               JEQ   OVLZ60
0196                    *                                  4=CMDPTR = LOC(COMMAND TABLE);
0197  0040  0205               LI    R5,CMDTBL
      0042  000E'
0198                    *                                  4=CCNT = NO CMDS
0199  0044  C120               MOV   @NOCMDS,R4
      0046  0012'
0200                    *                                  4=DO WHILE CCNT>0;
0201        0048' OVLZ20  EQU   $
0202  0048  1315               JEQ   OVLZ50
0203                    *                                  5=IF COMMAND STRING .EQ. CMDPT
0204                    *                                  5=THEN DO;
0205  004A  8556               C     *R6,*R5
0206  004C  1304               JEQ   OVLZ30
0207                    *                                  5=CMDPTR = CMDPTR + LEN(COMMAN
0208  004E  0225               AI    R5,CMDLEN
      0050  0006
```

```
0209                    *                                   5=CCNT = CCNT -1;
0210  0052  0604               DEC   R4
0211                    *                                   4=END;
0212  0054  10F9               JMP   OVLZ20
0213        0056' OVLZ30  EQU   $
0214                    *                                   6=CMDPTR,COMMAND SVC ROUTINE =
0215  0056  C960               MOV   @CDRTN(R6),@CMDRTN(R5)
      0058  0002
      005A  0004
0216  005C  06A0               BL    @PRCRLF
      005E  0000
0217  0060  1000               NOP
0218  0062  C285               MOV   R5,R10
0219  0064  0209               LI    R9,2
      0066  0002
0220  0068  06A0               BL    @PRINT
      006A  0000
0221  006C  1000               NOP
0222                    *                                   4=CPTR = CPTR + LEN(COMMAND EN
0223  006E  0226               AI    R6,CDRTN+2
      0070  0004
0224                    *                                   3=END;
0225  0072  10E4               JMP   OVLZ10
0226                    *                                   2=END;ELSE DO;
0227        0074' OVLZ50  EQU   $
0228                    *                                   3=/* INDICATE NO COMMANDS */
0229                    *                                   3=LOC(TRANSIENT AREA) = 0;
0230  0074  04E0               CLR   @TRNARA
      0076  003A'
0231                    *                                   2=END;
0232        0078' OVLZ60  EQU   $
0233  0078  0420               BLWP  @RR
      007A  0000
0234  007C  0458               RT
0235                      END
0000 ERS
```

$		0112	0153	0161	0173	0179	0185	0193	0201	0213
		0227	0232							
ACL		0078	0114							
CDRTN	0139	0176	0215	0223						
CDSTR	013A									
CMDLEN	0137	0168	0208							
CMDRTN	0136	0174	0215							
CMDTBL		0085	0157	0197						
GETBUF		0074								
INIT		0084	0182							
LOADOV		0082	0184							
LWP		0076								
NOCMDS		0086	0159	0199						
OVLRET	0185	0081								
OVLY	0112	0080								
OVLY10	0153	0178								
OVLY20	0161	0172								
OVLY40	0173	0166								
OVLY50	0179	0155	0162							
OVLZ10	0193	0225								
OVLZ20	0201	0212								
OVLZ30	0213	0206								
OVLZ50	0227	0188	0202							
OVLZ60	0232	0195								
PRCRLF		0087	0216							
PRINT		0088	0220							
R0	0092	0187	0187							
R1	0093	0182								
R10	0102	0218								
R11	0103									
R12	0104									
R13	0105									
R14	0106									
R15	0107									
R2	0094									
R3	0095									
R4	0096	0159	0170	0199	0210					
R5	0097	0157	0165	0168	0174	0197	0205	0208	0215	0218
R6	0098	0151	0154	0154	0165	0176	0191	0194	0194	0205
		0215	0223							
R7	0099									
R8	0100									
R9	0101	0219								
RETBUF		0075								
RR		0079	0233							
RWP		0077								
TRNARA		0083	0151	0191	0230					

THERE ARE 0045 SYMBOLS

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV	
SHEET	

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN — DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, PCOUT, OUTPUT DATA FOR FRONT PANEL LIGHTS-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945369-9901

SCALE | REV. | SHEET 1 of 3

TI-0023-C

```
0003                        IDT  'PCOUT'
0004                  *  TITLE:  OUTPUT USRPC
0005                  * REVISION:
0006                  *             ORIGINAL
0007                  *  COMPUTER: 990,ASM
0008                  *  ABSTRACT: OUTPUTS USER PC TO FRONT PANEL LIGHTS.
0009                  *  CALLING SEQUENCE :
0010                  *             BL   @PCOUT
0011                  *
0012                  *             DESTROYS R12,R10
0013                  *
0014                  *           REFS AND DEFS
0015                        DEF  PCOUT
0016                        REF  CRUOFF
0017                        REF  USRPC
0018                  *
0019        000C      R12   EQU  12
0020        000A      R10   EQU  10
0021                  *
0022        0000'     PCOUT EQU  $
0023  0000  020C            LI   R12,CRUOFF
      0002  0000
0024  0004  C2A0            MOV  @USRPC,R10
      0006  0000
0025  0008  320A            LDCR R10,8
0026  000A  06CA            SWPB R10
0027  000C  320A            LDCR R10,8
0028  000E  045B            RT
0029                        END
0000 ERS
```

```
960 - 980  CONCORDANCE
$                   0022
CRUOFF              0016  0023
PCOUT    0022       0015
R10      0020       0024  0025  0026  0027
R12      0019       0023
USRPC               0017  0024

THERE ARE    0006  SYMBOLS
```

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																		
SHEET																																		
REV STATUS OF SHEETS	REV																																	
	SHEET																																	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DWN	DATE
	CHK	
	ENGR	
	QA	
	APVD	
	CONTR NO	
	DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PRTMMC, PRINT MEMORY AND CRU RANGES-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945370-9901

SCALE	REV.	SHEET 1 of 8

TI-0023.C

```
0003              *    PROCEDURE PRTMEM (START ADDR, END ADDR);
0004              *       DECLARE (START ADDR, END ADDR) BIT (16);
0005              *       DECLARE MULTIPLIER LITERALLY '2';
0006              *       DECLARE NOT CRU LITERALLY '0';
0007              *
0008              *       CALL PRT000 (START ADDR, END ADDR, NOT CRU,
0009              *                MULTIPLIER);
0010              *       RETURN;
0011              *       END PRTMEM;
0012              *    PROCEDURE PRTCRU (WIDTH, START ADDR,
0013              *                END ADDR);
0014              *       DECLARE (START ADDR, END ADDR) BIT (16);
0015              *       DECLARE WIDTH FIXED;
0016              *       DECLARE MULTIPLIER LITERALLY '32';
0017              *       DECLARE CRU LITERALLY '-1';
0018              *
0019              *       CALL PRT000 (START ADDR, END ADDR, CRU,
0020              *                MULTIPLIER, WIDTH);
0021              *       RETURN;
0022              *       END PRTCRU;
0023              *    PROCEDURE PRT000 (START, END, CRU FLAG,
0024              *                MULTIPLIER, WIDTH)
0025              *       DECLARE (START,END, MULTIPLIER, WIDTH,
0026              *                CRU FLAG) BIT (16);
0027              *       DECLARE I BIT (16);
0028              *       DECLARE ADDR POINTER;
0029              *       DECLARE MEMORY WORD BIT (16) CONTROL (ADDR);
0030              *       I = 0;
0031              *       DO WHILE (START + I * MULTIPLIER) <= END;
0032              *          ADDR = START + I * MULTIPLIER
0033              *          IF MOD (I, 8) = 0 THEN DO;
0034              *                CALL PRCRLF;
0035              *                CALL PRNTHX (ADDR);
0036              *                CALL PRNTC ('=');
0037              *                END;
0038              *             ELSE IF MOD (I, 4) = 0 THEN CALL PRNTC ('>');
0039              *          IF CRU FLAG = 0
0040              *             THEN DATA = MEMORY WORD
0041              *             ELSE DATA = CRU (ADDR);
0042              *          CALL PRNTHX (DATA);
0043              *          I = I + 1;
0044              *        END;
0045              *       RETURN;
0046              *       END PRT000;
0047                      IDT  'PRTMMCR'
0048              * TITLE:    PRTMEMCR
0049              *           PRINT MEMORY AND CRU RANGES
0050              * REVISION:
0051              *           ORIGINAL
0052              * COMPUTER: 990
0053              * ABSTRACT:
0054              *           THIS SUBROUTINE PRINTS MEMORY AND CRU RANGES.
0055              *           THERE ARE TWO ENTRY POINTS: PRTMEM AND PRTCRU.
0056              *
```

```
0057                    DEF  PRTMEM
0058                    DEF  PRTCRU
0059              *
0060                    REF  PRNTHN
0061                    REF  PRNTC
0062                    REF  PRNTHX
0063                    REF  ACL,RR
0064                    REF  PRCRLF
0065                    REF  EQSIGN,DMPSPR
0066              *
0067              *WORKSPACE REGISTER DEFINITIONS
0068              *
0069        0000  R0      EQU  0
0070        0001  R1      EQU  1
0071        0002  R2      EQU  2
0072        0003  R3      EQU  3
0073        0004  R4      EQU  4
0074        0005  R5      EQU  5
0075        0006  R6      EQU  6
0076        0007  R7      EQU  7
0077        0008  R8      EQU  8
0078        0009  R9      EQU  9
0079        000A  R10     EQU  10
0080        000B  R11     EQU  11
0081        000C  R12     EQU  12
0082        000D  R13     EQU  13
0083        000E  R14     EQU  14
0084        000F  R15     EQU  15
0085              *
```

```
0087                    * TITLE:    PRTMEM
0088                    *           PRINT MEMORY RANGE
0089                    * REVISION:
0090                    *           ORIGINAL
0091                    * COMPUTER: 990
0092                    * ABSTRACT:
0093                    *           THIS SUBROUTINE PRINTS THE CONTENTS OF MEMORY,
0094                    *           EIGHT WORDS PER LINE, FOR THE INCLUSIVE RANGE
0095                    *           <START ADDR> TO <END ADDR>.  THIS DATA IS
0096                    *           PRINTED AS FOUR HEXADECIMAL DIGITS.
0097                    * CALLING SEQUENCE:
0098                    *           MOV  <START ADDR>,R9
0099                    *           MOV  <END ADDR>,R10
0100                    *           BL   @PRTMEM
0101                    *           JMP                 *** ERROR EXIT
0102                    *           R0 AND R1 ARE DESTROYED
0103                    *                               1=PROCEDURE PRTMEM (START ADDR
0104                    *                               2=DECLARE (START ADDR, END ADD
0105                    *                               2=DECLARE MULTIPLIER LITERALLY
0106                    *                               2=DECLARE NOT CRU LITERALLY '0
0107          0002  WORD   EQU  2                        MULTIPLIER = 2
0108          0000' PRTMEM EQU  $
0109  0000    0420        BLWP @ACL
      0002    0000
0110                    *                               2=
0111                    *                               2=CALL PRT000 (START ADDR, END
0112                    *                               5=MULTIPLIER);
0113  0004    0249        ANDI R9,>FFFE                   ASSURE WORD ADDRESS
      0006    FFFE
0114  0008    04C3        CLR  R3                         FLAG NOT CRU
0115  000A    0204        LI   R4,WORD                    MULTIPLIER
      000C    0002
0116  000E    100B        JMP  PRT000
0117                    *                               2=RETURN;
0118                    *                               2=END PRTMEM;
```

```
0120                    * TITLE:     PRTCRU
0121                    *            PRINT CRU RANGE
0122                    * REVISION:
0123                    *            ORIGINAL
0124                    * COMPUTER:  990
0125                    * ABSTRACT:
0126                    *            THIS SUBROUTINE PRINTS THE VALUES OF THE CRU
0127                    *            INPUT LINES.  THE CRU INPUT LINES ARE TREATED
0128                    *            IN GROUPS OF SIXTEEN, ALTHOUGH NO BOUNDARY
0129                    *            RESTRICTION IS PLACED UPON THE STARTING ADDRESS
0130                    *            OF A GROUP.  ONLY THE FIRST <WIDTH> LINES OF
0131                    *            EACH SIXTEEN LINE GROUP ARE READ AND DISPLAYED.
0132                    * CALLING SEQUENCE:
0133                    *            MOV  <WIDTH>,R8
0134                    *            MOV  <START ADDR>,R9
0135                    *            MOV  <END ADDR>,R10
0136                    *            BL   @PRTCRU
0137                    *            JMP                 *** ERROR EXIT
0138                    *            R0 AND R1 ARE DESTROYED
0139                    *                                1=PROCEDURE PRTCRU (WIDTH, STA
0140                    *                                5=END ADDR);
0141                    *                                2=DECLARE (START ADDR, END ADD
0142                    *                                2=DECLARE WIDTH FIXED;
0143                    *                                2=DECLARE MULTIPLIER LITERALLY
0144                    *                                2=DECLARE CRU LITERALLY '=1';
0145          0020  CRUMLT EQU  16*2             INC CRU BASE BY SIXTEEN BITS
0146  0010    340A  STCR   STCR R10,0
0147          0012' PRTCRU EQU  $
0148  0012    0420         BLWP @ACL
      0014    0002'
0149                    *                                2=
0150                    *                                2=CALL PRT000 (START ADDR, END
0151                    *                                5=MULTIPLIER, WIDTH);
0152  0016    C088         MOV  R8,R2            GENERATE STCR INSTRUCTION
0153  0018    0AC2         SLA  R2,12
0154  001A    0962         SRL  R2,6
0155  001C    A0A0         A    @STCR,R2
      001F    0010'
0156  0020    0204         LI   R4,CRUMLT         MULTIPLIER
      0022    0020
0157  0024    0703         SETO R3                CRU FLAG
0158                    *                                2=RETURN;
0159                    *                                2=END PRTCRU;
```

```
0161                    *                               1-PROCEDURE PRT000 (START, END
0162                    *                               5-MULTIPLIER, WIDTH)
0163          0026' PRT000 EQU  $
0164                    *                               2-DECLARE (START,END, MULTIPLI
0165                    *                               5-CRU FLAG) BIT (16);
0166                    *                               2-DECLARE I BIT (16);
0167                    *                               2-DECLARE ADDR POINTER;
0168                    *                               2-DECLARE MEMORY WORD BIT (16)
0169                    *                                R2 = STCR INSTRUCTION FOR CRU
0170                    *                                R3 = CRU FLAG (NONZERO = CRU)
0171                    *                                R4 = MULTIPLIER
0172  0026  C149              MOV   R9,R5                R5 = START
0173  0028  C18A              MOV   R10,R6               R6 = END
0174                    *                               2-I = 0;
0175  002A  04C7              CLR   R7                   R7 = I
0176                    *                               2-DO WHILE (START + I * MULTIP
0177          002C' PRT010 EQU  $
0178  002C  C2C7              MOV   R7,R11               I * MULTIPLIER
0179  002E  3AC4              MPY   R4,R11
0180  0030  A305              A     R5,R12               I * MULTIPLIER + START
0181  0032  182C              JOC   PRT200
0182  0034  818C              C     R12,R6
0183  0036  182A              JH    PRT200               IF END.OF.LOOP
0184                    *                               3-ADDR = START + I * MULTIPLIE
0185                    *                               3-IF MOD (I, 8) = 0 THEN DO;
0186  0038  C247              MOV   R7,R9
0187  003A  0249              ANDI  R9,7
      003C  0007
0188  003E  160D              JNE   PRT030               IF I MOD 8 NON-ZERO
0189                    *                               5-CALL PRCRLF;
0190  0040  06A0              BL    @PRCRLF              NEXT LINE
      0042  0000
0191  0044  1025              JMP   PRT500               **** ERROR EXIT
0192                    *                               5-CALL PRNTHX (ADDR);
0193  0046  C28C              MOV   R12,R10              PRINT STARTING ADDR OF LINE
0194  0048  06A0              BL    @PRNTHN
      004A  0000
0195  004C  1021              JMP   PRT500               **** ERROR EXIT
0196                    *                               5-CALL PRNTC ('=');
0197  004E  020A              LI    R10,EQSIGN           PRINT EQUAL SIGN
      0050  0000
0198  0052  06A0              BL    @PRNTC
      0054  0000
0199  0056  101C              JMP   PRT500               **** ERROR EXIT
0200                    *                               5-END;
0201  0058  1008              JMP   PRT050
0202                    *                               4-ELSE IF MOD (I, 4) = 0 THEN
0203          005A' PRT030 EQU  $
0204  005A  0249              ANDI  R9,3
      005C  0003
0205  005E  1605              JNE   PRT050               IF I MOD 4 NON-ZERO
0206  0060  020A              LI    R10,DMPSPR           PRINT SEPARATOR
      0062  0000
0207  0064  06A0              BL    @PRNTC
```

```
                0066  0054'
0208            0068  1013          JMP   PRT500             **** ERROR EXIT
0209                                *                        3-IF CRU FLAG = 0
0210                  006A' PRT050  EQU   $
0211            006A  C0C3          MOV   R3,R3
0212            006C  1602          JNE   PRT070             IF CRU FLAG NON-ZERO
0213                                *                        4-THEN DATA = MEMORY WORD
0214            006E  C29C          MOV   *R12,R10
0215            0070  1006          JMP   PRT110
0216                                *                        4-ELSE DATA = CRU (ADDR);
0217                  0072' PRT070  EQU   $
0218                                *
0219            0072  04CA          CLR   R10
0220            0074  0482          X     R2                 STCR INTO R10
0221            0076  0288          CI    R8,8               IF BYTE OR LESS, RIGHT
                0078  0008
0222            007A  1501          JGT   PRT110               JUSTIFY
0223            007C  06CA          SWPB  R10
0224                                *                        3-CALL PRNTHX (DATA);
0225                  007E' PRT110  EQU   $
0226            007E  06A0          BL    @PRNTHX
                0080  0000
0227            0082  1006          JMP   PRT500             **** ERROR EXIT
0228                                *                        3-I = I + 1;
0229            0084  0587          INC   R7
0230                                *                        3-END;
0231            0086  028C          CI    R12,>FFFE
                0088  FFFE
0232            008A  1AD0          JL    PRT010
0233                                *                        2-RETURN;
0234                  008C' PRT200  EQU   $
0235            008C  05ED          INCT  @R11*2(R13)        INDICATE NORMAL EXIT
                008E  0016
0236                  0090' PRT500  EQU   $
0237            0090  0420          BLWP  @RR                BACK TO OLD WKSP
                0092  0000
0238            0094  045B          RT
0239                                *                        2-END PRT000;
0240                                END
0000 ERS
```

```
960 = 980  CONCORDANCE
$                 0108  0147  0163  0177  0203  0210  0217  0225  0234
                  0236
ACL               0063  0109  0148
CRUMLT  0145      0156
DMPSPR            0065  0206
EQSIGN            0065  0197
PRCRLF            0064  0190
PRNTC             0061  0198  0207
PRNTHN            0060  0194
PRNTHX            0062  0226
PRT000  0163      0116
PRT010  0177      0232
PRT030  0203      0188
PRT050  0210      0201  0205
PRT070  0217      0212
PRT110  0225      0215  0222
PRT200  0234      0181  0183
PRT500  0236      0191  0195  0199  0208  0227
PRTCRU  0147      0058
PRTMEM  0108      0057
R0      0069
R1      0070
R10     0079      0146  0173  0193  0197  0206  0214  0219  0223
R11     0080      0178  0179  0235
R12     0081      0180  0182  0193  0214  0231
R13     0082      0235
R14     0083
R15     0084
R2      0071      0152  0153  0154  0155  0220
R3      0072      0114  0157  0211  0211
R4      0073      0115  0156  0179
R5      0074      0172  0180
R6      0075      0173  0182
R7      0076      0175  0178  0186  0229
R8      0077      0152  0221
R9      0078      0113  0172  0186  0187  0204
RR                0063  0237
STCR    0146      0155
WORD    0107      0115

THERE ARE   0038  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV	
SHEET	

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ANGLES ±1° 3 PLACE DECIMAL ±.010 2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS INCORPORATED
Equipment Group *Dallas, Texas*

AL, PRTSSN, PRINT SNAPSHOT-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945371-9901

SCALE	REV.	SHEET 1 of 5

TI-0023-C

```
0003                *    PROCEDURE PRINT SNAPSHOT(SS#);
0004                *       /* PRINT SNAPSHOT FINDS THE SNAPSHOT
0005                *          ENTRY ASSOCIATED WITH A SNAPSHOT
0006                *          AN PRINTS THE CONTENTS OF THE
0007                *          SNAPSHOT
0008                *       */
0009                *       DECLARE
0010                *          1 SNAPSHOT CONTROL SNAP ENTRY;
0011                *             2 SSFLAG BIT(8),
0012                *             2 FILLER(8),
0013                *             2 LOW REG FIXED(8)
0014                *             2 HIGH REG FIXED(8)
0015                *             2 MEM RANGE 1(2) FIXED(16),
0016                *             2 MEM RANGE 2(2) FIXED(16);
0017                *       DECLARE
0018                *          SFDEF LITERALLY '80',  /* SS DEFINED */
0019                *          SFMR1 LITERALLY '40',  /* MEM RNG 1  */
0020                *          SFMR2 LITERALLY '20'   /* MEM RNG 2  */
0021                *       SNAP ENTRY = LOC(SNAP TABLE) + SS#*SSLEN;
0022                *       IF (SSDEF &SSFLAG) .EQ. 0 THEN RETURN;
0023                *       CALL PRINT( 'SNAP ',SS#);
0024                *       CALL PRINT REG(LOW REG,HIGH REG);
0025                *       CALL PRINT MEMORY(MEM RANGE 1);
0026                *       END;
0027                *    END PRINT SNAPSHOT
0028                          IDT  'PRTSSN'
0029                *TITLE;     PRTSS
0030                *           PRINT SNAPSHOT
0031                * REVISION;
0032                *           ORIGINAL
0033                *COMPUTER;  990,ASM
0034                *ABSTRACT;  PRINT SNAPSHOT PRINTS THE CONTENTS OF A
0035                *           SNAPSHOT. IT IS ASSUMED THAT THE SNAPSHOT
0036                *           IS PROPERLY DEFINED
0037                *CALLING SEQUENCE;
0038                *           BL @PRTSS
0039                *           R10 = SNAPSHOT #
0040                *
0041                *       REF'S AND DEF'S
0042                *
0043                          DEF  PRTSS
0044                          REF  SNPENT
0045                          REF  SNPTAB
0046                          REF  CLST
0047                          REF  PRTMEM
0048                          REF  IWP
0049                          REF  PRNTC
0050                          REF  SSSTR
0051                          REF  SSSTRB
0052                          REF  GETBUF
0053                          REF  RETBUF
0054                          REF  ACL
0055                          REF  RR
0056                *
```

```
0057                    *WORKSPACE REGISTER DEFINITIONS
0058                    *
0059          0000      R0      EQU  0
0060          0001      R1      EQU  1
0061          0002      R2      EQU  2
0062          0003      R3      EQU  3
0063          0004      R4      EQU  4
0064          0005      R5      EQU  5
0065          0006      R6      EQU  6
0066          0007      R7      EQU  7
0067          0008      R8      EQU  8
0068          0009      R9      EQU  9
0069          000A      R10     EQU  10
0070          000B      R11     EQU  11
0071          000C      R12     EQU  12
0072          000D      R13     EQU  13
0073          000E      R14     EQU  14
0074          000F      R15     EQU  15
0075                    *
0076                    *                              1*PROCEDURE PRINT SNAPSHOT(SS#
0077                    *                              2*/* PRINT SNAPSHOT FINDS THE
0078                    *                              2*    ENTRY ASSOCIATED WITH A S
0079                    *                              2*    AN PRINTS THE CONTENTS OF
0080                    *                              2*    SNAPSHOT
0081                    *                              2*/
0082                    *                              2*DECLARE
0083                    *                              3*1 SNAPSHOT CONTROL SNAP ENTR
0084                    *                              4*2 SSFLAG BIT(8),
0085                    *                              4*2 FILLER(8),
0086                    *                              4*2 LOW REG FIXED(8)
0087                    *                              4*2 HIGH REG FIXED(8)
0088                    *                              4*2 MEM RANGE 1(2) FIXED(16),
0089                    *                              4*2 MEM RANGE 2(2) FIXED(16);
0090          0000      SSFLAG  EQU  0                      FLAGS
0091          0002      SSLORG  EQU  2                      LOW REG
0092          0003      SSHIRG  EQU  3                      HIGH REG
0093                    *                              2*DECLARE
0094                    *                              3*SFDEF LITERALLY '80',   /* SS
0095                    *                              3*SFMR1 LITERALLY '40',   /* ME
0096                    *                              3*SFMR2 LITERALLY '20'    /* ME
0097          0080      SSDEF   EQU  >80
0098          0004      SSMR1   EQU  4                      MEM RNG 1
0099          0100      LBYTE   EQU  256
0100          0000'     PRTSS   EQU  $
0101  0000    0420              BLWP @ACL
      0002    0000
0102                    *                              2*SNAP ENTRY = LOC(SNAP TABLE)
0103  0004    C18A              MOV  R10,R6             SS#
0104  0006    0205              LI   R5,SNPENT          ENTRY LENGTH
      0008    0000
0105  000A    3985              MPY  R5,R6              R6 = SNAP ENTRY PTR
0106  000C    C187              MOV  R7,R6
0107  000E    0226              AI   R6,SNPTAB
      0010    0000
0108                    *                              2*IF (SSDEF &SSFLAG) .EQ. 0 TH
```

```
0109  0012  C016          MOV  *R6,R0              FLAGS
0110  0014  0240          ANDI R0,SSDEF*LBYTE
      0016  8000
0111  0018  1325          JEQ  PRT020
0112                *                              2=CALL PRINT( 'SNAP ',SS#);
0113  001A  022A          AI   R10,CLST
      001C  0000
0114  001E  D81A          MOVB *R10,@SSSTRB
      0020  0000
0115  0022  020A          LI   R10,SSSTR
      0024  0000
0116  0026  06A0          BL   @PRNTC
      0028  0000
0117  002A  1000          NOP
0118                *                              2=CALL PRINT REG(LOW REG,HIGH
0119                *    *GETBUF
0120  002C  0420            BLWP @GETBUF
      002E  0000
0121  0030  C10A          MOV  R10,R4               SAVE BUFFER PTR
0122  0032  0209          LI   R9,2*LBYTE+>C0        AND USE INSPECT WORKSPACE
      0034  02C0
0123  0036  CE89          MOV  R9,*R10+
0124  0038  D266          MOVB @SSLORG(R6),R9       LOW REGISTER
      003A  0002
0125  003C  0989          SRL  R9,8
0126  003E  CE89          MOV  R9,*R10+
0127  0040  D266          MOVB @SSHIRG(R6),R9       HIGH REGISTER
      0042  0003
0128  0044  0989          SRL  R9,8
0129  0046  C689          MOV  R9,*R10
0130  0048  022A          AI   R10,-4               PTR TO CPL
      004A  FFFC
0131  004C  06A0          BL   @IWP
      004E  0000
0132  0050  C284          MOV  R4,R10               RESTORE BUF PTR
0133                *    *RETBUF
0134  0052  0420            BLWP @RETBUF
      0054  0000
0135                *                              2=CALL PRINT MEMORY(MEM RANGE
0136  0056  C266          MOV  @SSMR1(R6),R9
      0058  0004
0137  005A  C2A6          MOV  @SSMR1+2(R6),R10
      005C  0006
0138  005E  06A0          BL   @PRTMEM
      0060  0000
0139  0062  1000          NOP
0140                *                              2=END;
0141          0064' PRT020 EQU  $
0142                *                              1=END PRINT SNAPSHOT
0143  0064  0420          BLWP @RR
      0066  0000
0144  0068  045B          RT
0145                      END
0000 ERS
```

960 - 980 CONCORDANCE

$		0100	0141							
ACL		0054	0101							
CLST		0046	0113							
GETBUF		0052	0120							
IWP		0048	0131							
LBYTE	0099	0110	0122							
PRNTC		0049	0116							
PRT020	0141	0111								
PRTMEM		0047	0138							
PRTSS	0100	0043								
R0	0059	0109	0110							
R1	0060									
R10	0069	0103	0113	0114	0115	0121	0123	0126	0129	0130
		0132	0137							
R11	0070									
R12	0071									
R13	0072									
R14	0073									
R15	0074									
R2	0061									
R3	0062									
R4	0063	0121	0132							
R5	0064	0104	0105							
R6	0065	0103	0105	0106	0107	0109	0124	0127	0136	0137
R7	0066	0106								
R8	0067									
R9	0068	0122	0123	0124	0125	0126	0127	0128	0129	0136
RETBUF		0053	0134							
RR		0055	0143							
SNPENT		0044	0104							
SNPTAB		0045	0107							
SSDEF	0097	0110								
SSFLAG	0090									
SSHIRG	0092	0127								
SSLORG	0091	0124								
SSMR1	0098	0136	0137							
SSSTR		0050	0115							
SSSTRB		0051	0114							

THERE ARE 0037 SYMBOLS

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV	
SHEET	

REV STATUS OF SHEETS	
REV	
SHEET	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS INCORPORATED
Equipment Group Dallas, Texas

AL, PXDATA, READ/WIRTE DATA AREA-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945372-9901

SCALE	REV.	SHEET 1 of 20

TI-0023-C

```
0003                    IDT   !PXDATA!
0004                *TITLE!   PX990 DATA SEGMENT
0005                *         PXDATA
0006                * REVISION!
0007                *         ORIGINAL
0008                *COMPUTER!  990,ASM
0009                *ABSTRACT!  THIS SEGMENT DEFINES THE PX990
0010                *         READ/WRITE DATA AREA
0011                *CALLING SEQUENCE!
0012                *         REF'S TO APPROPRIATE VARIABLES
0013        0005    R5    EQU   5
0014        0007    R7    EQU   7
0015        000A    R10   EQU   10
0016        000D    R13   EQU   13
```

```
0019                    *
0020                    *       DEFINITION'S FOR SVC'S AND PRB I/O CODES
0021                    *
0022          0000      IO      EQU   0
0023                            DEF   BHA
0024          000C      BHA     EQU   >C              BINARY-HEX ASCII
0025                            DEF   BDA
0026          000A      BDA     EQU   >A              BINARY-DECIMAL ASCII
0027                            DEF   HAB
0028          000D      HAB     EQU   >D              HEX ASCII - BINARY
0029                            DEF   DAB
0030          000B      DAB     EQU   >B              DECIMAL ASCII - BINARY
0031                    *
0032                    *
0033                            DEF   OPEN
0034          0000      OPEN    EQU   0               OPEN LUNO
0035                            DEF   RDASC
0036          0009      RDASC   EQU   9               READ ASCII
0037                            DEF   WTASC
0038          000B      WTASC   EQU   >B              WRITE ASCII
```

```
0041                    *
0042                    *       HEX TO BINARY
0043                    *
0044                            DEF   CHB
0045                            DEF   CHBVAL
0046          0000' CHB         EQU   $
0047  0000      0D              BYTE HAB                SVC ID
0048  0001      00              BYTE 0                  ERR RETURN
0049  0002      20  CHBVAL      TEXT '    '             STRING TO BE CONVERTED
0050                            EVEN
0051                    *
0052                    *       BINARY TO HEX
0053                    *
0054                            DEF   CBH
0055                            DEF   CBH0
0056                            DEF   CBH2
0057          0006' CBH         EQU   $
0058  0006      0C              BYTE BHA                SVC ID
0059  0007      00              BYTE 0                  ERR RETURN
0060  0008      20  CBH0        TEXT '  '
0061  000A      20  CBH2        TEXT '    '
0062                            EVEN
0063                    *
0064                    *       BINARY TO DECIMAL ASCII
0065                    *
0066                            DEF   CBD
0067                            DEF   CBDSTG
0068  000E      0A  CBD         BYTE BDA          SVC ID
0069  000F      00              BYTE 0                  ERR RETURN
0070  0010          CBDSTG      BSS  6                  CONVERTED RESULT
0071                            EVEN
```

```
0074                    *
0075                    *       INPUT ONE CHARCTER FROM KEYBOARD
0076                    *       IN CHARACTER MODE.  TEST FOR ESC
0077                    *       INPUT.
0078                    *
0079                            DEF   SVCPRB        TOP OF SVC BLOCK
0080                            DEF   SVCSFG        SYSTEM FLAGS BYTE
0081                            DEF   INCHAR        ONE CHAR INPUT BUFFER
0082                            DEF   SVCUSR        USER FLAGS BYTE
0083                    *
0084  0016    00        SVCPRB BYTE IO                  I/O SVC
0085  0017    00               BYTE 0                   ERR RETURN
0086  0018    09               BYTE RDASC               READ ASCII
0087  0019    00               BYTE 0                   LUNO = KEYBOARD
0088  001A    00        SVCSFG BYTE 0                   SYSTEM FLAGS
0089  001B    10        SVCUSR BYTE >10                 USER FLAGS = CHARACTER MODE
0090  001C  0022'              DATA INCHAR              BUFFER
0091  001E  0001               DATA 1                   BUFFER LENGTH
0092  0020  0000               DATA 0                   CHARACTER COUNT
0093  0022    00        INCHAR BYTE 0
0094                            EVEN
0095                    *
0096                    *
0097                    *       OUTPUT CHARACTER STRING TO 733 PRINTER
0098                    *
0099                            DEF   SVCWRT        TOP OF SVC BLOCK
0100                            DEF   PRBBUF        BUFFER ADDRESS
0101                            DEF   PRBCC         CHARACTER COUNT
0102                            DEF   PRBUER        SYSTEM FLAGS (UNRECOVERABLE ERR BI
0103                    *
0104  0024    00        SVCWRT BYTE IO                  I/O SVC
0105  0025    00               BYTE 0                   ERR RETURN
0106  0026    08               BYTE WTASC               WRITE ASCII
0107  0027    00               BYTE 0                   LUNO = 733 PRINTER
0108  0028    00        PRBUER BYTE 0                   SYSTEM FLAGS
0109  0029    00               BYTE 0                   USER FLAGS
0110  002A              PRBBUF BSS  2                   BUFFER ADDRESS
0111  002C  0000               DATA 0                   UNUSED (BUFFER LENGTH)
0112  002E              PRBCC  BSS  2                   CHARACTER COUNT
0113                    *
0114                    *
0115                    *       SUPERVISOR CALL ENTRY
0116                    *
0117                            DEF SVCALT
0118                            DEF   SVCWP
0119                            REF   SVCSRA
0120  0030  01A6' SVCALT DATA SVCWP
0121  0032  0000               DATA SVCSRA
0122                    *
0123                    *       LOADER PRB & BUFFER
0124                    *
0125                            DEF   LDPRB         TOP FO SVC BLOCK
0126                            DEF   LDOPCD        IO OP CODE
0127                            DEF   LDLUN         IO LUNO
```

```
0128                        DEF   LDFLG        SYSTEM FLAGS
0129                        DEF   LDADDR       BUFFER POINTER
0130                        DEF   LDCC         CHARACTER COUNT
0131                        DEF   LDCBA        CURRENT BUFFER ADDRESS
0132                        DEF   LDBUF        BUFFER
0133                        DEF   LDCBBC
0134                  *
0135  0034    00    LDPRB BYTE  IO                 IO SVC
0136  0035    00          BYTE  0                  ERR RETURN
0137  0036        LDOPCD  BSS   1                  OP CODE
0138  0037        LDLUN   BSS   1                  LUNO
0139  0038    00  LDFLG   BYTE  0                  SYSTEM FLAGS
0140  0039    00          BYTE  0                  USER FLAGS
0141  003A  0044' LDADDR  DATA  LDBUF              BUFFER POINTER
0142  003C  0050          DATA  80                 BUFFER LENGTH
0143  003E  0000  LDCC    DATA  0                  CHARACTER COUNT
0144                  *
0145  0040        LDCBA   BSS   2                  CURRENT BUFFER ADDRESS
0146  0042        LDCBBC  BSS   2                  CURRENT BYTE BIT COUNT
0147                  *
0148  0044        LDBUF   BSS   82                 BUFFER
0149                  *
0150                        EVEN
```

```
0153                    *       LOADER PARM TABLE
0154                    *
0155                            DEF   LDTBL         START OF TABLE
0156                            DEF   LDENTY        ENTRY POINT
0157                            DEF   LDLDPT        LOAT POINT
0158                            DEF   LDNMCC        PROGRAM NAME CHARACTER COUNT
0159                            DEF   LDPGNM        PROGRAM NAME
0160                            DEF   DFBIAS             DEFAULT LOAD BIAS
0161                    *
0162            0096'   LDTBL   EQU   $
0163    0096    0000    LDENTY  DATA  0
0164    0098    0000    LDLDPT  DATA  0
0165    009A      00    LDNMCC  BYTE  0
0166    009B            LDPGNM  BSS   23            (MAX NAME = 23 CHAR)
0167                    *
0168    00B2    00A0    DFBIAS  DATA  >00A0
0169                            EVEN
```

```
0172                    *
0173                    *       CHARACTER DEFINITIONS
0174                    *
0175                            DEF  ZERO
0176                            DEF  BLANK
0177                            DEF  MINUS
0178                            DEF  COMMA
0179                            DEF  PLUS
0180                            DEF  EOR
0181                            DEF  CR
0182                            DEF  PERIOD
0183                            DEF  LF
0184                            DEF  BS
0185                            DEF  ESC
0186                            DEF  RUBOUT
0187                            DEF  TERMCR
0188                    *
0189          000D      CRC     EQU  >D                  CARRIAGE RETURN
0190          000A      LFC     EQU  >A                  LINE FEED
0191          0100      LBYTE   EQU  >100                SHIFT FACTOR TO LEFT BYTE
0192          0020      SP      EQU  ' '
0193  00B4      20      BLANK   TEXT ' '
0194  00B5      30      ZERO    TEXT '0'
0195  00B6      2D      MINUS   TEXT '-'
0196  00B7      2C      COMMA   TEXT ','
0197  00B8      2B      PLUS    TEXT '+'
0198  00B9      2E      PERIOD  TEXT '.'
0199          00BA'     EOR     EQU  $
0200  00BA      0D      CR      BYTE CRC
0201  00BB      0A      LF      BYTE LFC
0202  00BC      08      BS      BYTE >8
0203  00BD      1B      ESC     BYTE >1B
0204  00BE      7F      RUBOUT  BYTE >7F
0205                    *
0206                    *       LAST COMMAND STRING TERMINATOR
0207                    *
0208  00BF              TERMCR  BSS  1
0209                    *
0210                    *       PRINTABLE STRINGS
0211                    *
0212                            EVEN
0213                            DEF  CRLF
0214  00C0      03      CRLF    BYTE 3,>D,>A
      00C1      0D
      00C2      0A
0215                    *
0216                            DEF  PROMPT              MONITOR PROMPT
0217  00C4    030D      PROMPT  DATA ST1-PROMPT-1*LBYTE+CRC
0218  00C6      0A              BYTE LFC,'.'
      00C7      2E
0219          00C8'     ST1     EQU  $
0220                    *
0221                    *
0222                    *
```

```
0223                            EVEN
0224                            DEF  ERSTR              ERROR CODES
0225                            DEF  ERBUF
0226  00C8  0A0D   ERSTR       DATA ST2-ERSTR-1*LBYTE+CRC
0227  00CA    0A                BYTE LFC
0228  00CB    2A                TEXT '**'
0229  00CD          ERBUF       BSS  4                 ERROR CODE
0230  00D1    2A                TEXT '**'
0231        00D3' ST2           EQU  $
0232              *
0233              *
0234              *
0235                            EVEN
0236                            DEF  PTSUM              HEX,DECIMAL SUM
0237  00D4  050A   PTSUM       DATA ST3-PTSUM-1*LBYTE+LFC
0238  00D6    53                TEXT 'SUM='
0239        00DA' ST3           EQU  $
0240              *
0241              *
0242                            EVEN
0243                            DEF  PTDIFF             HEX,DECIMAL DIFFERENCE
0244  00DA  0720   PTDIFF      DATA ST4-PTDIFF-1*LBYTE+SP
0245  00DC    20                TEXT ' DIFF='
0246        00E2' ST4           EQU  $
0247              *
0248              *
0249                            EVEN
0250                            DEF  EQSIGN             EQUAL SYMBOL
0251  00E2  013D   EQSIGN      DATA ST5-EQSIGN-1*LBYTE+'='
0252        00E4' ST5           EQU  $
0253              *
0254              *
0255              *
0256                            EVEN
0257                            DEF  DMPSPR             DUMP SEPARATOR
0258  00E4  013E   DMPSPR      DATA ST6-DMPSPR-1*LBYTE+'>'
0259        00E6' ST6           EQU  $
0260              *
0261              *
0262              *
0263                            EVEN
0264                            DEF  PRTPRD
0265  00E6  012E   PRTPRD      DATA ST7-PRTPRD-1*LBYTE+'.'
0266        00E8' ST7           EQU  $
0267              *
0268              *
0269              *
0270                            EVEN
0271                            DEF  BKSTR
0272                            DEF  BKSTRB
0273  00E8  080D   BKSTR       DATA ST8-BKSTR-1*LBYTE+CRC
0274  00EA    0A                BYTE LFC
0275  00EB    42                TEXT 'BKPT#'
0276  00F0          BKSTRB      BSS  1
0277        00F1' ST8           EQU  $
```

```
0278                    *
0279                    *
0280                    *
0281                            EVEN
0282                            DEF   RGSTR
0283                            DEF   RGSTRB
0284   00F2   0352     RGSTR   DATA  ST9-RGSTR-1*LBYTE+'R'
0285   00F4            RGSTRB  BSS   1
0286   00F5     3D             TEXT  '='
0287           00F6'   ST9     EQU   $
0288                    *
0289                    *
0290                            DEF   SSSTR
0291                            DEF   SSSTRB
0292   00F6   070D     SSSTR   DATA  ST10-SSSTR-1*LBYTE+CRC
0293   00F8     0A             BYTE  LFC
0294   00F9     53             TEXT  'SNAP'
0295   00FD            SSSTRB  BSS   1
0296           00FE'   ST10    EQU   $
0297                            EVEN
```

```
0300                    *
0301                    **TRACE ENTRY DEFINITION
0302                    *
0303                            DEF  TRACE
0304                            DEF  TRTBL
0305                            DEF  TRFLGS,TRLOW,TRHIGH,TRTYPE,TRV1
0306                            DEF  TRLNTH,ISRFLG
0307                            DEF  NUMTR
0308                    *                                DISPLACEMENTS IN TRACE REGION
0309                    *                                  TABLE
0310           0000     TRFLGS EQU  0
0311           0002     TRLOW  EQU  2
0312           0004     TRHIGH EQU  4
0313           0006     TRTYPE EQU  6
0314           0008     TRV1   EQU  8
0315           0004     NUMTR  EQU  4                    NUMBER OF TRACE REGIONS
0316                    *
0317           00FE'    TRTBL  EQU  $
0318   00FE    0000             DATA 0,0,0,0,0,0,0,0
       0100    0000
       0102    0000
       0104    0000
       0106    0000
       0108    0000
       010A    0000
0319           010C'    TRTBL2 EQU  $
0320   010C    0000             DATA 0,0,0,0,0,0,0,0
       010E    0000
       0110    0000
       0112    0000
       0114    0000
       0116    0000
       0118    0000
0321   011A    0000             DATA 0,0,0,0,0,0,0,0
       011C    0000
       011E    0000
       0120    0000
       0122    0000
       0124    0000
       0126    0000
0322   0128    0000             DATA 0,0,0,0,0,0,0,0
       012A    0000
       012C    0000
       012E    0000
       0130    0000
       0132    0000
       0134    0000
0323   0136    00FE'    TRACE  DATA TRTBL,NUMTR
       0138    0004
0324   013A    000E     TRLNTH DATA TRTBL2-TRTBL
0325   013C    0800     ISRFLG DATA >0800                INSTRUCTIION STEP REGION FLAG
0326                    *
0327                            DEF  TTTBL,NUMTTE
0328           013E'    TTTBL  EQU  $
```

```
0329  013E  8000          DATA >8000              PC TRACE
0330  0140  D954          DATA >D954              PC INST WP SE SA DE DA
0331  0142  0800          DATA >0800                 AND BRANCH TARGET
0332  0144  FFFE          DATA >FFFE              EVERYTHING
0333        0004  NUMTTE EQU  $-TTTBL/2
```

```
0336                    *
0337                    **SNAPSHOT DEFNITION
0338                    *
0339                            DEF   SNAPS
0340                            DEF   NUMSNP
0341                            DEF   SNPENT
0342                            DEF   SNPTAB
0343           0008     SNPENT EQU   8                    SNAPSHOT ENTRY SIZE
0344           0004     NUMSNP EQU   4                    NUMBER SNAPSHOTS
0345                    *
0346    0146   014C!    SNAPS  DATA SNPTAB,NUMSNP,SNPENT  SNAPSHOT DEFINITION TABLE
        0148   0004
        014A   0008
0347                    *
0348           014C!    SNPTAB EQU  $
0349    014C   0000             DATA 0,0,0,0
        014E   0000
        0150   0000
        0152   0000
0350    0154   0000             DATA 0,0,0,0
        0156   0000
        0158   0000
        015A   0000
0351    015C   0000             DATA 0,0,0,0
        015E   0000
        0160   0000
        0162   0000
0352    0164   0000             DATA 0,0,0,0
        0166   0000
        0168   0000
        016A   0000
```

```
0355                    *
0356                    **BKPT DEFINITION
0357                    *
0358                            DEF   NUMBPS
0359                            DEF   ENTSIZ
0360                            DEF   BKPT
0361                            DEF   BPTAB
0362          000A      ENTSIZ EQU   10                  BREAKPOINT ENTRY SIZE
0363          0004      NUMBPS EQU   4                   NUMBER BREAKPOINTS
0364                    *
0365  016C    0172'     BKPT   DATA BPTAB,NUMBPS,ENTSIZ   BREAKPOINT DEFINITION TAB
      016E    0004
      0170    000A
0366                    *
0367          0172'     BPTAB  EQU  $
0368  0172    0000            DATA 0,0,0,0,0
      0174    0000
      0176    0000
      0178    0000
      017A    0000
0369  017C    0000            DATA 0,0,0,0,0
      017E    0000
      0180    0000
      0182    0000
      0184    0000
0370  0186    0000            DATA 0,0,0,0,0
      0188    0000
      018A    0000
      018C    0000
      018E    0000
0371  0190    0000            DATA 0,0,0,0,0
      0192    0000
      0194    0000
      0196    0000
      0198    0000
0372                    *
0373                    *       STORE STATUS AND WORKSPACE
0374                    *
0375                            DEF   SIEWP
0376                            DEF   SIEST
0377                            DEF   SIINPG              SIE INPROGRESS FLAG
0378                            DEF   SAVWP
0379  019A    0000      SIEWP  DATA 0
0380  019C    0000      SIEST  DATA 0
0381  019E    0000      SIINPG DATA 0
0382  01A0    0000      SAVWP  DATA 0
0383                    *
0384                    *       COMMAND PARMETER LIST ADDR SAVE WORD
0385                    *
0386                            DEF   CPLSAV
0387  01A2              CPLSAV BSS  2
0388                    *
0389                    *             SAVE WRITE PROTECT BOUNDS TO RESTORE AFTER VIOL
0390                    *
```

```
0391                        DEF  STRPRT
0392  01A4  0000  STRPRT    DATA 0
0393                        *
0394                        *       BACKSPACE FLAG
0395                        *
0396                        DEF  BSFLG
0397  01A6  0000  BSFLG     DATA 0                    BACKSPACE FLAG
```

```
0400                    *
0401                    *       RUN COMMAND INSTRUCTION COUNT
0402                    *
0403                            DEF  INSCNT
0404  01A8  0000       INSCNT DATA 0
0405                    *
0406                    *       USER PROGRAM START VECTOR
0407                    *
0408                            DEF  USRWSP
0409                            DEF  USRWP
0410                            DEF  USRPC
0411                            DEF  USRST
0412                            REF  INIT
0413          0190'     USRWSP EQU  $-R13-R13          WORKSPACE BASE FOR USE
0414  01AA    0000      USRWP  DATA 0                   WITH 'RTWP'
0415  01AC    0000      USRPC  DATA INIT
0416  01AE    0000      USRST  DATA 0
0417                    *
0418                    *       MONITOR PC LIMITS
0419                    *
0420                            DEF  LOWLIM
0421                            DEF  HILIM
0422  01B0    01AC'     LOWLIM DATA INIT
0423                            DEF  TRNARA
0424  01B2    031C'     HILIM  DATA TRNARA
```

```
0427                    *
0428                    *       SUPERVISOR CALL WORKSPACE
0429                    *          (USED TILL DYNAMIC WKSP CAN BE ALLOCATED)
0430                    *
0431                            DEF   SVCW7
0432                            DEF   SVCW10
0433          01A6' SVCWP   EQU   $-R7-R7
0434          01B4' SVCW7   EQU   $
0435          01BA' SVCW10  EQU   SVCWP+R10+R10
0436   01B4                     BSS   32-R7-R7
0437                    *
0438                    *       MEMORY MANAGEMENT WORKSPACE
0439                    *
0440                    *            R5-R15 ONLY
0441                    *
0442                            DEF   MEMWP
0443                    *
0444          01BC' MEMWP   EQU   $-R5-R5
0445   01C6                     BSS   22
0446                    *
0447                    *
0448                    *
0449                    *
0450                    *       FREE MEMORY POOL
0451                    *
0452                            DEF   FREMEM
0453                            DEF   NOCELS
0454                    *
0455          000A  NOCELS  EQU   10
0456   01DC         FREMEM  BSS   32*NOCELS
0457                    *
0458                    *       TRANSIENT AREA
0459                    *
0460                            EVEN
0461          031C' TRNARA  EQU   $
0462                            END
0000 ERS
```

960 - 980 CONCORDANCE

$		0046	0057	0162	0199	0219	0231	0239	0246	0252
		0259	0266	0277	0287	0296	0317	0319	0328	0333
		0348	0367	0413	0433	0434	0444	0461		
BDA	0026	0025	0068							
BHA	0024	0023	0058							
BKPT	0365	0360								
BKSTR	0273	0271	0273							
BKSTRB	0276	0272								
BLANK	0193	0176								
BPTAB	0367	0361	0365							
BS	0202	0184								
BSFLG	0397	0396								
CBD	0068	0066								
CBDSTG	0070	0067								
CBH	0057	0054								
CBH0	0060	0055								
CBH2	0061	0056								
CHB	0046	0044								
CHBVAL	0049	0045								
COMMA	0196	0178								
CPLSAV	0387	0386								
CR	0200	0181								
CRC	0189	0200	0217	0226	0273	0292				
CRLF	0214	0213								
DAB	0030	0029								
DFBIAS	0168	0160								
DMPSPR	0258	0257	0258							
ENTSIZ	0362	0359	0365							
EOR	0199	0180								
EQSIGN	0251	0250	0251							
ERBUF	0229	0225								
ERSTR	0226	0224	0226							
ESC	0203	0185								
FREMEM	0456	0452								
HAB	0028	0027	0047							
HILIM	0424	0421								
INCHAR	0093	0081	0090							
INIT		0412	0415	0422						
INSCNT	0404	0403								
IO	0022	0084	0104	0135						
ISRFLG	0325	0306								
LBYTE	0191	0217	0226	0237	0244	0251	0258	0265	0273	0284
		0292								
LDADDR	0141	0129								
LDBUF	0148	0132	0141							
LDCBA	0145	0131								
LDCBBC	0146	0133								
LDCC	0143	0130								
LDENTY	0163	0156								
LDFLG	0139	0128								
LDLOPT	0164	0157								
LDLUN	0138	0127								
LDNMCC	0165	0158								
LDOPCD	0137	0126								
LDPGNM	0166	0159								
LDPRB	0135	0125								
LDTBL	0162	0155								
LF	0201	0183								
LFC	0190	0201	0218	0227	0237	0274	0293			
LOWLIM	0422	0420								

```
TRTBL   0317        0304   0323   0324
TRTBL2  0319        0324
TRTYPE  0313        0305
TRV1    0314        0305
TTTBL   0328        0327   0333
USRPC   0415        0410
USRST   0416        0411
USRWP   0414        0409
USRWSP  0413        0408
WTASC   0038        0037   0106
ZERO    0194        0175

THERE ARE   0130  SYMBOLS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																														
SHEET																														
REV STATUS OF SHEETS	REV																													
	SHEET																													

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, PXLDDR, LOADER DIRVER-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945373-9901

SCALE	REV.	SHEET
		1 of 10

TI-9923-C

```
0003          *   PROCEDURE LOADER DRIVER(CPL);
0004          *      /* LOADER DRIVER WILL ACCEPT THE
0005          *         INPUT LUNO AND CALL THE LOADER
0006          *         TO LOAD THE USER'S PROGRAM.
0007          *      */
0008          *      LOADER LUNO = 7;  /* DEF = CS1 */
0009          *      IF CPL.PARM1 .NE. NULL THEN
0010          *         LOADER LUNO = CPL.PARM1
0011          *      LOAD PRB LUNO = LOADER LUNO;
0012          *      IF DEVICE TYPE .NE. CS SIGNAL ERROR
0013          *      LOAD PRB OPCODE = 'OPEN';
0014          *      CALL SVC(SVC CALL BLOCK);
0015          *      IF(LD FLAG.LE. 1) THEN
0016          *         IF(CPL PARM2 .NE. NULL)THN
0017          *            BIAS= PARM2;
0018          *         END IF;
0019          *         ELSE BIAS=DEFAULT BIAS;
0020          *      CALL SET LOADER WORKSPACE;
0021          *      CALL SET WORKSPACE(R11,LOC(LOADER IO));
0022          *      CALL SETWORKSPACE(R9,BIAS)
0023          *      CALL LOADER(LOADER IO,
0024          *         BIAS,CASS FLAG)
0025          *      IF ERROR .NE. 0 THEN DO;
0026          *         IF PROGRAM NAME CHAR COUNT .NE. 0
0027          *            THEN CALL PRINT(PROGRAM NAME);
0028          *         IF PROGRAM ENTRY .NE. 0 THEN
0029          *            IF CALLER NE OVLY THEN
0030          *               USER PC = PROGRAM ENTRY
0031          *               IF CALLER EQ LOADUPFRONT THEN
0032          *                  CALL UPFRONT LOADER
0033          *      END;ELSE
0034          *         CALL ERROR('LD00');
0035          *      END;
0036          *      RETURN;
0037          *      PROCEDURE LOADER IO;
0038          *         /* LOADER IO PROVIDES IO SUPPORT
0039          *            FOR THE ROM LOADER.  THE OPEN
0040          *            HAS BEEN PROCESSED BY THE MAIN
0041          *            DRIVER.
0042          *         */
0043          *               CHAR COUNT=CHAR COUNT -1;
0044          *            IF CHAR COUNT =-1 THEN DO;
0045          *               LOAD PRB OPCODE = 'READ';
0046          *               CALL SVC(SVC CALL BLOCK);
0047          *               IF ERROR THEN EXIT;
0048          *               IF EOF THEN EXIT;
0049          *               CURR BUFF ADDR = LOADER BUFFER ADDR;
0050          *               OUTPUT LOAD ADDRESS TO LIGHTS
0051          *            END;
0052          *            CALL MOVE BIT(RETURN VALUE,CURR BUF ADDR);
0053          *               CURR BUFF ADDR = CURR BUFF ADDR+1;
0054          *         END;
0055          *      END LDIO;
0056          *      DO FOREVER;
```

```
0057                *          CALL GETCHAR;
0058                *       END
0059                *    END LOADER DRIVER;
0060                       IDT  'PXLDDR'
0061                * TITLE:    PX990 LOADER DRIVER
0062                *           PXLDDR
0063                * REVISION:
0064                *           ORIGINAL
0065                * COMPUTER: 990,ASM
0066                * ABSTRACT: PXLDDR IS RESPONSIBLE FOR FOUR FUNCTIONS:
0067                *             1) PROCESS LP,LA KEYBOARD COMMANDS AND
0068                *                CALL ROM LOADER AND ABS LOADER RESPECTIVEL
0069                *             2) INTERFACE TO THE ROM LOADER
0070                *             3) PERFORM IO TO SUPPORT THE ROM LOADER.
0071                *             4) CALL ROM LOADER TO LOAD OVERLAY
0072                * CALLING SEQUENCE:
0073                *           1= KEYBOARD COMMAND
0074                *             R10 = POINTER TO COMMAND PARAMETER LIST
0075                *              BL @LOAD
0076                *                LOAD PROGRAM
0077                *              BL   @LDABS
0078                *                LOAD ABSOLUTE
0079                *              BL   @LOADOV
0080                *                LOAD OVERLAY
0081                *              ENTRY
0082                *                R10 = POINTER TO COMMAND PARM LIST
0083                *                R1  = LOAD BIAS FOR OV COMMAND ONLY
0084                *              EXIT
0085                *                R0 NOT EQUAL TO 0 FOR ERROR
0086                *           2= LOADER IO
0087                *              BL @R11          (MAIN DRIVER SETS POINTER
0088                *                                   IN R11)
0089                *
0090                *       REF'S AND DEF'S
0091                *
0092                       DEF  LDUFL
0093                       DEF  LDABS
0094                       DEF  LOAD
0095                       DEF  LOADOV
0096                       REF  SVCALT
0097                       REF  SVCWP
0098                       REF  SVCSR
0099                       REF  SAVWP
0100                       REF  CS
0101                       REF  CLDT
0102                       REF  OVLRET
0103                       REF  LDPRB
0104                       REF  LDOPCD
0105                       REF  LDLUN
0106                       REF  LDFLG
0107                       REF  LDADDR
0108                       REF  LDCC
0109                       REF  LDCBA
0110                       REF  LDBUF
0111                       REF  CR
```

```
0112                    REF   ABSLDR
0113                    REF   ROMLDR
0114                    REF   LDTBL
0115                    REF   PRNTC
0116                    REF   PRCRLF
0117                    REF   TRNARA
0118                    REF   USRPC
0119                    REF   ERROR
0120                    REF   LDCBBC
0121                    REF   LDNMCC
0122                    REF   DFBIAS
0123                    REF   INIT
0124                    REF   CRUOFF
0125                    REF   GETBUF
0126                    REF   RETBUF
0127                    REF   LWP
0128                    REF   RWP
0129                    REF   ACL
0130                    REF   RR
0131              *
0132              *WORKSPACE REGISTER DEFINITIONS
0133              *
0134        0000  R0      EQU   0
0135        0001  R1      EQU   1
0136        0002  R2      EQU   2
0137        0003  R3      EQU   3
0138        0004  R4      EQU   4
0139        0005  R5      EQU   5
0140        0006  R6      EQU   6
0141        0007  R7      EQU   7
0142        0008  R8      EQU   8
0143        0009  R9      EQU   9
0144        000A  R10     EQU   10
0145        000B  R11     EQU   11
0146        000C  R12     EQU   12
0147        000D  R13     EQU   13
0148        000E  R14     EQU   14
0149        000F  R15     EQU   15
0150              *
0151                    DXOP  SVC,15
0152        0100  LBYTE   EQU   256
0153        0000  OPN     EQU   0
0154        0009  READ    EQU   9
0155        2400  LDERR   EQU   >2400
0156        2401  LUNERR  EQU   >2401             INVALID LUNO
0157              *
0158              *       UPFRONT LOADER DEFAULT BIAS
0159              *
0160        FE50  UFLBIA  EQU   ->1B0
0161              *
0162              *                               1*PROCEDURE LOADER DRIVER(CPL)
0163              *                               2*/* LOADER DRIVER WILL ACCEPT
0164              *                               2*   INPUT LUNO AND CALL THE L
0165              *                               2*   TO LOAD THE USER'S PROGRA
0166              *                               2*/
```

```
0167                    *
0168                    ***** UP FRONT LOADER ENTRY POINT ********************
0169          0000' LDUFL   EQU  $
0170  0000  04CE          CLR  R14
0171  0002  1005          JMP  LOA002
0172                    ********* ROM LOADER ENTRY POINT ************
0173          0004' LOAD    EQU  $
0174  0004  020E          LI   R14,1
      0006  0001
0175  0008  1002          JMP  LOA002
0176                    ***** ABS LOADER ENTRY POINT *************
0177          000A' LDABS   EQU  $
0178  000A  020E          LI   R14,2
      000C  0002
0179                    *
0180                    *       XOP INTERRUPT VECTOR
0181          000E' LOA002  EQU  $
0182  000E  0209          LI   R9,SVCWP
      0010  0000
0183  0012  C809          MOV  R9,@>7C
      0014  007C
0184  0016  0209          LI   R9,SVCSR
      0018  0000
0185  001A  C809          MOV  R9,@>7E
      001C  007E
0186                    *
0187  001E  1002          JMP  LOA005
0188                    ****************** OVERLAY ENTRY POINT  ******************
0189          0020' LOADOV  EQU  $
0190  0020  020E          LI   R14,3
      0022  0003
0191          0024' LOA005  EQU  $
0192                    *     *ALLOC,COPY,LINK
0193  0024  0420          BLWP @ACL
      0026  0000
0194  0028  C80D          MOV  R13,@SAVWP
      002A  0000
0195  002C  C0ED          MOV  @R14*2(R13),R3    RETRIEVE R14 FROM OLD WS
      002E  001C
0196  0030  C10A          MOV  R10,R4
0197                    *                                  2=LOADER LUNO = 7;  /* DEF = C
0198  0032  0209          LI   R9,7*LBYTE
      0034  0700
0199                    *                                  2=IF CPL.PARM1 .NE. NULL THEN
0200                    *                                  3=LOADER LUNO = CPL.PARM1
0201  0036  C034          MOV  *R4+,R0
0202  0038  0A90          SLA  R0,9               CHECK PRESENCE BITS
0203  003A  1702          JNC  LOA010
0204  003C  C274          MOV  *R4+,R9
0205  003E  0A89          SLA  R9,8               BYTE SIZE PARM
0206          0040' LOA010  EQU  $
0207                    *                                  2=LOAD PRB LUNO = LOADER LUNO;
0208  0040  D809          MOVB R9,@LDLUN
      0042  0000
0209                    *                                  2=IF DEVICE TYPE .NE. CS SIGNA
```

```
0210  0044  C280          MOV   R9,R10
0211  0046  098A          SRL   R10,8
0212  0048  06A0          BL    @CLDT
      004A  0000
0213  004C  0208          LI    R8,CS
      004E  0000
0214  0050  0989          SRL   R9,8
0215  0052  020A          LI    R10,LUNERR
      0054  2401
0216  0056  8209          C     R9,R8
0217  0058  164C          JNE   LOA052
0218                *                                 2=LOAD PRB OPCODE = 'OPEN';
0219  005A  0209          LI    R9,OPN*LBYTE
      005C  0000
0220  005E  D809          MOVB  R9,@LDOPCD
      0060  0000
0221                *                                 2=CALL SVC(SVC CALL BLOCK);
0222  0062  020A          LI    R10,LDPRB
      0064  0000
0223  0066  0420          BLWP  @SVCALT
      0068  0000
0224                *                                 2=IF(LD FLAG.LE. 1) THEN
0225  006A  0283          CI    R3,1
      006C  0001
0226  006E  180C          JH    LOA014
0227                *                                 3=IF(CPL PARM2 .NE. NULL)THN
0228                *                                 4=BIAS= PARM2;
0229                *                                 3=END IF;
0230                *                                 3=ELSE BIAS=DEFAULT BIAS;
0231  0070  0A10          SLA   R0,1                    CHECK PRESENCE BIT
0232  0072  1809          JOC   LOA012
0233  0074  C0C3          MOV   R3,R3
0234  0076  1605          JNE   LOA011
0235  0078  0201          LI    R1,INIT
      007A  0000
0236  007C  0221          AI    R1,UFLBIA
      007E  FE50
0237  0080  1003          JMP   LOA014
0238        0082' LOA011  EQU   $
0239  0082  C520          MOV   @DFBIAS,*R4
      0084  0000
0240        0086' LOA012  EQU   $
0241  0086  C074          MOV   *R4+,R1
0242        0088' LOA014  EQU   $
0243                *                                 2=CALL SET LOADER WORKSPACE;
0244  0088  04E0          CLR   @LDCC
      008A  0000
0245  008C  0283          CI    R3,2
      008E  0002
0246  0090  1307          JEQ   LOA030
0247                *                                 2=CALL SET WORKSPACE(R11,LOC(L
0248  0092  020B          LI    R11,LDIO
      0094  0102'
0249                *                                 2=CALL SETWORKSPACE(R9,BIAS)
0250  0096  C241          MOV   R1,R9
```

```
0251  0098  04C0          CLR  R0                    SET FLAG FOR CASSETTE LOAD
0252                      *                          2=CALL LOADER(LOADER IO,
0253                      *                          3=BIAS,CASS FLAG)
0254  009A  C0A0          MOV  @ROMLDR,R2
      009C  0000
0255  009E  0452          B    *R2
0256        0000' LOA034  EQU  $
0257  00A0  06A0          BL   @ABSLDR
      00A2  0000
0258  00A4  0000          DATA LDTBL
0259                      *******RETURN FROM ABSLOADER **********
0260  00A6  1023          JMP  LOA050                ERROR RETURN
0261                      *                          2=IF ERROR .NE. 0 THEN DO;
0262                      *                          3=IF PROGRAM NAME CHAR COUNT .
0263                      *                          4=THEN CALL PRINT(PROGRAM NAME
0264  00A8  D0A0          MOVB @LDNMCC,R2
      00AA  0000
0265  00AC  1308          JEQ  LOA035
0266  00AE  06A0          BL   @PRCRLF
      00B0  0000
0267  00B2  1000          NOP
0268  00B4  020A          LI   R10,LDNMCC
      00B6  00AA'
0269  00B8  06A0          BL   @PRNTC
      00BA  0000
0270  00BC  1000          NOP
0271        00BE' LOA035  EQU  $
0272  00BE  C0E0          MOV  @LDTBL,R3
      00C0  00A4'
0273                      ********* RETURN FROM CASSETTE LOADER  ****
0274        00C2' RETLDR  EQU  $
0275  00C2  C360          MOV  @SAVWP,R13
      00C4  002A'
0276  00C6  040D          CLR  *R13
0277  00C8  C0C3          MOV  R3,R3
0278  00CA  1318          JEQ  LOA060
0279                      *                          3=IF PROGRAM ENTRY .NE. 0 THEN
0280                      *                          4=IF CALLER NE OVLY THEN
0281  00CC  C2AD          MOV  @R14*2(R13),R10
      00CE  001C
0282  00D0  028A          CI   R10,3
      00D2  0003
0283  00D4  1313          JEQ  LOA060
0284                      *                          5=USER PC = PROGRAM ENTRY
0285  00D6  C803          MOV  R3,@USRPC
      00D8  0000
0286                      *                          5=IF CALLER EQ LOADUPFRONT THE
0287                      *                          6=CALL UPFRONT LOADER
0288  00DA  C28A          MOV  R10,R10
0289  00DC  160F          JNE  LOA060
0290  00DE  05C3          INCT R3
0291  00E0  0693          BL   *R3
0292  00E2  1005          JMP  LOA050                ERROR RETURN
0293  00E4  C809          MOV  R9,@USRPC
      00E6  00D8'
```

```
0294  00E8  C360          MOV   @SAVWP,R13
      00EA  00C4'
0295  00EC  1007          JMP   LOA060
0296                *                                  2=END;ELSE
0297        00EE' LOA050  EQU   $
0298                *                                  3=CALL ERROR('L000');
0299  00EF  020A          LI    R10,LDERR
      00F0  2400
0300        00F2' LOA052  EQU   $
0301  00F2  06A0          BL    @ERROR
      00F4  0000
0302  00F6  C360          MOV   @SAVWP,R13
      00F8  00EA'
0303  00FA  071D          SETO  *R13
0304                *                                  2=END;
0305        00FC' LOA060  EQU   $
0306                *                                  2=RETURN;
0307                *     *LINK TO PREV WKSP, RET CURR WKSP
0308  00FC  0420            BLWP @RR
      00FE  0000
0309  0100  045B          RT
```

```
0311                    *                                  2=PROCEDURE LOADER IO;
0312                    *                                  3=/* LOADER IO PROVIDES IO SUP
0313                    *                                  3=   FOR THE ROM LOADER.  THE
0314                    *                                  3=   HAS BEEN PROCESSED BY THE
0315                    *                                  3=   DRIVER.
0316                    *                                  3=*/
0317                    *
0318          0102' LDIO    EQU  $
0319                    *          MAY USE R8  THIS SUBROUTINE IS
0320                    *          NON-STANDARD.
0321                    *                                  5=CHAR COUNT=CHAR COUNT -1;
0322  0102  0620             DEC  @LDCC
      0104  008A'
0323  0106  151F             JGT LDI050
0324  0108  131E             JEQ LDI050
0325                    *                                  4=IF CHAR COUNT =-1 THEN DO;
0326                    *                                  5=LOAD PRB OPCODE = 'READ';
0327  010A  0208             LI   R8,READ*LBYTE
      010C  0900
0328  010E  D808             MOVB R8,@LDOPCD
      0110  0060'
0329                    *                                  5=CALL SVC(SVC CALL BLOCK);
0330  0112  020A             LI   R10,LDPRB
      0114  0064'
0331  0116  0420             BLWP @SVCALT
      0118  0068'
0332                    *                                  5=IF ERROR THEN EXIT;
0333  011A  D220             MOVB @LDFLG,R8
      011C  0000
0334  011E  0A28             SLA  R8,2
0335  0120  18E6             JOC  LOA050
0336                    *                                  5=IF EOF THEN EXIT;
0337  0122  0A18             SLA  R8,1
0338  0124  18CE             JOC  RETLDR
0339                    *                                  5=CURR BUFF ADDR = LOADER BUFF
0340  0126  C820             MOV  @LDADDR,@LDCBA
      0128  0000
      012A  0000
0341  012C  C220             MOV  @LDCC,R8
      012E  0104'
0342  0130  DA20             MOVB @CR,@LDBUF(R8)
      0132  0000
      0134  0000
0343                    *                                  5=OUTPUT LOAD ADDRESS TO LIGHT
0344  0136  C28C             MOV  R12,R10
0345  0138  020C             LI   R12,CRUOFF
      013A  0000
0346  013C  3205             LDCR R5,8                    LOAD ADDR TO FP
0347  013E  06C5             SWPB R5
0348  0140  3205             LDCR R5,8
0349  0142  06C5             SWPB R5
0350  0144  C30A             MOV  R10,R12
0351                    *                                  4=END;
0352          0146' LDI050 EQU  $
```

```
 0353                    *                                  4=CALL MOVE BIT(RETURN VALUE,C
 0354  0146  C2A0               MOV   @LDCBA,R10
       0148  012A'
 0355                    *                                  5=CURR BUFF ADDR = CURR BUFF A
 0356  014A  05A0               INC   @LDCBA
       014C  0148'
 0357  014E  D29A               MOVB  *R10,R10
 0358  0150  098A               SRL   R10,8
 0359  0152  A1CA               A     R10,R7
 0360  0154  028A               CI    R10,>3A
       0156  003A
 0361  0158  130D               JEQ   LDI060
 0362  015A  028A               CI    R10,>D
       015C  000D
 0363  015E  1307               JEQ   GETRT
 0364  0160  022A               AI    R10,->30
       0162  FFD0
 0365  0164  028A               CI    R10,>11
       0166  0011
 0366  0168  1102               JLT   GETRT
 0367  016A  022A               AI    R10,-7
       016C  FFF9
 0368        016E' GETRT  EQU   $
 0369  016E  069B               BL    *R11
 0370  0170  10C8               JMP   LDIO
 0371  0172  10BD               JMP   LOA050                RETURN HERE FOR ERROR
 0372                    *                                 3=END;
 0373                    *                                 2=END LDIO;
 0374                    *                                 2=DO FOREVER;
 0375                    *                                 3=CALL GETCHAR;
 0376        0174' LDI060 EQU   $
 0377  0174  06A0               BL    @LDIO                 FLUSH TO END OF RECORD
       0176  0102'
 0378  0178  10FD               JMP   LDI060
 0379                    *                                 2=END
 0380                    *                                 1=END LOADER DRIVER;
 0381                          END
0000 ERS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																		
SHEET																																		
REV STATUS OF SHEETS	REV																																	
	SHEET																																	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DATE
DWN	
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, PX9IO, I/O DISPATCHER-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945374-9901

SCALE	REV.	SHEET
		1 of 21

TI-0923-C

```
0002          *      TITL 'PX990 IO EXECUTIVE
0003          *      DECLARE NUM DEVTYPE LITERALLY '3';
0004          *      DECLARE NUM LUNOS LITERALLY '16';
0005          *      DECLARE DEVNAME TABLE (NUM DEVICES)
0006          *          CHARACTER (3) INITIAL ('LOG', 'NUL',
0007          *          'CS1', 'CS2')
0008          *      DECLARE LOG LITERALLY '0',
0009          *      DECLARE DUM LITERALLY '127';
0010          *      DECLARE CS  LITERALLY '1';
0011          *      DECLARE NUM DEVICES LITERALLY '4';
0012          *      DECLARE DVNAME TO DVTYPE (NUM DEVICES)
0013          *          BIT (8) INITIAL (LOG, DUM, CS, CS);
0014          *      DECLARE DVNAME TO UNITNO (NUM DEVICES)
0015          *          BIT (8) INITIAL (0, 0, 0, 1);
0016          *      DECLARE LUNO TO DVTYPE (NUM LUNOS)
0017          *          BIT (8) INITIAL (LOG,DUM,DUM,DUM,
0018          *          DUM,DUM,LOG,CS,CS,NUL,NUL,NUL,
0019          *          DUM,DUM)
0020          *      DECLARE LUNO TO UNITNO (NUM LUNOS)
0021          *          BIT (8) INITIAL (0, 0, 0, 0, 0, 0, 0,
0022          *    PROCEDURE CONDT (OPID, DEV TYPE, UNIT NO)
0023          *      DECLARE OPID CHARACTER (3);
0024          *      DECLARE DEV TYPE BIT (8);
0025          *      DECLARE UNIT NO BIT (8);
0026          *      DECLARE ERROR TYPE LITERALLY '-1';
0027          *      DECLARE MINUS ONE LITERALLY '-1';
0028          *
0029          *      DEV TYPE = ERROR TYPE;
0030          *      DO I FROM NUM DEVICES-1 TO 0 BY MINUS ONE;
0031          *         IF OPID = DEVNAME TABLE (I)
0032          *            THEN DO;
0033          *               UNIT NO = DVNAME TO UNITNO (I);
0034          *               DEV TYPE = DVNAME TO DVTYPE (I);
0035          *               EXIT DO;
0036          *               END;
0037          *         END;
0038          *      RETURN;
0039          *      END CONDT;
0040          *    PROCEDURE SETLUN (LUNO, DEV TYPE, UNIT NO,
0041          *       ERROR CODE);
0042          *      DECLARE LUNO FIXED (16);
0043          *      DECLARE DEV TYPE BIT (8);
0044          *      DECLARE UNIT NO BIT (8);
0045          *      DECLARE ERROR CODE FIXED (16);
0046          *
0047          *      ERROR CODE = ERROR;
0048          *      IF LUNO > 0 & LUNO <= NUM LUNOS-1 THEN DO;
0049          *         LUNO TO DVTYPE (LUNO) = DEV TYPE;
0050          *         LUNO TO UNITNO (LUNO) = UNIT NO;
0051          *         ERROR CODE = 0;
0052          *         END;
0053          *      RETURN;
0054          *      END SETLUN;
0055          *    PROCEDURE CLDT (LUNO, DEV TYPE, UNIT NO);
```

```
0056                *       DECLARE LUNO FIXED (16);
0057                *       DECLARE (DEV TYPE, UNIT NO) BIT (8);
0058                *
0059                *       DEV TYPE = DUM
0060                *       IF LUNO < NUM LUNOS THEN DO;
0061                *          DEV TYPE = LUNO TO DVTYPE (LUNO);
0062                *          UNIT NO = LUNO TO UNITNO (LUNO);
0063                *          END;
0064                *       RETURN;
0065                *       END CLDT;
0066                *    PROCEDURE IO (SVC PTR);
0067                *       DECLARE SVC PTR POINTER;
0068                *       DECLARE PRB PTR POINTER;
0069                *       DECLARE (1 PHYSICAL RECORD BLOCK,
0070                *                  3 PRB CONTROL (PRB PTR),
0071                *                     5 IO OP BIT (8),
0072                *                     5 LUNO BIT (8),
0073                *                     5 SYSTEM FLAGS,
0074                *                        7 FILLER 1 BIT (1),
0075                *                        7 UNRECOVERABLE ERROR BIT (1),
0076                *                        7 EOF FLAG BIT (1),
0077                *                        7 FILLER 2 BIT (5),
0078                *                     5 USER FLAGS,
0079                *                        7 FILLER 3 BIT (3),
0080                *                        7 CHARACTER IO BIT (1),
0081                *                        7 FILLER 4 BIT (4),
0082                *                     5 BUFFER ADDRESS POINTER,
0083                *                     5 BUFFER LENGTH FIXED (16),
0084                *                     5 CHAR COUNT FIXED (16),
0085                *                     5 PRB RESERVED BIT (32));
0086                *       DECLARE DCT DEV TABLE (NUM DEVICES) POINTER;
0087                *       DECLARE DCT PTR POINTER;
0088                *       DECLARE DCT UNIT TABLE (1) POINTER
0089                *           CONTROL (DCT PTR);
0090                *       DECLARE DCT CRU BASE FIXED (16)
0091                *           CONTROL (DCT PTR);
0092                *       DECLARE READ ASCII CODE LITERALLY '9';
0093                *       DECLARE FWD SPACE CODE LITERALLY '6';
0094                *       DECLARE MAX IO OP LITERALLY '15';
0095                *       DECLARE MIN IO OP LITERALLY '0';
0096                *       DECLARE (DEV TYPE, UNIT NO) BIT (8);
0097                *       DECLARE PARM LIST LITERALLY
0098                *           'PRB PTR, DEV TYPE, UNIT NO, DCT PTR';
0099                *       PRB PTR = SVC PTR + 2;
0100                *       UNRECOVERABLE ERROR = 0;
0101                *       EOF FLAG = 0;
0102                *       CALL CLDT (LUNO, DEV TYPE, UNIT NO)
0103                *       IF DEV TYPE = NUL THEN DO;
0104                *          IF IO OP = READ ASCII CODE
0105                *             IO=OP = FWD SPACE CODE THEN DO;
0106                *             EOF FLAG = 1;
0107                *             CHAR COUNT = 0;
0108                *             END;
0109                *          RETURN;
0110                *          END;
```

```
0111                *       IF IO OP > MAX IO OP THEN RETURN;
0112                *       DCT PTR = DCT DEV TABLE (DEV TYPE);
0113                *       DCT PTR = DCT UNIT TABLE (UNIT NO);
0114                *       DO CASE IO OP;
0115                *          DO CASE DEV TYPE;     /* OPEN */
0116                *             NULL;
0117                *             CALL CSOPEN (PARM LIST);
0118                *             END;
0119                *          NULL;
0120                *          NULL;
0121                *          NULL;
0122                *          NULL;
0123                *          NULL;
0124                *          DO CASE DEV TYPE;     /* FORWARD SPACE */
0125                *             NULL;
0126                *             CALL CSFWD  (PARM LIST);
0127                *             END;
0128                *          DO CASE DEV TYPE;     /* BACKSPACE */
0129                *             NULL
0130                *             CALL CSBACK (PARM LIST);
0131                *             END;
0132                *          NULL;
0133                *          DO CASE DEV TYPE;     /* READ ASCII */
0134                *             CALL LGRASC (PARM LIST);
0135                *             CALL CSRASC (PARM LIST);
0136                *             END;
0137                *          NULL;
0138                *          DO CASE DEV TYPE;     /* WRITE ASCII */
0139                *             CALL LGWASC (PARM LIST);
0140                *             CALL CSWASC (PARM LIST);
0141                *             END;
0142                *          NULL;
0143                *          DO CASE DEV TYPE;     /* WRITE END OF FILE */
0144                *             NULL;
0145                *             CALL CSWEOF (PARM LIST);
0146                *             END;
0147                *          DO CASE DEV TYPE;     /* REWIND */
0148                *             NULL;
0149                *             CALL CSRWND (PARM LIST);
0150                *             END;
0151                *          DO CASE DEV TYPE;     /* UNLOAD */
0152                *             NULL;
0153                *             CALL CSUNLD (PARM LIST);
0154                *             END;
0155                *          END;
0156                *       RETURN;
0157                *       END IO;
0158                        IDT  'PX9IO'
0159                * TITLE:    IOPX9
0160                * REVISION:
0161                *           ORIGINAL
0162                * COMPUTER: 990
0163                * ABSTRACT:
0164                *           A GROUPING OF SUBROUTINES WHICH CONSTITUTE
0165                *           THE PX990-1 IO EXECUTIVE.
```

```
0166                    *           TABLE OF CONTENTS:
0167                    *               1.  LUNO/DEVICE TYPE/DEVICE NAME CONVERSION
0168                    *                       A.  DEVICE NAME TO DEVICE TYPE
0169                    *                       B.  SET DEVICE TYPE IN LUNO TABLE
0170                    *                       C.  LUNO TO DEVICE TYPE
0171                    *                   2.  IO SERVICE DISPATCHER
0172                    *
0173                    *WORKSPACE REGISTER DEFINITIONS
0174                    *
0175          0000      R0      EQU  0
0176          0001      R1      EQU  1
0177          0002      R2      EQU  2
0178          0003      R3      EQU  3
0179          0004      R4      EQU  4
0180          0005      R5      EQU  5
0181          0006      R6      EQU  6
0182          0007      R7      EQU  7
0183          0008      R8      EQU  8
0184          0009      R9      EQU  9
0185          000A      R10     EQU  10
0186          000B      R11     EQU  11
0187          000C      R12     EQU  12
0188          000D      R13     EQU  13
0189          000E      R14     EQU  14
0190          000F      R15     EQU  15
0191                    *
```

```
0194                    *                                   2*DECLARE NUM DEVTYPE LITERALL
0195          0003      NDVTYP EQU  3                        NUMBER OF DEVICE TYPES
0196                    *                                   2*DECLARE NUM LUNOS LITERALLY
0197          0010      NUMLUN EQU  16                       NUMBER OF LUNOS, ZERO-RELATIV
0198                    *                                   2*DECLARE DEVNAME TABLE (NUM D
0199                    *                                   2*    CHARACTER (3) INITIAL ('
0200                    *                                   2*    'CS1', 'CS2')
0201          0000'     DVNAME EQU  $                        DEVICE NAME TABLE
0202  0000      4C              TEXT 'LOG'
0203  0003      44              TEXT 'DUM'
0204  0006      43              TEXT 'CS1'
0205  0009      43              TEXT 'CS2'
0206                    *                                   2*DECLARE LOG LITERALLY '0',
0207                    *                                   2*DECLARE DUM LITERALLY '127';
0208                    *                                   2*DECLARE CS  LITERALLY '1';
0209                    *                                   2*DECLARE NUM DEVICES LITERALL
0210                    *                                   2*DECLARE DVNAME TO DVTYPE (NU
0211                    *                                   2*    BIT (8) INITIAL (LOG, DU
0212                          DEF   LOG,DUM,CS
0213          007F      DUM   EQU   127
0214          0001      CS    EQU   1
0215          0004      NUMDEV EQU  4                        TOTAL NUMBER OF PERIPHERALS
0216                    *                                   CONNECTED
0217                    *
0218          000C'     DVNDVT EQU  $                        DEVICE NAME TO DEVICE TYPE
0219                    *                                   (ORDER MATCHES DVNAME)
0220  000C      00              BYTE LOG
0221  000D      7F              BYTE DUM
0222  000E      01              BYTE CS
0223  000F      01              BYTE CS
0224                    *                                   2*DECLARE DVNAME TO UNITNO (NU
0225                    *                                   2*    BIT (8) INITIAL (0, 0, 0
0226          0010'     DVNUNT EQU  $                        DEVICE NAME TO UNIT NUMBER
0227                    *                                   (ORDER MATCHES DVNAME)
0228  0010      00              BYTE 0
0229  0011      00              BYTE 0
0230  0012      00              BYTE 0
0231  0013      01              BYTE 1
0232                    *                                   2*DECLARE LUNO TO DVTYPE (NUM
0233                    *                                   2*    BIT (8) INITIAL (LOG,DUM
0234                    *                                   2*    DUM,DUM,LOG,CS,CS,NUL,NU
0235                    *                                   2*    DUM,DUM)
0236          0014'     LUNDVT EQU  $                        LOGICAL UNIT NO TO DEV TYPE
0237  0014      00              BYTE LOG,DUM,DUM,DUM
      0015      7F
      0016      7F
      0017      7F
0238  0018      7F              BYTE DUM,DUM,LOG,CS
      0019      7F
      001A      00
      001B      01
0239  001C      01              BYTE CS,DUM,DUM,DUM
      001D      7F
      001E      7F
```

```
        001F    7F
0240    0020    7F          BYTE DUM,DUM,DUM,DUM
        0021    7F
        0022    7F
        0023    7F
0241                *                                   2-DECLARE LUNO TO UNITNO (NUM
0242                *                                   2-    BIT (8) INITIAL (0, 0, 0
0243          0024' LUNUNT EQU  $                        LOGICAL UNIT NO TO PHYS UNIT
0244    0024    00          BYTE 0,0,0,0                (MATCHES LUNDVT)
        0025    00
        0026    00
        0027    00
0245    0028    00          BYTE 0,0,0,0
        0029    00
        002A    00
        002B    00
0246    002C    01          BYTE 1,0,0,0
        002D    00
        002E    00
        002F    00
0247    0030    00          BYTE 0,0,0,0
        0031    00
        0032    00
        0033    00
0248                        EVEN
0249                *
0250          0100  LBYTE   EQU  >100                   MOVE BYTE TO LEFT BYTE
** ERR 2 - RCD 0251 **
0251    0034  1000          TITL'CNVRT DEVICE NAME TO DEVICE TYPE/UNIT 945374-99
```

```
0253                *                                   1*PROCEDURE CDNDT (OPID, DEV T
0254                *                                   2*DECLARE OPID CHARACTER (3);
0255                *                                   2*DECLARE DEV TYPE BIT (8);
0256                *                                   2*DECLARE UNIT NO BIT (8);
0257                *                                   2*DECLARE ERROR TYPE LITERALLY
0258                *                                   2*DECLARE MINUS ONE LITERALLY
0259                * TITLE:     CDNDT
0260                *            CONVERT DEVICE NAME TO DEVICE TYPE / UNIT NO.
0261                * REVISION:
0262                *            ORIGINAL
0263                * COMPUTER:  990
0264                * ABSTRACT:
0265                *            THIS SUBROUTINE CONVERTS A THREE-CHARACTER
0266                *            DEVICE NAME TO A DEVICE TYPE AND PHYSICAL
0267                *            UNIT NUMBER.
0268                * CALLING SEQUENCE:
0269                *            MOV  «ADDRESS OF DEVICE NAME STRING»,R10
0270                *            BL   CDNDT
0271                *            UPON RETURN:
0272                *                 R9L: DEVICE TYPE (ERROR IF = -1)
0273                *                 R8L: PHYSICAL UNIT NUMBER
0274                *
```

```
0276                          DEF  CDNDT
0277          0036' CDNDT     EQU  $                    CONVERT DVNAME TO DVTYPE
0278                *                                   2-
0279                *                                   2-DEV TYPE = ERROR TYPE)
0280                *                                   2-DO I FROM NUM DEVICES-1 TO 0
0281  0036  0209           LI   R9,NUMDEV               SET LOOP VARIABLE
      0038  0004
0282          003A' CDNDT2  EQU  $
0283  003A  0609           DEC  R9                      IF END.OF.LOOP
0284  003C  1110           JLT  CDNDT8
0285                *
0286                *                                   3-IF OPID = DEVNAME TABLE (I)
0287  003E  C009           MOV  R9,R0                   DEVOLOP DVNAME (I)
0288  0040  A009           A    R9,R0
0289  0042  A009           A    R9,R0
0290  0044  0220           AI   R0,DVNAME
      0046  0000'
0291                *
0292  0048  C04A           MOV  R10,R1                  IF NOT EQUAL
0293  004A  9C70           CB   *R0+,*R1+
0294  004C  16F6           JNE  CDNDT2
0295  004E  9C70           CB   *R0+,*R1+
0296  0050  16F4           JNE  CDNDT2
0297  0052  9450           CB   *R0,*R1
0298  0054  16F2           JNE  CDNDT2
0299                *
0300                *                                   4-THEN DO;
0301                *                                   5-UNIT NO = DVNAME TO UNITNO (
0302  0056  D229           MOVB @DVNUNT(R9),R8          SET UNIT NUMBER
      0058  0010'
0303                *                                   5-DEV TYPE = DVNAME TO DVTYPE
0304  005A  D269           MOVB @DVNDVT(R9),R9
      005C  000C'
0305                *                                   5-EXIT DO;
0306                *                                   5-END;
0307                *                                   3-END;
0308          005E' CDNDT8  EQU  $
0309                *                                   2-RETURN;
0310  005E  045B           RT
0311                *                                   2-END CDNDT;
** ERR 2 - RCD 0312 **
0312  0060  1000           TITL'SET DEVICE TYPE/UNIT IN LUNO TABLE 945374-9901*
```

```
0314                *                                   1=PROCEDURE SETLUN (LUNO, DEV
0315                *                                   1=    ERROR CODE);
0316                *                                   2=DECLARE LUNO FIXED (16);
0317                *                                   2=DECLARE DEV TYPE BIT (8);
0318                *                                   2=DECLARE UNIT NO BIT (8);
0319                *                                   2=DECLARE ERROR CODE FIXED (16
0320                *
0321                * TITLE:     SETLUN
0322                *            SET DEVICE TYPE/UNIT IN LUNO TABLE
0323                * REVISION:
0324                *            ORIGINAL
0325                * COMPUTER:  990
0326                * ABSTRACT:
0327                *            THIS SUBROUTINE SETS THE DEVICE TYPE AND
0328                *            PHYSICAL UNIT NUMBER FOR THE INDICATED LOGICAL
0329                *            UNIT NUMBER.
0330                * CALLING SEQUENCE:
0331                *            MOV  <LUNO>,R10
0332                *            MOVB <DEVICE TYPE>,R9
0333                *            MOVB <UNIT NUMBER>,R8
0334                *            BL   SETLUN
0335                *            UPON RETURN:
0336                *                 R7:  ERROR CODE (0 = NO ERROR, =1 = ERROR)
```

```
0338                        DEF  SETLUN
0339          0062' SETLUN  EQU  $
0340                *                                  2*
0341                *                                  2*ERROR CODE = ERROR;
0342  0062  0707           SETO R7
0343                *                                  2*IF LUNO > 0 & LUNO <= NUM LU
0344  0064  C28A           MOV  R10,R10                 IF LUNO OUT OF RANGE
0345  0066  1308           JEQ  SLUN90
0346  0068  028A           CI   R10,NUMLUN
      006A  0010
0347  006C  1405           JHE  SLUN90
0348                *                                  3*LUNO TO DVTYPE (LUNO) = DEV
0349  006E  DA89           MOVB R9,@LUNDVT(R10)         SET DEVICE TYPE
      0070  0014'
0350                *                                  3*LUNO TO UNITNO (LUNO) = UNIT
0351  0072  DA88           MOVB R8,@LUNUNT(R10)         SET PHYSICAL UNIT NUMBER
      0074  0024'
0352                *                                  3*ERROR CODE = 0;
0353  0076  04C7           CLR  R7                      CLEAR ERROR CODE
0354                *                                  3*END;
0355                *                                  2*RETURN;
0356                *                                  2*END SETLUN;
0357          0078' SLUN90  EQU  $
0358  0078  045B           RT
```

```
0361                *                                   1=PROCEDURE CLDT (LUNO, DEV TY
0362                *                                   2=DECLARE LUNO FIXED (16);
0363                *                                   2=DECLARE (DEV TYPE, UNIT NO)
0364                * TITLE:     CLDT
0365                *            CONVERT LUNO TO DEVICE TYPE AND UNIT
0366                * REVISION:
0367                *            ORIGINAL
0368                * COMPUTER:  990
0369                * ABSTRACT:
0370                *            THIS SUBROUTINE RETURNS THE DEVICE TYPE AND
0371                *            PHYSICAL UNIT NUMBER FOR THE GIVEN LUNO.
0372                * CALLING SEQUENCE:
0373                *            MOV   <LUNO>,R10
0374                *            BL    CLDT
0375                *            UPON RETURN:
0376                *                  R9L:  DEVICE TYPE (ERROR = -1)
0377                *                  R8L:  UNIT NUMBER
0378                *
```

```
0380                    *                                   2*
0381                            DEF   CLDT
0382          007A'     CLDT    EQU   $                      CONVERT LUN TO DEVICE TYPE
0383                    *
0384                    *                                   2*DEV TYPE = DUM
0385    007A  0209              LI    R9,DUM*LBYTE
        007C  7F00
0386                    *                                   2*IF LUNO < NUM LUNOS THEN DO;
0387    007E  028A              CI    R10,NUMLUN             IF NOT IN RANGE
        0080  0010
0388    0082  1404              JHE   CLDT90
0389                    *
0390                    *                                   3*DEV TYPE = LUNO TO DVTYPE (L
0391    0084  D26A              MOVB  @LUNDVT(R10),R9        SET DEVICE TYPE
        0086  0014'
0392                    *                                   3*UNIT NO = LUNO TO UNITNO (LU
0393    0088  D22A              MOVB  @LUNUNT(R10),R8        SET PHYSICAL UNIT NUMBER
        008A  0024'
0394                    *                                   3*END;
0395          008C'     CLDT90  EQU   $
0396                    *                                   2*RETURN;
0397    008C  045B              RT
0398                    *                                   2*END CLDT;
```

```
0401                    *
0402                    *
0403                    *
0404                    *
0405                    *
0406                    *
0407                    *
0408                    *
0409                    *
0410                    *
0411                    *
0412                    *
0413                    *
0414                    *
0415                    *
0416                    *
0417                    *
0418                    *
0419                    *
0420                    *
0421                    *
0422                    *
0423                    *
0424                    *
0425                    *
0426                    *
0427                    *
0428          0000      IOOP    EQU   0
0429          0001      LUNO    EQU   1
0430          0002      SYSFLG  EQU   2
0431          0003      USEFLG  EQU   3
0432          0004      BFADDR  EQU   4
0433          0006      BFLNTH  EQU   6
0434          0008      CCOUNT  EQU   8
0435                    *
0436          000F      UE      EQU   16-1
0437          000E      EOFLG   EQU   16-2
0438          000D      CHRMOD  EQU   16-3
0439                    *
0440                            DEF   UEMASK
0441  008E      40      UEMASK  BYTE  >40
0442                            DEF   EFMASK
0443  008F      20      EFMASK  BYTE  >20
0444                            DEF   CMDMSK
0445  0090      10      CMDMSK  BYTE  >10
0446                            DEF   UEFMSK
0447  0091      60      UEFMSK  BYTE  >40+>20
0448                    *
0449                    *
0450          0000      DCTCRU  EQU   0
0451                    *
0452                    *
0453                    *
0454                    *
```

```
1-PROCEDURE IO (SVC PTR);
2-DECLARE SVC PTR POINTER;
2-DECLARE PRB PTR POINTER;
2-DECLARE (1 PHYSICAL RECORD B
3-          3 PRB CONTROL (PRB
4-          5 IO OP BIT (8),
4-          5 LUNO BIT (8),
4-          5 SYSTEM FLAGS,
5-          7 FILLER 1 BIT (1),
5-          7 UNRECOVERABLE ERR
5-          7 EOF FLAG BIT (1),
5-          7 FILLER 2 BIT (5),
4-          5 USER FLAGS,
5-          7 FILLER 3 BIT (3),
5-          7 CHARACTER IO BIT
5-          7 FILLER 4 BIT (4),
4-          5 BUFFER ADDRESS PO
4-          5 BUFFER LENGTH FIX
4-          5 CHAR COUNT FIXED
4-          5 PRB RESERVED BIT
2-DECLARE DCT DEV TABLE (NUM D
2-DECLARE DCT PTR POINTER;
2-DECLARE DCT UNIT TABLE (1) P
2-    CONTROL (DCT PTR);

 PRB EQUATES
 DISPLACEMENTS:
     IO OP CODE
     LOGICAL UNIT NUMBER
     SYSTEM FLAGS
     USER FLAGS
     BUFFER ADDRESS
     BUFFER LENGTH
     CHARACTER COUNT
 FLAG SHIFT COUNTS:
     UNRECOVERABLE ERROR
     END.OF.FILE FLAG
     CHARACTER MODE
 FLAG SET MASKS:
     UNRECOVERABLE ERROR

     END.OF.FILE FLAG

     CHARACTER MODE

     UN.ERROR AND EOF FLAG

2-DECLARE DCT CRU BASE FIXED (
2-    CONTROL (DCT PTR);

 DCT MAPPING TABLES

 MAP DEVICE TYPE
```

```
0455                    *
0456          0092' DCTDT   EQU  $
0457    0092    02          BYTE UTLOG-DCTDT
0458    0093    04          BYTE UTCST-DCTDT
0459                        EVEN
0460                    *
0461                    *
0462                    *
0463          0094' UTLOG   EQU  $
0464                        REF  DC7331
0465    0094  0000          DATA DC7331
0466          0096' UTCST   EQU  $
0467    0096  0094'          DATA DC7331
0468    0098  0096'          DATA DC7331
0469                    *
0470                    *
0471          0009  RDACOD  EQU  9
0472                    *
0473          0006  FSPCOD  EQU  6
0474                    *
0475          000F  MXIOOP  EQU  15
0476                    *
0477                    *
0478                    *
0479                    *
0480                    *
```

```
    (ONE ENTRY PER DEV TYPE)

    LOG
    CASSETTE

MAP UNIT TO DCT
    (ONE TABLE PER DEV TYPE)
    (ONE ENTRY PER PHYS UNIT)
LOG

    UNIT 0
CASSETTE
    UNIT 0
    UNIT 1

2-DECLARE READ ASCII CODE LITE
 READ ASCII IO OP CODE
2-DECLARE FWD SPACE CODE LITER
 FORWARD SPACE IO OP CODE
2-DECLARE MAX IO OP LITERALLY
 MAXIMUM IO OP
2-DECLARE MIN IO OP LITERALLY
 (IMPLICIT)
2-DECLARE (DEV TYPE, UNIT NO)
2-DECLARE PARM LIST LITERALLY
2-    'PRB PTR, DEV TYPE, UNIT
```

```
0482                          DEF  IO                    IO SERVICE DISPATCHER
0483          009A' IO        EQU  $
0484                *                                   2=PRB PTR = SVC PTR + 2;
0485  009A  05CA             INCT R10                    SET PRB ADDRESS
0486  009C  C08B             MOV  R11,R2
0487  009E  C0CA             MOV  R10,R3
0488                *                                   2=UNRECOVERABLE ERROR = 0;
0489                *                                   2=EOF FLAG = 0;
0490  00A0  58E0             SZCB @UEFMSK,@SYSFLG(R3)
      00A2  0091'
      00A4  0002
0491                *                                   2=CALL CLDT (LUNO, DEV TYPE, U
0492  00A6  D2A3             MOVB @LUNO(R3),R10          SET LUNO
      00A8  0001
0493  00AA  098A             SRL  R10,8
0494  00AC  06A0             BL   @CLDT
      00AE  007A'
0495                *                                   2=IF DEV TYPE = NUL THEN DO;
0496  00B0  0989             SRL  R9,8                   (DEVICE TYPE)
0497  00B2  0988             SRL  R8,8                   (UNIT NUMBER)
0498  00B4  D1E3             MOVB @IOOP(R3),R7           (IO OP CODE)
      00B6  0000
0499  00B8  0987             SRL  R7,8
0500  00BA  0289             CI   R9,DUM
      00BC  007F
0501  00BE  160C             JNE  IO020
0502                *                                   3=IF IO OP = READ ASCII CODE
0503                *                                   3=   IO=OP = FWD SPACE CODE TH
0504  00C0  0287             CI   R7,RDACOD              IF NOT READ OR FORWARD SPACE
      00C2  0009
0505  00C4  1303             JEQ  LL2
0506  00C6  0287             CI   R7,FSPCOD
      00C8  0006
0507  00CA  1605             JNE  IO010
0508          00CC' LL2       EQU  $
0509                *
0510                *                                   4=EOF FLAG = 1;
0511  00CC  F8E0             SOCB @EFMASK,@SYSFLG(R3)
      00CE  008F'
      00D0  0002
0512                *                                   4=CHAR COUNT = 0;
0513  00D2  04E3             CLR  @CCOUNT(R3)
      00D4  0008
0514                *                                   4=END;
0515          00D6' IO010     EQU  $
0516                *                                   3=RETURN;
0517  00D6  0452             B    *R2
0518                *                                   3=END;
0519          00D8' IO020     EQU  $
0520                *                                   2=IF IO OP > MAX IO OP THEN RE
0521  00D8  0287             CI   R7,MXIOOP
      00DA  000F
0522  00DC  1201             JLE  LL4
0523  00DE  0452             B    *R2
```

```
0524          00E0' LL4     EQU  $
0525                 *
0526                 *                                  2=DCT PTR = DCT DEV TABLE (DEV
0527  00E0  D1A9            MOVB @DCTDT(R9),R6          (DISP TO BEGINNING OF TABLE)
      00E2  0092'
0528  00E4  0986            SRL  R6,8
0529                 *                                  2=DCT PTR = DCT UNIT TABLE (UN
0530  00E6  C148            MOV  R8,R5                  (UNIT*2)
0531  00E8  0A15            SLA  R5,1
0532  00EA  A185            A    R5,R6
0533  00EC  C1A6            MOV  @DCTDT(R6),R6
      00EE  0092'
0534                 *                                  2=DO CASE IO OP;
0535                 *                                  3=DO CASE DEV TYPE;     /* OPEN
0536                 *                                  4=NULL;
0537                 *                                  4=CALL CSOPEN (PARM LIST);
0538                 *                                  4=END;
0539                 *                                  3=NULL;
0540                 *                                  3=NULL;
0541                 *                                  3=NULL;
0542                 *                                  3=NULL;
0543                 *                                  3=NULL;
0544                 *                                  3=DO CASE DEV TYPE;     /* FORW
0545                 *                                  4=NULL;
0546                 *                                  4=CALL CSFWD  (PARM LIST);
0547                 *                                  4=END;
0548                 *                                  3=DO CASE DEV TYPE;     /* BACK
0549                 *                                  4=NULL
0550                 *                                  4=CALL CSBACK (PARM LIST);
0551                 *                                  4=END;
0552                 *                                  3=NULL;
0553                 *                                  3=DO CASE DEV TYPE;     /* READ
0554                 *                                  4=CALL LGRASC (PARM LIST);
0555                 *                                  4=CALL CSRASC (PARM LIST);
0556                 *                                  4=END;
0557                 *                                  3=NULL;
0558                 *                                  3=DO CASE DEV TYPE;     /* WRIT
0559                 *                                  4=CALL LGWASC (PARM LIST);
0560                 *                                  4=CALL CSWASC (PARM LIST);
0561                 *                                  4=END;
0562                 *                                  3=NULL;
0563                 *                                  3=DO CASE DEV TYPE;     /* WRIT
0564                 *                                  4=NULL;
0565                 *                                  4=CALL CSWEOF (PARM LIST);
0566                 *                                  4=END;
0567                 *                                  3=DO CASE DEV TYPE;     /* REWI
0568                 *                                  4=NULL;
0569                 *                                  4=CALL CSRWND (PARM LIST);
0570                 *                                  4=END;
0571                 *                                  3=DO CASE DEV TYPE;     /* UNLO
0572                 *                                  4=NULL;
0573                 *                                  4=CALL CSUNLD (PARM LIST);
0574                 *                                  4=END;
0575                 *                                  3=END;
0576  00F0  D167            MOVB @IOPTBL(R7),R5         DISP TO DEV TYPE TABLE BY OPC
```

```
      00F2  011A'
0577  00F4  1311          JEQ   IO050               IF NULL ON IO OP CODE
0578  00F6  0985          SRL   R5,8
0579  00F8  C109          MOV   R9,R4
0580  00FA  0A14          SLA   R4,1                (DEV TYPE * 2)
0581  00FC  A144          A     R4,R5
0582  00FE  C165          MOV   @IOPTBL(R5),R5      ADDRESS OF CALLED SUBROUTINE
      0100  011A'
0583  0102  130A          JEQ   IO050               IF NULL
0584                      REF   ACL
0585  0104  0420          BLWP  @ACL
      0106  0000
0586  0108  C283          MOV   R3,R10              MOVE PRB POINTER TO NEW WKSP
0587  010A  C240          MOV   R6,R9               SET CRU BASE ADDRESS
0588  010C  C329          MOV   @DCTCRU(R9),R12
      010E  0000
0589  0110  C045          MOV R5,R1                 CALL SUBROUTINE
0590  0112  0691          BL    *R1
0591                      REF   RR
0592  0114  0420          BLWP  @RR
      0116  0000
0593                *
0594        0118' IO050   EQU   $
0595                *                               2=RETURN;
0596  0118  0452          B     *R2
0597                *                               2=END IO;
0598        011A' IOPTBL  EQU   $                     KEYED BY IO OP CODE
0599  011A    10          BYTE  OPEN-IOPTBL         00: OPEN
0600  011B    00          BYTE  0,0,0,0,0
      011C    00
      011D    00
      011E    00
      011F    00
0601  0120    14          BYTE  FSPACE-IOPTBL       06: FORWARD SPACE
0602  0121    18          BYTE  BSPACE-IOPTBL       07: BACK SPACE
0603  0122    00          BYTE  0
0604  0123    1C          BYTE  RASCII-IOPTBL       09: READ ASCII
0605  0124    00          BYTE  0
0606  0125    20          BYTE  WASCII-IOPTBL       11: WRITE ASCII
0607  0126    00          BYTE  0
0608  0127    24          BYTE  WEOF-IOPTBL         13: WRITE END.OF.FILE
0609  0128    28          BYTE  REWIND-IOPTBL       14: REWIND
0610  0129    2C          BYTE  UNLOAD-IOPTBL       15: UNLOAD
0611        012A' OPEN    EQU   $
0612  012A  0000          DATA  0
0613                      REF   CSOPEN
0614  012C  0000          DATA  CSOPEN
0615        012E' FSPACE  EQU   $
0616  012E  0000          DATA  0
0617                      REF   CSFWD
0618  0130  0000          DATA  CSFWD
0619        0132' BSPACE  EQU   $
0620  0132  0000          DATA  0
0621                      REF   CSBACK
0622  0134  0000          DATA  CSBACK
```

```
 0623                          EVEN
 0624          0136' RASCII    EQU  $
 0625                          REF  LGRASC
 0626  0136  0000              DATA LGRASC
 0627                          REF  CSRASC
 0628  0138  0000              DATA CSRASC
 0629          013A' WASCII    EQU  $
 0630                          REF  LGWASC
 0631  013A  0000              DATA LGWASC
 0632                          REF  CSWASC
 0633  013C  0000              DATA CSWASC
 0634          013E' WEOF      EQU  $
 0635  013E  0000              DATA 0
 0636                          REF  CSWEOF
 0637  0140  0000              DATA CSWEOF
 0638          0142' REWIND    EQU  $
 0639  0142  0000              DATA 0
 0640                          REF  CSRWND
 0641  0144  0000              DATA CSRWND
 0642          0146' UNLOAD    EQU  $
 0643  0146  0000              DATA 0
 0644                          REF  CSUNLD
 0645  0148  0000              DATA CSUNLD
 0646                          END
LOG
 0002 ERS
```

960 - 980 CONCORDANCE

```
$                    0201  0218  0226  0236  0243  0277  0282  0308  0339
                     0357  0382  0395  0456  0463  0466  0483  0508  0515
                     0519  0524  0594  0598  0611  0615  0619  0624  0629
                     0634  0638  0642
ACL                  0584  0585
BFADDR  0432
BFLNTH  0433
BSPACE  0619         0602
CCOUNT  0434         0513
CDNDT   0277         0276
CDNDT2  0282         0294  0296  0298
CDNDT8  0308         0284
CHRMOD  0438
CLDT    0382         0381  0494
CLDT90  0395         0388
CMDMSK  0445         0444
CS      0214         0212  0222  0223  0238  0239
CSBACK               0621  0622
CSFWD                0617  0618
CSOPEN               0613  0614
CSRASC               0627  0628
CSRWND               0640  0641
CSUNLD               0644  0645
CSWASC               0632  0633
CSWEOF               0636  0637
DC7331               0464  0465  0467  0468
DCTCRU  0450         0588
DCTDT   0456         0457  0458  0527  0533
DEVICE               0251  0312
DUM     0213         0212  0221  0237  0237  0237  0238  0238  0239  0239
                     0239  0240  0240  0240  0240  0385  0500
DVNAME  0201         0290
DVNDVT  0218         0304
DVNUNT  0226         0302
EFMASK  0443         0442  0511
EOFLG   0437
FSPACE  0615         0601
FSPCOD  0473         0506
IO      0483         0482
IO010   0515         0507
IO020   0519         0501
IO050   0594         0577  0583
IOOP    0428         0498
IOPTBL  0598         0576  0582  0599  0601  0602  0604  0606  0608  0609
                     0610
LBYTE   0250         0385
LGRASC               0625  0626
LGWASC               0630  0631
LL2     0508         0505
LL4     0524         0522
LOG                  0212  0220  0237  0238
LUNDVT  0236         0349  0391
LUNO    0429         0492
LUNUNT  0243         0351  0393
MXIOOP  0475         0521
NDVTYP  0195
NUMDEV  0215         0281
NUMLUN  0197         0346  0387
OPEN    0611         0599
R0      0175         0287  0288  0289  0290  0293  0295  0297
```

```
R1      0176      0292  0293  0295  0297  0589  0590
R10     0185      0292  0344  0344  0346  0349  0351  0387  0391  0393
                  0485  0487  0492  0493  0586
R11     0186      0486
R12     0187      0588
R13     0188
R14     0189
R15     0190
R2      0177      0486  0517  0523  0596
R3      0178      0487  0490  0492  0498  0511  0513  0586
R4      0179      0579  0580  0581
R5      0180      0530  0531  0532  0576  0578  0581  0582  0582  0589
R6      0181      0527  0528  0532  0533  0533  0587
R7      0182      0342  0353  0498  0499  0504  0506  0521  0576
R8      0183      0302  0351  0393  0497  0530
R9      0184      0281  0283  0287  0288  0289  0302  0304  0304  0349
                  0385  0391  0496  0500  0527  0579  0587  0588
RASCII  0624      0604
RDACOD  0471      0504
REWIND  0638      0609
RR                0591  0592
SETLUN  0339      0338
SLUN90  0357      0345  0347
SYSFLG  0430      0490  0511
UE      0436
UEFMSK  0447      0446  0490
UEMASK  0441      0440
UNLOAD  0642      0610
USEFLG  0431
UTCST   0466      0458
UTLOG   0463      0457
WASCII  0629      0606
WEOF    0634      0608

THERE ARE   0086  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV

SHEET

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS INCORPORATED
Equipment Group *Dallas, Texas*

AL, RANGE, EXTRACT RANGE OF PARAMETERS-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945375-9901

SCALE | REV. | SHEET 1 of 5

TI-0023-C

```
0003                    IDT  'RANGE'
0004                * TITLE:    RANGE
0005                *           RANGE EXTRACTER AND VERIFICATION ROUTINE
0006                * REVISION:
0007                *           ORIGINAL
0008                * COMPUTER: 990,ASM
0009                * ABSTRACT: THIS ROUTINE, WHEN GIVEN A PARM LIST, WILL SET
0010                *           UP THE APPROPRIATE LIMITS IN R9,R10.  IT ALSO
0011                *           VERIFIES,THAT LOWER LIMIT < UPPER LIMIT.  IF
0012                *           THIS ISN'T TRUE, THEN EXIT IS TO
0013                *           WD FOLLOWING BL.
0014                *           1) BOTH PARMS NULL = R9=0 AND R10=MAXIMUM VALUE
0015                *           2) LOWER NULL,UPPER ENTERED = R9=0 AND R10=PARM
0016                *           3) LOWER NONNULL,UPPER NULL = R9 AND R10=PARM
0017                *           4) BOTH PARMS NON-NULL = R9=PARM AND R10=PARM'
0018                *
0019                * CALLING SEQUENCE:
0020                *           R10=PARM LIST
0021                *           R8=NUMBER OF VALUES (0 >= NO MAX CHECK)
0022                *           BL   @RANGEX
0023                *           JMP  ERROR          ERROR EXIT
0024                *
0025                *           USES R1,R9,R10
0026                *
0027                *  REFS, EQU'S ,DATA
0028                *
0029                        DEF  RANGEX
0030                        REF  ERROR
0031          1113  RNGERR  EQU  >1113              UPPER LIMIT > MAX
0032          0005  PRBITS  EQU  5
0033          0006  PRBTSM  EQU  6
0034          0001  R1      EQU  1
0035          0008  R8      EQU  8
0036          0009  R9      EQU  9
0037          000A  R10     EQU  10
0038          000B  R11     EQU  11
```

```
0040          0000' JMPTAB EQU  $
0041  0000  1003         JMP  RAN000
0042  0002  100C         JMP  RAN010
0043  0004  100E         JMP  RAN020
0044  0006  1010         JMP  RAN030
0045                RANGEX
0046                *
0047                *  SET UP AND B TO APPROPRIATE PARAMETER EXTRACTER
0048                *
0049  0008  C07A         MOV  *R10+,R1          R1=PRESENCE FLAGS
0050  000A  0851         SRA  R1,PRBITS
0051  000C  0241         ANDI R1,PRBTSM
      000E  0006
0052  0010  0461         B    @JMPTAB(R1)
      0012  0000'
0053                *
0054                *  BOTH PARMS NULL - INDIC EVERYTHING(0 = MAX VAL )
0055                *
0056  0014  04C9  RAN000 CLR  R9
0057  0016  C288         MOV  R8,R10
0058  0018  060A         DEC  R10
0059  001A  100E         JMP  RAN045
0060                *
0061                *  ONLY UPPER LIMIT PRESENT - ZERO LOWER LIMIT
0062                *
0063  001C  04C9  RAN010 CLR  R9
0064  001E  C29A         MOV  *R10,R10
0065  0020  1005         JMP  RAN040
0066                *
0067                *  ONLY LOWER LIMIT PRESENT - SINGLE CONSTRUCT ONLY
0068                *
0069  0022  C25A  RAN020 MOV  *R10,R9
0070  0024  C289         MOV  R9,R10
0071  0026  1002         JMP  RAN040
0072                *
0073                *  BOTH LIMITS PRESENT - LOWER AND UPPER
0074                *
0075  0028  C27A  RAN030 MOV  *R10+,R9
0076  002A  C29A         MOV  *R10,R10
0077                *
0078                *  CHECK LIMIT ORDER AND RETURN
0079                *
0080          002C' RAN040 EQU  $
0081  002C  8289         C    R9,R10
0082  002E  1806         JH   RAN050
0083  0030  C208         MOV  R8,R8
0084  0032  1302         JEQ  RAN045
0085  0034  820A         C    R10,R8
0086  0036  1402         JHE  RAN050
0087          0038' RAN045 EQU  $
0088  0038  05CB         INCT R11
0089  003A  045B         RT
0090          003C' RAN050 EQU  $
0091  003C  020A         LI   R10,RNGERR
```

```
        003E  1113
 0092   0040  0460          R     *ERROR                AND RETURN TO CALLER
        0042  0000
 0093          0044' RAN060 EQU   $
 0094   0044  045D          RT
 0095                       END
0000 ERS
```

```
960 - 980  CONCORDANCE
$                       0039   0079   0086   0089   0092
ERROR                   0029   0091
JMPTAB  0039            0051
PRBITS  0031            0049
PRBTSM  0032            0050
R1      0033            0048   0049   0050   0051
R10     0036            0048   0056   0057   0063   0063   0068   0069   0074   0075
                        0075   0080   0084   0090
R11     0037            0087
R8      0034            0056   0082   0082   0084
R9      0035            0055   0062   0068   0069   0074   0080
RAN000  0055            0040
RAN010  0062            0041
RAN020  0068            0042
RAN030  0074            0043
RAN040  0079            0064   0070
RAN045  0086            0058   0083
RAN050  0089            0081   0085
RAN060  0092
RANGEX  0044            0028
RNGERR  0030            0090

THERE ARE   0020  SYMBOLS
```

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																														
SHEET																														
REV STATUS OF SHEETS	REV																													
	SHEET																													

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, SB, SET BREAKPOINT-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945377-9901

SCALE	REV.	SHEET 1 of 5

TI-0023-C

```
0003                        IDT  'SB'
0004                 * TITLE:    SBP
0005                 *           SET BREAKPOINT CMD PROCESSOR
0006                 * REVISION:
0007                 *           ORIGINAL
0008                 * COMPUTER: 990,ASM
0009                 * ABSTRACT: THIS ROUTINE PROCESSES THE SB COMMAND.  IT FIRS
0010                 *           VERIFIES AND OBTAINS THE BREAKPT NUMBER AND
0011                 *           PC LOCATION.  THESE ARE PLACED IN THE BRKPT
0012                 *           TABLE.  THEN, A REFERENCE COUNT IS ENTERED INTO
0013                 *           THE TABLE ENTRY.  IF NULL, A 1 IS ASSUMED.  THE
0014                 *           LAST PARM,IF PRESENT, IS A SNAP #.  IT MUST
0015                 *           BE DEFINED.  AFTER THESE DATA ITEMS ARE PLACED
0016                 *           IN THE TABLE, THE BP IS MARKED AS DEFINED.  ANY
0017                 *           ERRORS RESULT IN A NASTY MESSAGE AND RETURN TO
0018                 *           COMMAND PROCESSOR.
0019                 * CALLING SEQUENCE:
0020                 *           R10=CMD PARM LIST PTR
0021                 *           CMD PARM LIST= PARM#/PRESENCE BITS
0022                 *                          PARMS
0023                 *           BL   @SBP
0024                 *
0025                 *           USES R0,R1,R2,R5,R9,R15
0026                 *
0027                 *
0028                 *  REFS, DEFS, EQUS
0029                 *
0030                       DEF  SBP
0031                       REF  BPTAB - BREAKPOINT DEFINITION TABLE
0032                       REF  NUMBPS - NUMBER OF BREAKPOINTS
0033                       REF  ENTSIZ - ENTRY BLOCK SIZE
0034                       REF  NUMSNP - NUMBER OF SNAPS
0035                       REF  SNPTAB - SNAPSHOT TABLE
0036                       REF  SNPENT - SNAPSHOT ENTRY SIZE
0037                       REF  ERROR  - ERROR PROCESSOR
0038  0000    C0   FLAGSN  BYTE >C0
0039  0001    80   FLAG    BYTE >80
0040        0008   PRBITS  EQU  8
0041        0001   S1      EQU  1
0042        1120   DP20    EQU  >1120              BREAKPOINT ERROR
0043        0000   R0      EQU  0
0044        0001   R1      EQU  1
0045        0002   R2      EQU  2
0046        0005   R5      EQU  5
0047        0009   R9      EQU  9
0048        000A   R10     EQU  10
0049        000B   R11     EQU  11
0050        000F   R15     EQU  15
```

```
0052                    SBP
0053                    *
0054                    *  GET INDEX NUMBER AND FIND ENTRY BLOCK
0055                    *
0056  0002  C03A               MOV   *R10+,R0          R0=# PARMS/PRESENCE FLAGS
0057  0004  0A80               SLA   R0,PRRITS
0058  0006  1524               JGT   SBP100            REQUIRED INDEX MISSING
0059  0008  C07A               MOV   *R10+,R1
0060                    *
0061                    *  CHECK INDEX NUMBER
0062                    *
0063  000A  0281               CI    R1,NUMBPS
      000C  0003
0064  000E  1420               JHE   SBP100            BAD INDEX NUMBER
0065  0010  0202               LI    R2,ENTSIZ
      0012  0000
0066  0014  3842               MPY   R2,R1
0067  0016  0222               AI    R2,BPTAB          R2=TABLE ENTRY POINTER
      0018  0000
0068  001A  C242               MOV   R2,R9             R9=TABLE ENTRY POINTER
0069                    *
0070                    *  INITIALIZE PC VALUE
0071                    *
0072  001C  05C2               INCT  R2
0073  001E  0A10               SLA   R0,S1
0074  0020  1517               JGT   SBP100            PC MISSING
0075  0022  1315               JEQ   SBP100
0076  0024  CC9A               MOV   *R10,*R2+
0077  0026  CCBA               MOV   *R10+,*R2+
0078                    *
0079                    *  INSERT REF CNT
0080                    *
0081  0028  0732               SETO  *R2+
0082  002A  0A10               SLA   R0,S1
0083  002C  1102               JLT   SBP030            NULL?
0084  002E  04D2               CLR   *R2
0085  0030  1002               JMP   SBP040
0086        0032' SBP030  EQU   $
0087  0032  C4BA               MOV   *R10+,*R2
0088  0034  0612               DEC   *R2
0089                    *
0090                    *     GET SNAPSHOT NUMBER
0091                    *
0092  0036  0A10  SBP040  SLA   R0,S1
0093  0038  1103               JLT   SBP050            NULL?
0094  003A  D660               MOVB  @FLAG,*R9         YES, MOVE DEF'D,NO SNAP FLAGS
      003C  0001'
0095  003E  1007               JMP   SBP090            EXIT
0096                    *
0097                    *  CHECK OUT SNAP #
0098                    *
0099  0040  C01A  SBP050  MOV   *R10,R0           R0=SNAP #
0100  0042  0280               CI    R0,NUMSNP
      0044  0000
```

```
 0101  0046  1404          JHE   SBP100              ILLEGAL SNAPSHOT NUMBER
 0102                *
 0103                *  INSERT SNAPSHOT NUMBER AND FLAGS
 0104                *
 0105  0048  C65A          MOV   *R10,*R9
 0106  004A  D660          MOVB  @FLAGSN,*R9
       004C  0000!
 0107                *
 0108                *  EXIT
 0109                *
 0110        004E! SBP090 EQU   $
 0111  004F  045B          RT
 0112                *
 0113                *  ERROR EXIT
 0114                *
 0115        0050! SBP100 EQU   $
 0116  0050  020A          LI    R10,DP20            BREAKPOINT SPECIFICATION ERRO
       0052  1120
 0117  0054  0460          B     @ERROR                (BR AND RETURN TO MY CALLER
       0056  0000
 0118                      END
0000 ERS
```

960 - 980 CONCORDANCE

$		0086	0110	0115						
BPTAB		0031	0067							
DP20	0042	0116								
ENTSIZ		0033	0065							
ERROR		0037	0117							
FLAG	0039	0094								
FLAGSN	003A	0106								
NUMBPS		0032	0063							
NUMSNP		0034	0100							
PRBITS	0040	0057								
R0	0043	0056	0057	0073	0082	0092	0099	0100		
R1	0044	0059	0063	0066						
R10	004B	0056	0059	0076	0077	0087	0099	0105	0116	
R11	0049									
R15	0050									
R2	0045	0065	0066	0067	0068	0072	0076	0077	0081	0084
		0087	0088							
R5	0046									
R9	0047	0068	0094	0105	0108					
S1	0041	0073	0082	0092						
SBP	0052	0030								
SBP030	0086	0083								
SBP040	0092	0085								
SBP050	0099	0093								
SBP090	0110	0095								
SBP100	0115	0058	0064	0074	0075	0101				
SNPENT		0036								
SNPTAB		0035								

THERE ARE 0027 SYMBOLS

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV	
SHEET	

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, SCNTBL, SCAN TABLES-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945378-9901

SCALE	REV.	SHEET 1 of 5

TI-0023-C

```
0003                *    PROCEDURE SCAN TABLE(TABL,VAL,RTN CODE,
0004                *       ENTRY NUM,ENTRY LOC);
0005                *       /* SCAN TABLE WILL SEARCH A STANDARD
0006                *          TABLE TO DETERMINE IF AN INPUT
0007                *          VALUE IS WITHIN THE RANGE OF
0008                *          DEFINITION OF SOME TABLE ENTRY.
0009                *       */
0010                *       ENTRY LOC = TABL(1);
0011                *       ENTRY MAX = TABL(2)
0012                *       ENTRY LEN = TABL(3)
0013                *       ENTRY NUM = 0
0014                *       DO WHILE ENTRY NUM .LT. ENTRY MAX
0015                *          FLAGS = ENTRY LOC.FLAG
0016                *          IF ENTRY DEFINED THEN DO;
0017                *             IF VAL .GE. ENTRY LOW LIM .AND.
0018                *             VAL .LE. ENTRY HIGH LIM THEN DO;
0019                *                RETURN (FOUND);
0020                *             END;
0021                *          END;
0022                *          ENTRY LOC = ENTRY LOC + ENTRY LEN;
0023                *          ENTRY NUM = ENTRY NUM +1;
0024                *       END;
0025                *    END SCAN TABLE;
0026                         IDT  'SCNTBL'
0027                * TITLE:    SCNTBL
0028                *           SCAN TABLES
0029                * REVISION:
0030                *           ORIGINAL
0031                * COMPUTER: 990,ASM
0032                * ABSTRACT: SCAN TABLES IS A UTILITY ROUTINE WHICH
0033                *           CAN BE USED TO FIND IF SOME VALUE FALLS
0034                *           WITHIN THE LIMITS OF A CONTROLLED DATA
0035                *           STRUCTURE.
0036                * CALLING SEQUENCE:
0037                *           BL @SCTB
0038                *           R10 = PTR TO LIST
0039                *                    WD 1 = ADDRESS OF TABLE
0040                *                    WD 2 = # OF ENTRIES IN TABLE
0041                *                    WD 3 = # OF WORDS PER ENTRY
0042                *           R9  = VALUE FOR TEST
0043                *           R8  = RESERVED
0044                *       RETURN
0045                *           R10 =  0 IF VALUE FOUND IN TABLE LIMIT
0046                *           R9  =  ENTRY NUMBER
0047                *           R8  =  ENTRY POINTER
0048                *       REF'S AND DEF'S
0049                *
0050                         DEF  SCTB
0051                         REF  GETBUF
0052                         REF  RETBUF
0053                         REF  LWP
0054                         REF  RWP
0055                         REF  ACL
0056                         REF  RR
```

```
0057                    *
0058                    *WORKSPACE REGISTER DEFINITIONS
0059                    *
0060          0000      R0      EQU  0
0061          0001      R1      EQU  1
0062          0002      R2      EQU  2
0063          0003      R3      EQU  3
0064          0004      R4      EQU  4
0065          0005      R5      EQU  5
0066          0006      R6      EQU  6
0067          0007      R7      EQU  7
0068          0008      R8      EQU  8
0069          0009      R9      EQU  9
0070          000A      R10     EQU  10
0071          000B      R11     EQU  11
0072          000C      R12     EQU  12
0073          000D      R13     EQU  13
0074          000E      R14     EQU  14
0075          000F      R15     EQU  15
0076                    *
0077          0002      ENLOLM  EQU  2                   TABLE LOW LIMIT
0078          0004      ENHILM  EQU  4                   TABLE HIGH LIMIT
0079                    *
0080                    *       REGISTER ASSIGNMENTS
0081                    *
0082                    *       R11   RETURN CODE
0083                    *       R15   CURRENT ENTRY #
0084                    *       R10   ENTRY LENGTH
0085                    *       R9    TEST VALUE
0086                    *       R8    CURRENT ENTRY POINTER
0087                    *       R1    NUMBER OF ENTRIES
0088                    *                                 1*PROCEDURE SCAN TABLE(TABL,VA
0089                    *                                 2*ENTRY NUM,ENTRY LOC);
0090          0000'     SCTB    EQU  $
0091                    *                                 2*/* SCAN TABLE WILL SEARCH A
0092                    *                                 2*   TABLE TO DETERMINE IF AN
0093                    *                                 2*   VALUE IS WITHIN THE RANGE
0094                    *                                 2*   DEFINITION OF SOME TABLE
0095                    *                                 2**/
0096                    *                                 2*ENTRY LOC = TABL(1);
0097  0000    C23A              MOV  *R10+,R8
0098                    *                                 2*ENTRY MAX = TABL(2)
0099  0002    C07A              MOV  *R10+,R1
0100                    *                                 2*ENTRY LEN = TABL(3)
0101  0004    C29A              MOV  *R10,R10
0102                    *                                 2*ENTRY NUM = 0
0103  0006    04CF              CLR  R15
0104                    *                                 2*DO WHILE ENTRY NUM .LT. ENTR
0105          0008'     SCT010  EQU  $
0106  0008    804F              C    R15,R1
0107  000A    1411              JHE  SCT030
0108                    *                                 3*FLAGS = ENTRY LOC.FLAG
0109  000C    C018              MOV  *R8,R0
0110                    *                                 3*IF ENTRY DEFINED THEN DO;
0111  000E    0A10              SLA  R0,1
```

```
 0112  0010  1708          JNC   SCT020                    4=IF VAL ,GE, ENTRY LOW LIM ,A
 0113                *                                     4=VAL ,LE, ENTRY HIGH LIM THEN
 0114                *
 0115  0012  C028          MOV   @ENLOLM(R8),R0
       0014  0002
 0116  0016  8009          C     R9,R0
 0117  0018  1A07          JL    SCT020
 0118  001A  C028          MOV   @ENHILM(R8),R0
       001C  0004
 0119  001E  8009          C     R9,R0
 0120  0020  1803          JH    SCT020
 0121                *                                     5=RETURN (FOUND);
 0122  0022  C24F          MOV   R15,R9                     ENTRY #
 0123  0024  04CA          CLR   R10                        FOUND
 0124  0026  045B          RT
 0125                *                                     4=END;
 0126                *                                     3=END;
 0127        0028' SCT020  EQU   $
 0128                *                                     3=ENTRY LOC = ENTRY LOC + ENTR
 0129  0028  A20A          A     R10,R8
 0130                *                                     3=ENTRY NUM = ENTRY NUM +1;
 0131  002A  058F          INC   R15
 0132                *                                     2=END;
 0133  002C  10ED          JMP   SCT010
 0134        002E' SCT030  EQU   $
 0135                *                                     1=END SCAN TABLE;
 0136  002E  045B          RT
 0137                      END
0000 ERS
```

```
$                       0090  0105  0127  0134
ACL                     0055
ENHILM  0078            0118
ENLOLM  0077            0115
GETBUF                  0051
LWP                     0053
R0      0060            0109  0111  0115  0116  0118  0119
R1      0061            0099  0106
R10     0070            0097  0099  0101  0101  0123  0129
R11     0071
R12     0072
R13     0073
R14     0074
R15     0075            0103  0106  0122  0131
R2      0062
R3      0063
R4      0064
R5      0065
R6      0066
R7      0067
R8      0068            0097  0109  0115  0118  0129
R9      0069            0116  0119  0122
RETBUF                  0052
RR                      0056
RWP                     0054
SCT010  0105            0133
SCT020  0127            0112  0117  0120
SCT030  0134            0107
SCT8    0090            0050

THERE ARE   0029  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV STATUS OF SHEETS

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ANGLES ±1° 3 PLACE DECIMAL ± 010 2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, SIEPR, SIE DRIVER-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945379-9901

SCALE | REV. | SHEET 1 of 6

TI-0023-C

```
0003                *    PROCEDURE SIE;
0004                *       * SIE CONTROLS THE LEVEL ZERO INTERRUPT
0005                *          AND PROGRAMMED SIE FACILITY.  WHEN
0006                *          CALLED FROM THE MONITOR, IT WILL
0007                *          EXECUTE ONE INSTRUCTION OF THE USER'S
0008                *          PROGRAM AND RETURN.  WHEN A FRONT PANEL
0009                *          HALT IS PUSHED, THE MONITOR IS
0010                *          RE-INITIALIZED.
0011                *       *1
0012                *       CALL SAVE CURRENT ENVIRONMENT;
0013                *       CALL SET FRONT PANEL WORKSPACE
0014                *       SIE FLAG = '9900'
0015                *       CALL SET USER ENVIRONMENT;
0016                *       SIE RETURN ADDRESS = SIEINT;
0017                *       SIE IN PROG = -1
0018                *       CRU BASE = FRONT PANEL CRU;
0019                *       CALL SIE EXEC;
0020                *       CALL SAVE USER ENVIRONMENT
0021                *       IF .NOT. SIE IN PROG THEN INITIALIZE MONITOR
0022                *       SIE IN PROG = 0;
0023                *       CALL RETURN MONITOR ENVIRONMENT
0024                *    END SIE
0025                         IDT  'SIEPR'
0026                * TITLE:     SIEPR
0027                *            SINGLE INSTRUCTION EXECUTION PROCESSOR
0028                * REVISION:
0029                *            ORIGINAL
0030                * COMPUTER: 990,ASM
0031                * ABSTRACT: SIE EXECUTES ONE INSTRUCTION USING THE
0032                *            SIE INTERRUPT.  ANY HALT INTERRUPT (LEVEL
0033                *            0 INTERRUPT NOT CAUSED BY MONITOR INITIATED
0034                *            SIE) WILL RE-INITIALIZE PX9
0035                * CALLING SEQUENCE:
0036                *            BL @SIE
0037                *
0038                *       REF'S AND DEF'S
0039                *
0040                         DEF  SIE
0041                         DEF  SIEINT
0042                         REF  FPWP                FRONT PANEL WORKSPACE
0043                         REF  USRWP
0044                         REF  USRPC
0045                         REF  USRST
0046                         REF  CRUOFF
0047                         REF  PCOUT
0048                         REF  INIT
0049                         REF  SIEWP
0050                         REF  SIEST
0051                         REF  SIINPG
0052                *
0053                *WORKSPACE REGISTER DEFINITIONS
0054                *
0055         0000   R0        EQU  0
0056         0001   R1        EQU  1
```

```
0057          0002  R2       EQU  2
0058          0003  R3       EQU  3
0059          0004  R4       EQU  4
0060          0005  R5       EQU  5
0061          0006  R6       EQU  6
0062          0007  R7       EQU  7
0063          0008  R8       EQU  8
0064          0009  R9       EQU  9
0065          000A  R10      EQU  10
0066          000B  R11      EQU  11
0067          000C  R12      EQU  12
0068          000D  R13      EQU  13
0069          000E  R14      EQU  14
0070          000F  R15      EQU  15
0071                *
0072          000E  SIEFLG   EQU  14
```

```
0074                    *                       1-PROCEDURE SIE;
0075          0000'     SIE     EQU   $
0076                    *                       2-** SIE CONTROLS THE LEVEL ZER
0077                    *                       2-   AND PROGRAMMED SIE FACILI
0078                    *                       2-   CALLED FROM THE MONITOR,
0079                    *                       2-   EXECUTE ONE INSTRUCTION O
0080                    *                       2-   PROGRAM AND RETURN.  WHEN
0081                    *                       2-   HALT IS PUSHED, THE MONIT
0082                    *                       2-   RE-INITIALIZED.
0083                    *                       2-*1
0084                    *
0085                    *                       2-CALL SAVE CURRENT ENVIRONMEN
0086  0000  02A0               STWP  R0
0087  0002  C800               MOV   R0,@SIEWP
      0004  0000
0088  0006  02C0               STST  R0
0089  0008  C800               MOV   R0,@SIEST
      000A  0000
0090                    *                       2-CALL SET FRONT PANEL WORKSPA
0091  000C  02E0               LWPI  FPWP
      000E  0000
0092                    *                       2-SIE FLAG = '9900'
0093  0010  0201               LI    R1,>9900    FLAG INDICATED MONITOR SIE
      0012  9900
0094                    *                       2-CALL SET USER ENVIRONMENT;
0095  0014  C360               MOV   @USRWP,R13
      0016  0000
0096  0018  C3A0               MOV   @USRPC,R14
      001A  0000
0097  001C  C3E0               MOV   @USRST,R15
      001E  0000
0098                    *                       2-SIE RETURN ADDRESS = SIEINT;
0099  0020  0200               LI    R0,SIEINT   FRONT PANEL RETURN TO MONITOR
      0022  0030'
0100                    *                       2-SIE IN PROG = -1
0101  0024  0720               SETO  @SIINPG
      0026  0000
0102                    *                       2-CRU BASE = FRONT PANEL CRU;
0103  0028  020C               LI    R12,CRUOFF
      002A  0000
0104                    *                       2-CALL SIE EXEC;
0105  002C  1D0E               SBO   SIEFLG      EXEC 2 INST AND INTERRUPT
0106  002E  0380               RTWP                INST#1 = OURS  #2 =USER
0107          0030' SIEINT EQU   $
0108                    *                       2-CALL SAVE USER ENVIRONMENT
0109  0030  C80D               MOV   R13,@USRWP
      0032  0016'
0110  0034  C80E               MOV   R14,@USRPC
      0036  001A'
0111  0038  C80F               MOV   R15,@USRST
      003A  001E'
0112  003C  06A0               BL    @PCOUT      OUTPUT USRPC TO PANEL
      003E  0000
0113                    *                       2-IF .NOT. SIE IN PROG THEN IN
```

```
 0114  0040  C0A0          MOV   @SIINPG,R2
       0042  0026'
 0115  0044  1602          JNE   SIE010
 0116  0046  0460          B     @INIT
       0048  0000
 0117        004A' SIE010  EQU   $
 0118              *                          2=SIE IN PROG = 0;
 0119  004A  04E0          CLR   @SIINPG
       004C  0042'
 0120              *                          2=CALL RETURN MONITOR ENVIRONM
 0121  004E  C360          MOV   @SIEWP,R13
       0050  0004'
 0122  0052  C3E0          MOV   @SIEST,R15
       0054  000A'
 0123  0056  020E          LI    R14,SIE020
       0058  005C'
 0124  005A  0380          RTWP
 0125        005C' SIE020  EQU   $
 0126              *                          1=END SIE
 0127  005C  045B          RT
 0128              *         END
0000 ERS
```

$		0075	0107	0117	0125	
CRUOFF		0046	0103			
FPWP		0042	0091			
INIT		0048	0116			
PCOUT		0047	0112			
R0	0055	0086	0087	0088	0089	0099
R1	0056	0093				
R10	0065					
R11	0066					
R12	0067	0103				
R13	0068	0095	0109	0121		
R14	0069	0096	0110	0123		
R15	0070	0097	0111	0122		
R2	0057	0114				
R3	0058					
R4	0059					
R5	0060					
R6	0061					
R7	0062					
R8	0063					
R9	0064					
SIE	0075	0040				
SIE010	0117	0115				
SIE020	0125	0123				
SIEFLG	0072	0105				
SIEINT	0107	0041	0099			
SIEST		0050	0089	0122		
SIEWP		0049	0087	0121		
SINPG		0051	0101	0114	0119	
USRPC		0044	0096	0110		
USRST		0045	0097	0111		
USRWP		0043	0095	0109		

THERE ARE 0032 SYMBOLS

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																												
SHEET																												

REV STATUS OF SHEETS	REV																						
	SHEET																						

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, SSNAP, SET SNAPSHOT-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945380-9901

SCALE	REV.	SHEET 1 of 8

TI-0023-C

```
0003              *    PROCEDURE SET SNAPSHOT(CPL);
0004              *       /* SET SNAPSHOT ACCEPTS AS INPUT A
0005              *          COMMAND PARAMETER LIST, VALIDATES
0006              *          OR DEFAULTS THE APPROPRIATE PARAMETERS
0007              *          AND BUILD'S A SNAPSHOT TABLE ENTRY.
0008              *       */
0009              *       FLAGS = CPL.FLAGS
0010              *       CPL = CPL + 2        /* 1ST PARM WORD */
0011              *       PRM DEFAULT = 0;
0012              *       PRM MAX = 3;
0013              *       PRM MIN = 0;
0014              *       CALL GET PARM(SS#,PRM DEFAULT,PRM MAX,
0015              *          PRM MIN,ERROR);
0016              *       IF .NOT. ERROR THEN DO;
0017              *          SNAP ENTRY = LOC(SNAPSHOT TABLE)+
0018              *             SS# * SNAP ENTRY LENGTH;
0019              *          IF SNAP ENTRY.SNAP FLAGS .AND. SSDEF .EQ. 0
0020              *          THEN DO;
0021              *          PRM MAX = 'F';
0022              *          CALL GET PARM(LOW REG,PRM DEFAULT,PRM MAX,
0023              *             PRM MIN,ERROR);
0024              *          IF .NOT. ERROR THEN DO;
0025              *             SNAP ENTRY.LOW REG = LOW REG;
0026              *             PRM DEFAULT = LOW REG;
0027              *             PRM MIN = LOW REG;
0028              *             CALL GET PARM(HIGH REG,PRM DEFAULT,PRM MAX,
0029              *                PRM MIN,ERROR);
0030              *             IF  .NOT. ERROR THEN DO;
0031              *                SNAP ENTRY.HIGH REG = HIGH REG;
0032              *                PRM.DEFAULT = 0;
0033              *                PRM.MAX = >FFFF
0034              *                PRM.MIN = 0;
0035              *                CALL GET PARM(MEM,PRM DEFAULT,PRM MAX,
0036              *                   PRM MIN,ERROR);
0037              *                IF .NOT. ERROR THEN DO;
0038              *                   SNAP ENTRY.MEM RNG   = MEM;
0039              *                   PRM.DEFAULT = MEM;
0040              *                   PRM.MIN = MEM,
0041              *                   CALL GET PARM(MEM,PRM DEFAULT,PRM MAX
0042              *                      PRM MIN,ERROR);
0043              *                   IF .NOT. ERROR  THEN DO;
0044              *                      SNAP ENTRY.MEM RNG+2 = MEM;
0045              *                      SNAP ENTRY.FLAGS = SNAP ENTRY.FLAG
0046              *                         .OR. SSDEF
0047              *                   END;
0048              *                END;
0049              *             END;
0050              *          RETURN
0051              *          END
0052              *       END;ELSE DO;
0053              *          CALL ERROR('DP04');
0054              *       END;
0055              *    END
0056              *  END SET SNAPSHOT;
```

```
0057                *       PROCEDURE GET PARM(PARM,DEF,MAX,MIN,ERR);
0058                *          /* GET PARM FINDS THE NEXT PARAMETER
0059                *             IN THE CPL IF PRESENT, AND VALIDATES
0060                *             IT; ELSE THE DEFAULT IS SUPPLIED,
0061                *             IF AN INVALID PARM IS SPECIFIED,
0062                *             AN ERROR MESSAGE IS PRINTED.
0063                *          */
0064                *          PARM = DEF;
0065                *          IF FLAGS .AND. NEXT PARM THEN DO;
0066                *             PARM = CPL.PARM;
0067                *             CPL = CPL + 2;
0068                *          END;
0069                *          ERR    = 0;
0070                *          IF PARM .LT. MIN THEN ERR    = !DP013!
0071                *          IF PARM .GT. MAX THEN ERR    = !DP03!;
0072                *          IF ERR    .NE. 0 THEN DO;
0073                *             CALL ERROR(ERR);
0074                *             RETURN(ERR)
0075                *          END;
0076                *          RETURN
0077                *       END GET PARM;
0078                       IDT  'SSNAP'
0079                * TITLE:     SSNAP
0080                *            SET SNAPSHOT
0081                * REVISION:
0082                *            ORIGINAL
0083                * COMPUTER: 990,ASM
0084                * ABSTRACT: SET SNAPSHOT ACCEPTS AS INPUT, A USER'S
0085                *            SNAPSHOT DEFINITION AND CONSTRUCTS THE
0086                *            APPROPRIATE SNAPSHOT ENTRY.
0087                * CALLING SEQUENCE:
0088                *            BL @SSS
0089                *            R10 = PTR TO COMMAND PARAMETER LIST
0090                *
0091                *       REF'S AND DEF'S
0092                *
0093                       DEF  SSS
0094                       REF  SNPTAB
0095                       REF  SNPENT
0096                       REF  NUMSNP
0097                       REF  ERROR
0098                       REF  GETBUF
0099                       REF  RETBUF
0100                       REF  LWP
0101                       REF  RWP
0102                       REF  ACL
0103                       REF  RR
0104                *
0105                *WORKSPACE REGISTER DEFINITIONS
0106                *
0107          0000  R0      EQU  0
0108          0001  R1      EQU  1
0109          0002  R2      EQU  2
0110          0003  R3      EQU  3
0111          0004  R4      EQU  4
```

```
0112          0005  R5      EQU  5
0113          0006  R6      EQU  6
0114          0007  R7      EQU  7
0115          0008  R8      EQU  8
0116          0009  R9      EQU  9
0117          000A  R10     EQU  10
0118          000B  R11     EQU  11
0119          000C  R12     EQU  12
0120          000D  R13     EQU  13
0121          000E  R14     EQU  14
0122          000F  R15     EQU  15
0123                *
0124          0003  MXSNP   EQU  3                  MAX SNAPSHOT #
0125          0100  LBYTE   EQU  256
0126          0080  SSDEF   EQU  >80                SNAPSHOT DEFINED FLAG
0127          0002  SSLORG  EQU  2
0128          0003  SSHIRG  EQU  3
0129          0004  SSMR1   EQU  4
0130          1103  DP03    EQU  >1103              PARM > MAX
0131          1104  DP04    EQU  >1104              SNAPSHOT ALREADY DEFINED
0132          1113  DP13    EQU  >1113              PARM< MIN
0133                *
0134                *       REGISTER ASSIGNMENTS
0135                *
0136                *       R9,R10 SUB CALL PARMS
0137                *       R8     CPL FLAGS
0138                *       R7     CPL CURRENT POINTER
0139                *       R6     SNAPSHOT ENTRY POINTER
0140                *       R5     CURRENT PARAMETER
0141                *       R4     PARAMETER DEFAULT VALUE *
0142                *       R3     PARAMETER MAX VALUE     ** UNCHANGED BY
0143                *       R2     PARAMETER MIN VALUE     *    SUBROUTINE
0144                *                                   1*PROCEDURE SET SNAPSHOT(CPL);
0145          0000' SSS     EQU  $
0146                *     *ALLOC,COPY,LINK
0147  0000  0420           BLWP @ACL
      0002  0000
0148                *                                   2*/* SET SNAPSHOT ACCEPTS AS I
0149                *                                   2*    COMMAND PARAMETER LIST, V
0150                *                                   2*    OR DEFAULTS THE APPROPRIA
0151                *                                   2*    AND BUILD'S A SNAPSHOT TA
0152                *                                   2**/
0153                *                                   2*FLAGS = CPL.FLAGS
0154                *                                   2*CPL = CPL + 2        /* 1ST P
0155  0004  C23A           MOV  *R10+,R8
0156  0006  0A88           SLA  R8,8
0157  0008  C1CA           MOV  R10,R7
0158                *                                   2*PRM DEFAULT = 0;
0159  000A  04C4           CLR  R4
0160                *                                   2*PRM MAX = 3;
0161  000C  0203           LI   R3,NUMSNP
      000E  0000
0162  0010  0603           DEC  R3
0163                *                                   2*PRM MIN = 0;
0164  0012  04C2           CLR  R2
```

```
0165                    *                               2=CALL GET PARM(SS#,PRM DEFAUL
0166                    *                               3=PRM MIN,ERROR);
0167  0014  06A0               BL    @GTPARM
      0016  0086'
0168                    *                               2=IF .NOT. ERROR THEN DO;
0169  0018  1033               JMP   SSS030
0170                    *
0171                    *                               3=SNAP ENTRY = LOC(SNAPSHOT TA
0172                    *                               4=SS# * SNAP ENTRY LENGTH;
0173  001A  0200               LI    R0,SNPENT
      001C  0000
0174  001E  3805               MPY   R5,R0
0175  0020  0221               AI    R1,SNPTAB
      0022  0000
0176  0024  C181               MOV   R1,R6
0177                    *                               3=IF SNAP ENTRY.SNAP FLAGS .AN
0178                    *                               4=THEN DO;
0179  0026  C016               MOV   *R6,R0
0180  0028  0240               ANDI  R0,SSDEF*LBYTE
      002A  8000
0181  002C  1625               JNE   SSS020
0182                    *                               4=PRM MAX = 'F';
0183  002E  0203               LI    R3,>F
      0030  000F
0184                    *                               4=CALL GET PARM(LOW REG,PRM DE
0185                    *                               5=PRM MIN,ERROR);
0186                    *                               4=IF .NOT. ERROR THEN DO;
0187  0032  06A0               BL    @GTPARM
      0034  0086'
0188  0036  1024               JMP   SSS030
0189                    *                               5=SNAP ENTRY.LOW REG = LOW REG
0190  0038  0A85               SLA   R5,8
0191  003A  D985               MOVB  R5,@SSLORG(R6)
      003C  0002
0192                    *                               5=PRM DEFAULT = LOW REG;
0193  003E  0885               SRA   R5,8
0194  0040  C105               MOV   R5,R4
0195                    *                               5=PRM MIN = LOW REG;
0196  0042  C085               MOV   R5,R2
0197                    *                               5=CALL GET PARM(HIGH REG,PRM D
0198                    *                               6=PRM MIN,ERROR);
0199                    *                               5=IF  .NOT. ERROR THEN DO;
0200  0044  06A0               BL    @GTPARM
      0046  0086'
0201  0048  1018               JMP   SSS030
0202                    *                               6=SNAP ENTRY.HIGH REG = HIGH R
0203  004A  0A85               SLA   R5,8
0204  004C  D985               MOVB  R5,@SSHIRG(R6)
      004E  0003
0205                    *                               6=PRM.DEFAULT = 0;
0206  0050  04C4               CLR   R4
0207                    *                               6=PRM.MAX = >FFFF
0208  0052  0203               LI    R3,>FFFF
      0054  FFFF
0209                    *                               6=PRM.MIN = 0;
```

```
0210  0056  04C2          CLR   R2
0211                *                                 6=CALL GET PARM(MEM,PRM DEFAUL
0212                *                                 7=PRM MIN,ERROR);
0213  0058  06A0          BL    @GTPARM
      005A  0086'
0214                *                                 6=IF .NOT. ERROR THEN DO;
0215  005C  1011          JMP   SSS030
0216                *                                 7=SNAP ENTRY.MEM RNG   = MEM;
0217  005E  C985          MOV   R5,@SSMR1(R6)
      0060  0004
0218                *                                 7=PRM.DEFAULT = MEM;
0219  0062  C105          MOV   R5,R4
0220                *                                 7=PRM.MIN = MEM,
0221  0064  C085          MOV   R5,R2
0222                *                                 7=CALL GET PARM(MEM,PRM DEFAUL
0223                *                                 8=PRM MIN,ERROR);
0224  0066  06A0          BL    @GTPARM
      0068  0086'
0225                *                                 7=IF .NOT. ERROR  THEN DO;
0226  006A  100A          JMP   SSS030
0227                *                                 8=SNAP ENTRY.MEM RNG+2 = MEM;
0228  006C  C985          MOV   R5,@SSMR1+2(R6)
      006E  0006
0229                *                                 8=SNAP ENTRY.FLAGS = SNAP ENTR
0230                *                                 9=.OR. SSDEF
0231  0070  0200          LI    R0,SSDEF*LBYTE
      0072  8000
0232  0074  E580          SOC   R0,*R6
0233                *                                 7=END;
0234                *                                 6=END;
0235                *                                 5=END;
0236                *                                 4=RETURN
0237        0076' SSS010 EQU   $
0238  0076  1004          JMP   SSS030
0239                *                                 4=END
0240                *                                 3=END;ELSE DO;
0241        0078' SSS020 EQU   $
0242                *                                 4=CALL ERROR('DP04');
0243  0078  020A          LI    R10,DP04
      007A  1104
0244  007C  06A0          BL    @ERROR
      007E  0000
0245                *                                 3=END;
0246        0080' SSS030 EQU   $
0247                *                                 2=END
0248                *                                 1=END SET SNAPSHOT;
0249                *     *LINK TO PREV WKSP, RET CURR WKSP
0250  0080  0420          BLWP @RR
      0082  0000
0251  0084  045B          RT
0252                *                                 2=PROCEDURE GET PARM(PARM,DEF,
0253        0086' GTPARM EQU   $
0254                *                                 3=/* GET PARM FINDS THE NEXT P
0255                *                                 3=    IN THE CPL IF PRESENT, AN
0256                *                                 3=    IT; ELSE THE DEFAULT IS S
```

```
 0257                    *                                  3=   IF AN INVALID PARM IS SPE
 0258                    *                                  3=   AN ERROR MESSAGE IS PRINT
 0259                    *                                  3=*/
 0260                    *                                  3=PARM = DEF;
 0261  0086  C144              MOV   R4,R5
 0262                    *                                  3=IF FLAGS .AND. NEXT PARM THE
 0263  0088  0A18              SLA   R8,1
 0264  008A  1701              JNC   GTP010                     BIT OF FLAGS)
 0265                    *                                  4=PARM = CPL,PARM;
 0266                    *                                  4=CPL = CPL + 2;
 0267  008C  C177              MOV   *R7+,R5
 0268                    *                                  3=END;
 0269        008E' GTP010 EQU   $
 0270                    *                                  3=ERR    = 0;
 0271  008E  04CA              CLR   R10
 0272                    *                                  3=IF PARM .LT. MIN THEN ERR
 0273  0090  8085              C     R5,R2
 0274  0092  1402              JHE   GTP020
 0275  0094  020A              LI    R10,DP13
       0096  1113
 0276                    *                                  3=IF PARM .GT. MAX THEN ERR
 0277        0098' GTP020 EQU   $
 0278  009A  80C5              C     R5,R3
 0279  009C  1202              JLE   GTP030
 0280  009E  020A              LI    R10,DP03
       00A0  1103
 0281        00A0' GTP030 EQU   $
 0282                    *                                  3=IF ERR    .NE. 0 THEN DO;
 0283  00A2  C28A              MOV   R10,R10
 0284  00A4  1304              JEQ   GTP040
 0285                    *                                  4=CALL ERROR(ERR);
 0286  00A6  C3CB              MOV   R11,R15
 0287  00A8  06A0              BL    @ERROR
       00AA  007E'
 0288                    *                                  4=RETURN(ERR)
 0289  00AC  045F              B     *R15
 0290        00AE' GTP040 EQU   $
 0291                    *                                  3=END;
 0292                    *                                  3=RETURN
 0293  00B0  05CB              INCT  R11
 0294  00B2  045B              RT
 0295                    *                                  2=END GET PARM;
 0296                          END
0000 ERS
```

960 - 980 CONCORDANCE

$		0144	0236	0240	0245	0252	0268	0276	0280	0289
ACL		0101	0146							
DP03	0129	0279								
DP04	0130	0242								
DP13	0131	0274								
ERROR		0096	0243	0286						
GETBUF		0097								
GTP010	0268	0263								
GTP020	0276	0273								
GTP030	0280	0278								
GTP040	0289	0283								
GTPARM	0252	0166	0186	0199	0212	0223				
LAYTE	0124	0179	0230							
LWP		0099								
MXSNP	0123									
NUMSNP		0095	0160							
R0	0106	0172	0173	0178	0179	0230	0231			
R1	0107	0174	0175							
R10	0116	0154	0156	0242	0270	0274	0279	0282	0282	
R11	0117	0285	0292							
R12	0118									
R13	0119									
R14	0120									
R15	0121	0285	0288							
R2	0108	0163	0195	0209	0220	0272				
R3	0109	0160	0161	0182	0207	0277				
R4	0110	0158	0193	0205	0218	0260				
R5	0111	0173	0189	0190	0192	0193	0195	0202	0203	0216
		0218	0220	0227	0260	0266	0272	0277		
R6	0112	0175	0178	0190	0203	0216	0227	0231		
R7	0113	0156	0266							
R8	0114	0154	0155	0262						
R9	0115									
RETBUF		0098								
RR		0102	0249							
RWP		0100								
SNPENT		0094	0172							
SNPTAB		0093	0174							
SSDEF	0125	0179	0230							
SSHIRG	0127	0203								
SSLORG	0126	0190								
SSMR1	0128	0216	0227							
SSS	0144	0092								
SSS010	0236									
SSS020	0240	0180								
SSS030	0245	0168	0187	0200	0214	0225	0237			

THERE ARE 0045 SYMBOLS

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																
SHEET																																
REV STATUS OF SHEETS	REV																															
	SHEET																															

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DWN	DATE
	CHK	
	ENGR	
	QA	
	APVD	
	CONTR NO	
	DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, SUPVSR, SUPERVISOR CALL INTERFACE-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945381-9901

SCALE	REV.	SHEET 1 of 7

TI-0023-C

```
0003                *    PROCEDURE SVC(SCB);
0004                *       /* SVC ALLOCATES A WORKSPACE TO REPLACE
0005                *         THE FIXED WORKSPACE OF THE XOP VECTOR
0006                *         AND SWITCHES TO IT,  THE SUPERVISOR CALL
0007                *         IS VALIDATED (FOR ID) AND THE APPROPRIATE
0008                *         SERVICE ROUTINE IS CALLED.  WHEN CONTROL
0009                *         RETURNS FROM THE SERVICE ROUTINE, THE
0010                *         CANCEL FLAG IS CHECKED FOR THE PROCESS.
0011                *         IF A USER PROCESS HAS BEEN CANCELLED,
0012                *         THE PROCESS IS TERMINATED. */
0013                *
0014                *       DECLARE
0015                *          FLAG BIT(8);   /* PROCESS FLAGS    */
0016                *       DECLARE
0017                *          (INTERRUPT MASK, CHANGE WORKSPACE,
0018                *            WORKSPACE RETURN) INTRINSIC
0019                *       CALL TURN OFF SIE
0020                *       CALL CHANGE WORKSPACE;
0021                *       IF SCB(SCB CMD) <= MAX CMD THEN DO;
0022                *          SCB(SCB RESULT) = 0;
0023                *          CALL @COMMAND LIST(SCB(SCB CMD));
0024                *       END;
0025                *       ELSE SCB(SCB RESULT) = 1;
0026                *       CALL SWITCH WORKSPACE
0027                *       IF SIE MODE THEN DO;
0028                *          CALL TURN ON SIE;
0029                *       END;
0030                *    END SVC;
0031                         IDT  'SUPVSR'
0032                * TITLE:    SUPVSR
0033                *           SUPERVISOR
0034                * REVISION:
0035                *           ORIGINAL
0036                * COMPUTER: 990,ASM
0037                * ABSTRACT: SVC ALLOCATES A WORKSPACE  O REPLACE
0038                *           THE FIXED WORKSPACE OF THE XOP VECTOR
0039                *           AND SWITCHES TO IT. THE SUPERVISOR
0040                *           IS VALIDATED (FOR ID) AND THE APPROPRIATE
0041                *           SERVICE ROUTINE CALLED. WHEN CONTROL
0042                *           RETURNS FROM THE SERVICE ROUTINE, THE
0043                *           CANCEL FLAG IS CHECKED FOR THE PROCESS.
0044                *           IF A USER PROCESS HAS BEEN CANCELED,
0045                *           THE PROCESS IS TERMINATED.
0046                * CALLING SEQUENCE:
0047                *           BLWP @SVCALT
0048                *             R10 = ADDRESS OF PRB BLOCK
0049                *
0050                *         XOP 15,=
0051                *              CALL FROM USER AREA PROGRAM
0052                *
0053                *                                1=PROCEDURE SVC(SCB);
0054                *                                2=/* SVC ALLOCATES A WORKSPACE
0055                *                                2=  THE FIXED WORKSPACE OF THE
0056                *                                2=  AND SWITCHES TO IT.  THE S
```

```
0057                *                                    2-  IS VALIDATED (FOR ID) AND
0058                *                                    2-  SERVICE ROUTINE IS CALLED.
0059                *                                    2-  RETURNS FROM THE SERVICE R
0060                *                                    2-  CANCEL FLAG IS CHECKED FOR
0061                *                                    2-  IF A USER PROCESS HAS BEEN
0062                *                                    2-  THE PROCESS IS TERMINATED.
0063                *                                    2-
0064                *                                    2-DECLARE
0065                *                                    3-FLAG BIT(8);   /* PROCESS FL
0066                *                                    2-DECLARE
0067                *                                    3-(INTERRUPT MASK, CHANGE WORK
0068                *                                    3-  WORKSPACE RETURN) INTRINSI
0069                *
0070                *WORKSPACE REGISTER DEFINITIONS
0071                *
0072          0000  R0      EQU  0
0073          0001  R1      EQU  1
0074          0002  R2      EQU  2
0075          0003  R3      EQU  3
0076          0004  R4      EQU  4
0077          0005  R5      EQU  5
0078          0006  R6      EQU  6
0079          0007  R7      EQU  7
0080          0008  R8      EQU  8
0081          0009  R9      EQU  9
0082          000A  R10     EQU  10
0083          000B  R11     EQU  11
0084          000C  R12     EQU  12
0085          000D  R13     EQU  13
0086          000E  R14     EQU  14
0087          000F  R15     EQU  15
0088                *
0089                *
0090                *       REF'S AND DEF'S
0091                *
0092                        DEF  SVC
0093                        DEF  SVCSR
0094                        DEF  SVCSRA
0095                        REF  SVCWP
0096                        REF  IO              USER'S I/O CALL
0097                        REF  TERM            END OF PROGRAM
0098                        REF  CNVRT           BINARY-ASCII CONVERSION ROUTINES
0099                        REF  FPWP
0100                        REF  SIEINT
0101                        REF  USRPC
0102                        REF  PCOUT
0103                        REF  SVCW10
0104                        REF  CRUOFF
0105                        REF  SVCW7
0106                        REF  GETBUF
0107                        REF  RETBUF
0108                        REF  LWP
0109                        REF  RWP
0110                        REF  ACL
0111                        REF  RR
```

```
0112         000E  SIEFLG EQU  14
0113               *
0114               *       SUPERVISOR CALL XOP 15 VECTOR
0115               *
0116         007C  SVC    EQU  >7C                    XOP 15
0117               *
0118               *       REGISTER ASSIGNMENTS
0119               *
0120               *       R13  CALLER'S WORKSPACE
0121               *       R11  LINKAGE
0122               *       R10  SUPERVISOR CALL BLOCK
0123  0000                 RORG
0124               *
0125               *       PX990 SUPERVISOR CALLS
0126               *
0127               *         SUPERVISOR CALL BLOCK
0128               *
0129         0000  SCBCMD EQU  0             FIXED(8) - SUPV CALL ID
0130         0001  SCBERR EQU  1             FIXED(8) - RETURN CODE
0131               *
0132               *         SVC SERVICE ROUTINES
0133               *
0134         0000' SCBLST EQU  $
0135  0000   0000          DATA IO                    I/O PACKAGE
0136  0002   0000          DATA 0
0137  0004   0000          DATA 0
0138  0006   0000          DATA 0
0139  0008   0000          DATA TERM                  END OF PROGRAM
0140  000A   0000          DATA 0
0141  000C   0000          DATA 0
0142  000E   0000          DATA 0
0143  0010   0000          DATA 0
0144  0012   0000          DATA 0
0145  0014   0000          DATA CNVRT                 ASCII <-> BINARY CONVERSION
0146  0016   0014'         DATA CNVRT
0147  0018   0016'         DATA CNVRT
0148  001A   0018'         DATA CNVRT
0149  001C   0000          DATA 0
0150         000F  MAXCMD EQU  $-SCBLST/2
```

```
0152          001E' SVCSRA EQU  $                    BLWP ENTRY
0153  001F    C2ED         MOV  @R10*2(R13),R11
      0020    0014
0154          0022' SVCSR  EQU  $
0155  0022    04C7         CLR  R7
0156                 *                                2=CALL TURN OFF SIE
0157  0024    0208         LI   R8,T1           IF IN SIE MODE, USER PC WILL
      0026    0028'
0158  0028    8808  T1     C    R8,@USRPC         POINT TO CURRENT INSTRUCTION
      002A    0000
0159  002C    160D         JNE  T3              IF NOT SIE MODE
0160  002E    0707         SETO R7                   (SIE MODE FLAG)
0161  0030    0209         LI   R9,T2           TAKE CONTROL FROM
      0032    0038'
0162  0034    C809         MOV  R9,@FPWP          FRONT PANEL (FPWP = PANEL R0)
      0036    0000
0163          0038' T2     EQU  $
0164                 *                              2=CALL CHANGE WORKSPACE;
0165  0038    02E0         LWPI SVCWP                  (CURRENT WKSP MAY BE FPWP)
      003A    0000
0166  003C    C80E         MOV  R14,@USRPC             SET USER PC TO INST AFTER XOP
      003E    002A'
0167  0040    C24B         MOV  R11,R9
0168  0042    06A0         BL   @PCOUT                 OUTPUT USRPC TO PANEL
      0044    0000
0169  0046    C2C9         MOV  R9,R11
0170          0048' T3     EQU  $
0171  0048    C24B         MOV  R11,R9                   XOP EFFECTIVE ADDRESS
0172                 *    *ALLOC,COPY,LINK
0173  004A    0420         BLWP @ACL
      004C    0000
0174                 *       MOV  @R11*2(R13),R9     XOP EFF ADDR * DONE BY ACL
0175  004E    C36D         MOV  @R13*2(R13),R13     USER'S WORKSPACE
      0050    001A
0176  0052    C289         MOV  R9,R10              XP EFF ADDR = SCB
0177                 *                              2=IF SCB(SCB CMD) <= MAX CMD T
0178  0054    04C9         CLR  R9
0179  0056    D25A         MOVB *R10,R9              (CM = SCBCMD=0)
0180  0058    06C9         SWPB R9
0181  005A    0289         CI   R9,MAXCMD
      005C    000F
0182  005E    1408         JHE  SVCS10               (IF INVALID COMMAND)
0183                 *                              3=SCB(SCB RESULT) = 0;
0184  0060    DA89         MOVB R9,@SCBERR(R10)      CLR ERR RETURN
      0062    0001
0185                 *                              3=CALL @COMMAND LIST(SCB(SCB C
0186  0064    0A19         SLA  R9,1                 (WORD OFFSET IN BRANCH TABLE)
0187  0066    C269         MOV  @SCBLST(R9),R9
      0068    0000'
0188  006A    1302         JEQ  SVCS10
0189  006C    0699         BL   *R9
0190                 *                              2=END;
0191  006E    1004         JMP  SVCS20
0192                 *                              2=ELSE SCB(SCB RESULT) = 1;
```

```
 0193                    *
 0194          0070' SVCS10 EQU   $
 0195  0070  0209          LI    R9,>100
       0072  0100
 0196  0074  DA89          MOVB  R9,@SCBERR(R10)
       0076  0001
 0197          0078' SVCS20 EQU   $
 0198                    *                                  2=CALL SWITCH WORKSPACE
 0199  0078  020D          LI    R13,SVCWP
       007A  003A'
 0200  007C  0420          BLWP  @RR
       007E  0000
 0201                    *                                  2=IF SIE MODE THEN DO;
 0202  0080  C1C7          MOV   R7,R7
 0203  0082  130C          JEQ   SVCS30
 0204                    *                                  3=CALL TURN ON SIE;
 0205  0084  0208          LI    R8,FPWP
       0086  0036'
 0206  0088  0209          LI    R9,SIEINT
       008A  0000
 0207  008C  CE09          MOV   R9,*R8+
 0208  008E  0209          LI    R9,>9900
       0090  9900
 0209  0092  C609          MOV   R9,*R8
 0210  0094  020C          LI    R12,CRUOFF
       0096  0000
 0211  0098  1D0E          SBO   SIEFLG
 0212  009A  1000          NOP
 0213                    *                                  2=END;
 0214          009C' SVCS30 EQU   $
 0215  009C  0380          RTWP
 0216                    *                                  1=END SVC;
 0217                    END
0000 ERS
```

```
990 - 980  CONCORDANCE
$                       0133  0149  0151  0153  0162  0169  0193  0196  0213
ACL                     0109  0172
CNVRT                   0097  0144  0145  0146  0147
CRUOFF                  0103  0209
FPWP                    0098  0161  0204
GETBUF                  0105
IO                      0095  0134
LWP                     0107
MAXCMD  0149            0180
PCOUT                   0101  0167
R0      0071
R1      0072
R10     0081            0152  0175  0178  0183  0195
R11     0082            0152  0166  0168  0170
R12     0083            0209
R13     0084            0152  0174  0174  0174  0198
R14     0085            0165
R15     0086
R2      0073
R3      0074
R4      0075
R5      0076
R6      0077
R7      0078            0154  0159  0201  0201
R8      0079            0156  0157  0204  0206  0208
R9      0080            0160  0161  0166  0168  0170  0175  0177  0178  0179
                        0180  0183  0185  0186  0186  0188  0194  0195  0205
                        0206  0207  0208
RETBUF                  0106
RR                      0110  0199
RWP                     0108
SCBCMD  0128
SCBERR  0129            0183  0195
SCBLST  0133            0149  0186
SIEFLG  0111            0210
SIEINT                  0099  0205
SVC     0115            0091
SVCS1W  0193            0181  0187
SVCS2W  0196            0190
SVCS3W  0213            0202
SVCSR   0153            0092
SVCSRA  0151            0093
SVCW10                  0102
SVCW7                   0104
SVCWP                   0094  0164  0198
T1      0157            0156
T2      0162            0160
T3      0169            015B
TERM                    0096  013B
USRPC                   0100  0157  0165

THERE ARE    0048  SYMBOLS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV	
SHEET	

REV STATUS OF SHEETS	REV	
	SHEET	

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, WKSPMG, WORKSPACE MANAGEMENT-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945382-9901
SCALE	REV.	SHEET 1 of 7

TI-0023-C

```
0003                * TITLE:     WKSPMG
0004                *            WORKSPACE MANAGEMENT ROUTINES
0005                * REVISION:
0006                *            ORIGINAL
0007                * COMPUTER:  990,ASM
0008                * ABSTRACT:  ALLOCATES AND DEALLOWACATES NEW BUFFERS AND
0009                *            WORKSPACES.
0010                *
0011                * CALLING SEQUENCE:
0012                *            BLWP @INIMEM
0013                *              LINK ALL CELLS TOGETHER IN A FREE POOL
0014                *
0015                *            BLWP @GETBUF
0016                *              ALLOCATE A BUFFER
0017                *              RETURN
0018                *                R10 = ADDRESS OF ALLOCATED BUFFER
0019                *
0020                *            BLWP @ RETBUF
0021                *              ENTRY
0022                *                R10 = POINTS TO BUFFER
0023                *
0024                *            BLWP @LWP
0025                *              LINK WORKSPACE
0026                *              ENTRY
0027                *                R10 = ADDR OF NEW WORKSPACE TO SWITCH TO
0028                *              RETURN
0029                *                R13 = ADDR OF OLD WSP
0030                *                R12 = COPY OF OLD R12
0031                *
0032                *            BLWP @RWP
0033                *              ENTRY
0034                *              RETURN WORKSPACE
0035                *                R13 = ADDR OF RETURN WORKSPACE
0036                *
0037                *            BLWP @ACL
0038                *              ALLOCATE,COPY AND LINK
0039                *              RETURN
0040                *                R13 = ADDR OF OLD WORKSPACE
0041                *
0042                *            BLWP @RR
0043                *              RETURN LINK WORKSPACE AND DEALLOCATE
0044                *              ENTRY
0045                *                R13 = ADDR OF WORKSPACE TO RETURN TO
0046                *              RETURN
0047                *                R10 = ADDR OF DEALLOCATED WORKSPACE
0048                *
```

```
0050                    *
0051                    *       WORKSPACE DEFINITIONS
0052                    *
0053        0000  R0    EQU  0
0054        0001  R1    EQU  1
0055        0002  R2    EQU  2
0056        0003  R3    EQU  3
0057        0004  R4    EQU  4
0058        0005  R5    EQU  5
0059        0006  R6    EQU  6
0060        0007  R7    EQU  7
0061        0008  R8    EQU  8
0062        0009  R9    EQU  9
0063        000A  R10   EQU  10
0064        000B  R11   EQU  11
0065        000C  R12   EQU  12
0066        000D  R13   EQU  13
0067        000E  R14   EQU  14
0068        000F  R15   EQU  15
0069                    *
0070                    *
0071                    *       REF'S AND DEF'S
0072                    *
0073                            DEF  GETBUF
0074                            DEF  RETBUF
0075                            DEF  LWP
0076                            DEF  RWP
0077                            DEF  ACL
0078                            DEF  RR
0079                            REF  MEMWP
0080                            REF  FREMEM
0081                            DEF  INIMEM
0082                    *
0083        000A  NOCELS EQU  10
```

```
0085                    *
0086                    *       LINKAGE VECTORS
0087                    *
0088  0000  0000               DATA 0
0089        0002' INIMEM EQU  $             INITIALIZE MEMORY
0090  0002  0000               DATA MEMWP
0091  0004  001E'               DATA MEMINI
0092                    *
0093        0006' GETBUF EQU  $             ALLOCATE BUFFER
0094  0006  0002'               DATA MEMWP
0095  0008  003A'               DATA MEMALC
0096                    *
0097        000A' RETBUF EQU  $             RETURN BUFFER
0098  000A  0006'               DATA MEMWP
0099  000C  0046'               DATA MEMRTN
0100                    *
0101        000E' LWP    EQU  $             LINK WORKSPACE
0102  000E  000A'               DATA MEMWP
0103  0010  0058'               DATA LWPA
0104                    *
0105        0012' RWP    EQU  $             RETURN WORKSPACE
0106  0012  000E'               DATA MEMWP
0107  0014  006E'               DATA RWPA
0108                    *
0109        0016' ACL    EQU  $             ALLOCATE COPY AND LINK WORKSPACE
0110  0016  0012'               DATA MEMWP
0111  0018  0074'               DATA ACLA
0112        001A' RR     EQU  $             RETURN LINK WORKSPACE AND DEALLOCA
0113  001A  0016'               DATA MEMWP
0114  001C  0050'               DATA RRA
```

```
0116                    *
0117                    *       MEMORY REGISTER CONTROL CONVENTION
0118                    *
0119                    *       R8  = POINTER TO TOP OF STACK
0120                    *
0121                    *
0122                    *
0123                    *       MEMORY INITIALIZATION = LINK ALL CELLS
0124                    *       TOGETHER IN FREE POOL
0125                    *
0126          001E'     MEMINI  EQU   $
0127  001E    0209              LI    R9,FREMEM     SET HDR, TAIL POINTERS
      0020    0000
0128  0022    0207              LI    R7,NOCELS-1   # OF AVAILABLE CELLS
      0024    0009
0129          0026'     M1      EQU   $
0130  0026    C209              MOV   R9,R8
0131  0028    0229              AI    R9,32         SET LINK TO NEXT
      002A    0020
0132  002C    C609              MOV   R9,*R8
0133  002E    04D9              CLR   *R9           CLEAR TAIL CELL
0134  0030    0607              DEC   R7
0135  0032    15F9              JGT   M1
0136  0034    0208              LI    R8,FREMEM     SET HEADER POINTER
      0036    0020'
0137  0038    0380              RTWP
0138                    *
0139                    *       ALLOCATE BUFFER
0140                    *
0141          003A'     MEMALC  EQU   $
0142  003A    CB48              MOV   R8,@20(R13)   ALLOCATE CELL
      003C    0014
0143  003E    1601              JNE   M2
0144  0040    10FF              JMP   $             IF OUT OF MEMORY
0145          0042'     M2      EQU   $
0146  0042    C218              MOV   *R8,R8        POP NEXT CELL
0147  0044    0380              RTWP
0148                    *
0149                    *       RETURN BUFFER
0150                    *
0151          0046'     MEMRTN  EQU   $
0152  0046    C1ED              MOV   @20(R13),R7   @ TO CELL TO BE RETURNED
      0048    0014
0153          004A'     MEMRT2  EQU   $
0154  004A    C5C8              MOV   R8,*R7
0155  004C    C207              MOV   R7,R8
0156  004E    0380              RTWP
0157                    *
0158                    *       RETURN LINK TO PREVIOUS WORKSPACE
0159                    *       DEALLOCATE CURRENT WORKSPACE
0160                    *
0161          0050'     RRA     EQU   $
0162  0050    C1CD              MOV   R13,R7        CELL TO BE RETURNED
0163  0052    C36D              MOV   @R13*2(R13),R13
```

```
       0054  001A
 0164  0056  10F9          JMP   MEMRT2
 0165                *
 0166                *     LINK  AND SWITCH TO WORKSPACE IN CALLER'S R10
 0167                *
 0168        0058' LWPA    EQU   $
 0169  0058  C1E0          MOV   @R10*2(R13),R7     NEW WORKSPACE
       005A  0014
 0170        005C' LWPB    EQU   $
 0171  005C  C14D          MOV   R13,R5             SAVE RETURN WORKSPACE
 0172  005E  C1A0          MOV   @R12*2(R13),R6     SAVE CRU BASE
       0060  0018
 0173  0062  C347          MOV   R7,R13         SET NEW WORKSPACE
 0174  0064  CB46          MOV   R6,@R12*2(R13)
       0066  0018
 0175  0068  CB45          MOV   R5,@R13*2(R13)
       006A  001A
 0176  006C  0380          RTWP
 0177                *
 0178                *     SWITCH TO WORKSPACE GIVEN IN CALLER'S WORKSPACE @
 0179                *
 0180        006E' RWPA    EQU   $
 0181  006E  C36D          MOV   @R13*2(R13),R13
       0070  001A
 0182  0072  0380          RTWP
 0183                *
 0184                *     ALLOCATE NEW WORKSPACE
 0185                *     COPY R0-R10
 0186                *     LINK WORKSPACE
 0187                *
 0188        0074' ACLA    EQU   $
 0189  0074  C288          MOV   R8,R10
 0190  0076  C1C8          MOV   R8,R7
 0191  0078  1601          JNE   M5
 0192  007A  10FF          JMP   $
 0193        007C' M5      EQU   $
 0194  007C  C218          MOV   *R8,R8
 0195  007E  020B          LI    R11,11
       0080  000B
 0196  0082  C30D          MOV   R13,R12
 0197        0084' M6      EQU   $
 0198  0084  CEBC          MOV   *R12+,*R10+
 0199  0086  060B          DEC   R11
 0200  0088  15FD          JGT   M6
 0201  008A  10E8          JMP   LWPB
 0202                      END
0000 ERS
```

```
$                       0089  0093  0097  0101  0105  0109  0112  0126  0129
                        0141  0144  0145  0151  0153  0161  0168  0170  0180
                        0188  0192  0193  0197
ACL         0109        0077
ACLA        0188        0111
FREMEM                  0080  0127  0136
GETBUF      0093        0073
INIMEM      0089        0081
LWP         0101        0075
LWPA        0168        0103
LWPB        0170        0201
M1          0129        0135
M2          0145        0143
M5          0193        0191
M6          0197        0200
MEMALC      0141        0095
MEMINI      0126        0091
MEMRT2      0153        0164
MEMRTN      0151        0099
MEMWP                   0079  0090  0094  0098  0102  0106  0110  0113
NOCELS      0083        0128
R0          0053
R1          0054
R10         0063        0169  0189  0198
R11         0064        0195  0199
R12         0065        0172  0174  0196  0198
R13         0066        0142  0152  0162  0163  0163  0163  0169  0171  0172
                        0173  0174  0175  0175  0181  0181  0181  0196
R14         0067
R15         0068
R2          0055
R3          0056
R4          0057
R5          0058        0171  0175
R6          0059        0172  0174
R7          0060        0128  0134  0152  0154  0155  0162  0169  0173  0190
R8          0061        0130  0132  0136  0142  0146  0146  0154  0155  0189
                        0190  0194  0194
R9          0062        0127  0130  0131  0132  0133
RETBUF      0097        0074
RR          0112        0078
RRA         0161        0114
RWP         0105        0076
RWPA        0180        0107

THERE ARE   0040  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV	
SHEET	

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN — DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, USRPGM, USER PROGRAM SUPERVISOR CALLS-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945383-9901

SCALE	REV.	SHEET 1 of 3

TI-0023-C

```
0003                    IDT  'USRPGM'
0004                * TITLE:    USRPGM
0005                *           USER PROGRAM CONTROL
0006                * REVISION:
0007                *           ORIGINAL
0008                * COMPUTER: 990,ASM
0009                * ABSTRACT: USER PROGRAM CNTROL IS A SET OF
0010                *           ROUTINES TO BE USED TO START AND
0011                *           TERMINATE USER PROGRAMS.
0012                *           1)START A USER PROGRAM AT LOCATION
0013                *             INDICATED BY THE START VECTOR.
0014                *           2)TERMINATE A USER PROGRAM IN RESPONSE
0015                *             TO AN END OF PROGRAM CALL.
0016                * CALLING SEQUENCE:
0017                *           EXECUTE USER PROGRAM
0018                *           BL   @EXCT
0019                *           R10 CONTAINS POINTER TO COMMAND PARAMETER LIST
0020                *
0021                *           TERMINATE USER PROGRAM
0022                *           BL   @TERM
0023                *           R10 CONTAINS POINTER TO SUPERVISOR CALL BLOCK.
0024                *
0025                *      REF'S AND DEF'S
0026                *
0027                      REF  USRWP              USER STATUS VECTOR
0028                      REF  USRPC
0029                      REF  USRST
0030                      REF  USRWSP
0031                      REF  INIT
0032                      DEF  EXCT
0033                      DEF  TERM
0034                *
0035                *     EXECUTE USER PROGRAM
0036                *
0037         0000' EXCT   EQU  $
0038  0000   02E0        LWPI USRWSP             SET WORKSPACE FOR RETURN
      0002   0000
0039  0004   0380        RTWP                    VECTOR TO POINT TO USER.
0040                *
0041                *     TERMINATE USER PROGRAM
0042                *
0043         0006' TERM   EQU  $
0044  0006   0460        B    @INIT              RE-INIT MONITOR
      0008   0000
0045                      END
0000 ERS
```

```
980 - 980  CONCORDANCE
$                       0037  0043
EXCT     0037           0032
INIT                    0031  0044
TERM     0043           0033
USRPC                   0028
USRST                   0029
USRWP                   0027
USRWSP                  0030  0038

THERE ARE    0008  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV STATUS OF SHEETS

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO.
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, SETREG, SET TRACE REGION-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945385-9901

SCALE | REV. | SHEET 1 of 8

TI-0923-C

```
0003          *   PROCEDURE SR (CPL POINTER);
0004          *      DECLARE NUMBER TABLE ENTRIES LITERALLY '4';
0005          *      DECLARE INSTRUCTION STEP KEY LITERALLY '8';
0006          *      DECLARE (1 TRACE REGION TABLES,
0007          *                  3 TRC REGION TBL (4),
0008          *                     5 TR FLAGS,
0009          *                        7 TF DEFINED BIT (1),
0010          *                        7 TF VARIABLE (3) BIT (1),
0011          *                        7 TF INST STEP BIT (1),
0012          *                        7 TF FILLER BIT (3),
0013          *                     5 TR FILLER BIT (8),
0014          *                     5 TR LOW BOUND POINTER,
0015          *                     5 TR HIGH BOUND POINTER,
0016          *                     5 TR TYPE INDEX FIXED (16),
0017          *                     5 TR VARIABLE (3) POINTER,
0018          *                     5 TR RESERVED BIT (16));
0019          *      DECLARE (REGION INDEX, TYPE INDEX, I)
0020          *           FIXED (16);
0021          *      DECLARE (LOW BOUND, HIGH BOUND) POINTER;
0022          *      DECLARE (1 COMMAND PARAMETER LIST,
0023          *                  3 CPL CONTROL (CPL POINTER),
0024          *                     5 NUMBER PARMS FIXED (8),
0025          *                     5 PARM PRESENT (8) BIT (1),
0026          *                     5 FIRST PARM FIXED (8));
0027          *      DECLARE NUMERIC PARAMETER FIXED (16)
0028          *           CONTROL (PARM POINTER);
0029          *      DECLARE (1 CHARACTER PARAMETER,
0030          *                  3 CHAR PARM CONTROL (PARM POINTER),
0031          *                     5 STRING LENGTH FIXED (8),
0032          *                     5 PARM STRING (1) CHARACTER (1));
0033          *      DECLARE (CPL POINTER, PARM POINTER) POINTER;
0034          *      PARM POINTER = ADDR (FIRST PARM)
0035          *      IF PARM PRESENT (0) = 0
0036          *         THEN CALL ERROR ('MS05'); /* NO RETURN */
0037          *      REGION INDEX = NUMERIC PARAMETER;
0038          *      PARM POINTER = PARM POINTER + 2;
0039          *      IF REGION INDEX >= NUM TRACE REGIONS
0040          *         THEN CALL ERROR ('DP10'); /* NO RETURN */
0041          *      IF PARM PRESENT (1) = 0
0042          *         THEN CALL ERROR ('MS05'); /* NO RETURN */
0043          *      LOW BOUND = NUMERIC PARAMETER;
0044          *      PARM POINTER = PARM POINTER + 2;
0045          *      IF PARM PRESENT (2) = 0
0046          *         THEN CALL ERROR ('MS05'); /* NO RETURN */
0047          *      HIGH BOUND = NUMERIC PARAMETER;
0048          *      PARM POINTER = PARM POINTER + 2;
0049          *      IF HIGH BOUND < LOW BOUND
0050          *         THEN CALL ERROR ('DP013); /* NO RETURN */
0051          *      IF PARM PRESENT (3) = 0
0052          *         THEN CALL ERROR ('MS05'); /* NO RETURN */
0053          *      TYPE INDEX = NUMERIC PARAMETER;
0054          *      PARM POINTER = PARM POINTER + 2;
0055          *      IF TYPE INDEX >= NUMBER TABLE ENTRIES
0056          *         THEN CALL ERROR ('DP26'); /* NO RETURN */
```

```
0057                *       TRC REGION TBL (REGION INDEX).TR LOW BOUND
0058                *          = LOW BOUND;
0059                *       TRC REGION TBL (REGION INDEX).TR HIGH BOUND
0060                *          = HIGH BOUND;
0061                *       TRC REGION TBL (REGION INDEX).TR TYPE INDEX
0062                *          = TYPE INDEX;
0063                *       TF DEFINED = 1;
0064                *       IF PARM PRESENT (4) = 1 THEN DO;
0065                *          IF PARM STRING (0) = INSTRUCTION STEP KEY
0066                *             THEN TF INST STEP = 1;
0067                *          PARM POINTER =
0068                *                ((PARM POINTER + STRING LENGTH + 2)/2)*2;
0069                *          END;
0070                *       DO I FROM 0 TO 2 BY 1;
0071                *          IF PARM PRESENT (I+5) = 1 THEN DO;
0072                *             TR VARIABLE (I) = NUMERIC PARAMETER;
0073                *             PARM POINTER = PARM POINTER + 1;
0074                *             TF VARIABLE (I) = 1;
0075                *             END;
0076                *       RETURN;
0077                *       END;
0078                        IDT   'SETREGN'
0079                * TITLE:    SETREGN
0080                *           PROCESS THE SET REGION COMMAND
0081                * REVISION:
0082                *           ORIGINAL
0083                * COMPUTER: 990
0084                * ABSTRACT:
0085                *           THIS SUBROUTINE INTERPRETS THE PARAMETERS OF
0086                *           THE SR (SET REGION) COMMAND TO MAKE AN ENTRY
0087                *           IN THE TRACE REGION TABLES.
0088                * CALLING SEQUENCE:
0089                *           STANDARD CALL FROM COMMAND STRING PROCESSOR.
0090                *           <R10> = COMMAND PARAMETER LIST POINTER
0091                *           BL    *SR
0092                *                                 1=PROCEDURE SR (CPL POINTER);
0093                *                                 2=DECLARE NUMBER TABLE ENTRIES
0094                *           REF   NUMTTE           NUMBER OF TRACE TYPE ENTRIES
0095                *                                 2=DECLARE INSTRUCTION STEP KEY
0096        0100    LBYTE EQU   >100
0097        5300    ISRKEY EQU  'S'*LBYTE
0098                *           REF   ERROR
0099                *                                 2=DECLARE (1 TRACE REGION TABL
0100                *                                 3=          3 TRC REGION TBL (4
0101                *                                 4=          5 TR FLAGS,
0102                *                                 5=          7 TF DEFINED BIT (1
0103                *                                 5=          7 TF VARIABLE (3) B
0104                *                                 5=          7 TF INST STEP BIT
0105                *                                 5=          7 TF FILLER BIT (3)
0106                *                                 4=          5 TR FILLER BIT (8)
0107                *                                 4=          5 TR LOW BOUND POIN
0108                *                                 4=          5 TR HIGH BOUND POI
0109                *                                 4=          5 TR TYPE INDEX FIX
0110                *                                 4=          5 TR VARIABLE (3) P
0111                *                                 4=          5 TR RESERVED BIT (
```

```
0112                        REF   TRTBL
0113                        REF   TRFLGS,TRLOW,TRHIGH,TRTYPE,TRV1
0114                        REF   TRLNTH
0115                        REF   ISRFLG,NUMTR
0116                  *                                 2*DECLARE (REGION INDEX, TYPE
0117                  *                                 4*FIXED (16));
0118                  *                                 2*DECLARE (LOW BOUND, HIGH BOU
0119                  *                                 2*DECLARE (1 COMMAND PARAMETER
0120                  *                                 3*          3 CPL CONTROL (CPL
0121                  *                                 4*          5 NUMBER PARMS FIXE
0122                  *                                 4*          5 PARM PRESENT (8)
0123                  *                                 4*          5 FIRST PARM FIXED
0124                  *                                 2*DECLARE NUMERIC PARAMETER FI
0125                  *                                 2*      CONTROL (PARM POINTER);
0126                  *                                 2*DECLARE (1 CHARACTER PARAMET
0127                  *                                 3*          3 CHAR PARM CONTROL
0128                  *                                 4*          5 STRING LENGTH FIX
0129                  *                                 4*          5 PARM STRING (1) C
0130                  *                                 2*DECLARE (CPL POINTER, PARM P
0131                  *
0132                  *WORKSPACE REGISTER DEFINITIONS
0133                  *
0134        0000  R0      EQU   0
0135        0001  R1      EQU   1
0136        0002  R2      EQU   2
0137        0003  R3      EQU   3
0138        0004  R4      EQU   4
0139        0005  R5      EQU   5
0140        0006  R6      EQU   6
0141        0007  R7      EQU   7
0142        0008  R8      EQU   8
0143        0009  R9      EQU   9
0144        000A  R10     EQU   10
0145        000B  R11     EQU   11
0146        000C  R12     EQU   12
0147        000D  R13     EQU   13
0148        000E  R14     EQU   14
0149        000F  R15     EQU   15
0150                  *
```

```
0152                        DEF   SR
0153          0000' SR      EQU   $
0154  0000    C08B          MOV   R11,R2          SAVE RETURN, CPL POINTER
0155                  *                           2-PARM POINTER = ADDR (FIRST P
0156  0002    C13A          MOV   *R10+,R4
0157                  *                           2-IF PARM PRESENT (0) = 0
0158                  *                           3-THEN CALL ERROR ('MS05'); /*
0159  0004    0A94          SLA   R4,9
0160  0006    1741          JNC   SRER1           IF FIRST PARAMETER MISSING
0161                  *                           2-REGION INDEX = NUMERIC PARAM
0162                  *                           2-PARM POINTER = PARM POINTER
0163  0008    C17A          MOV   *R10+,R5
0164                  *                           2-IF REGION INDEX >= NUM TRACE
0165                  *                           3-THEN CALL ERROR ('DP10'); /*
0166  000A    0285          CI    R5,NUMTR
      000C    0000
0167  000E    1440          JHE   SRER2
0168  0010    3960          MPY   @TRLNTH,R5
      0012    0000
0169  0014    C146          MOV   R6,R5
0170                  *                           2-IF PARM PRESENT (1) = 0
0171                  *                           3-THEN CALL ERROR ('MS05'); /*
0172  0016    0A14          SLA   R4,1
0173  0018    1738          JNC   SRER1
0174                  *                           2-LOW BOUND = NUMERIC PARAMETE
0175                  *                           2-PARM POINTER = PARM POINTER
0176  001A    C1BA          MOV   *R10+,R6
0177                  *                           2-IF PARM PRESENT (2) = 0
0178                  *                           3-THEN CALL ERROR ('MS05'); /*
0179  001C    0A14          SLA   R4,1
0180  001E    1735          JNC   SRER1
0181                  *                           2-HIGH BOUND = NUMERIC PARAMET
0182                  *                           2-PARM POINTER = PARM POINTER
0183  0020    C1FA          MOV   *R10+,R7
0184                  *                           2-IF HIGH BOUND < LOW BOUND
0185                  *                           3-THEN CALL ERROR ('DP013'); /*
0186  0022    8187          C     R7,R6
0187  0024    1A38          JL    SRER3
0188                  *                           2-IF PARM PRESENT (3) = 0
0189                  *                           3-THEN CALL ERROR ('MS05'); /*
0190  0026    0A14          SLA   R4,1
0191  0028    1730          JNC   SRER1
0192                  *                           2-TYPE INDEX = NUMERIC PARAMET
0193                  *                           2-PARM POINTER = PARM POINTER
0194  002A    C23A          MOV   *R10+,R8
0195                  *                           2-IF TYPE INDEX >= NUMBER TABL
0196                  *                           3-THEN CALL ERROR ('DP26'); /*
0197  002C    0288          CI    R8,NUMTTE
      002E    0000
0198  0030    1435          JHE   SRER4
0199  0032    0225          AI    R5,TRTBL
      0034    0000
0200                  *                           2-TRC REGION TBL (REGION INDEX
0201                  *                           3-= LOW BOUND);
```

```
0202  0036  C945          MOV   R6,@TRLOW(R5)
      0038  0000
0203                *                              2=TRC REGION TBL (REGION INDEX
0204                *                              3=* HIGH BOUND);
0205  003A  C947          MOV   R7,@TRHIGH(R5)
      003C  0000
0206                *                              2=TRC REGION TBL (REGION INDEX
0207                *                              3=* TYPE INDEX);
0208  003E  C948          MOV   R8,@TRTYPE(R5)
      0040  0000
0209                *                              2=TF DEFINED = 1;
0210  0042  0206          LI    R6,>8000
      0044  8000
0211                *                              2=IF PARM PRESENT (4) = 1 THEN
0212  0046  0A14          SLA   R4,1
0213  0048  170D          JNC   SR020
0214                *                              3=IF PARM STRING (0) = INSTRUC
0215  004A  D1FA          MOVB  *R10+,R7
0216  004C  0987          SRL   R7,8                   SAVE CHARACTER COUNT
0217  004E  04C8          CLR   R8
0218  0050  D21A          MOVB  *R10,R8                ISOLATE FIRST CHARACTER
0219  0052  0288          CI    R8,ISRKEY
      0054  5300
0220  0056  1602          JNE   SR010                  IF FLAG .NE. 'S'
0221                *                              4=THEN TF INST STEP = 1;
0222  0058  F1A0          SOCB  @ISRFLG,R6
      005A  0000
0223        005C' SR010   EQU   $
0224                *                              3=PARM POINTER =
0225                *                              5=((PARM POINTER + STRING LENG
0226  005C  A287          A     R7,R10
0227  005E  058A          INC   R10
0228  0060  081A          SRA   R10,1
0229  0062  0A1A          SLA   R10,1
0230        0064' SR020   EQU   $
0231                *                              3=END;
0232                *                              2=DO I FROM 0 TO 2 BY 1;
0233  0064  0207          LI    R7,>8000
      0066  8000
0234  0068  0208          LI    R8,3
      006A  0003
0235  006C  C045          MOV   R5,R1
0236  006E  0641          DECT  R1
0237        0070' SR030   EQU   $
0238  0070  05C1          INCT  R1
0239  0072  0608          DEC   R8
0240  0074  1107          JLT   SR040                  IF END.OF.LOOP
0241                *                              3=IF PARM PRESENT (I+5) = 1 TH
0242  0076  0917          SRL   R7,1
0243  0078  0A14          SLA   R4,1
0244  007A  17FA          JNC   SR030                  IF PARAMETER NOT PRESENT
0245                *                              4=TR VARIABLE (I) = NUMERIC PA
0246                *                              4=PARM POINTER = PARM POINTER
0247  007C  C87A          MOV   *R10+,@TRV1(R1)
      007E  0000
```

```
 0248                    *                                  4=TF VARIABLE (I) = 1;
 0249  0080  F187          SOCB R7,R6
 0250                    *                                  4=END;
 0251  0082  10F6          JMP  SR030
 0252        0084' SR040   EQU  $
 0253  0084  C946          MOV  R6,@TRFLGS(R5)     SET FLAGS WORD
       0086  0000
 0254                    *                                  2=RETURN;
 0255        0088' SR100   EQU  $
 0256  0088  0452          B    *R2
 0257                    *
 0258                    *
 0259        008A' SRER1   EQU  $
 0260  008A  020A          LI   R10,>0205          "MS05" REQD PARM MISSING
       008C  0205
 0261  008E  1003          JMP  SRERR
 0262        0090' SRER2   EQU  $
 0263  0090  020A          LI   R10,>1110          "DP10" INV. TRACE REGION INDE
       0092  1110
 0264  0094  1005          JMP  SRERR
 0265        0096' SRER3   EQU  $
 0266  0096  020A          LI   R10,>1113          "DP13" HIGH < LOW
       0098  1113
 0267  009A  1002          JMP  SRERR
 0268        009C' SRER4   EQU  $
 0269  009C  020A          LI   R10,>1126          "DP26" INV. TRACE TYPE INDEX
       009E  1126
 0270        00A0' SRERR   EQU  $
 0271  00A0  06A0          BL   @ERROR
       00A2  0000
 0272  00A4  10F1          JMP  SR100
 0273                    *                                  2=END;
 0274                          END
0000 ERS
```

```
$                       0153  0223  0230  0237  0252  0255  0259  0262  0265
                        0268  0270
ERROR                   0098  0271
ISRFLG                  0115  0222
ISRKEY   0097           0219
LBYTE    0096           0097
NUMTR                   0115  0166
NUMTTE                  0094  0197
R0       0134
R1       0135           0235  0236  0238  0247
R10      0144           0156  0163  0176  0183  0194  0215  0218  0226  0227
                        0228  0229  0247  0260  0263  0266  0269
R11      0145           0154
R12      0146
R13      0147
R14      0148
R15      0149
R2       0136           0154  0256
R3       0137
R4       0138           0156  0159  0172  0179  0190  0212  0243
R5       0139           0163  0166  0168  0169  0199  0202  0205  0208  0235
                        0253
R6       0140           0169  0176  0186  0202  0210  0222  0249  0253
R7       0141           0183  0186  0205  0215  0216  0226  0233  0242  0249
R8       0142           0194  0197  0208  0217  0218  0219  0234  0239
R9       0143
SR       0153           0152
SR010    0223           0220
SR020    0230           0213
SR030    0237           0244  0251
SR040    0252           0240
SR100    0255           0272
SRER1    0259           0160  0173  0180  0191
SRER2    0262           0167
SRER3    0265           0187
SRER4    0268           0198
SRERR    0270           0261  0264  0267
TRFLGS                  0113  0253
TRHIGH                  0113  0205
TRLNTH                  0114  0168
TRLOW                   0113  0202
TRTBL                   0112  0199
TRTYPE                  0113  0208
TRV1                    0113  0247

THERE ARE   0041  SYMBOLS
```

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																														
SHEET																														
REV STATUS OF SHEETS	REV																													
	SHEET																													

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DWN	DATE
	CHK	
	ENGR	
	QA	
	APVD	
	CONTR NO	
	DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, SETRACE, SET TRACE-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945386-9901

SCALE	REV.	SHEET
		1 of 9

TI-0929-C

```
0003          *   PROCEDURE STRACE (CPL POINTER);
0004          *      DECLARE TRACE TABLE NUMBER FIXED (16);
0005          *      DECLARE NUMBER TABLE ENTRIES LITERALLY '4';
0006          *      DECLARE NUM LETTERS INPUT FIXED (16);
0007          *      DECLARE BIT MASK LITERALLY '"8000"';
0008          *      DECLARE MASK BIT (16);
0009          *      DECLARE FOUND BIT (1);
0010          *      DECLARE (1 COMMAND PARAMETER LIST,
0011          *                  3 CPL CONTROL (CPL POINTER),
0012          *                     5 NUMBER PARMS FIXED (8),
0013          *                     5 PARM PRESENT (8) BIT (1),
0014          *                     5 FIRST PARM FIXED (8));
0015          *      DECLARE NUMERIC PARAMETER FIXED (16)
0016          *           CONTROL (PARM POINTER);
0017          *      DECLARE (CPL POINTER, PARM POINTER) POINTER;
0018          *      DECLARE NUM PARAMETER FIXED (8)
0019          *           CONTROL (PARM POINTER);
0020          *      DECLARE (I, BIAS) FIXED (16);
0021          *      DECLARE CHAR PARAMETER CHARACTER (1)
0022          *           CONTROL (PARM POINTER);
0023          *      DECLARE CHAR UNDER TEST CHARACTER (1);
0024          *      DECLARE KEY LETTER (13) CHARACTER (1)
0025          *         INITIAL ('P', 'I', 'M', 'W', 'T', 'C', 'X',
0026          *                  'S', 'D', 'E', 'B', 'A', 'R');
0027          *      DECLARE LETTER TYPE (13) FIXED (4)
0028          *         INITIAL (0, 0, 0, 0, 0, 0, 0, 1, 1,
0029          *                  2, 2, 2, 2);
0030          *         /* NORMAL = 0,  BIAS = 1, BIASED = 2 */
0031          *      DECLARE LETTER BIT BIAS (13) FIXED (4)
0032          *         INITIAL (0, 1, 2, 3, 4, 5, 6, 7, 11,
0033          *                  0, 1, 2, 3);
0034          *         /* FOR TYPE 0, BIT NUMBER
0035          *               TYPE 1, BIAS
0036          *               TYPE 2, DISPLACEMENT FROM BIAS */
0037          *      DECLARE NUM KEYS LITERALLY '13';
0038          *      DECLARE TRACE TYPE TABLE (4) BIT (16);
0039          *      IF PARM PRESENT (1) = 0
0040          *           THEN CALL ERROR ('MS05'); /* NO RETURN */
0041          *      PARM POINTER = ADDR (FIRST PARM);
0042          *      TRACE TABLE NUMBER = NUMERIC PARAMETER;
0043          *      PARM POINTER = PARM POINTER + 2;
0044          *      IF TRACE TABLE NUMBER >= NUMBER TABLE ENTRIES
0045          *           THEN CALL ERROR ('DP26'); /* NO RETURN */
0046          *      IF PARM PRESENT (2) = 0
0047          *           THEN CALL ERROR ('MS05'); /* NO RETURN */
0048          *      NUM LETTERS INPUT = NUM PARAMETER;
0049          *      PARM POINTER = PARM POINTER + 1;
0050          *      BIAS = 0;
0051          *      MASK = 0;
0052          *      DO WHILE NUM LETTERS INPUT > 0;
0053          *         NUM LETTERS INPUT = NUM LETTERS INPUT - 1;
0054          *         CHAR UNDER TEST = CHAR PARAMETER;
0055          *         PARM POINTER = PARM POINTER + 1;
0056          *         I = NUM KEYS;
```

```
0057                *          FOUND = 0;
0058                *          DO WHILE I > 0  &  FOUND = 0;
0059                *             I = I - 1;
0060                *             IF CHAR UNDER TEST = KEY LETTER (I) THEN DO;
0061                *                FOUND = 1;
0062                *                DO CASE LETTER TYPE (I);
0063                *                   MASK = OR (MASK  RSHIFT (BIT MASK,
0064                *                         LETTER BIT BIAS (I)));  /* NORMAL E
0065                *                   BIAS = LETTER BIT BIAS (I); /* SET BIAS
0066                *                   DO;              /* BIASED ENTRY */
0067                *                      IF BIAS = 0 THEN CALL ERROR ('DP23');
0068                *                            /* NO RETURN */
0069                *                      MASK = OR (MASK, RSHIFT (BIT MASK,
0070                *                            LETTER BIT BIAS (I) + BIAS));
0071                *                      END;
0072                *                   END;
0073                *                END;
0074                *             END;
0075                *          IF FOUND = 0 THEN CALL ERROR ('DP23');
0076                *                /* NO RETURN */
0077                *          END;
0078                *       TRACE TYPE TABLE (TRACE TABLE NUMBER) = MASK;
0079                *       RETURN;
0080                *       END STRACE;
0081                       IDT  'SETRACE'
0082                * TITLE:    SETRACE
0083                *           PROCESS THE SET TRACE COMMAND
0084                * REVISION:
0085                *           ORIGINAL
0086                * COMPUTER: 990
0087                * ABSTRACT:
0088                *           THIS SUBROUTINE PROCESSES THE SET TRACE COMMAND
0089                *           THE CHARACTER STRING SECOND PARAMETER IS
0090                *           PARSED LEFT TO RIGHT, GENERATING A MASK OF
0091                *           TRACE ENTITIES WHICH IS PLACED IN THE TRACE
0092                *           TYPE TABLE ACCORDING TO THE NUMERIC FIRST
0093                *           PARAMETER.  BOTH PARAMETERS ARE REQUIRED.
0094                *
0095                *           THE SYNTAX OF THE STRING IS SIMPLY A STRING
0096                *           OF CHARACTERS IN THE SET:
0097                *               (P, I, M, W, T, C, X, S, D, E, B, A, R).
0098                *           IN ADDITION, THE SUBSET (E, B, A, R) IS VALID
0099                *           ONLY TO THE RIGHT OF AN ELEMENT OF (S, D).
0100                *
0101                *              KEYLETTER       MEANING
0102                *
0103                *                  P           PROGRAM COUNTER
0104                *                  I           INSTRUCTION AND FORMAT
0105                *                  M           STATUS MASK
0106                *                  W           WORKSPACE POINTER CHANGES
0107                *                  T           BRANCH TARGETS
0108                *                  C           CRU ADDRESS
0109                *                  X           XOP LEVEL
0110                *                  N           TRACE NOTHING
0111                *
```

```
0112                    *                   S          SOURCE PREFIX
0113                    *                   D          DESTINATION PREFIX
0114                    *                   E            = EFFECTIVE ADDRESS (EA)
0115                    *                   B            = (EA) BEFORE EXECUTION
0116                    *                   A            = (EA) AFTER EXECUTION
0117                    *                   R            = WKSP REGISTER CHANGES
0118                    *                                    WHEN TS/TD .NE. 0
0119                    * CALLING SEQUENCE:
0120                    *           STANDARD CALL FROM COMMAND STRING PROCESSOR.
0121                    *           <R10> = COMMAND PARAMETER LIST POINTER
0122                    *           BL    @STRACE
0123                    *
0124                    *                                1=PROCEDURE STRACE (CPL POINTE
0125                    *                                2=DECLARE TRACE TABLE NUMBER F
0126                    *                                2=DECLARE NUMBER TABLE ENTRIES
0127                          REF   NUMTTE
0128                    *                                2=DECLARE NUM LETTERS INPUT FI
0129                    *                                2=DECLARE BIT MASK LITERALLY '
0130                    *                                2=DECLARE MASK BIT (16);
0131                    *                                2=DECLARE FOUND BIT (1);
0132                    *                                2=DECLARE (1 COMMAND PARAMETER
0133                    *                                3=          3 CPL CONTROL (CPL
0134                    *                                4=          5 NUMBER PARMS FIXE
0135                    *                                4=          5 PARM PRESENT (8)
0136                    *                                4=          5 FIRST PARM FIXED
0137                    *                                2=DECLARE NUMERIC PARAMETER FI
0138                    *                                2=      CONTROL (PARM POINTER);
0139                    *                                2=DECLARE (CPL POINTER, PARM P
0140                    *                                2=DECLARE NUM PARAMETER FIXED
0141                    *                                2=      CONTROL (PARM POINTER);
0142                    *                                2=DECLARE (I, BIAS) FIXED (16)
0143                    *                                2=DECLARE CHAR PARAMETER CHARA
0144                    *                                2=      CONTROL (PARM POINTER);
0145                    *                                2=DECLARE CHAR UNDER TEST CHAR
0146                    *                                2=DECLARE KEY LETTER (13) CHAR
0147                    *                                3=INITIAL ('P', 'I', 'M', 'W',
0148                    *                                3=          'S', 'D', 'E', 'B',
0149                    *                                2=DECLARE LETTER TYPE (13) FIX
0150                    *                                3=INITIAL (0, 0, 0, 0, 0, 0, 0
0151                    *                                3=          2, 2, 2, 2);
0152                    *                                3=/* NORMAL = 0,  BIAS = 1, BI
0153                    *                                2=DECLARE LETTER BIT BIAS (13)
0154                    *                                3=INITIAL (0, 1, 2, 3, 4, 5, 6
0155                    *                                3=          0, 1, 2, 3);
0156                    *                                3=/* FOR TYPE 0, BIT NUMBER
0157                    *                                3=        TYPE 1, BIAS
0158                    *                                3=        TYPE 2, DISPLACEMENT
0159  0000     50   KEYLTR TEXT 'PIMWTCXSDEBARN'
0160           000E  NUMKEY EQU  $-KEYLTR
0161  000F     00   LTRTYP BYTE 0,1,2,3,4,5,6      P I M W T C X
      000F     01
      0010     02
      0011     03
      0012     04
      0013     05
```

```
                0014    06
0160            0015    17          BYTE >17,>1B            S D
                0016    1B
0163            0017    20          BYTE >20,>21,>22,>23    E B A R
                0018    21
                0019    22
                001A    23
0164            001B    30          BYTE >30                N
0165                        *                               2=DECLARE NUM KEYS LITERALLY '
0166                        *                               2=DECLARE TRACE TYPE TABLE (4)
0167                                REF   TTTBL
0168                        *
0169                        *WORKSPACE REGISTER DEFINITIONS
0170                        *
0171                0000    R0      EQU   0
0172                0001    R1      EQU   1
0173                0002    R2      EQU   2
0174                0003    R3      EQU   3
0175                0004    R4      EQU   4
0176                0005    R5      EQU   5
0177                0006    R6      EQU   6
0178                0007    R7      EQU   7
0179                0008    R8      EQU   8
0180                0009    R9      EQU   9
0181                000A    R10     EQU   10
0182                000B    R11     EQU   11
0183                000C    R12     EQU   12
0184                000D    R13     EQU   13
0185                000E    R14     EQU   14
0186                000F    R15     EQU   15
0187                        *
0188                                REF   ERROR
```

```
0190                        DEF  STRACE
0191        001C' STRACE    EQU  $                 PROCESS SET TRACE COMMAND
0192  001C  C085           MOV  R11,R2            SAVE RETURN, CPL
0193                  *                            2-IF PARM PRESENT (1) = 0
0194                  *                            4-THEN CALL ERROR ('MS05'); /*
0195  001E  C13A           MOV  *R10+,R4
0196  0020  0A94           SLA  R4,9
0197  0022  1732           JNC  STRER1             IF PARM 1 MISSING
0198                  *                            2-PARM POINTER = ADDR (FIRST P
0199                  *                            2-TRACE TABLE NUMBER = NUMERIC
0200                  *                            2-PARM POINTER = PARM POINTER
0201  0024  C18A           MOV  *R10+,R6
0202                  *                            2-IF TRACE TABLE NUMBER >= NUM
0203                  *                            4-THEN CALL ERROR ('DP26'); /*
0204  0026  0286           CI   R6,NUMTTE
      0028  0000
0205  002A  1431           JHE  STRER2
0206  002C  0A16           SLA  R6,1
0207                  *                            2-IF PARM PRESENT (2) = 0
0208                  *                            4-THEN CALL ERROR ('MS05'); /*
0209  002E  0A14           SLA  R4,1
0210  0030  1720           JNC  STRER1             IF PARM 2 MISSING
0211                  *                            2-NUM LETTERS INPUT = NUM PARA
0212                  *                            2-PARM POINTER = PARM POINTER
0213  0032  D13A           MOVB *R10+,R4
0214  0034  0984           SRL  R4,8
0215                  *                            2-BIAS = 0;
0216  0036  04CB           CLR  R11
0217                  *                            2-MASK = 0;
0218  0038  04C7           CLR  R7
0219                  *                            2-DO WHILE NUM LETTERS INPUT >
0220                  *                            3-NUM LETTERS INPUT = NUM LETT
0221        003A' STR010    EQU  $
0222  003A  C104           MOV  R4,R4
0223  003C  1322           JEQ  STR080             IF END.OF.LOOP
0224  003E  0604           DEC  R4
0225                  *                            3-CHAR UNDER TEST = CHAR PARAM
0226                  *                            3-PARM POINTER = PARM POINTER
0227  0040  D17A           MOVB *R10+,R5
0228                  *                            3-I = NUM KEYS;
0229  0042  0201           LI   R1,NUMKEY
      0044  000E
0230                  *                            3-FOUND = 0;
0231        0046' STR020    EQU  $
0232                  *                            3-DO WHILE I > 0  &  FOUND = 0
0233  0046  C041           MOV  R1,R1
0234  0048  1325           JEQ  STRER3             IF NO KEYWORD MATCHES
0235                  *                            4-I = I - 1;
0236  004A  0601           DEC  R1
0237                  *                            4-IF CHAR UNDER TEST = KEY LET
0238  004C  9845           CB   R5,@KEYLTR(R1)
      004E  0000'
0239  0050  16FA           JNE  STR020
0240                  *                            5-FOUND = 1;
```

```
0241                    *                                  5=DO CASE LETTER TYPE (I);
0242  0052  D161              MOVB @LTRTYP(R1),R5
      0054  000E'
0243  0056  D005              MOVB R5,R0                     ISOLATE LETTER BIT BIAS
0244  0058  0A40              SLA  R0,4
0245  005A  09C0              SRL  R0,12
0246  005C  09C5              SRL  R5,12                     ISOLATE LETTER TYPE
0247  005E  0A15              SLA  R5,1
0248  0060  0465              B    @STR040(R5)
      0062  0064'
0249        0064' STR040  EQU  $
0250  0064  1003              JMP  STR050                    NORMAL ENTRY
0251  0066  1007              JMP  STR060                    DEFINE BIAS
0252  0068  1008              JMP  STR070                    BIASED ENTRY
0253  006A  10E7              JMP  STR010                    NO-OP
0254                    *                                  6=MASK = OR (MASK  RSHIFT (BIT
0255                    *                                  8=LETTER BIT BIAS (I)));  /* NO
0256        006C' STR050  EQU  $
0257  006C  0201              LI   R1,>8000
      006E  8000
0258  0070  0B01              SRC  R1,0
0259  0072  E1C1              SOC  R1,R7
0260  0074  10E2              JMP  STR010
0261                    *                                  6=BIAS = LETTER BIT BIAS (I);
0262        0076' STR060  EQU  $
0263  0076  C2C0              MOV  R0,R11
0264  0078  10E0              JMP  STR010
0265                    *                                  6=DO;              /* BIASED ENT
0266                    *                                  7=IF BIAS = 0 THEN CALL ERROR
0267                    *                                  1=/* NO RETURN */
0268        007A' STR070  EQU  $
0269  007A  C2CB              MOV  R11,R11
0270  007C  130B              JEQ  STRER3                    IF (E, B, A, R) LEFT OF
0271                    *                                    FIRST (S, D)
0272                    *                                  7=MASK = OR (MASK, RSHIFT (BIT
0273                    *                                  9=LETTER BIT BIAS (I) + BIAS))
0274                    *                                  7=END;
0275  007E  A00B              A    R11,R0
0276  0080  10F5              JMP  STR050
0277                    *                                  6=END;
0278                    *                                  5=END;
0279                    *                                  4=END;
0280                    *                                  3=IF FOUND = 0 THEN CALL ERROR
0281                    *                                  6=/* NO RETURN */
0282                    *                                  3=END;
0283        0082' STR080  EQU  $
0284                    *                                  2=TRACE TYPE TABLE (TRACE TABL
0285  0082  C987              MOV  R7,@TTTBL(R6)             SET TRACE TYPE
      0084  0000
0286        0086' STR100  EQU  $
0287                    *                                  2=RETURN;
0288  0086  0452              B    *R2
0289                    *                                  2=END STRACE;
0290        0088' STRER1  EQU  $
0291  0088  020A              LI   R10,>0205                 MS05 = REQ PARM MISSING
```

```
       008A  0205
 0292  008C  1005          JMP   STRERR
 0293        008E' STRER2  EQU   $
 0294  008E  020A          LI    R10,>1126        DP26 = SELECT NUMBER RANGE ER
       0090  1126
 0295  0092  1002          JMP   STRERR
 0296        0094' STRER3  EQU   $
 0297  0094  020A          LI    R10,>1123        DP23 = STRING SYNTAX ERROR
       0096  1123
 0298        0098' STRERR  EQU   $
 0299  0098  06A0          BL    @ERROR
       009A  0000
 0300  009C  10F4          JMP   STR100
 0301                      END
0000 ERS
```

960 - 980 CONCORDANCE

$		0160	0191	0221	0231	0249	0256	0262	0268	0283
		0286	0290	0293	0296	0298				
ERROR		0188	0299							
KEYLTR	0159	0160	0238							
LTRTYP	0161	0242								
NUMKEY	0160	0229								
NUMTTE		0127	0204							
R0	0171	0243	0244	0245	0263	0275				
R1	0172	0229	0233	0233	0236	0238	0242	0257	0258	0259
R10	0181	0195	0201	0213	0227	0291	0294	0297		
R11	0182	0192	0216	0263	0269	0269	0275			
R12	0183									
R13	0184									
R14	0185									
R15	0186									
R2	0173	0192	0288							
R3	0174									
R4	0175	0195	0196	0209	0213	0214	0222	0222	0224	
R5	0176	0227	0236	0242	0243	0246	0247	0248		
R6	0177	0201	0204	0206	0285					
R7	0178	0218	0259	0285						
R8	0179									
R9	0180									
STR010	0221	0253	0260	0264						
STR020	0231	0239								
STR040	0249	0248								
STR050	0256	0250	0276							
STR060	0262	0251								
STR070	0268	0252								
STR080	0283	0223								
STR100	0286	0300								
STRACE	0191	0190								
STRER1	0290	0197	0210							
STRER2	0293	0205								
STRER3	0296	0234	0270							
STRERR	0298	0292	0295							
TTTBL		0167	0285							

THERE ARE 0036 SYMBOLS

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																												
SHEET																												

REV STATUS OF SHEETS	REV																						
	SHEET																						

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS INCORPORATED
Equipment Group *Dallas, Texas*

TRACEMOD, TRACE INTERPRETER-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945387-9901
SCALE	REV.	SHEET 1 of 46

TI-0023-C

```
0003                    IDT  'TRACEMOD'
0004              *     DECLARE PCCURSOR POINTER;
0005              *     DECLARE MEMWORD2 BIT (16) CONTROL (PTR);
0006              *     DECLARE PTR POINTER;
0007              *     DECLARE NEWUSRREG (16) BIT (16)
0008              *        CONTROL (NEWWP);
0009              *     DECLARE EXECUTEDINSTRUCTION (3) BIT (16);
0010              *     DECLARE MEMWORD BIT (16) CONTROL (PCCURSOR)
0011              *     DECLARE PRINTFLAG BIT (16) INITIAL ("E000");
0012              *     DECLARE CLEARFLAG BIT (16) INITIAL ("E000");
0013              *     DECLARE NOPINSTRUCTION BIT (16)
0014              *        INITIAL ("1000");
0015              *     DECLARE I FIXED (16);
0016              *     DECLARE (FOUND, CONDITIONMET) FIXED;
0017              *     DECLARE XINSTPOINTER POINTER;
0018              *     DECLARE XEQINST BIT (16)
0019              *        CONTROL (XINSTPOINTER);
0020              *     DECLARE (SRCREG, DSTREG) POINTER;
0021              *     DECLARE SRCREGCONTENTS BIT (16)
0022              *           CONTROL (SRCREG);
0023              *     DECLARE USRREG (16) BIT (16) CONTROL (USRWP)
0024              *     DECLARE CRUFLAG BIT (16) INITIAL ("0400");
0025              *     DECLARE XOPFLAG BIT (16) INITIAL ("0200");
0026              *     DECLARE DISTANCE BIT (16);
0027              *     DECLARE BRANCHFLAG BIT (16)
0028              *        INITIAL ("0800");
```

```
0031              *       DECLARE PRVOPF LITERALLY '1'; /* PRIVILEGED*/
0032              *       DECLARE NPROPF LITERALLY '0'; /* NON-PRIV  */
0033              *       DECLARE ILLOPF LITERALLY '1'; /* ILLEGAL   */
0034              *       DECLARE LEGOPF LITERALLY '0'; /* LEGAL     */
0035              *       DECLARE GAGRPA LITERALLY '0'; /* ADDRESS,  */
0036              *       DECLARE GAGRPB LITERALLY '1'; /*   GROUPS  */
0037              *       DECLARE GAGRPC LITERALLY '2'; /*
0038              *       DECLARE GAGRPD LITERALLY '3';
0039              *       DECLARE GAGRPE LITERALLY '4';
0040              *       DECLARE GROUPLOWERBOUND (26) BIT (16)
0041              *          INITIAL ("4000", "3800", "3000", "2C00",
0042              *                   "2000", "1D00", "1000", "0C00",
0043              *                   "0800", "0780", "06C0", "0680",
0044              *                   "04C0", "0480", "0440", "0400",
0045              *                   "03E0", "03A0", "0380", "0360",
0046              *                   "0340", "0320", "02E0", "02A0",
0047              *                   "0200", "0000");
0048              *       DECLARE (1 TRACEDECODETABLE,
0049              *                   3 DECODEENTRY (26),
0050              *                      5 LEGALITY   BIT (1),
0051              *                      5 PRIVILEGEDOP BIT (1),
0052              *                      5 DEFILLER BIT (2),
0053              *                      5 ADDRESSGROUP BIT (4));
0054              *       DECLARE INSTFORMAT (26) CHARACTER (1)
0055              *          INITIAL ('1', '9', '4', '9', '3', '2', '2',
0056              *                   ' ', '5', ' ', '6', '6', '6', '6',
0057              *                   '6', '6', '7', '7', '7', '7', '7',
0058              *                   'A', '8', '8', '8', ' ');
0059              *       DECLARE PTABLE (15) FIXED (16);
0060              *       DECLARE PRINTTAGS (15) CHARACTER (2)
0061              *          INITIAL (CRLF, 'X-', 'ST', 'WP', 'BT', 'CR',
0062              *                   'XL', 'SE', 'SB', 'SA', 'SR', 'DE',
0063              *                   'DB', 'DA', 'DR');
0064              *       DECLARE FORMAT   LITERALLY PRINTTAGS (1);
0065              *       DECLARE STACKINDEX FIXED (16);
0066              *       DECLARE ADDRSTACK (4) POINTER;
0067              *       DECLARE BEFORECONTENTS (4) BIT (16);
0068              *       DECLARE (1 TRACEREGIONTABLES,
0069              *                   3 TRCREGIONTBL
0070              *                        CONTROL (TRCENTRYADDR),
0071              *                      5 TRFLAGS,
0072              *                        7 TFDEFINED BIT (1),
0073              *                        7 TFVARIABLE (3) BIT (1),
0074              *                        7 TFINSTSTEP BIT (1),
0075              *                        7 TFFILLER BIT (3),
0076              *                      5 TRFILLER BIT (8),
0077              *                      5 TRLOWBOUND POINTER,
0078              *                      5 TRHIGHBOUND POINTER,
0079              *                      5 TRTYPEINDEX FIXED (16),
0080              *                      5 TRVARIABLE (3) POINTER,
0081              *                      5 TRRESERVED BIT (16));
0082              *       DECLARE TRACETYPES (4) BIT (16);
```

```
0084          *       PTABLE (PWP) = USRWP;
0085          *       NEWWP = USRWP;
0086          *       PCCURSOR = USRPC;
0087          *       NEWST = USRST;
0088          *       NEWPC = PCCURSOR;
0089          *       PTABLE (PPC) = USRPC;
0090          *       EXECUTEDINSTRUCTION (0) = MEMWORD;
0091          *       PCCURSOR = PCCURSOR + 2;
0092          *       TRCXINST:    /* REENTRY FOR X INST */
0093          *       STACKINDEX = 0;  /* CLEAR STACK */
0094          *       DO I FROM 0 TO 2;
0095          *          IF TFVARIABLE (I) = 1
0096          *             THEN CALL PUSHSTACK (TRVARIABLE (I));
0097          *          END;
0098          *       PRINTFLAG = AND (PRINTFLAG, CLEARFLAG);
0099          *       PTABLE (PINST) = EXECUTEDINSTRUCTION (0);
0100          *       EXECUTEDINSTRUCTION (1) = NOPINSTRUCTION;
0101          *       EXECUTEDINSTRUCTION (2) = NOPINSTRUCTION;
0102          *       I = 0;
0103          *       FOUND = 0;
0104          *       DO WHILE FOUND = 0;
0105          *          IF EXECUTEDINSTRUCTION (0) <
0106          *                GROUPLOWERBOUND (I)
0107          *             THEN I = I + 1
0108          *             ELSE FOUND = 1;
0109          *          END;
0110          *       FORMAT = INSTFORMAT (I);
0111          *       IF DECODEENTRY (I).LEGALITY = LEGOPF
0112          *             THEN DO;
0113          *          XINSTPOINTER = ADDR (EXECUTEDINSTRUCTION
0114          *                (1));
0115          *          DO CASE DECODEENTRY (I).ADDRESSGROUP;
0116          *             DO; END;          /* GROUP A */
0117          *             DO;               /* GROUP B */
0118          *                CALL PGADDR (AND (EXECUTEDINSTRUCTION (0);
0119          *                                  "F"),
0120          *                             XINSTPOINTER, PTABLE (PSOURC)
0121          *                             SRCREG, PCCURSOR);
0122          *                END;
0123          *             DO;               /* GROUP C */
0124          *                CALL PGADDR (EXECUTEDINSTRUCTION (0),
0125          *                             XINSTPOINTER, PTABLE (PSOURC)
0126          *                             SRCREG, PCCURSOR);
0127          *                END;
0128          *             DO;               /* GROUP D */
0129          *                CALL PGADDR (EXECUTEDINSTRUCTION (0),
0130          *                             XINSTPOINTER, PTABLE (PSOURC)
0131          *                             SRCREG, PCCURSOR);
0132          *                CALL PGADDR (AND ("F", RSHIFT (
0133          *                              EXECUTEDINSTRUCTION (0), 6))
0134          *                             XINSTPOINTER, PTABLE (PDEST),
0135          *                             DSTREG, PCCURSOR);
0136          *                END;
0137          *             DO;               /*GROUP E */
```

```
0138                *                  CALL PGADDR (EXECUTEDINSTRUCTION (0),
0139                *                               XINSTPOINTER, PTABLE (PSOURC)
0140                *                               SRCREG, PCCURSOR);
0141                *                  CALL PGADDR (RSHIFT (EXECUTEDINSTRUCTION(0
0142                *                                      6),
0143                *                               XINSTPOINTER, PTABLE (PDEST),
0144                *                               DSTREG, PCCURSOR);
0145                *                  IF DSTREG = SRCREG THEN CALL PUSH (DSTREG);
0146                *                  END;
0147                *               END;
0148                *            DO CASE I;     /* DO INSTRUCTION DEPENDANT
0149                *               CALL INSTEXEC; /* FORMAT I */
0150                *               DO;            /* MPY DIV */
0151                *                  CALL PUSH (PTABLE (PDE) + 2);
0152                *                  CALL INSTEXEC;
0153                *                  END;
0154                *               DO;             /* LDCR STCR */
0155                *                  PTABLE (PCRU) = USRREG (12);
0156                *                  PRINTFLAG = OR (PRINTFLAG, CRUFLAG),
0157                *                  CALL INSTEXEC;
0158                *                  END;
0159                *               DO;             /* XOP */
0160                *                  PTABLE (PXL) = LSHIFT (AND ("00C0",
0161                *                        EXECUTEDINSTRUCTION (0)), 6);
0162                *                  PRINTFLAG = OR (PRINTFLAG, XOPFLAG);
0163                *                  CALL INSTEXEC;
0164                *                  END;
0165                *               CALL INSTEXEC; /* COC  CZC  XOR */
0166                *               DO;             /* SBO  SBZ  TB */
0167                *                  PTABLE (PCRU) = USRREG (12) +
0168                *                     AND (EXECUTEDINSTRUCTION, "FF") *2;
0169                *                  PRINTFLAG = OR (PRINTFLAG, CRUFLAG);
0170                *                  CALL INSTEXEC;
0171                *                  END;
0172                *               DO;             /* JUMPS */
0173                *                  CALL XEQJUMPINST (EXECUTEDINSTRUCTION,
0174                *                        CONDITIONMET);
0175                *                  IF CONDITIONMET = 1 /* YES */ THEN DO;
0176                *                     DISTANCE = ARSHIFT (LSHIFT
0177                *                        (EXECUTEDINSTRUCTION (0), 8), 7)
0178                *                     IF DISTANCE .NE. 0 THEN DO;
0179                *                        PCCURSOR = PCCURSOR + DISTANCE;
0180                *                        PTABLE (PBT) = PCCURSOR;
0181                *                        PRINTFLAG = OR (PRINTFLAG, BRANCHF
0182                *                        END;
0183                *                     END;
0184                *                  NEWPC = PCCURSOR;
0185                *                  PTABLE (PST) = USRST;
0186                *                  END;
0187                *               DO; END;         /* ILLEGAL OPCODE */
0188                *               CALL INSTEXEC; /* SHIFTS */
0189                *               DO; END;         /* ILLEGAL OPCODE */
0190                *               CALL INSTEXEC; /* SWPB SETO ABS */
0191                *               DO;              /* BL */
0192                *                  USRREG (11) = PCCURSOR;
```

```
0193                *          NEWPC = PTABLE(PSOURC+PSE)
0194                *          PTABLE (PBT) = NEWPC;
0195                *          CALL AUTOINCTEST;
0196                *          PRINTFLAG = OR (PRINTFLAG, BRANCHFLAG);
0197                *          END;
0198                *       CALL INSTEXEC; /* CLR NEG INV INC/T DEC/T */
0199                *       DO;            /* X */
0200                *          PTR = PTABLE (PSOURC+PSE);
0201                *          EXECUTEDINSTRUCTION (0) = MEMWORD2;
0202                *          CALL AUTOINCTEST;
0203                *          CALL TRCPRINT (PRINTFLAG, TRCENTRYADDR,
0204                *                         PTABLE)
0205                *          GO TO TRCXINST;
0206                *          END;
0207                *       DO;            /* B */
0208                *          NEWPC = PTABLE (PSOURC+PSE);
0209                *          PTABLE (PBT) = NEWPC;
0210                *          PRINTFLAG = OR (PRINTFLAG, BRANCHFLAG)
0211                *          END;
0212                *       DO;            /* BLWP */
0213                *          PTR = PTABLE (PSOURC+PSE);
0214                *          NEWWP = MEMWORD2;
0215                *          PTR = PTR + 2;
0216                *          NEWPC = MEMWORD2;
0217                *          NEWUSRREG (13) = USRWP;
0218                *          NEWUSRREG (14) = PCCURSOR;
0219                *          NEWUSRREG (15) = USRST;
0220                *          PTABLE (PBT) = NEWPC;
0221                *          PRINTFLAG = OR (PRINTFLAG, BRANCHFLAG);
0222                *          END;
0223                *       DO; END;       /* LREX ILLEGAL */
0224                *       CALL INSTEXEC; /* CKON CKOF */
0225                *       DO;            /* RTWP */
0226                *          NEWWP = USRREG (13);
0227                *          NEWPC = USRREG (14);
0228                *          NEWST = USRREG (15);
0229                *          PTABLE (PBT) = NEWPC;
0230                *          PRINTFLAG = OR (PRINTFLAG, BRANCHFLAG);
0231                *          END;
0232                *       CALL INSTEXEC; /* RSET */
0233                *       DO; END;       /* IDLE */
0234                *       DO; END;       /* MAP FILE ILLEGAL */
0235                *       DO;            /* LWPI LIMI */
0236                *          EXECUTEDINSTRUCTION (1) = MEMWORD;
0237                *          PCCURSOR = PCCURSOR + 2;
0238                *          CALL INSTEXEC;
0239                *          END;
0240                *       CALL INSTEXEC; /* STST STWP */
0241                *       DO;            /* LI AI ANDI ORI CI */
0242                *          EXECUTABLEINSTRUCTION (1) = MEMWORD;
0243                *          PCCURSOR = PCCURSOR + 2;
0244                *          CALL INSTEXEC;
0245                *          END;
0246                *       DO; END;       /* ILLEGAL OPCODE */
0247                *       END;           /* END OF CASE I */
```

```
0250          *          ERRORCODE = 0;
0251          *          END;
0252          *       ELSE DO;        /* ILLEGAL OP CODE */
0253          *          CALL PRNTC ('IDLE OR ILOP DETECTED')
0254          *          ERRORCODE = 1;
0255          *          END;
0256          *       CALL PRTTRACE (PRINTFLAG, TRCENTRYADDR,
0257          *                      PTABLE)
0258          *       USRPC = NEWPC;
0259          *       USRWP = NEWWP;
0260          *       USRST = NEWST;
0261          *       IF TFINSTSTEP = 1 THEN DO;
0262          *          CALL GETCHR (CHAR);
0263          *          IF CHAR = ESC THEN ERRORCODE = 1;
0264          *          END;
0265          *       RETURN;
0266          *       END TRACER;
```

```
0269              *   PROCEDURE PGADDR (INST, XINSTPOINTER,
0270              *                     SUBPTABLE, REG, PCCURSOR,
0271              *                     REGNUMBER);
0272              *      DECLARE (XINSTPOINTER, PCCURSOR) POINTER;
0273              *      DECLARE WORDOFINST BIT (16)
0274              *         CONTROL (XINSTPOINTER);
0275              *      DECLARE WORDOFSIMINST BIT (16)
0276              *         CONTROL (PCCURSOR);
0277              *      DECLARE SUBPTABLE (4) BIT (16);
0278              *      DECLARE REG POINTER;
0279              *      DECLARE REGNUMBER BIT (16);
0280              *      DECLARE WREGADDR POINTER;
0281              *      DECLARE TSTD BIT (16);
0282              *      DECLARE MEMWORD BIT (16) CONTROL (WREGADDR)
0283              *      DECLARE MASK BIT (16);
0284              *      DECLARE PSRMSK LITERALLY "1000";
0285              *      DECLARE PEBAMK LITERALLY "E000";
0286              *      REGNUMBER = AND (INST, "3F");
0287              *      WREGADDR = AND (INST, "F") * 2 + USRWP
0288              *      REG = WREGADDR;
0289              *      SUBPTABLE (PSR) = MEMWORD;
0290              *      TSTD = RSHIFT (AND (INST, >30), 4);
0291              *      DO CASE TSTD;
0292              *         DO;              /* DIRECT */
0293              *            EFFECTIVEADDR = WREGADDR;
0294              *            MASK = 0;
0295              *            END;
0296              *         DO;              /* INDIRECT */
0297              *            EFFECTIVEADDR = MEMWORD;
0298              *            MASK = PSRMSK;
0299              *            END;
0300              *         DO;              /* INDEXED */
0301              *            EFFECTIVEADDR = MEMWORD;
0302              *            IF AND (REGNUMBER, "F") = 0
0303              *               THEN EFFECTIVEADDR = 0;
0304              *            EFFECTIVEADDR = EFFECTIVEADDR
0305              *               + WORDOFSIMINST;
0306              *            WORDOFINST = WORDOFSIMINST;
0307              *            XINSTPOINTER = XINSTPOINTER + 2;
0308              *            PCCURSOR = PCCURSOR + 2;
0309              *            MASK = PSRMSK;
0310              *            END;
0311              *         DO;              /* INDIRECT AUTO-INC */
0312              *            EFFECTIVEADDR = MEMWORD;
0313              *            MASK = PSRMSK;
0314              *            END;
0315              *         END;             /* END CASE TSTD */
0316              *      SUBPTABLE (PSE) = EFFECTIVEADDR;
0317              *      SUBPTABLE (PSB) = MEMWORD;
0318              *      MASK = RSHIFT (OR (MASK, PEBAMK),
0319              *           DISP (SUBPTABLE));
0320              *      PRINTFLAG = OR (PRINTFLAG, MASK);
0321              *      RETURN;
0322              *      END PGADDR;
```

```
0325              *   PROCEDURE TRCPRINT (PRINTFLAG,
0326              *         TRCENTRYADDR, PTABLE);
0327              *      DECLARE TRCENTRYADDR POINTER;
0328              *      DECLARE PTABLE (16) BIT (16);
0329              *      DECLARE (1 PRINTFLAG,
0330              *                  3 PRTBIT (16) BIT (1));
0331              *      DECLARE SOURCEVALUE BIT (16)
0332              *            CONTROL (SRCREG);
0333              *      DECLARE DESTVALUE BIT (16)
0334              *            CONTROL (DSTREG);
0335              *      DECLARE (I, LINELENGTH) FIXED;
0336              *      DECLARE MAXLENGTH LITERALLY (65);
0337              *      DECLARE ELEMENTLENGTH LITERALLY (9);
0338              *      DECLARE PTR POINTER;
0339              *      DECLARE MEMWORD BIT (16) CONTROL (PTR);
0340              *      DECLARE VARLN LITERALLY '11';
0341              *      PTABLE (PST) = NEWST;
0342              *      IF USRWP .NE. NEWWP THEN DO;
0343              *         PTABLE (PWP) = NEWWP;
0344              *         PRINTFLAG = OR (PRINTFLAG, PWPMSK)
0345              *         END;
0346              *      CALL UPDATEREG (PRINTFLAG, PTABLE,
0347              *            PSOURC, SRCREG);
0348              *      CALL UPDATEREG (PRINTFLAG, PTABLE,
0349              *            PDEST, DSTREG);
0350              *      PRINTFLAG = AND (PRINTFLAG,
0351              *            TRACETYPES (TRACETYPEINDEX));
0352              *      IF PRINTFLAG .NE. 0
0353              *         THEN PRTBIT (0) = 1; /* ASSURE PC PRINT */
0354              *      LINELENGTH = MAXLENGTH;
0355              *      DO I FROM 0 TO 15;
0356              *         IF PRTBIT (I) .NE. 0 THEN DO;
0357              *            LINELENGTH = LINELENGTH - ELEMENTLENGTH;
0358              *            IF LINELENGTH <= 0 THEN DO;
0359              *               CALL NEWLIN;
0360              *               END;
0361              *            CALL PRINT (PRINTTAGS (I), 2);
0362              *            IF I > PINST THEN CALL PRNTC (EQSIGN);
0363              *            CALL PRNTHX (PTABLE (I));
0364              *            END;
0365              *         END;
0366              *      DO WHILE STACKINDEX > 0;
0367              *         STACKINDEX = STACKINDEX - 1;
0368              *      LTR = ADDRSTACK (STACKINDEX);
0369              *      IF MEMWORD .NE. BEFORECONTENTS
0370              *            (STACKINDEX) THEN DO;
0371              *         LINELENGTH = LINELENGTH - VARLN;
0372              *         IF LINELENGTH < 0 THEN CALL NEWLIN;
0373              *         CALL PRNTHN (PTR);
0374              *         CALL PRNTC (EQSIGN);
0375              *         CALL PRNTHX (MEMWORD);
0376              *         END;
0377              *      RETURN;
0378              *      END PRTTRACE;
```

```
0381          *   PROCEDURE UPDATEREG (PRINTFLAG, PTABLE,
0382          *             BASE, REG);
0383          *      DECLARE REG POINTER;
0384          *      DECLARE MEMWORD BIT (16) CONTROL (REG);
0385          *      DECLARE BASE FIXED (16);
0386          *      DECLARE (1 PRINTFLAG,
0387          *                 3 PRTBIT (16) BIT (1));
0388          *      DECLARE PTABLE (16) BIT (16);
0389          *      DECLARE PTR POINTER;
0390          *      DECLARE MWORD BIT (16) CONTROL (PTR);
0391          *      IF PRTBIT (BASE+PSE) = 1 THEN DO;
0392          *         PTR = PTABLE (BASE+PSE);
0393          *         PTABLE (BASE+PSA) = MWORD;
0394          *         END;
0395          *      IF PRTBIT (BASE+PSR) = 1 THEN DO;
0396          *         PRTBIT (BASE+PSR) = 0;
0397          *         IF MEMWORD .NE. PTABLE (BASE+PSR) THEN DO;
0398          *            PRTBIT (BASE+PSR) = 1;
0399          *            PTABLE (BASE+PSR) = MEMWORD;
0400          *            END;
0401          *         END;
0402          *      RETURN;
0403          *      END UPDATEREG;
```

```
0406          *    PROCEDURE PUSHSTACK (ADDR);
0407          *       DECLARE ADDR POINTER;
0408          *       DECLARE STACKMAX LITERALLY '4';
0409          *       DECLARE MEMWORD BIT (16) CONTROL (ADDR);
0410          *       IF STACKINDEX >= STACKMAX THEN RETURN;
0411          *       ADDRSTACK (STACKINDEX) = ADDR;
0412          *       BEFORECONTENTS (STACKINDEX) = MEMWORD;
0413          *       STACKINDEX = STACKINDEX + 1;
0414          *       RETURN;
```

```
0417                *   PROCEDURE AUTOINCTEST;
0418                *      IF  AND (SRCRNM, "30") = "30"
0419                *         THEN SRCREGCONTENTS =
0420                *              SRCREGCONTENTS + 2;
0421                *      END;
```

```
0424                    *
0425                    *       WORKSPACE REGISTER DEFINITIONS
0426                    *
0427           0000     R0      EQU  0
0428           0001     R1      EQU  1
0429           0002     R2      EQU  2
0430           0003     R3      EQU  3
0431           0004     R4      EQU  4
0432           0005     R5      EQU  5
0433           0006     R6      EQU  6
0434           0007     R7      EQU  7
0435           0008     R8      EQU  8
0436           0009     R9      EQU  9
0437           000A     R10     EQU  10
0438           000B     R11     EQU  11
0439           000C     R12     EQU  12
0440           000D     R13     EQU  13
0441           000E     R14     EQU  14
0442           000F     R15     EQU  15
0443                    *
0444                            REF  USRPC,USRWP,USRST
0445                            REF  ISRFLG,ESC
0446           0100     LBYTE   EQU  >0100
0447           000D     CR      EQU  >0D
0448           000A     LF      EQU  >0A
0449                    *
0450                    *      OVERLAY PARAMETERS
0451                    *
0452                            REF  SR
0453                            REF STRACE
0454  0000       53             TEXT 'SR'
0455  0002  0000                 DATA SR
0456  0004       53             TEXT 'ST'
0457  0006  0000                 DATA STRACE
0458  0008  0000                 DATA 0
0459  000A  0000     NEWWP   DATA 0
0460  000C  0000     NEWST   DATA 0
0461  000E  0000     NEWPC   DATA 0
0462  0010  003B     R11ATO  DATA >3B
0463  0012           TEADDR  BSS  1*2
0464  0014           DSTRNM  BSS  1*2
0465  0016           SRCRNM  BSS  1*2
0466  0018  0C0D     ILMSG   DATA IM1-ILMSG*LBYTE+CR
0467  001A    0A             BYTE LF
0468  001B    49             TEXT 'IDLE/ILOP'
0469          0024' IM1      EQU  $
0470  0024  090D     NULINE  DATA NL1-NULINE*LBYTE+CR
0471  0026    0A             BYTE LF
0472  0027    20             TEXT '      '
0473          002D' NL1      EQU  $
0474                         EVEN
0475                         REF  ACL,RR
0476                         REF  PRINT,PRNTHX,PRNTC,PRNTHN
0477                         REF  TTTBL
```

```
0478                        REF   GETCHR
0479                        REF   EQSIGN
0480  002E  1000  PWPMSK DATA  >1000
0481        0020  SRCR   EQU   >0020
0482        0002  DSTR   EQU   >0002
0483        0100  SRCE   EQU   >0100
0484        0010  DSTE   EQU   >0010
0485        8000  PPCMSK EQU   >8000
0486              *
0487              *
0488              *
0489              *
0490              *
0491              *
0492              *
0493              *
0494  0030  E000  PRTFLG DATA  >E000
0495              *
0496  0032  1FFF  CPFLG  DATA  >1FFF
0497              *
0498              *
0499  0034  1000  NOP    NOP
0500              *
0501              *
0502              *
0503              *
0504              *
0505              *
0506              *
0507              *
0508  0036        SRCREG BSS   1*2
0509  0038        DSTREG BSS   1*2
0510              *
0511              *
0512  003A  0400  CRUFLG DATA  >0400
0513              *
0514  003C  0200  XOPFLG DATA  >0200
0515              *
0516              *
0517              *
0518  003E  0800  BRNFLG DATA  >0800
```

```
                        SOURCE EFF ADDR MASK
                        DEST EFF ADDR MASK

2-DECLARE PCCURSOR POINTER;
2-DECLARE MEMWORD2 BIT (16) CONTRO
2-DECLARE PTR POINTER;
2-DECLARE NEWUSRREG (16) BIT (16)
3-CONTROL (NEWWP);
2-DECLARE EXECUTEDINSTRUCTION (3) B
2-DECLARE MEMWORD BIT (16) CONTROL
2-DECLARE PRINTFLAG BIT (16) INITIA
      PC, INST, AND STATUS
2-DECLARE CLEARFLAG BIT (16) INITIA

2-DECLARE NOPINSTRUCTION BIT (16)
3-INITIAL ("1000");

2-DECLARE I FIXED (16);
2-DECLARE (FOUND, CONDITIONMET) FIX
2-DECLARE XINSTPOINTER POINTER;
2-DECLARE XEQINST BIT (16)
3-CONTROL (XINSTPOINTER);
2-DECLARE (SRCREG, DSTREG) POINTER;
2-DECLARE SRCREGCONTENTS BIT (16)
4-CONTROL (SRCREG);


2-DECLARE USRREG (16) BIT (16) CONT
2-DECLARE CRUFLAG BIT (16) INITIAL
      PRINT CRU FLAG
2-DECLARE XOPFLAG BIT (16) INITIAL
      PRINT XOP LEVEL FLAG
2-DECLARE DISTANCE BIT (16);
2-DECLARE BRANCHFLAG BIT (16)
3-INITIAL ("0800");
      PRINT BRANCH TARGET FLAG
```

```
0520                    *                       2-DECLARE PRVOPF LITERALLY '1'; /* P
0521          0040      PRVOPF EQU  >40                 PRIVILEGED OP CODE
0522                    *                       2-DECLARE NPROPF LITERALLY '0'; /* N
0523          0000      NPROPF EQU  0                   NON-PRIVILEGED CODE
0524                    *
0525                    *                       2-DECLARE ILLOPF LITERALLY '1'; /* I
0526          0080      ILLOPF EQU  >80                 ILLEGAL OP CODE
0527                    *                       2-DECLARE LEGOPF LITERALLY '0'; /* L
0528          0000      LEGOPF EQU  0                   LEGAL OP CODE
0529                    *
0530                    *                           ADDRESS GROUPS
0531                    *                       2-DECLARE GAGRPA LITERALLY '0'; /* A
0532          0000      GAGRPA EQU  0                     FORMAT II, VII, X
0533                    *                       2-DECLARE GAGRPB LITERALLY '1'; /*
0534          0001      GAGRPB EQU  1                     FORMAT V, VIII
0535                    *                       2-DECLARE GAGRPC LITERALLY '2'; /*
0536          0002      GAGRPC EQU  2                     FORMAT IV, VI, IX(MPY, DIV)
0537                    *                       2-DECLARE GAGRPD LITERALLY '3';
0538          0003      GAGRPD EQU  3                     FORMAT III, IX(XOP)
0539                    *                       2-DECLARE GAGRPE LITERALLY '4',
0540          0004      GAGRPE EQU  4                     FORMAT I
0541                    *                       2-DECLARE GROUPLOWERBOUND (26) BIT
0542                    *                       3-INITIAL ("4000", "3800", "3000", "
0543                    *                       3-          "2000", "1D00", "1000", "
0544                    *                       3-          "0800", "0780", "06C0", "
0545                    *                       3-          "04C0", "0480", "0440", "
0546                    *                       3-          "03E0", "03A0", "0380", "
0547                    *                       3-          "0340", "0320", "02E0", "
0548                    *                       3-          "0200", "0000");
0549          0040'     GLBTBL EQU  $                   GROUP LOWER BOUND TABLE
0550    0040  4000              DATA >4000,>3800,>3000,>2C00
        0042  3800
        0044  3000
        0046  2C00
0551    0048  2000              DATA >2000,>1D00,>1000,>0C00
        004A  1D00
        004C  1000
        004E  0C00
0552    0050  0800              DATA >0800,>0780,>06C0,>0680
        0052  0780
        0054  06C0
        0056  0680
0553    0058  04C0              DATA >04C0,>0480,>0440,>0400
        005A  0480
        005C  0440
        005E  0400
0554    0060  03E0              DATA >03E0,>03A0,>0380,>0360
        0062  03A0
        0064  0380
        0066  0360
0555    0068  0340              DATA >0340,>0320,>02E0,>02A0
        006A  0320
        006C  02E0
        006E  02A0
```

```
0556  0070  0200          DATA >0200,>0000
      0072  0000
0557                *
0558                *                          2-DECLARE (1 TRACEDECODETABLE,
0559                *                          3-          3 DECODEENTRY (26),
0560                *                          4-            5 LEGALITY   BIT (1),
0561                *                          4-            5 PRIVILEGEDOP BIT (1),
0562                *                          4-            5 DEFILLER BIT (2),
0563                *                          4-            5 ADDRESSGROUP BIT (4));
0564          0074' TDTBL   EQU  $                      TRACE DECODE TABLE
0565  0074      04          BYTE LEGOPF+NPROPF+GAGRPE    FORMAT I INSTRUCTIONS
0566  0075      03          BYTE LEGOPF+NPROPF+GAGRPD    DIV, MPY
0567  0076      02          BYTE LEGOPF+NPROPF+GAGRPC    STCR, LDCR
0568  0077      02          BYTE LEGOPF+NPROPF+GAGRPC    XOP GAGRPC
0569  0078      03          BYTE LEGOPF+NPROPF+GAGRPD    XOR, CZC, COC
0570  0079      00          BYTE LEGOPF+NPROPF+GAGRPA    TB, SBZ, SBO
0571  007A      00          BYTE LEGOPF+NPROPF+GAGRPA    JUMPS
0572  007B      80          BYTE ILLOPF+NPROPF+GAGRPA    *** ILLEGAL
0573  007C      01          BYTE LEGOPF+NPROPF+GAGRPB    SHIFTS
0574  007D      82          BYTE ILLOPF+NPROPF+GAGRPC    *** RESERVED FOR LDS LDD
0575  007E      02          BYTE LEGOPF+NPROPF+GAGRPC    ABS, SETO, SWPB
0576  007F      02          BYTE LEGOPF+NPROPF+GAGRPC    BL
0577  0080      02          BYTE LEGOPF+NPROPF+GAGRPC    INC, DEC, INV, NEG, CLR
0578  0081      02          BYTE LEGOPF+NPROPF+GAGRPC    X
0579  0082      02          BYTE LEGOPF+NPROPF+GAGRPC    B
0580  0083      02          BYTE LEGOPF+NPROPF+GAGRPC    BLWP
0581  0084      C0          BYTE ILLOPF+PRVOPF+GAGRPA    LREX
0582  0085      40          BYTE LEGOPF+PRVOPF+GAGRPA    CKON, CKOF
0583  0086      00          BYTE LEGOPF+NPROPF+GAGRPA    RTWP
0584  0087      40          BYTE LEGOPF+PRVOPF+GAGRPA    RSET
0585  0088      C0          BYTE ILLOPF+PRVOPF+GAGRPA    IDLE
0586  0089      C0          BYTE ILLOPF+PRVOPF+GAGRPA    *** MAP FILE
0587  008A      01          BYTE LEGOPF+NPROPF+GAGRPB    LIMI, LWPI
0588  008B      01          BYTE LEGOPF+NPROPF+GAGRPB    STST, STWP
0589  008C      01          BYTE LEGOPF+NPROPF+GAGRPB    IMMEDIATES
0590  008D      81          BYTE ILLOPF+NPROPF+GAGRPB    *** ILLEGAL
0591                *                          2-DECLARE INSTFORMAT (26) CHARACTER
0592                *                          3-INITIAL ('1', '9', '4', '9', '3',
0593                *                          3-          ' ', '5', ' ', '6', '6',
0594                *                          3-          '6', '6', '7', '7', '7',
0595                *                          3-          'A', '8', '8', '8', ' ');
0596          008E' INSFMT  EQU  $                   INSTRUCTION FORMAT NUMBERS
0597  008E      31          TEXT '1949322X5X6666667777A888X'
0598                *
0599                *                          2-DECLARE PTABLE (15) FIXED (16);
0600                        EVEN
0601  00A8          PTABLE  BSS  15*2                VALUES TO BE PRINTED
0602          0000  PPC     EQU  0*2                   PROGRAM COUNTER
0603          0002  PINST   EQU  1*2                   INSTRUCTION
0604          0004  PST     EQU  2*2                   STATUS
0605          0006  PWP     EQU  3*2                   WORKSPACE POINTER CHANGES
0606          0008  PBT     EQU  4*2                   BRANCH TARGET
0607          000A  PCRU    EQU  5*2                   CRU ADDRESS
0608          000C  PXL     EQU  6*2                   XOP LEVEL
0609          000E  PSOURC  EQU  7*2                   SOURCE
```

```
0610          0000  PSE     EQU  0*2                    EFFECTIVE ADDRESS
0611          0002  PSB     EQU  1*2                    (EA) BEFORE EXECUTION
0612          0004  PSA     EQU  2*2                    (EA) AFTER EXECUTION
0613          0006  PSR     EQU  3*2                    WORKSPACE REGISTER CHANGE
0614                *                                      FOR TS/TD .NE. 0
0615          0016  PDEST   EQU  11*2                 DESTINATION
0616          0000  PDE     EQU  PSE
0617          0002  PDB     EQU  PSB
0618          0004  PDA     EQU  PSA
0619          0006  PDR     EQU  PSR
0620                *                          2-DECLARE PRINTTAGS (15) CHARACTER
0621                *                          3-INITIAL (CRLF, 'X-', 'ST', 'WP', '
0622                *                          3-         'XL', 'SE', 'SB', 'SA', '
0623                *                          3-         'DB', 'DA', 'DR'));
0624                *                          2-DECLARE FORMAT   LITERALLY PRINTT
0625          00C6' PRTTAG  EQU  $                       PRINT TAGS
0626  00C6    0D0A          DATA CR*LBYTE+LF
0627          00C8' FORMAT  EQU  $
0628  00C8      58          TEXT 'X-'
0629  00CA      53          TEXT 'ST'
0630  00CC      57          TEXT 'WP'
0631  00CE      42          TEXT 'BT'
0632  00D0      43          TEXT 'CR'
0633  00D2      58          TEXT 'XL'
0634  00D4      53          TEXT 'SE'
0635  00D6      53          TEXT 'SB'
0636  00D8      53          TEXT 'SA'
0637  00DA      53          TEXT 'SR'
0638  00DC      44          TEXT 'DE'
0639  00DE      44          TEXT 'DB'
0640  00E0      44          TEXT 'DA'
0641  00E2      44          TEXT 'DR'
0642                *                          2-DECLARE STACKINDEX FIXED (16);
0643  00E4        STINX BSS  1*2                       STACK INDEX
0644          0008  STKMAX EQU  4*2                    STACK SIZE
0645                *                          2-DECLARE ADDRSTACK (4) POINTER;
0646  00E6        ADRSTK BSS  STKMAX                   ADDRESS STACK
0647                *                          2-DECLARE BEFORECONTENTS (4) BIT (1
0648  00EE        B4CNTS BSS  STKMAX                   BEFORE CONTENTS STACK
0649                *                          2-DECLARE (1 TRACEREGIONTABLES,
0650                *                          3-          3 TRCREGIONTBL
0651                *                          5-          CONTROL (TRCENTRYADDR),
0652                *                          4-          5 TRFLAGS,
0653                *                          5-          7 TFDEFINED BIT (1),
0654                *                          5-          7 TFVARIABLE (3) BIT (1)
0655                *                          5-          7 TFINSTSTEP BIT (1),
0656                *                          5-          7 TFFILLER BIT (3),
0657                *                          4-          5 TRFILLER BIT (8),
0658                *                          4-          5 TRLOWBOUND POINTER,
0659                *                          4-          5 TRHIGHBOUND POINTER,
0660                *                          4-          5 TRTYPEINDEX FIXED (16
0661                *                          4-          5 TRVARIABLE (3) POINTER
0662                *                          4-          5 TRRESERVED BIT (16));
0663                      REF  TRFLGS,TRLOW,TRHIGH,TRTYPE,TRV1
0664          0001  TFV1   EQU  1
```

```
0665                *                2-DECLARE TRACETYPES (4) BIT (16);
```

```
0668                        DEF  TRACER
0669        00F6' TRACER    EQU  $
0670  00F6  0420           BLWP @ACL                GET WKSP FOR TRACER
      00F8  0000
0671  00FA  C808           MOV  R8,@TEADDR            AND POINTER
      00FC  0012'
0672  00FE  0209           LI   R9,USRWP
      0100  0000
0673  0102  0208           LI   R8,NEWWP
      0104  000A'
0674                  *                              2-PTABLE (PWP) = USRWP;
0675  0106  C819           MOV  *R9,@PTABLE+PWP
      0108  00AE'
0676                  *                              2-NEWWP = USRWP;
0677  010A  CE39           MOV  *R9+,*R8+
0678                  *                              2-PCCURSOR = USRPC;
0679  010C  C179           MOV  *R9+,R5
0680                  *                              2-NEWST = USRST;
0681  010E  CE19           MOV  *R9,*R8+
0682                  *                              2-NEWPC = PCCURSOR;
0683  0110  C605           MOV  R5,*R8
0684                  *                              2-PTABLE (PPC) = USRPC;
0685  0112  C805           MOV  R5,@PTABLE+PPC
      0114  00A8'
0686                  *                              2-EXECUTEDINSTRUCTION (0) = MEMWOR
0687                  *                              2-PCCURSOR = PCCURSOR + 2;
0688  0116  C835           MOV  *R5+,@XINST           MOVE FIRST WORD OF INSTRUCTIO
      0118  0212'
0689                  *                              2-TRCXINST:    /* REENTRY FOR X IN
0690        011A' TRC010    EQU  $                       REENTRY POINT FOR 'X' INSTR.
0691                  *                              2-STACKINDEX = 0;  /* CLEAR STACK *
0692  011A  04E0           CLR  @STINX
      011C  00E4'
0693                  *                              2-DO I FROM 0 TO 2;
0694                  *                              3-IF TFVARIABLE (I) = 1
0695                  *                              4-THEN CALL PUSHSTACK (TRVARIABLE
0696                  *                              3-END;
0697  011E  C260           MOV  @TEADDR,R9
      0120  0012'
0698  0122  D229           MOVB @TRFLGS(R9),R8
      0124  0000
0699  0126  0A18           SLA  R8,TFV1
0700  0128  0248           ANDI R8,>E000              ISOLATE THE VARIABLE TRACE BI
      012A  E000
0701  012C  0229           AI   R9,TRV1
      012E  0000
0702        0130' TRC012    EQU  $
0703  0130  C208           MOV  R8,R8
0704  0132  1307           JEQ  TRC014                IF ALL BITS PROCESSED
0705  0134  0A18           SLA  R8,1
0706  0136  1703           JNC  TRC013                IF THIS VARIALE NOT PRESENT
0707  0138  C299           MOV  *R9,R10
0708  013A  06A0           BL   @PUSH
      013C  052E'
```

```
0709         013E' TRC013 EQU  $
0710  013E   05C9         INCT R9
0711  0140   10F7         JMP  TRC012
0712         0142' TRC014 EQU  $
0713               *                          2-PRINTFLAG = AND (PRINTFLAG, CLEA
0714  0142   4820         SZC  @CPFLG,@PRTFLG
      0144   0032'
      0146   0030'
0715  0148   0206         LI   R6,XINST
      014A   0212'
0716               *                          2-PTABLE (PINST) = EXECUTEDINSTRUCT
0717  014C   C296         MOV  *R6,R10
0718  014E   C836         MOV  *R6+,@PTABLE+PINST
      0150   00AA'
0719               *                          2-EXECUTEDINSTRUCTION (1) = NOPINS
0720               *                          2-EXECUTEDINSTRUCTION (2) = NOPINS
0721  0152   CDA0         MOV  @NOP,*R6+
      0154   0034'
0722  0156   C5A0         MOV  @NOP,*R6
      0158   0034'
0723               *                          2-I = 0;
0724  015A   04C6         CLR  R6
0725               *                          2-FOUND = 0;
0726               *                          2-DO WHILE FOUND = 0;
0727         015C' TRC020 EQU  $
0728               *                          3-IF EXECUTEDINSTRUCTION (0) <
0729               *                          5-GROUPLOWERBOUND (I)
0730               *                          4-THEN I = I + 1
0731               *                          4-ELSE FOUND = 1;
0732  015C   898A         C    R10,@GLBTBL(R6)
      015E   0040'
0733  0160   1402         JHE  TRC030             IF INST CATAGORY FOUND
0734  0162   05C6         INCT R6
0735  0164   10FB         JMP  TRC020
0736               *                          3-END;
0737         0166' TRC030 EQU  $
0738               *                          2-FORMAT = INSTFORMAT (I);
0739  0166   C046         MOV  R6,R1              SET INSTRUCTION FORMAT
0740  0168   0911         SRL  R1,1
0741  016A   D821         MOVB @INSFMT(R1),@FORMAT
      016C   008E'
      016E   00C8'
0742               *                          2-IF DECODEENTRY (I).LEGALITY = LEG
0743  0170   D1E1         MOVB @TDTBL(R1),R7
      0172   0074'
0744  0174   0A17         SLA  R7,1
0745  0176   1702         JNC  TRC035             IF LEGAL OP CODE
0746  0178   0460         B    @TRC210            IF ILLEGAL OP CODE
      017A   035E'
0747         017C' TRC035 EQU  $
0748               *                          5-THEN DO;
0749               *                          3-XINSTPOINTER = ADDR (EXECUTEDINS
0750               *                          6-(1));
0751               *                          3-DO CASE DECODEENTRY (I).ADDRESSG
0752  017C   0209         LI   R9,XINST+2
```

```
       017E  0214'
0753   0180  0A37          SLA   R7,3
0754   0182  09B7          SRL   R7,11
0755   0184  0467          B     @TRC040(R7)        CASE OF ADDRESS GROUP
       0186  0188'
0756         0188' TRC040  EQU   $
0757   0188  102D          JMP   TRC150             GRP A - NO ADDRESSES TO DECOD
0758   018A  1003          JMP   TRC100             GRP B - W
0759   018C  1004          JMP   TRC110             GRP C - GA(S)
0760   018E  1009          JMP   TRC120             GRP D - GA(S), D
0761   0190  100A          JMP   TRC130             GRP E - GA(S), GA(D)
0762                 *                          4-DO; END;        /* GROUP A */
0763                 *                          4-DO;             /* GROUP B */
0764                 *                          5-CALL PGADDR (AND (EXECUTEDINSTRUC
0765                 *                          5-                   "F"),
0766                 *                          5-                XINSTPOINTER, PTABLE
0767                 *                          5-                SRCREG, PCCURSOR);
0768                 *                          5-END;
0769         0192' TRC100  EQU   $
0770   0192  024A          ANDI  R10,>000F          MASK W
       0194  000F
0771                 *                          4-DO;             /* GROUP C */
0772                 *                          5-CALL PGADDR (EXECUTEDINSTRUCTION
0773                 *                          5-                XINSTPOINTER, PTABLE
0774                 *                          5-                SRCREG, PCCURSOR);
0775                 *                          5-END;
0776         0196' TRC110  EQU   $
0777   0196  06A0          BL    @PGADDR
       0198  03AA'
0778   019A  000E          DATA  PSOURC,SRCRNM,SRCREG
       019C  0016'
       019E  0036'
0779   01A0  1021          JMP   TRC150
0780                 *                          4-DO;             /* GROUP D */
0781                 *                          5-CALL PGADDR (EXECUTEDINSTRUCTION
0782                 *                          5-                XINSTPOINTER, PTABLE
0783                 *                          5-                SRCREG, PCCURSOR);
0784                 *                          5-CALL PGADDR (AND ("F", RSHIFT (
0785                 *                          5-                 EXECUTEDINSTRUCTION
0786                 *                          5-                XINSTPOINTER, PTABLE
0787                 *                          5-                DSTREG, PCCURSOR);
0788         01A2' TRC120  EQU   $
0789   01A2  024A          ANDI  R10,>03FF
       01A4  03FF
0790                 *                          5-END;
0791                 *                          4-DO;             /*GROUP E */
0792                 *                          5-CALL PGADDR (EXECUTEDINSTRUCTION
0793                 *                          5-                XINSTPOINTER, PTABLE
0794                 *                          5-                SRCREG, PCCURSOR);
0795         01A6' TRC130  EQU   $
0796   01A6  06A0          BL    @PGADDR
       01A8  03AA'
0797   01AA  000E          DATA  PSOURC,SRCRNM,SRCREG
       01AC  0016'
       01AE  0036'
```

```
0798                    *                       5-CALL PGADDR (RSHIFT (EXECUTEDINST
0799                    *                       5-                    6),
0800                    *                       5-                XINSTPOINTER, PTABLE
0801                    *                       5-                DSTREG, PCCURSOR);
0802  01B0  096A                SRL   R10,6
0803  01B2  06A0                BL    @PGADDR
      01B4  03AA'
0804  01B6  0016                DATA  PDEST,DSTRNM,DSTREG
      01B8  0014'
      01BA  0038'
0805                    *                       5-IF DSTREG = SRCREG THEN CALL PUSH
0806  01BC  8820                C     @DSTREG,@SRCREG   WHEN *RI+,*RI+
      01BE  0038'
      01C0  0036'
0807  01C2  1610                JNE   TRC135                RI+1 MODIFIED
0808  01C4  0201                LI    R1,>30
      01C6  0030
0809  01C8  4060                SZC   @SRCRNM,R1
      01CA  0016'
0810  01CC  160B                JNE   TRC135
0811  01CE  0201                LI    R1,PTABLE+PDEST+PDE
      01D0  00BE'
0812  01D2  C2A0                MOV   @PTABLE+PINST,R10
      01D4  00AA'
0813  01D6  0A4A                SLA   R10,4
0814  01D8  1801                JOC   TRC133
0815  01DA  0591                INC   *R1
0816        01DC' TRC133  EQU   $
0817  01DC  0591                INC   *R1
0818  01DE  C051                MOV   *R1,R1
0819  01E0  C811                MOV   *R1,@PTABLE+PDEST+PDB
      01E2  00C0'
0820        01E4' TRC135  EQU   $
0821                    *                       5-END;
0822                    *                       4-END;
0823                    *                       3-DO CASE I;      /* DO INSTRUCTION D
0824                    *                                   FIXUPS */
0825        01E4' TRC150  EQU   $
0826  01E4  04C1                CLR   R1
0827  01E6  0916                SRL   R6,1
0828  01E8  D066                MOVB  @CDISP(R6),R1
      01EA  0340'
0829  01EC  0971                SRL   R1,7
0830  01EE  0204                LI    R4,NEWPC
      01F0  000E'
0831  01F2  0206                LI    R6,NEWWP
      01F4  000A'
0832  01F6  0209                LI    R9,XINST
      01F8  0212'
0833  01FA  0461                B     @CATBL(R1)
      01FC  01FE'
0834                    *                       4-CALL INSTEXEC; /* FORMAT I */
0835        01FE' CATBL   EQU   $                       BRANCH BY CATAGORY OF INST.
0836                    ********************
0837        01FE' CATA    EQU   $
```

```
0838  01FE  C3E0          MOV   @USRST,R15        RESTORE USER CONTEXT
      0200  0000
0839  0202  C389          MOV   R9,R14
0840  0204  C00D          MOV   R13,R0            SAVE WKSP LINKAGE
0841  0206  C360          MOV   @USRWP,R13
      0208  0100'
0842  020A  02A1          STWP  R1                SET UP BLWP TO TRACER CONTEXT
0843  020C  C801          MOV   R1,@XBLWP
      020E  021C'
0844  0210  0380          RTWP
0845  0212        XINST   BSS   3*2               **** EXECUTED INSTRUCTION
0846  0218  0420          BLWP  @XBLWP            RETURN TO TRACER CONTEXT
      021A  021C'
0847  021C        XBLWP   BSS   1*2
0848  021E  0220'         DATA  CATA10
0849        0220' CATA10  EQU   $                 RETURN HERE AFTER EXECUTED
0850  0220  0300          LIMI  0                   INSTRUCTION
      0222  0000
0851  0224  C58D          MOV   R13,*R6           SET NEW WP
0852  0226  C505          MOV   R5,*R4
0853  0228  C80F          MOV   R15,@NEWST
      022A  000C'
0854  022C  C340          MOV   R0,R13            RESTORE WKSP LINKAGE
0855  022E  0460          B     @TRC200
      0230  035A'
0856                      ********************
0857                      *                             4-DO;          /* MPY DIV */
0858        0232' CATB    EQU   $
0859                      *                             5-CALL PUSH (PTABLE (PDE) + 2);
0860  0232  C2A0          MOV   @PTABLE+PDEST+PDE,R10
      0234  00BE'
0861  0236  05CA          INCT  R10               (DE) + 2 AFFECTED BY MRY, DIV
0862  0238  06A0          BL    @PUSH
      023A  052E'
0863                      *                             5-CALL INSTEXEC;
0864  023C  10E0          JMP   CATA
0865                      *                             5-END;
0866                      ********************
0867                      *                             4-DO;           /* LDCR STCR */
0868        023E' CATC    EQU   $
0869                      *                             5-PTABLE (PCRU) = USRREG (12);
0870  023E  C060          MOV   @USRWP,R1
      0240  0208'
0871  0242  C821          MOV   @R12*2(R1),@PTABLE+PCRU
      0244  0018
      0246  00B2'
0872                      *                             5-PRINTFLAG = OR (PRINTFLAG, CRUF
0873  0248  E820          SOC   @CRUFLG,@PRTFLG
      024A  003A'
      024C  0030'
0874                      *                             5-CALL INSTEXEC;
0875  024E  10D7          JMP   CATA
0876                      *                             5-END;
0877                      ********************
0878                      *                             4-DO;           /* XOP */
```

```
0879          02?0' CATD   EQU  $
0880                *                          5-PTABLE (PXL) = LSHIFT (AND ("00C0"
0881                *                          7-EXECUTEDINSTRUCTION (0)), 6);
0882  0250    C0 9          MOV  *R9,R0
0883  0252    0A?0          SLA  R0,6
0884  0254    09?0          SRL  R0,12
0885  0256    C8?0          MOV  R0,@PTABLE+PXL
      0258    00?4'
0886                *                          5-PRINTFLAG = OR (PRINTFLAG, XOPF
0887  025A    E8?0          SOC  @XOPFLG,@PRTFLG
      025C    00?C'
      025E    00?0'
0888                *                          5-CALL INSTEXEC;
0889  0260    10?E          JMP  CATA
0890                *                          5-END;
0891                **********
0892                *                          4-CALL INSTEXEC; /* COC  CZC  XOR *
0893          01?E' CATE   EQU  CATA
0894                **********
0895                *                          4-DO;              /* SBO  SBZ  TB */
0896          02?2' CATF   EQU  $
0897                *                          5-PTABLE (PCRU) = USRREG (12) +
0898                *                          6-AND (EXECUTEDINSTRUCTION, "FF") *
0899  0262    C0 9          MOV  *R9,R0
0900  0264    0A?0          SLA  R0,8
0901  0266    09?0          SRL  R0,7
0902  0268    C0?0          MOV  @USRWP,R1
      026A    0240'
0903  026C    A0?1          A    @R12*2(R1),R0
      026E    00?8
0904  0270    C8?0          MOV  R0,@PTABLE+PCRU
      0272    00?2'
0905                *                          5-PRINTFLAG = OR (PRINTFLAG, CRUF
0906  0274    E8?0          SOC  @CRUFLG,@PRTFLG
      0276    003A'
      0278    0030'
0907                *                          5-CALL INSTEXEC;
0908  027A    10?1          JMP  CATA
0909                *                          5-END;
0910                **********
0911                *                          4-DO;              /* JUMPS */
0912          02?C' CATG   EQU  $
0913                *                          5-CALL XEQJUMPINST (EXECUTEDINSTR
0914                *                          7-CONDITIONMET);
0915  027C    D8?9          MOVB *R9,@JINST
      027E    0298'
0916  0280    C3E0          MOV  @USRST,R15        GET USER STATUS
      0282    0200'
0917  0284    024F          ANDI R15,>FFF0           AT LEVEL ZERO
      0286    FFF0
0918  0288    020E          LI   R14,CATG10        RTWP TO CATG10
      028A    0293'
0919  028C    02A0          STWP R0
0920  028E    C800          MOV  R0,@JBLWP1
      0290    029E'
```

```
0921  0292  C800          MOV  R0,@JBLWP2
      0294  02A6'
0922  0296  0380          RTWP
0923        0298' CATG10  EQU  $
0924        0298' JINST   EQU  $
0925  0298  1004          JMP  CATG20            **** MODIFIED INSTRUCTION
0926  029A  0420          BLWP @JBLWP1           IF JUMP NOT TAKEN
      029C  029E'
0927        029E' JBLWP1  EQU  $
0928  029E  0000          DATA 0
0929  02A0  02BE'         DATA CATG40
0930        02A2' CATG20  EQU  $
0931  02A2  0420          BLWP @JBLWP2           IF JUMP TAKEN
      02A4  02A6'
0932        02A6' JBLWP2  EQU  $
0933  02A6  0000          DATA 0
0934  02A8  02AA'         DATA CATG30
0935              *                            5-IF CONDITIONMET = 1 /* YES */ THE
0936              *                            6-DISTANCE = ARSHIFT (LSHIFT
0937              *                            7-(EXECUTEDINSTRUCTION (0), 8), 7)
0938        02AA' CATG30  EQU  $
0939  02AA  C059          MOV  *R9,R1
0940  02AC  0A81          SLA  R1,8
0941  02AE  0871          SRA  R1,7
0942              *                            6-IF DISTANCE .NE. 0 THEN DO;
0943  02B0  1306          JEQ  CATG40                  IF NOP
0944              *                            7-PCCURSOR = PCCURSOR + DISTANCE;
0945  02B2  A141          A    R1,R5
0946              *                            7-PTABLE (PBT) = PCCURSOR;
0947  02B4  C805          MOV  R5,@PTABLE+PBT
      02B6  00B0'
0948              *                            7-PRINTFLAG = OR (PRINTFLAG, BRANC
0949  02B8  E820          SOC  @BRNFLG,@PRTFLG
      02BA  003E'
      02BC  0030'
0950              *                            7-END;
0951              *                            6-END;
0952              *                            5-NEWPC = PCCURSOR;
0953        02BE' CATG40  EQU  $
0954  02BE  C505          MOV  R5,*R4
0955              *                            5-PTABLE (PST) = USRST;
0956  02C0  C80F          MOV  R15,@PTABLE+PST
      02C2  00AC'
0957              *                            5-END;
0958  02C4  104A          JMP  TRC200
0959              *
0960              ********************
0961              *                            4-DO; END;          /* ILLEGAL OPCODE
0962        02C6' CATH    EQU  $
0963  02C6  10FF          JMP  $
0964              ********************
0965              *                            4-CALL INSTEXEC; /* SHIFTS */
0966        01FE' CATI    EQU  CATA
0967              ********************
0968              *                            4-DO; END;          /* ILLEGAL OPCODE
```

```
0969          02C6' CATJ   EQU  CATH
0970                ********************
0971                *                          4-CALL INSTEXEC; /* SWPB SETO ABS *
0972          01FE' CATK   EQU  CATA
0973                ********************
0974                *                          4-DO;              /* BL */
0975          02C8' CATL   EQU  $
0976                *                          5-USRREG (11) = PCCURSOR;
0977  02C8  C060         MOV  @USRWF,R1
      02CA  026A'
0978  02CC  C845         MOV  R5,@R11*2(R1)
      02CE  0016
0979                *                          5-NEWPC = PTABLE(PSOURC+PSE)
0980  02D0  C520         MOV  @PTABLE+PSOURC+PSE,*R4
      02D2  00B6'
0981  02D4  8820         C    @SRCRNM,@R11ATO
      02D6  0016'
      02D8  0010'
0982  02DA  1302         JEQ  CATL20
0983                *                          5-PTABLE (PBT) = NEWPC;
0984          02DC' CATL10 EQU  $
0985                *                          5-CALL AUTOINCTEST;
0986  02DC  06A0         BL   @ATOINC
      02DE  0546'
0987          02E0' CATL20 EQU  $
0988  02E0  C814         MOV  *R4,@PTABLE+PBT
      02E2  00B0'
0989                *                          5-PRINTFLAG = OR (PRINTFLAG, BRANC
0990  02E4  E820         SOC  @BRNFLG,@PRTFLG
      02E6  003E'
      02E8  0030'
0991                *                          5-END;
0992  02EA  1037         JMP  TRC200
0993                ********************
0994                *                          4-CALL INSTEXEC; /* CLR NEG INV INC
0995          01FE' CATM   EQU  CATA
0996                ********************
0997                *                          4-DO;              /* X */
0998                *                          5-PTR = PTABLE (PSOURC+PSE),
0999          02EC' CATN   EQU  $
1000  02EC  C060         MOV  @PTABLE+PSOURC+PSE,R1
      02EE  00B6'
1001                *                          5-EXECUTEDINSTRUCTION (0) = MEMWOR
1002  02F0  C651         MOV  *R1,*R9
1003                *                          5-CALL AUTOINCTEST,
1004  02F2  06A0         BL   @ATOINC
      02F4  0546'
1005                *                          5-CALL TRCPRINT (PRINTFLAG, TRCEN
1006                *                          5-                  PTABLE)
1007  02F6  06A0         BL   @PRTRC
      02F8  0418'
1008                *                          5-GO TO TRCXINST,
1009  02FA  0460         B    @TRC010
      02FC  011A'
1010                *                          5-END;
```

```
1011                ********************
1012                *                              4-DO;              /* B */
1013                *                              5-NEWPC = PTABLE (PSOURC+PSE);
1014          02FE' CAT0    EQU  $
1015  02FE  C520          MOV  @PTABLE+PSOURC+PSE,*R4
      0300  00B6'
1016                *                              5-PTABLE (PBT) = NEWPC;
1017                *                              5-PRINTFLAG = OR (PRINTFLAG, BRANC
1018                *                              5-END;
1019  0302  10EC          JMP  CATL10
1020                ********************
1021                *                              4-DO;              /* BLWP */
1022                *                              5-PTR = PTABLE (PSOURC+PSE);
1023          0304' CATP    EQU  $
1024  0304  C060          MOV  @PTABLE+PSOURC+PSE,R1
      0306  00B6'
1025                *                              5-NEWWP = MEMWORD2;
1026                *                              5-PTR = PTR + 2;
1027  0308  C5B1          MOV  *R1+,*R6
1028                *                              5-NEWPC = MEMWORD2;
1029  030A  C511          MOV  *R1,*R4
1030                *                              5-NEWUSRREG (13) = USRWP;
1031  030C  C056          MOV  *R6,R1
1032  030E  C860          MOV  @USRWP,@R13*2(R1)
      0310  02CA'
      0312  001A
1033                *                              5-NEWUSRREG (14) = PCCURSOR;
1034  0314  C845          MOV  R5,@R14*2(R1)
      0316  001C
1035                *                              5-NEWUSRREG (15) = USRST;
1036  0318  C860          MOV  @USRST,@R15*2(R1)
      031A  0282'
      031C  001E
1037                *                              5-PTABLE (PBT) = NEWPC;
1038                *                              5-PRINTFLAG = OR (PRINTFLAG, BRANC
1039                *                              5-END;
1040  031E  10DE          JMP  CATL10
1041                ********************
1042                *                              4-DO; END;          /* LREX ILLEGAL */
1043          02C6' CATQ    EQU  CATH
1044                ********************
1045                *                              4-CALL INSTEXEC; /* CKON CKOF */
1046          01FE' CATR    EQU  CATA
1047                ********************
1048                *                              4-DO;              /* RTWP */
1049          0320' CATS    EQU  $
1050                *                              5-NEWWP = USRREG (13);
1051  0320  C060          MOV  @USRWP,R1
      0322  0310'
1052  0324  C5A1          MOV  @R13*2(R1),*R6
      0326  001A
1053                *                              5-NEWPC = USRREG (14);
1054  0328  C521          MOV  @R14*2(R1),*R4
      032A  001C
1055                *                              5-NEWST = USRREG (15);
```

```
1056  032C  C821          MOV   @R15*2(R1),@NEWST
      032E  001E
      0330  000C'
1057                *                          5-PTABLE (PBT) = NEWPC;
1058                *                          5-PRINTFLAG = OR (PRINTFLAG, BRANC
1059                *                          5-END;
1060  0332  04E0          CLR   @SRCREG
      0334  0036'
1061  0336  10D2          JMP   CATL10
1062                ********************
1063                *                          4-CALL INSTEXEC; /* RSET */
1064        01FE' CATT    EQU   CATA
1065                ********************
1066                *                          4-DO; END;          /* IDLE */
1067        02C6' CATU    EQU   CATH
1068                ********************
1069                *                          4-DO; END;          /* MAP FILE ILLEGA
1070        02C6' CATV    EQU   CATH
1071                ********************
1072        0338' CATW    EQU   $
1073                *                          4-DO;              /* LWPI LIMI */
1074                *                          5-EXECUTEDINSTRUCTION (1) = MEMWOR
1075                *                          5-PCCURSOR = PCCURSOR + 2;
1076  0338  C835          MOV   *R5+,@XINST+2      MOVE IMMEDIATE VALUE
      033A  0214'
1077                *                          5-CALL INSTEXEC;
1078  033C  0460          B     @CATA
      033E  01FE'
1079                *                          5-END;
1080                ********************
1081                *                          4-CALL INSTEXEC; /* STST STWP */
1082        01FE' CATX    EQU   CATA
1083                ********************
1084                *                          4-DO;              /* LI AI ANDI ORI
1085                *                          5-EXECUTABLEINSTRUCTION (1) = MEMW
1086                *                          5-PCCURSOR = PCCURSOR + 2;
1087        0338' CATY    EQU   CATW
1088                *                          5-CALL INSTEXEC;
1089                *                          5-END;
1090                *                          4-DO; END;          /* ILLEGAL OPCODE
1091        02C6' CATZ    EQU   CATH
1092                *                          4-END;              /* END OF CASE I *
```

```
1094          0340' CDISP  EQU  $                 DISPLACEMENTS TO CATAGORIES
1095  0340      00         BYTE CATA-CATBL/2        FORMAT I INSTRUCTIONS
1096  0341      1A         BYTE CATB-CATBL/2        MPY DIV
1097  0342      20         BYTE CATC-CATBL/2        LDCR STCR
1098  0343      29         BYTE CATD-CATBL/2        XOP
1099  0344      00         BYTE CATE-CATBL/2        COC CZC XOR
1100  0345      32         BYTE CATF-CATBL/2        SBO SBZ TB
1101  0346      3F         BYTE CATG-CATBL/2        JUMPS
1102  0347      64         BYTE CATH-CATBL/2        ILLEGAL OPCODES
1103  0348      00         BYTE CATI-CATBL/2        SHIFTS
1104  0349      64         BYTE CATJ-CATBL/2        ILLEGAL OPCODES
1105  034A      00         BYTE CATK-CATBL/2        SWPB SETO ABS
1106  034B      65         BYTE CATL-CATBL/2        BL
1107  034C      00         BYTE CATM-CATBL/2        CLR NEG INV INC(T) DEC(T)
1108  034D      77         BYTE CATN-CATBL/2        X
1109  034E      30         BYTE CATO-CATBL/2        B
1110  034F      33         BYTE CATP-CATBL/2        BLWP
1111  0350      64         BYTE CATQ-CATBL/2        LREX
1112  0351      00         BYTE CATR-CATBL/2        CKON CKOF
1113  0352      91         BYTE CATS-CATBL/2        RTWP
1114  0353      00         BYTE CATT-CATBL/2        RSET
1115  0354      64         BYTE CATU-CATBL/2        IDLE
1116  0355      64         BYTE CATV-CATBL/2        *** MAP FILE
1117  0356      9D         BYTE CATW-CATBL/2        LIMI LWPI
1118  0357      00         BYTE CATX-CATBL/2        STST STWP
1119  0358      9D         BYTE CATY-CATBL/2        IMMEDIATES EXCEPT LIMI LWPI
1120  0359      64         BYTE CATZ-CATBL/2        *** ILLEGAL
```

```
1122                    TRC200
1123                    *                               3-ERRORCODE = 0;
1124  035A  04C4               CLR   R4
1125                    *                               3-END;
1126  035C  1006               JMP   TRC220
1127                    *                               2-ELSE DO;       /* ILLEGAL OP CODE
1128        035E'       TRC210 EQU   $                          ILLEGAL OP CODE DETECTED
1129                    *                               3-CALL PRNTC ('IDLE OR ILOP DETECTED
1130  035E  020A               LI    R10,ILMSG
      0360  0018'
1131  0362  06A0               BL    @PRNTC
      0364  0000
1132  0366  1000               NOP
1133                    *                               3-ERRORCODE = 1;
1134  0368  0704               SETO  R4
1135                    *                               3-END;
1136        036A'       TRC220 EQU   $
1137                    *                               2-CALL PRTTRACE (PRINTFLAG, TRCEN
1138                    *                               2-                  PTABLE)
1139  036A  06A0               BL    @PRTRC
      036C  0418'
1140  036E  0704               SETO  R4                        **** ERROR RETURN
1141                    *                               2-USRPC = NEWPC;
1142  0370  C820               MOV   @NEWPC,@USRPC
      0372  000E'
      0374  0000
1143                    *                               2-USRWP = NEWWP;
1144  0376  C820               MOV   @NEWWP,@USRWP
      0378  000A'
      037A  0322'
1145                    *                               2-USRST = NEWST;
1146  037C  C820               MOV   @NEWST,@USRST
      037E  000C'
      0380  031A'
1147                    *                               2-IF TFINSTSTEP = 1 THEN DO;
1148  0382  C2A0               MOV   @TEADDR,R10
      0384  0012'
1149  0386  C2AA               MOV   @TRFLGS(R10),R10
      0388  0124'
1150  038A  22A0               COC   @ISRFLG,R10
      038C  0000
1151  038E  1606               JNE   TRC230
1152                    *                               3-CALL GETCHR (CHAR);
1153  0390  06A0               BL    @GETCHR
      0392  0000
1154                    *                               3-IF CHAR = ESC THEN ERRORCODE = 1;
1155  0394  980A               CB    R10,@ESC
      0396  0000
1156  0398  1601               JNE   TRC230
1157  039A  0704               SETO  R4
1158                    *                               3-END;
1159        039C'       TRC230 EQU   $
1160                    *                               2-RETURN;
1161  039C  C104               MOV   R4,R4
```

```
 1162  039E  1602          JNE  TRC250              IF ERROR CODE IND. ERROR
 1163  03A0  05ED          INCT @R11*2(R13)         NORMAL EXIT
       03A2  0016
 1164        03A4' TRC250 EQU  $
 1165  03A4  0420          BLWP @RR                 BACK TO PREVIOUS WKSP
       03A6  0000
 1166  03A8  045B          B    *R11
 1167              *                        2-END TRACER;
```

```
1170                    * TITLE:     PGADDR
1171                    *            POST A GENERAL ADDRESS
1172                    * REVISION:
1173                    *            ORIGINAL
1174                    * ABSTRACT:
1175                    *            THE LEFTMOST SIX BITS OF THE INSTRUCTION ARE
1176                    *            INTERPRETTED AS A GENERAL ADDRESS.  THE
1177                    *            ADDRESS OF THE WORKSPACE REGISTER, THE CONTENTS
1178                    *            OF THE WORKSPACE REGISTER. THE EFFECTIVE
1179                    *            ADDRESS AND THE CONTENTS OF THE EFFECTIVE
1180                    *            ADDRESS ARE POSTED.
1181                    * CALLING SEQUENCE:
1182                    *            <R10> ::= TD/TS, D/S RIGHT JUSTIFIED
1183                    *            <R9>  ::= XEQ INSTRUCTION POINTER
1184                    *            <R5>  ::= PC CURSOR
1185                    *            BL    @PGADDR
1186                    *            DATA <PTABLE DISPLACEMENT>
1187                    *            DATA <PTR TO WORD TO GET TS/TD S/D VALUES>
1188                    *            DATA <PTR TO WORD TO GET WKSP REG ADDRESS>
1189                    *                        1-PROCEDURE PGADDR (INST, XINSTPOIN
1190                    *                        1-                  SUBPTABLE, REG.
1191                    *                        1-                  REGNUMBER);
1192                    *                        2-DECLARE (XINSTPOINTER, PCCURSOR)
1193                    *                        2-DECLARE WORDOFINST BIT (16)
1194                    *                        3-CONTROL (XINSTPOINTER);
1195                    *                        2-DECLARE WORDOFSIMINST BIT (16)
1196                    *                        3-CONTROL (PCCURSOR);
1197                    *                        2-DECLARE SUBPTABLE (4) BIT (16);
1198                    *                        2-DECLARE REG POINTER;
1199                    *                        2-DECLARE REGNUMBER BIT (16);
1200                    *                        2-DECLARE WREGADDR POINTER;
1201                    *                        2-DECLARE TSTD BIT (16);
1202                    *                        2-DECLARE MEMWORD BIT (16) CONTROL
1203                    *                        2-DECLARE MASK BIT (16);
1204                    *                        2-DECLARE PSRMSK LITERALLY "1000";
1205          1000      PSRMSK EQU   >1000
1206                    *                        2-DECLARE PEBAMK LITERALLY "E000";
1207          E000      PEBAMK EQU   >E000
1208          03AA'     PGADDR EQU   $
1209    03AA  C03B              MOV   *R11+,R0          GET DISPLACEMENT IN PTABLE
1210                    *                                 = SHIFT COUNT * 2
1211    03AC  020E              LI    R14,PTABLE
        03AE  00A8'
1212    03B0  A380              A     R0,R14            ADDR OF SUB PTABLE
1213                    *
1214    03B2  0810              SRA   R0,1              ADJUST SHIFT COUNT
1215                    *                       2-REGNUMBER = AND (INST, "3F");
1216    03B4  C04A              MOV   R10,R1
1217    03B6  0241              ANDI  R1,>3F
        03B8  003F
1218    03BA  C33B              MOV   *R11+,R12
1219    03BC  C701              MOV   R1,*R12           SET TS/TD S/D
1220                    *                       2-WREGADDR = AND (INST, "F") * 2 +
1221    03BE  0241              ANDI  R1,>F
```

```
       03C0  000F
1222   03C2  C3E0          MOV   @USRWP,R15
       03C4  037A'
1223   03C6  A3C1          A     R1,R15
1224   03C8  A3C1          A     R1,R15              ADDRESS OF WKSP REGISTER
1225                 *                          2-REG = WREGADDR;
1226   03CA  C33B          MOV   *R11+,R12
1227   03CC  C70F          MOV   R15,*R12            SET WKSP REGISTER ADDRESS
1228                 *                          2-SUBPTABLE (PSR) = MEMWORD;
1229   03CE  CB9F          MOV   *R15,@PSR(R14)
       03D0  0006
1230                 *                          2-TSTD = RSHIFT (AND (INST, >30), 4)
1231   03D2  C30A          MOV   R10,R12
1232   03D4  024C          ANDI  R12,>30
       03D6  0030
1233   03D8  093C          SRL   R12,3
1234                 *                          2-DO CASE TSTD;
1235   03DA  046C          B     @PGA010(R12)
       03DC  03DE'
1236         03DE' PGA010  EQU   $
1237   03DE  1003          JMP   PGA100              DIRECT
1238   03E0  1004          JMP   PGA200              INDIRECT
1239   03E2  1007          JMP   PGA300              INDEXED
1240   03E4  1002          JMP   PGA400              INDIRECT, AUTO-INCREMENT
1241                 *                          3-DO;                /* DIRECT */
1242                 *                          4-EFFECTIVEADDR = WREGADDR;
1243         03E6' PGA100  EQU   $
1244                 *                          4-MASK = 0;
1245   03E6  04CC          CLR   R12
1246                 *                          4-END;
1247   03E8  100D          JMP   PGA500
1248                 *                          3-DO;                /* INDIRECT */
1249         03EA' PGA200  EQU   $
1250                 *                          4-EFFECTIVEADDR = MEMWORD;
1251   03EA  C3DF          MOV   *R15,R15
1252                 *                          4-MASK = PSRMSK;
1253   03EC  020C          LI    R12,PSRMSK
       03EE  1000
1254   03F0  1003          JMP   PGA500
1255                 *                          4-END;
1256                 *                          3-DO;                /* INDEXED */
1257         03F2' PGA300  EQU   $                   INDEXED
1258                 *                          4-EFFECTIVEADDR = MEMWORD;
1259   03F2  C3DF          MOV   *R15,R15
1260                 *                          4-IF AND (REGNUMBER, "F") = 0
1261   03F4  C041          MOV   R1,R1
1262   03F6  1601          JNE   PGA310
1263                 *                          5-THEN EFFECTIVEADDR = 0;
1264   03F8  04CF          CLR   R15
1265                 *                          4-EFFECTIVEADDR = EFFECTIVEADDR
1266                 *                          5-+ WORDOFSIMINST;
1267         03FA' PGA310  EQU   $
1268   03FA  A3D5          A     *R5,R15
1269                 *                          4-WORDOFINST = WORDOFSIMINST;
1270                 *                          4-XINSTPOINTER = XINSTPOINTER + 2;
```

```
1271                    *                               4-PCCURSOR = PCCURSOR + 2;
1272  03FC  CE75               MOV   *R5+,*R9+               MOVE EXTENDED PART OF INST
1273                    *                               4-MASK = PSRMSK;
1274  03FE  020C               LI    R12,PSRMSK
      0400  1000
1275                    *                               4-END;
1276  0402  1000               JMP   PGA500
1277                    *                               3-DO;                 /* INDIRECT AUTO-IN
1278                    *                               4-EFFECTIVEADDR = MEMWORD;
1279                    *                               4-MASK = PSRMSK;
1280                    *                               4-END;
1281        03EA' PGA400 EQU   PGA200                    INDIRECT AUTO INCREMENT
1282                    *                               3-END;                /* END CASE TSTD */
1283                    *                               2-SUBPTABLE (PSE) = EFFECTIVEADDR;
1284        0404' PGA500 EQU   $
1285  0404  CB8F               MOV   R15,@PSE(R14)
      0406  0000
1286                    *                               2-SUBPTABLE (PSB) = MEMWORD;
1287  0408  CB9F               MOV   *R15,@PSB(R14)
      040A  0002
1288                    *                               2-MASK = RSHIFT (OR (MASK, PEBAMK),
1289                    *                               4-DISP (SUBPTABLE));
1290  040C  026C               ORI   R12,PEBAMK
      040E  E000
1291  0410  090C               SRL   R12,0
1292                    *                               2-PRINTFLAG = OR (PRINTFLAG, MASK)
1293  0412  E80C               SOC   R12,@PRTFLG
      0414  0030'
1294                    *                               2-RETURN;
1295  0416  045B               B     *R11
1296                    *                               2-END PGADDR;
```

```
1299                    * TITLE:     PRTRC
1300                    *            TRACE PRINT SUBROUTINE
1301                    * REVISION:
1302                    *            ORIGINAL
1303                    * ABSTRACT:
1304                    *            THE VALUES OF PRINT FLAGS AND PTABLE ARE USED
1305                    *            TO PRINT A SEMI-FREE FORMAT TRACE OUTPUT.
1306                    *
1307                    * CALLING SEQUENCE:
1308                    *            BL   @PRTRC
1309                    *            JMP  <ERROR/ESC ACTION>
1310                    *            R0, R1, R2, R6, R7, R8, R9, R10, R12, R14, R15
1311                    *            R3
1312                    *            ARE DESTROYED
1313                    *                         1-PROCEDURE TRCPRINT (PRINTFLAG,
1314                    *                         3-TRCENTRYADDR, PTABLE);
1315                    *                         2-DECLARE TRCENTRYADDR POINTER;
1316                    *                         2-DECLARE PTABLE (16) BIT (16);
1317                    *                         2-DECLARE (1 PRINTFLAG,
1318                    *                         3-          3 PRTBIT (16) BIT (1));
1319                    *                         2-DECLARE SOURCEVALUE BIT (16)
1320                    *                         4-CONTROL (SRCREG);
1321                    *                         2-DECLARE DESTVALUE BIT (16)
1322                    *                         4-CONTROL (DSTREG);
1323                    *                         2-DECLARE (I, LINELENGTH) FIXED;
1324                    *                         2-DECLARE MAXLENGTH LITERALLY (65);
1325                    *                         2-DECLARE ELEMENTLENGTH LITERALLY (
1326          0041      MAXLN  EQU  65
1327          0009      ELEMLN EQU  9
1328                    *                         2-DECLARE PTR POINTER;
1329                    *                         2-DECLARE MEMWORD BIT (16) CONTROL
1330                    *                         2-DECLARE VARLN LITERALLY '11';
1331          000B      VARLN  EQU  11
1332          0418'     PRTRC  EQU  $
1333  0418    C08B            MOV  R11,R2              SAVE RETURN
1334                    *                         2-PTABLE (PST) = NEWST;
1335  041A    C820            MOV  @NEWST,@PTABLE+PST
      041C    000C'
      041E    00AC'
1336                    *                         2-IF USRWP .NE. NEWWP THEN DO;
1337  0420    8820            C    @USRWP,@NEWWP
      0422    03C4'
      0424    000A'
1338  0426    1306            JEQ  PT010
1339                    *                         3-PTABLE (PWP) = NEWWP;
1340  0428    C820            MOV  @NEWWP,@PTABLE+PWP
      042A    000A'
      042C    00AE'
1341                    *                         3-PRINTFLAG = OR (PRINTFLAG, PWPMS
1342  042E    E820            SOC  @PWPMSK,@PRTFLG
      0430    002E'
      0432    0030'
1343                    *                         3-END;
1344          0434'     PT010  EQU  $
```

```
1345                    *                       2-CALL UPDATEREG (PRINTFLAG, PTABL
1346                    *                       4-PSOURC, SRCREG);
1347  0434  C2A0        PT030   MOV   @PRTFLG,R10       UPDATE SOURCE REGS
      0436  0030'
1348  0438  0640              BL    @UDREG
      043A  0536'
1349  043C  0036'             DATA PTABLE+PSOURC,SRCE,SRCR,SRCREG
      043E  0100
      0440  0020
      0442  0036'
1350                    *                       2-CALL UPDATEREG (PRINTFLAG, PTABL
1351                    *                       4-PDEST, DSTREG);
1352  0444  0640              BL    @UDREG            UPDATE DESTINATION REGS
      0446  0506'
1353  0448  00BE'             DATA PTABLE+PDEST,DSTE,DSTR,DSTREG
      044A  0010
      044C  0002
      044E  0038'
1354  0450  C0CA              MOV   R10,R1            SAVE PRINT FLAG
1355        0452' PT030   EQU   $
1356                    *                       2-PRINTFLAG = AND (PRINTFLAG,
1357                    *                       4-TRACETYPES (TRACETYPEINDEX));
1358  0452  C2F0              MOV   @TEADDR,R10
      0454  0012'
1359  0456  C2FA              MOV   @TRTYPE(R10),R10  TRACE TYPE INDEX
      0458  0000
1360  045A  0A1A              SLA   R10,1
1361  045C  C2FA              MOV   @TTTBL(R10),R10   TRACE TYPE
      045E  0000
1362  0460  054A              INV   R10
1363  0462  404A              SZC   R10,R1
1364                    *                       2-IF PRINTFLAG .NE. 0
1365  0464  04C0              CLR   R0
1366  0466  1326              JEQ   PT090             NOTHING TO PRINT
1367                    *                       3-THEN PRTBIT (0) = 1; /* ASSURE PC
1368        0468' PT035   EQU   $
1369  0468  0261              ORI   R1,PPCMSK         ASSURE PC PRINTED
      046A  8000
1370                    *                       2-LINELENGTH = MAXLENGTH;
1371  046C  0206              LI    R6,MAXLN
      046E  0041
1372                    *                       2-DO I FROM 0 TO 15;
1373  0470  04C8              CLR   R8                LOOP COUNT
1374  0472  C1C1              MOV   R1,R7
1375                    *                       3-IF PRTBIT (I) .NE. 0 THEN DO;
1376        0474' PT040   EQU   $
1377  0474  0A17              SLA   R7,1
1378  0476  171A              JNC   PT080
1379                    *                       4-LINELENGTH = LINELENGTH - ELEMEN
1380  0478  0226              AI    R6,-ELEMLN
      047A  FFF7
1381                    *                       4-IF LINELENGTH <= 0 THEN DO;
1382  047C  1502              JGT   PT050
1383                    *                       5-CALL NEWLIN;
1384  047E  06A0              BL    @NEWLIN
```

```
      0480  04F4'
1385                *                               5-END;
1386        0482' PT050  EQU  $
1387                *                               4-CALL PRINT (PRINTTAGS (I), 2);
1388  0482  020A         LI   R10,PRTTAG
      0484  00C6'
1389  0486  A288         A    R8,R10
1390  0488  0209         LI   R9,2
      048A  0002
1391  048C  06A0         BL   @PRINT
      048E  0000
1392  0490  1030         JMP  PT800                  **** ERROR/ESC RETURN
1393                *                               4-IF I > PINST THEN CALL PRNTC (EQSI
1394  0492  0288         CI   R8,PINST
      0494  0002
1395  0496  1205         JLE  PT070
1396  0498  020A         LI   R10,EQSIGN
      049A  0000
1397  049C  06A0         BL   @PRNTC
      049E  0364'
1398  04A0  1023         JMP  PT800                  **** ERROR/ESC RETURN
1399        04A2' PT070  EQU  $
1400                *                               4-CALL PRNTHX (PTABLE (I));
1401  04A2  C2A8         MOV  @PTABLE(R8),R10
      04A4  00A8'
1402  04A6  06A0         BL   @PRNTHX
      04A8  0000
1403  04AA  1023         JMP  PT800                  **** ERROR/ESC RETURN
1404                *                               4-END;
1405                *                               3-END;
1406        04AC' PT080  EQU  $
1407  04AC  05C8         INCT R8
1408  04AE  C1C7         MOV  R7,R7
1409  04B0  16E1         JNE  PT040
1410  04B2  0580         INC  R0
1411                *                               2-DO WHILE STACKINDEX > 0;
1412                *                               3-STACKINDEX = STACKINDEX - 1;
1413        04B4' PT090  EQU  $
1414  04B4  C1E0         MOV  @STINX,R7
      04B6  00E4'
1415        04B8' PT100  EQU  $
1416  04B8  0647         DECT R7
1417  04BA  111A         JLT  PT190
1418                *                               2-LTR = ADDRSTACK (STACKINDEX);
1419  04BC  C227         MOV  @ADRSTK(R7),R8
      04BE  00E6'
1420                *                               2-IF MEMWORD .NE. BEFORECONTENTS
1421                *                               4-(STACKINDEX) THEN DO;
1422  04C0  8627         C    @B4CNTS(R7),*R8
      04C2  00EE'
1423  04C4  13F9         JEQ  PT100
1424  04C6  C000         MOV  R0,R0
1425  04C8  13CF         JEQ  PT035
1426                *                               3-LINELENGTH = LINELENGTH - VARLN;
1427  04CA  0226         AI   R6,-VARLN
```

```
      04CC  FFF5
1428              *                          3-IF LINELENGTH < 0 THEN CALL NEWLI
1429  04CE  1502           JGT   PT110
1430  04D0  06A0           BL    @NEWLIN
      04D2  04F4'
1431        04D4' PT110     EQU   $
1432              *                          3-CALL PRNTHN (PTR);
1433  04D4  C288           MOV   R8,R10
1434  04D6  06A0           BL    @PRNTHN
      04D8  0000
1435  04DA  100F           JMP   PT800              **** ERROR/ESC RETURN
1436              *                          3-CALL PRNTC (EQSIGN);
1437  04DC  020A           LI    R10,EQSIGN
      04DE  049A'
1438  04E0  06A0           BL    @PRNTC
      04E2  049E'
1439  04E4  1006           JMP   PT800              **** ERROR/ESC RETURN
1440              *                          3-CALL PRNTHX (MEMWORD);
1441  04E6  C298           MOV   *R8,R10
1442  04E8  06A0           BL    @PRNTHX
      04EA  04A2'
1443  04EC  1002           JMP   PT800              **** ERROR/ESC RETURN
1444              *                          3-END;
1445  04EE  10E4           JMP   PT100
1446        04F0' PT190     EQU   $
1447              *                          2-RETURN;
1448        04F0' PT700     EQU   $
1449  04F0  05C2           INCT  R2
1450        04F2' PT800     EQU   $
1451  04F2  0452           B     *R2
1452              *
1453              *
1454              *                                PRINT A NEW LINE
1455        04F4' NEWLIN    EQU   $
1456  04F4  C0CB           MOV   R11,R3
1457  04F6  020A           LI    R10,NULINE
      04F8  0024'
1458  04FA  06A0           BL    @PRNTC
      04FC  04E2'
1459  04FE  10F9           JMP   PT800              **** ERROR/ESC RETURN
1460  0500  0206           LI    R6,MAXLN           RE-ESTABLISH LINE COUNT
      0502  0041
1461  0504  0453           B     *R3
1462              *                          2-END PRTTRACE;
```

```
1465                    * TITLE:     UDREG
1466                    *            UPDATE REGISTERS
1467                    * REVISION:
1468                    *            ORIGINAL
1469                    * ABSTRACT:
1470                    *            THE INDICATED SOURCE (DESTINATION) EFFECTIVE
1471                    *            ADDRESS IS USED TO SET THE 'AFTER' CONTENTS
1472                    *            OF THE EFFECTIVE ADDRESS.  IN ADDITION, IF
1473                    *            'REGISTER' FLAG IS SET IN PRINT FLAG, THEN
1474                    *            IF THE CONTENTS OF THE REGISTER HAVE CHANGED,
1475                    *            THEN THE NEW CONTENTS ARE SET IN PTABLE, ELSE
1476                    *            THE 'REGISTER' FLAG IS RESET
1477                    * CALLING SEQUENCE:
1478                    *            (R10) := PRINT FLAG
1479                    *            BL    @UDREG
1480                    *            DATA  (BASE ADDRESS OF REGISTER SET IN PTABLE)
1481                    *            DATA  (EFFECTIVE ADDRESS MASK)
1482                    *            DATA  (REGISTER CONTENTS MASK)
1483                    *            DATA  (ADDRESS OF REGISTER)
1484                    *
1485                    *                        1-PROCEDURE UPDATEREG (PRINTFLAG,
1486                    *                        4-BASE, REG);
1487                    *                        2-DECLARE REG POINTER;
1488                    *                        2-DECLARE MEMWORD BIT (16) CONTROL
1489                    *                        2-DECLARE BASE FIXED (16),
1490                    *                        2-DECLARE (1 PRINTFLAG,
1491                    *                        3-          3 PRTBIT (16) BIT (1));
1492                    *                        2-DECLARE PTABLE (16) BIT (16);
1493                    *                        2-DECLARE PTR POINTER;
1494                    *                        2-DECLARE MWORD BIT (16) CONTROL (P
1495          0506' UDREG   EQU   $
1496  0506    C3FB          MOV   *R11+,R15           PTABLE BASE ADDRESS
1497                    *                        2-IF PRTBIT (BASE+PSE) = 1 THEN DO;
1498  0508    22BB          COC   *R11+,R10
1499  050A    1604          JNE   UDREG2              IF THIS REG UNUSED
1500                    *                        3-PTR = PTABLE (BASE+PSE);
1501  050C    C3AF          MOV   @PSE(R15),R14
      050E    0000
1502                    *                        3-PTABLE (BASE+PSA) = MWORD;
1503  0510    CBDE          MOV   *R14,@PSA(R15)
      0512    0004
1504                    *                        3-END;
1505          0514' UDREG2  EQU   $
1506  0514    C3BB          MOV   *R11+,R14           REG CHANGE MASK
1507  0516    C03B          MOV   *R11+,R0            REG ADDR
1508                    *                        2-IF PRTBIT (BASE+PSR) = 1 THEN DO;
1509  0518    228E          COC   R14,R10
1510  051A    1608          JNE   UDREG4              IF DONT CARE ABOUT REG CHANGE
1511                    *                        3-PRTBIT (BASE+PSR) = 0;
1512  051C    428E          SZC   R14,R10
1513                    *                        3-IF MEMWORD .NE. PTABLE (BASE+PSR)
1514  051E    C010          MOV   *R0,R0
1515  0520    8BD0          C     *R0,@PSR(R15)
      0522    0006
```

```
 1516  0524  1303          JEQ   UDREG4              IF NO CHANGE
 1517                *                          4-PRTBIT (BASE+PSR) = 1;
 1518  0526  E28E          SOC   R14,R10
 1519                *                          4-PTABLE (BASE+PSR) = MEMWORD;
 1520  0528  CBD0          MOV   *R0,@PSR(R15)
       052A  0006
 1521                *                          4-END;
 1522                *                          3-END;
 1523        052C' UDREG4  EQU   $
 1524                *                          2-RETURN;
 1525  052C  045B          B     *R11
 1526                *                          2-END UPDATEREG;
```

```
1529                    * TITLE:     PUSH
1530                    *            PLACE ADDRESS AND (ADDRESS) ON STACK
1531                    * REVISION:
1532                    *            ORIGINAL
1533                    * ABSTRACT:
1534                    *            THE PASSED ADDRESS AND ITS CONTENTS ARE PLACED
1535                    *            ON THE STACK.  STACK OVERFLOW IS DETECTED
1536                    *            AND IGNORED.
1537                    * CALLING SEQUENCE:
1538                    *            (R10) ::= ADDRESS
1539                    *            BL    @PUSH
1540                    *                        1-PROCEDURE PUSHSTACK (ADDR);
1541                    *                        2-DECLARE ADDR POINTER;
1542                    *                        2-DECLARE STACKMAX LITERALLY '4';
1543                    *                        2-DECLARE MEMWORD BIT (16) CONTROL
1544                    *
1545           052E'    PUSH    EQU   $
1546                    *                        2-IF STACKINDEX >= STACKMAX THEN R
1547    052E   C060             MOV   @STINX,R1
       0530   00E4'
1548    0532   0281             CI    R1,STKMAX
       0534   0008
1549    0536   1406             JHE   PUSH90
1550                    *                        2-ADDRSTACK (STACKINDEX) = ADDR;
1551    0538   C84A             MOV   R10,@ADRSTK(R1)
       053A   00E6'
1552                    *                        2-BEFORECONTENTS (STACKINDEX) = ME
1553    053C   C85A             MOV   *R10,@B4CNTS(R1)
       053E   00EE'
1554                    *                        2-STACKINDEX = STACKINDEX + 1;
1555    0540   05E0             INCT  @STINX
       0542   00E4'
1556                    *                        2-RETURN;
1557           0544'    PUSH90  EQU   $
1558    0544   045B             B     *R11
```

```
 1561                   * TITLE:     ATOINC
 1562                   *            AUTO INCREMENT TEST
 1563                   * REVISION:
 1564                   *            ORIGINAL
 1565                   * ABSTRACT:
 1566                   *            THE SOURCE REGISTER IS INCREMNTED BY TWO
 1567                   *            IF NECESSARY.
 1568                   * CALLING SEQUENCE:
 1569                   *            BL   @ATOINC
 1570                   *                           1-PROCEDURE AUTOINCTEST;
 1571                   *                           2-IF  AND (SRCRNM, "30") = "30"
 1572          0546' ATOINC EQU  $
 1573   0546   0201          LI   R1,>30
        0548   0030
 1574   054A   4060          SZC  @SRCRNM,R1
        054C   0016'
 1575   054E   1603          JNE  ATOI90
 1576                   *                           3-THEN SRCREGCONTENTS =
 1577                   *                           3-     SRCREGCONTENTS + 2;
 1578   0550   C060          MOV  @SRCREG,R1
        0552   0036'
 1579   0554   05D1          INCT *R1
 1580                   *                           2-END;
 1581          0556' ATOI90 EQU  $
 1582   0556   045B          RT
 1583                        END
0000 ERS
```

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																
SHEET																																
REV STATUS OF SHEETS	REV																															
	SHEET																															

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DWN	DATE
	CHK	
	ENGR	
	QA	
	APVD	
	CONTR NO	
	DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, ABSDMP, DUMP ABSOLUTE-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945388-9901

SCALE	REV.	SHEET 1 of 15

TI-0023-C

```
0003                * TITLE:    DPRG
0004                *           DUMP MEMORY TO CASSETTE IN ABS DATA FORMAT
0005                *
0006                * REVISION:
0007                *           ORIGINAL
0008                *
0009                * ABSTRACT: DPRG IS A PX990 MONITOR COMMAND PROCESSOR WHICH
0010                *           TRANSLATES THE MEMORY AREA DEFINED BY "START"
0011                *           AND "END" INTO THE ABSOLUTE LOAD DATA FORMAT
0012                *           AND SAVES THE PROGRAM IMAGE ON CASSETTE TAPE.
0013                *
0014                *           IF THE OPERATOR WANTS THE OUTPUT DIRECTED TO
0015                *           OTHER THAN THE DEFAULT OUTPUT CASSETTE UNIT,
0016                *           LUNO 7 MUST BE ATTACHED TO THE DESIRED DEVICE
0017                *           PRIOR TO ENTERING THE "DP" COMMAND.  THE TRANS-
0018                *           FER WILL BEGIN AT THE CURRENT TAPE POSITION.
0019                *           THE COMMAND WILL ABORT IF THE SELECTED UNIT IS
0020                *           NOT READY OR ENCOUNTERS AN UNRECOVERABLE ERROR.
0021                *
0022                *           IF "ENTRY" AND/OR "NAME" PARAMETERS ARE ENTERED
0023                *           THEY ARE APPROPRIATELY TAGGED FOR THE LOADER
0024                *           AND WRITTEN WITH THE ABLOLUTE LOAD ADDRESS AS
0025                *           THE FIRST DATA RECORD ON THE FILE.
0026                *
0027                *           CONTIGUOUS WORDS HAVING THE SAME DATA VALUE
0028                *           WILL BE COMPRESSED BEFORE DUMPING
0029                *
0030                *           IF "PARTIAL" PARAMETER ENTERED, EOM TAG AND END
0031                *           OF FILE WILL NOT BE OUTPUT ON TAPE.
0032                *
0033                * CALLING SEQUENCE:
0034                *           DP  START,END,ENTRY,NAME,PARTIAL
0035                *
0036                *           START(R!QD): 1-4 HEX CH. REPRESENTING THE ADDR.
0037                *              OF THE FIRST BYTE TO BE DUMPED
0038                *           END(R!QD): 1-4 HEX CH. REPRESENTING THE ADDR.
0039                *              OF THE BYTE AFTER THE LAST BYTE DUMPED.
0040                *           ENTRY(OPT): 1-4 HEX CH. REPRESENTING THE ADDR.
0041                *              AT WHICH EXECUTION IS TO BEGIN WHEN THE
0042                *              PROG. IS RELOADED(NO DEFAULT SUPPLIED).
0043                *           NAME(OPT):1-23 ALPHANUMERICS, THE NAME BY WHICH
0044                *              THE PROGRAM IS TO BE KNOWN(NO DFLT SUPPLIED)
0045                *           PARTIAL(OPT): 1 CH. IF CHAR EQUALS 'P' THEN
0046                *              PARTIAL DUMP OCCURS.
0047                *
0048                *
0049                *           ERROR CODES
0050                *              MS01 SYNTAX ERROR                    -ABORT
0051                *              MS05 REQUIRED PARM NOT ENTERED       -ABORT
0052                *              MP00 PARAMETER SPECIFICATION ERROR   -ABORT
0053                *              MX01 UNRECOVERABLE I/O ERROR -       -ABORT
0054                *              DP13 INCORRECT LIMITS
0055                *
0056                ****** ALGORITHM:
```

```
0057                *           NOTE: IN-LINE COMMENTS CORRES. W/STATEMENTS
0058                *
0059                *
0060                      IDT  'ABSDMP'
0061                ****REGISTER EQUATES*****
0062                *     ALL REGISTER REFERENCES IN ABS DUMP ARE VIA ACRONYMS
0063                *     INDICATIVE OF THE CURRENT CONTENTS OF THE REGISTER.
0064                *     THE FOLLOWING DEFINES THE ACRONYMS. WHEN MULTIPLE
0065                *     ACRONYMS ARE ASSIGNED TO A REGISTER, THE ORDER OF
0066                *     THEIR USE CORRESPONDS TO THE ORDER GIVEN BELOW. ANY
0067                *     TIME A REGISTER FREES UP, OR GETS RE-ASSIGNED, ONE
0068                *     OF THE FOLLOWING COMMENT CARDS WILL SO INDICATE.
0069                *
0070                *** REGISTER REASSIGNMENT:PRIOR WPX='PTR1', NEW WPX='FREE'
0071                *** REGISTER REASSIGNMENT:PRIOR WPX='PTR1', NEW WPX='PTR2'
0072                *
0073          0000  R0      EQU  0                 SCRATCH
0074          0000  DATA    EQU  0                 16 BIT DATA VALUE
0075          0001  R1      EQU  1                 SCRATCH
0076          0002  CPRM    EQU  2                 ADDR. OF COMMAND PARM. TABLE
0077          0003  HDR     EQU  3                 HEADER FROM COMMAND PARM. TAB
0078          0004  ST      EQU  4                 DUMP START ADDR.
0079          0005  LN      EQU  5                 DUMP LENGTH
0080          0006  EP      EQU  6                 DUMP ENTRY POINT
0081          0006  DUPC    EQU  6                 COMPRESSION REPEAT COUNT
0082          0007  PRBA    EQU  7                 PRB ADDRESS
0083          0008  BFRA    EQU  8                 OUTPUT BFR INDEX
0084          0009  BFRE    EQU  9                 END OF OUTPUT BFR
0085          000A  R10     EQU  >A                SCRATCH
0086          000B  R11     EQU  >B                RETURN PC
0087          000D  R13     EQU  >D                ADDR. OF PREVIOUS WORKSPACE
0088                *** ABSOLUTE DATA FORMAT EQUATES
0089                *
0090          0004  ABS     EQU  4                 16 BIT ABS DATA TAG
0091          0070  ABEND   EQU  >70               END OF MODULE TAG
0092          0073  ABEP    EQU  >73               16 BIT ABS ENTRY POINT
0093          0006  ABLP    EQU  6                 16 BIT ABS LOAD POINT
0094          0071  ABPN    EQU  >71               1-8 CH. PROGRAM NAME
0095          0076  ABRP    EQU  >76               ABS DATA REPEAT(COMPRESS)
0096          0005  ABSB    EQU  >5                8 BIT ABS DATA
0097          0075  ABCKSM  EQU  >75               16 BIT CHECKSUM(ABS)
0098                *
0099                *** SERVICE CALL EQUATES
0100                *
0101          0000  OPEN    EQU   0                OPEN FILE
0102          000D  EOF     EQU   13               WRITE END-OF-FILE
0103          000B  WRTA    EQU  11                WRITE ASCII
0104                *
0105                *** ERROR MESSAGE EQUATES
0106                *
0107          0205  MS05    EQU  >0205             REQUIRED PARAMETER MISSING
0108          0100  MP00    EQU  >0100             NUMERIC VALUE OUT OF RANGE
0109          0001  MX01    EQU  >0001             UNRECOVERABLE I/O ERROR
0110          1113  DP13    EQU  >1113             LIMIT ERROR
0111                *** EXTERNAL REFERENCES
```

```
0112                    *
0113                            REF   ACL
0114                            REF   RETWSP
0115                            REF   RETBUF
0116                            REF   ERROR
0117                            REF   RR
0118                            REF   SVCALT
0119                    *** DATA SECTIONS
0120                    *
0121                    *
0122                    *       OVERLAY PARAMETERS
0123                    *
0124                            DEF   DPRG
0125                            REF   LDABS
0126  0000     44               TEXT 'DP'
0127  0002  0072'              DATA DPRG
0128  0004     4C               TEXT 'LA'
0129  0006  0000               DATA LDABS
0130  0008  0000               DATA 0
0131                    *
0132                    *** PERIPHERAL REQUEST BLOCK(PRB)
0133                    *
0134          000A'     PRB     EQU  $
0135  000A     00       SVC     BYTE 0
0136  000B     00               BYTE 0
0137  000C     00       PRIOP   BYTE 0
0138  000D     07       PRIOLN  BYTE 7
0139  000E     00       PRSFG   BYTE 0
0140  000F     00       PRUFG   BYTE 0
0141  0010  001A'       PRBFA   DATA BFR
0142  0012  0056       PRBLN   DATA 86
0143  0014  0000       PRCNT   DATA 0
0144  0016  0000               DATA 0
0145  0018  0000               DATA 0
0146                    *
0147                    *** OUTPUT BUFFER(BFR)
0148  001A              BFR     BSS  86
0149                    *
0150  0070     50       PARTIL TEXT 'P '
0151                    *
```

```
0153                        DEF  DPRG
0154          0072' DPRG    EQU  $                    ABS DUMP ENTRY POINT
0155
0156                  *   SAVE CALLING WP & SWITCH TO LOCAL WP
0157  0072  C08A             MOV  R10,CPRM            SAVE PARM ADDR IN ORIG WP
0158  0074  0420             BLWP @ACL
      0076  0000
0159                  *-1 IF((STRB.AND.LNGB).EQ.0)ABORT(MS05)
0160  0078  C0F2             MOV  *CPRM+,HDR
0161  007A  0A93             SLA  HDR,9               CHECK FOR START PARM PRESENT
                                                         AND ABORT IF NOT
0162  007C  1702             JNC  ABRT1
0163  007E  0A13             SLA  HDR,1               CHECK FOR LENGTH PRESENT
0164  0080  1804             JOC  DPRG01                JUMP IF OK
0165          0082' ABRT1   EQU  $
0166  0082  020A             LI   R10,MS05
      0084  0205
0167          0086' ABRT2   EQU  $
0168  0086  0460             B    @ABORT
      0088  01BE'
0169          008A' DPRG01  EQU  $
0170                  *1  CALL TRPLT(ABLP,ST)
0171  008A  C132             MOV  *CPRM+,ST           GET THE START AND LENGTH PARM
0172  008C  C172             MOV  *CPRM+,LN
0173  008E  6144             S    ST,LN               FIND LENGTH
0174  0090  0585             INC  LN
0175  0092  1503             JGT  DP001               IF LEN LE 0 THEN LIMIT ERROR
0176  0094  020A             LI   R10,DP13
      0096  1113
0177  0098  10F6             JMP  ABRT2
0178          009A' DP001   EQU  $
0179  009A  C004             MOV  ST,R0               MOVE THE LOAD POINT INTO THE
0180  009C  0201             LI   R1,ABLP               OUTPUT BUFFER.
      009E  0006
0181  00A0  0208             LI   BFRA,BFR
      00A2  001A'
0182  00A4  0209             LI   BFRE,BFR+80-5-5     LEAVE ROOM FOR CHECKSUM
      00A6  0060'
0183  00A8  06A0             BL   @TRPLT
      00AA  01CA'
0184                    *1  IF(EPB)
0185                    *2    THEN:
0186                    *2     CALL TRPLT(ABEP,EP)
0187                    *2     IF((EP-ST).GE.LN)CALL ABORT(MP02)
0188                    *2    ELSE:
0189                    *1  ENDIF
0190                    *1
0191                    *1
0192                    *1
0193  00AC  0A13             SLA  HDR,1               TEST FOR EP PRESENT
0194  00AE  170E             JNC  DPRG02                JUMP IF NOT
0195  00B0  0200             LI   R0,ABEP*256
      00B2  7300
0196  00B4  DE00             MOVB R0,*BFRA+
0197  00B6  C012             MOV  *CPRM,R0            MOVE EP TO OUTPUT BFR
```

```
0198  00B8  0201          LI   R1,ABS
      00BA  0004
0199  00BC  06A0          BL   @TRPLT
      00BE  01CA'
0200  00C0  C032          MOV  *CPRM+,R0          TEST FOR EP IN DUMP AREA
0201  00C2  6004          S    ST,R0
0202  00C4  020A          LI   R10,MP00           IF EP NOT IN RANGE,
      00C6  0100
0203  00C8  8140          C    R0,LN
0204  00CA  14DD          JHE  ABRT2              LOAD ERROR MSG, AND ABORT
0205
0206        00CC' DPRG02 EQU  $
0207
0208              *1  IF(NMB)OUT(NAME)
0209  00CC  0A13          SLA  HDR,1              CHECK FOR PROG NAME PRESENT
0210  00CE  1711          JNC  DPRG04               AND JUMP IF NOT
0211  00D0  04C1          CLR  R1                 GET PROGRAM NAME TAG
0212  00D2  D072          MOVB *CPRM+,R1            AND CHARACTER COUNT
0213  00D4  0261          ORI  R1,ABS*4096+ABPN
      00D6  4071
0214  00D8  06C1          SWPB R1                   INTO OUTPUT BFR.
0215  00DA  DE01          MOVB R1,*BFRA+
0216  00DC  06C1          SWPB R1
0217  00DE  DE01          MOVB R1,*BFRA+
0218  00E0  0A31          SLA  R1,3               CLEAR OFF ALL BUT LOW ORDER
0219  00E2  09B1          SRL  R1,11                SIX BITS OF CHARACTER COUNT
0220  00E4  A042          A    CPRM,R1              & ADJUST TO END OF STRING.
0221
0222        00E6' DPRG03 EQU  $
0223
0224  00E6  DE32          MOVB *CPRM+,*BFRA+      MOVE PROGRAM NAME INTO
0225  00E8  8042          C    CPRM,R1              OUTPUT BUFFER
0226  00EA  11FD          JLT  DPRG03               LOOP BACK TILL DONE
0227  00EC  0911          SRL  R1,1               CHECK IF ON A EVEN BOUNDARY
0228  00EE  1701          JNC  DPRG04
0229  00F0  0582          INC  CPRM               MOVE POINTER TO WORD BOUNDARY
0230        00F2' DPRG04 EQU  $                  PARAMETER VALIDATION COMPLETE
0231
0232              *1  CALL OPEN(PRB)
0233              *       (IN LINE EXPANSION)
0234  00F2  0207          LI   PRBA,PRB           GET PRB ADDR IN REG
      00F4  000A'
0235  00F6  0200          LI   R0,OPEN            OPEN THE DEVICE
      00F8  0000
0236  00FA  D800          MOVB R0,@PRSFG          CLEAR SYSTEM FLAGS THEN
      00FC  000E'
0237  00FE  D800          MOVB R0,@PRIOP
      0100  000C'
0238  0102  C287          MOV  PRBA,R10
0239  0104  0420          BLWP @SVCALT
      0106  0000
0240  0108  D020          MOVB @PRSFG,R0
      010A  000E'
0241  010C  16F2          JNE  DPRG04               LOOP UNTIL DEVICE IS READY
0242              *1  CALL LABEL(PRBA,BFRA,BFRE)
```

```
0243  010E  0200          LI   R0,WRTA*256       SET PRB I/O CODE TO
      0110  0B00
0244  0112  D800          MOVB R0,@PRIOP           WRITE ASCII
      0114  000C'
0245                *1  CALL CKSM(BFRA,BFR)
0246  0116  06A0          BL   @CKSM               CHECKSUM THE DATA
      0118  021A'
0247                *1= CALL WRITE(BFRA,PRBA,STATUS)
0248  011A  06A0          BL   @WRITE
      011C  01EE'
0249  011E  164D          JNE  DPRG18                JUMP IF WRITE FAILED
0250                *1= IF(MOD(ST,2),NE,0)CALL BYTOUT(ST,BFRA)
0251  0120  C004          MOV  ST,R0
0252  0122  0B10          SRC  R0,1                PUT LSB OF START IN CARRY BIT
0253  0124  1703          JNC  DPRG10
0254  0126  06A0          BL   @BYTOUT             CALL BYTOUT.
      0128  0206'
0255  012A  0605          DEC  LN
0256                ***  NOW READY TO START DATA TRANSFER--PROCESS CODED IN LINE
0257                *REG REASSIGN: PRIOR R6=EP , NEW R6=DUPC(COMPRESSION CNTR)
0258                *
0259        012C' DPRG10 EQU  $
0260                *
0261  012C  04C6          CLR  DUPC                CLEAR REPEAT COUNT
0262  012E  0285          CI   LN,1                HAVE ALL BYTES BEEN DUMPED?
      0130  0001
0263  0132  1321          JEQ  DPRG15              JUMP IF ON BYTE REMAINING
0264  0134  1122          JLT  DPRG16              JUMP IF ALL DONE
0265        0136' DPRG11 EQU  $
0266  0136  C034          MOV  *ST+,DATA           GET THE NEXT WORD AND CHECK
0267  0138  0645          DECT LN                    FOR POSSIBLE COMPRESSION.
0268  013A  0285          CI   LN,1
      013C  0001
0269  013E  1308          JEQ  DPRG12              JUMP IF END OF DATA
0270  0140  1107          JLT  DPRG12              JUMP IF END OF DATA
0271  0142  8500          C    DATA,*ST            JUMP IF NO COMPRESSION
0272  0144  1605          JNE  DPRG12                REQUIRED
0273  0146  0586          INC  DUPC                INCR REPEAT COUNT AND CHECK
0274  0148  0286          CI   DUPC,63               FOR MAX COMPRESSION
      014A  003F
0275  014C  11F4          JLT  DPRG11              JUMP TO CONTINUE COMPRESSION
0276  014E  0606          DEC  DUPC                  (ADJUST REP COUNT)
0277        0150' DPRG12 EQU  $
0278  0150  0286          CI   DUPC,0              CHECK FOR COMPRESSION AND
      0152  0000
0279  0154  1305          JEQ  DPRG13                JUMP IF NONE OCCURED
0280  0156  0226          AI   DUPC,ABRP*16+ABS*16+1  ELSE, GENERATE THE
      0158  7641
0281  015A  DE06          MOVB DUPC,*BFRA+
0282  015C  06C6          SWPB DUPC
0283  015E  DE06          MOVB DUPC,*BFRA+
0284                *                                CHARACTER COUNT, AND
0285                *                                STORE IN THE OUTPUT
0286                *                                BUFFER.
0287        0160' DPRG13 EQU  $
```

```
                   ** TRANSMIT THE DATA TRIPLET **
0160  0201                LI   R1,ABS          SET TAG TO 16 BIT ABS DATA
0162  0004
0164  06A0                BL   @TRPLT            AND TRANSLATE INTO OUT BFR.
0166  01CA'
0168  12E1                JLE  DPRG10          REPEAT IF BFR NOT FULL.
      016A' DPRG14 EQU  $
                   ** OUTPUT BUFFER IS FULL.---DUMP IT TO TAPE
016A  06A0                BL   @CKSM           CHECKSUM THE DATA AND
016C  021A'
016E  06A0                BL   @WRITE          WRITE THE BUFFER
0170  01EE'
0172  1623                JNE  DPRG18          JUMP IF WRITE FAILED
0174  10DB                JMP  DPRG10          ELSE, REPEAT FROM TOP
      0176' DPRG15 EQU  $
0176  06A0                BL   @BYTOUT           ONE BYTE LEFT. OUTPUT IT.
0178  0206'
      017A' DPRG16 EQU  $
017A  0A13                SLA  HDR,1
017C  170D                JNC  DP020
017E  C012                MOV  *CPRM,R0        IF PARM .EQ. PARTIAL
0180  0A80                SLA  R0,8
0182  9800                CB   R0,@PARTIL       THEN EXIT WITHOUT WRITTING
0184  0070'
0186  1608                JNE  DP020            EOF.
0188  8808                C    BFRA,@PRBFA
018A  0010'
018C  131A                JEQ  DPRG19
018E  06A0                BL   @CKSM
0190  021A'
0192  06A0                BL   @WRITE
0194  01EE'
0196  1015                JMP  DPRG19          ELSE
      0198' DP020  EQU  $
0198  0200                LI   R0,ABEND*256    ADD EOM TAG TO DATA
019A  7000
019C  DE00                MOVB R0,*BFRA+
019E  06A0                BL   @CKSM           ELSE, CHEKSUM THE DATA AND
01A0  021A'
01A2  06A0                BL   @WRITE            WRITE IT OUT
01A4  01EE'
                   *
      01A6' DPRG17 EQU  $
                   *
                   ** THE DUMP IS FINISHED---CLOSE THE FILE AND TERMINATE
                   *
01A6  0200                LI   R0,EOF*256      WRITE EOF
01A8  0D00
01AA  D800                MOVB R0,@PRIOP
01AC  000C'
01AE  C287                MOV  PRBA,R10
01B0  0420                BLWP @SVCALT
01B2  0106'
01B4  D020                MOVB @PRSFG,R0
01B6  000E'
```

```
0327  01B8  1304          JEQ   DPRG19          JUMP IF WEOF SUCCESSFUL
0328                *
0329        01BA' DPRG18 EQU   $
0330  01BA  020A          LI    R10,MX01        LOAD I/O ABORT MSGID
      01BC  0001
0331                *
0332        01BE' ABORT  EQU   $
0333                *
0334  01BE  06A0          BL    @ERROR          OUTPUT THE ERROR MESSAGE
      01C0  0000
0335                *
0336        01C2' DPRG19 EQU $
0337                *
0338  01C2  0420          BLWP @RR
      01C4  0000
0339  01C6  C282          MOV   CPRM,R10        RESET R10 FOR CMD PRCSR
0340  01C8  045B          RT
```

```
0343                * TITLE:    TRPLT
0344                *           TRANSLATE AND TRANSFER 16 DATA BITS
0345                * REVISION:
0346                *           ORIGINAL
0347                * ABSTRACT: TRPLT TRANSLATES A 16 BIT WORD INTO 3 BYTES,
0348                *           A TRIPLET,AND MOVES THE BYTES INTO THE SYSTEM
0349                *           BUFFER.
0350                * CALLING SEQUENCE:
0351                *           ENTRY: R1 = R1  =THE 4 BIT ABS. LOAD TAG
0352                *                  R8 =BFRA =ADDRESS OF CURRENT BFR
0353                *                  R0 =DATA =WORD TO BE TRANSLATED
0354                *                  R9 =BFRE =END OF DATA AREA IN BFR.
0355                *
0356                *           EXIT :  R8 =BFRA =CURRENT BYTE ADDR IN BFR
0357                *                   R0,R1 =DESTROYED
0358                *                   ST = .LT. IF BFR NOT FULL
0359                *                      = .GE. IF BFR FULL
0360         01CA'  TRPLT  EQU  $
0361   01CA  D040          MOVB R0,R1              FORMAT BYTE 0 AND STORE
0362   01CC  0B41          SRC  R1,4                 IN OUTPUT BUFFER
0363   01CE  DE01          MOVB R1,*BFRA+            (SOURCE BITS 0-3)
0364   01D0  06C0          SWPB R0                 FORMAT 2ND BYTE
0365   01D2  0941          SRL  R1,4                 (SOURCE BITS 4-9)
0366   01D4  D040          MOVB R0,R1
0367   01D6  0B51          SRC  R1,5
0368   01D8  0261          ORI  R1,ABS*8192
       01DA  8000
0369   01DC  0911          SRL  R1,1
0370   01DE  DE01          MOVB R1,*BFRA+
0371   01E0  0A71          SLA  R1,7
0372   01E2  0261          ORI  R1,ABS*8192          SOURCE BITS10-15
       01E4  8000
0373   01E6  0911          SRL  R1,1
0374   01E8  DE01          MOVB R1,*BFRA+
0375   01EA  8248          C    BFRA,BFRE          SET STATUS BITS AND RETURN
0376   01EC  045B          RT
```

```
0379                    * TITLE:    WRITE
0380                    * REVISION:
0381                    *           ORIGINAL
0382                    * ABSTRACT: THE OUTPUT BUFFER POINTED AT BY «PRBA»
0383                    *           IS PRESENTED TO MONITOR VIA A WRITE SERVICE
0384                    *           CALL.
0385                    * CALLING SEQUENCE:
0386                    *           ENTRY:
0387                    *             R7 =PRBA=ADDRESS OF PRB
0388                    *             R8 =BFRA=ADDR OF NEXT CH. POS IN BFR.
0389                    *
0390                    *           EXIT:
0391                    *                BFRA=ADDR OF 1ST CH. POS. IN BFR
0392                    *                R1B0=STATUS OF WRITE(SYS.FLAGS FROM PRB)
0393                    *                STATUS REG = .EQ. FOR SUCCESSFUL
0394                    *                           = .NE. FOR TROUBLE
0395                    *                ALL OTHER REGS UNCHANGED.
0396                    *
0397         01EE' WRITE  EQU  $              ****ENTRY POINT****
0398  01EE   0220         S    @PRBFA,BFRA      COMPUTE CHAR. COUNT AND
       01F0  0010'
0399  01F2   C808         MOV  BFRA,@PRCNT        STORE IN PRB.
       01F4  0014'
0400  01F6   C287         MOV  PRBA,R10
0401  01F8   0420         BLWP @SVCALT
       01FA  01B2'
0402  01FC   0208         LI   BFRA,BFR         RESET THE BUFFER POINTER
       01FE  001A'
0403  0200   D060         MOVB @PRSFG,R1        GET THE I/O STATUS FLAGS AND
       0202  000E'
0404                    *                          RETURN TO CALLER WITH STATU
0405                    *                          SET.
0406  0204   045B         RT
```

```
0409                    * TITLE   : BYTOUT
0410                    *
0411                    * REVISION:
0412                    *           ORIGINAL
0413                    *
0414                    * ABSTRACT:
0415                    *           GET A BYTE OF DATA FROM THE DUMP AREA, FORMAT
0416                    *           WITH ABSOLUTE BYTE TAG, AND STORE IN THE
0417                    *           OUTPUT BUFFER.
0418                    *
0419                    * CALLING SEQUENCE:
0420                    *           ENTRY:
0421                    *              R4 = ST  =ADDRESS OF BYTE TO BE FORMATTED
0422                    *              R8 = BFRA=PLACE TO PUT REFORMATTED BYTE
0423                    *           EXIT :
0424                    *                 (ST)=(ST)+1
0425                    *                 (BFRA)=(BFRA)+1
0426                    *                 R1 DESTROYED
0427                    *              ALL OTHER REGS UNTOUCHED
0428          0206'     BYTOUT EQU  $               ****ENTRY POINT****
0429    0206  0201             LI   R1,ABSB            GET THE ABS BYTE TAG
        0208  0005
0430    020A  D074             MOVB *ST+,R1              AND CONCAT. WITH BITS 0-3
0431    020C  0B41             SRC  R1,4                 OF THE DATA BYTE.
0432    020E  DE01             MOVB R1,*BFRA+          MOVE INTO OUTPUT BFR.
0433    0210  0A61             SLA  R1,6               LEFT JUSTIFY BITS 4-7, CNC
0434    0212  0261             ORI  R1,ABS*4096          WITH ABS TAG, AND
        0214  4000
0435    0216  DE01             MOVB R1,*BFRA+          MOVE INTO OUTPUT BFR.
0436    0218  045B             RT                      RETURN
```

```
0439                    * TITLE   : CKSM
0440                    *
0441                    * REVISION:
0442                    *           ORIGINAL
0443                    *
0444                    * ABSTRACT:
0445                    *           THE EXCLUSIVE OR CHECKSUM OF EACH EVEN NUMBERED
0446                    *           BYTE AND THAT OF EACH ODD NUMBERED BYTE ARE
0447                    *           COMPUTED AND RETURNED IN THE LEFT AND RIGHT
0448                    *           HALVES OF A 16 BIT WORD.
0449                    *           NOTE THAT IF THE LAST
0450                    *           BYTE IS ON AN EVEN ADDRESS, THE RIGHT HALF OF
0451                    *           THE CHECKSUM IS PRESERVED SO THAT THE NUMBER
0452                    *           OF EVEN BYTES XOR'ED MAY BE 1 GREATER THAN THE
0453                    *           NUMBER OF ODD BYTES XOR'ED.  WHEN CKSM FINISHES
0454                    *           IT FALLS INTO TRPLT TO STASH THE CHECKSUM.
0455                    * CALLING SEQUENCE
0456                    *           ENTRY:THE VALUE OF 'BFR' IS THE STRING ADDRESS
0457                    *                 R8 =BFRA=THE NEXT AVAILABLE BYTE ADDR.
0458                    *           EXIT: R0 =COMPUTED CHECKSUM
0459                    *                 R1 =DESTROYED
0460                    *                 ALL OTHER REGS UN=ALTERED
0461                    *
0462          021A' CKSM    EQU  $              ****ENTRY
0463   021A   04C0         CLR  R0                 CLEAR CHECKSUM ACCUMULATOR
0464   021C   C048         MOV  BFRA,R1            CHECK NUMBER OF BYTES AND
0465   021E   0B11         SRC  R1,1                 JUMP IF EVEN
0466   0220   1701         JNC  CKSM1
0467                    *
0468   0222   D600         MOVB R0,*BFRA           CLR RIGHT HALF OF ODD BYTE WD
0469                    *
0470          0224' CKSM1   EQU  $
0471                    *
0472   0224   0201         LI   R1,BFR             GET THE STARTING ADDRESS
       0226   001A'
0473                    *
0474          0228' CKSM2   EQU  $
0475                    *
0476   0228   2831         XOR  *R1+,R0            CHECKSUM ANOTHER WORD, AND
0477   022A   8201         C    R1,BFRA              IF NOT DONE,
0478   022C   1AFD         JL   CKSM2                LOOP BACK FOR ANOTHER TIME
0479   022E   1301         JEQ  CKSM3
0480   0230   06C0         SWPB R0
0481          0232' CKSM3   EQU  $
0482   0232   0201         LI   R1,ABCKSM*256      CHECKSUM THE CKSM TAG
       0234   7500
0483   0236   2801         XOR  R1,R0
0484   0238   0221         AI   R1,ABS             STORE THE CKSM TAG IN THE
       023A   0004
0485   023C   DE01         MOVB R1,*BFRA+            OUTPUT BUFFER AND RETURN
0486   023E   0460         B    @TRPLT
       0240   01CA'
0487                       END
0000 ERS
```

$		0134	0154	0165	0167	0169	0178	0206	0222	0230
		0259	0265	0277	0287	0292	0298	0300	0312	0318
		0329	0332	0336	0360	0397	0428	0462	0470	0474
		0481								
ABCKSM	0097	0482								
ABEND	0091	0313								
ABEP	0092	0195								
ABLP	0093	0180								
ABORT	0332	0168								
ABPN	0094	0213								
ABRP	0095	0280								
ABRT1	0165	0162								
ABRT2	0167	0177	0204							
ABS	0090	0198	0213	0280	0289	0368	0372	0434	0484	
ABSB	0096	0429								
ACL		0113	0158							
BFR	0148	0141	0181	0182	0402	0472				
BFRA	0083	0181	0196	0215	0217	0224	0281	0283	0307	0314
		0363	0370	0374	0375	0398	0399	0402	0432	0435
		0464	0468	0477	0485					
BFRE	0084	0182	0375							
BYTOUT	0428	0254	0299							
CKSM	0462	0246	0294	0309	0315					
CKSM1	0470	0466								
CKSM2	0474	0478								
CKSM3	0481	0479								
CPRM	0076	0157	0160	0171	0172	0197	0200	0212	0220	0224
		0225	0229	0303	0339					
DATA	0074	0266	0271							
DP001	0178	0175								
DP020	0312	0302	0306							
DP13	0110	0176								
DPRG	0154	0124	0127	0153						
DPRG01	0169	0164								
DPRG02	0206	0194								
DPRG03	0222	0226								
DPRG04	0230	0210	0228	0241						
DPRG10	0259	0253	0291	0297						
DPRG11	0265	0275								
DPRG12	0277	0269	0270	0272						
DPRG13	0287	0279								
DPRG14	0292									
DPRG15	0298	0263								
DPRG16	0300	0264								
DPRG17	0318									
DPRG18	0329	0249	0296							
DPRG19	0336	0308	0311	0327						
DUPC	0081	0261	0273	0274	0276	0278	0280	0281	0282	0283
EOF	0102	0322								
EP	0080									
ERROR		0116	0334							
HDR	0077	0160	0161	0163	0193	0209	0301			
LDABS		0125	0129							
LN	0079	0172	0173	0174	0203	0255	0262	0267	0268	
MP00	0108	0202								
MS05	0107	0166								
MX01	0109	0330								
OPEN	0101	0235								
PARTIL	0150	0305								

```
PRB      0134      0234
PRBA     0082      0234  0238  0324  0400
PRBFA    0141      0307  0398
PRBLN    0142
PRCNT    0143      0399
PRIOLN   0138
PRIOP    0137      0237  0244  0323
PRSFG    0139      0236  0240  0326  0403
PRUFG    0140
R0       0073      0179  0195  0196  0197  0200  0201  0203  0235  0236
                   0237  0240  0243  0244  0251  0252  0303  0304  0305
                   0313  0314  0322  0323  0326  0361  0364  0366  0463
                   0468  0476  0480  0483
R1       0075      0180  0198  0211  0212  0213  0214  0215  0216  0217
                   0218  0219  0220  0225  0227  0289  0361  0362  0363
                   0365  0366  0367  0368  0369  0370  0371  0372  0373
                   0374  0403  0429  0430  0431  0432  0433  0434  0435
                   0464  0465  0472  0476  0477  0482  0483  0484  0485
R10      0085      0157  0166  0176  0202  0238  0324  0330  0339  0400
R11      0086
R13      0087
RETBUF             0115
RETWSP             0114
RR                 0117  0338
ST       0078      0171  0173  0179  0201  0251  0266  0271  0430
SVC      0135
SVCALT             0118  0239  0325  0401
TRPLT    0360      0183  0199  0290  0486
WRITE    0397      0248  0295  0310  0316
WRTA     0103      0243

THERE ARE   0076  SYMBOLS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																													
SHEET																													

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN — DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, ABSLD, ABSOLUTE LOADER-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945389-9901

SCALE — REV. — SHEET 1 of 19

TI-0023-C

```
0003                *   PROCEDURE:ROMBUT
0004                *   GETBIT = A(GETBIT)
0005                *   CALL ABSLDR(>A0)
0006                *   IF(ERROR)CALL SPIN(S)
0007                *   IF(PROG EP.NE.0)
0008                *      THEN
0009                *         GO TO C(EP)
0010                *      ELSE
0011                *         GO TO FRONT PANEL
0012                *   ENDIF
0013                *   END ROMBOOT
0014                *   SUBROUTINE ABSLDR
0015                *   DO UNTIL END OF MODULE
0016                *   REPEAT CNT=0
0017                *   CKSM=OLD CKSM
0018                *      CALL GETBIT(VALUE,3,CLR)
0019                *      DO CASE
0020                *         C1:DO VALUE.EQ.XTNDED TAG
0021                *            CALL GETBIT(VALUE,4,CLR)
0022                *            DO CASE
0023                *               C1:DO VALUE.EQ.PGM NAME
0024                *                  IF(PARM ADDR.NE.0)
0025                *                     PARM PGN=PARM ADDR+PHMDSPL
0026                *                  LIMIT=23
0027                *                  CALL GETBIT(VALUE,7,CLR)
0028                *                  NMBR CHAR=AND(VALUE,>3F)
0029                *                  DO IF PARM ADDR.NE.0
0030                *                     C(PARM PGN)=NMBR CHAR
0031                *                     PARM PGN=PARM PGN+1
0032                *                     DO WHILE(NMBR CHAR.GE.0).AND
0033                *                              (LIMIT.GE.0)
0034                *                        CALL GETBIT(VALUE,7,CLR)
0035                *                        C(PARM PGN)=VALUE;
0036                *                        PARM PGN=PARM PGN+1;
0037                *                        LIMIT=LIMIT-1;
0038                *                     END DO;
0039                *                     DO WHILE NMBR CHAR.GE.0
0040                *                        CALL GETBIT(VALUE,7,CLR)
0041                *                        NMBR CHAR=NMBR CHAR-1
0042                *                     END DO;
0043                *                  END DO;
0044                *               END C1;
0045                *               C2:DO VALUE.EQ.ETRY POINT
0046                *                  CALL GET19(VALUE)
0047                *                  EP=VALUE
0048                *                  IF(PARM ADDR.NE.0)
0049                *                    C(PARM ADDR)=EP
0050                *               END C2
0051                *               C3:DO VALUE.EQ.CHECKSUM
0052                *                  TEMP=CKSM
0053                *                  CALL GET19,VALUE
0054                *                  IF(NOT(TEMP.EQ.VALUE).AND.
0055                *                     NOT(SHC(TEMP,8).EQ.VALUE)
0056                *                     ERROR RETURN
```

```
0057                *                  END IF
0058                *                END C3
0059                *                C4:DO VALUE.EQ.END OF MODULE
0060                *                   RETURN
0061                *                END C4
0062                *                C5:DO VALUE.EQ.REPEAT TRIPLET
0063                *                   CALL GETBIT(VALUE,7,CLR)
0064                *                   REPEAT COUNT=AND(VALUE,>3F)
0065                *                   CALL GET16(VALUE)
0066                *                   DO WHILE REPEAT COUNT.NE.0
0067                *                      C(CURR ADDR)=VALUE
0068                *                      CURR ADDR=CURR ADD+2
0069                *                      REPEAT COUNT=REPEAT COUNT-1
0070                *                   END DO
0071                *                END C5
0072                *                      ***NOTE: IN THE INTEREST OF SPACE,
0073                *                      ***       THE CODE FOR THE ABOVE
0074                *                      ***       IS INCLUDED IN THE
0075                *                      ***       PROCESSING FOR ABSOLUTE
0076                *                      ***       DATA TRIPLETS.
0077                *             END CASE
0078                *          C2:DO VALUE,EQ,ABS WORD
0079                *             CALL GET16(VALUE)
0080                *             DO WHILE REPEAT COUNT,GT,0
0081                *                C(CURR ADDR)=VALUE
0082                *                CURR ADDR=CUR ADDR+2
0083                *                REPEAT COUNT=REPEAT COUNT-1
0084                *             END DO
0085                *          END C2
0086                *          C3:DO VALUE,EQ,ABS BYTE
0087                *             CALL GETBIT(VALUE,5,CLR)
0088                *             VALUE=SRL(VALUE,1)
0089                *             CALL GETBIT(VALUE,6,OR)
0090                *             C(CURR ADDR)=SHL(VALUE,2)
0091                *             CUR ADDR=CURR ADDR+1
0092                *          END C3
0093                *          C4:DO VALUE,EQ,LOAD ADDR
0094                *             CALL GET16(VALUE)
0095                *             IF(PARM ADDR,NE,0)
0096                *             CUR,ADDR=VALUE
0097                *               C(PARM ADDR+LDPT)=VALUE
0098                *          END C4
0099                *          C5:DO VALUE,LT,>4
0100                *             ERROR RETURN
0101                *    END ABSLDR
0102                *    SUBROUTINE GET16
0103                *    END GET16
0104                *    SUBROUTINE GETBIT
0105                *    ***START INPUT STREAM***
0106                *    CRUBASE=100
0107                *     BIT,COUNT=0
0108                *    CKSM=0
0109                *    INBITS=0
0110                *    INB=0
0111                *    RE ENTRY=A(GETB2)
```

```
0112                *      ***NEWRECORD***
0113                *       DO WHILE CHAR.EQ.LF
0114                *          CHAR=IN733
0115                *          END DO
0116                *    IF(CHAR,EQ,DEL)CHAR=IN733
0117                *    *****GETCHAR*****
0118                *    CHAR = IN733
0119                *    IF CHAR,EQ,'CR'  NEWRECORD
0120                *    CKSM=CKSM,XOR,CHAR                3
0121                *    CBS=CNC(CBS,CHAR,CBC)
0122                *    CBC=CBC+7
0123                *    ****GETBT2*******
0124                *    IF(MODE,EQ,CLR)RBS =0
0125                *    IF(RBC,GT,CBC)GETCHAR
0126                *    DO WHILE RBC,GT,0
0127                *       CALL SLA(RBS,1)
0128                *       CALL SLA(CBS,1)
0129                *       IF(CARRY)RBS=RBS+1
0130                *       CBC=CBC-1
0131                *       RBC=RBC-1
0132                *    END DO
0133                *    RETURN
0134                *       PROCEDURE GET BIT (MONITOR CONTROL)
0135                *          /* GET BIT PROVIDES IO SUPPORT
0136                *           FOR THE ABSLOADER, THE OPEN
0137                *             HAS BEEN PROCESSED BY THE MAIN
0138                *             DRIVER.
0139                *          */
0140                *          IF CLEAR FLAG=0 THEN
0141                *             RETURN CALUE = 0;
0142                *          DO WHILE REQUEST BIT COUNT>0;
0143                *          IF CURRENT BIT COUNT > 0 THEN DO;
0144                *             CURRENT BIT COUNT = CURRENT BIT COUNT-1;
0145                *             CALL MOVE BIT(RETURN VALUE,CURR BUF ADDR);
0146                *             REQUEST BIT COUNT = REQUEST BIT COUNT -1;
0147                *          END;ELSE
0148                *             IF CHAR COUNT >0 THEN DO;
0149                *                CHAR COUNT=CHAR COUNT -1;
0150                *                CURRENT BIT COUNT = 7;
0151                *                CALL SLA(CURR BUFF ADDR,CHAR,1);
0152                *                CKSM = SWPB(
0153                *                CHECKSUM = SWPB(CHECKSUM,XOR(CHECKSUM,
0154                *                   CURR BUFF ADDR,CHAR))
0155                *                CURR BUFF ADDR = CURR BUFF ADDR+1;
0156                *             END;ELSE D0;
0157                *                LOAD PRB OPCODE = 'READ';
0158                *                CALL SVC(SVC CALL BLOCK);
0159                *                CURR BUFF ADDR = LOADER BUFFER ADDR;
0160                *             END;
0161                *          END;
0162                *       END GTBITM;
0163                       IDT  'ABSLD'
0164                * TITLE:    'ABSLD'
0165                *           ABSOLUTE LOADER
0166                * REVISION:
```

```
0167                *              ORIGINAL
0168                * COMPUTER:    990,ASM
0169                * ABSTRACT:    CALLED BY THE MONITOR AS AN OVERLAY TO
0170                *              PERFORM LOADS OF ABSOLUTE CODE MODULES.
0171                *              SET CALL+2 TO ZERO TO INHIBIT RETURN PARAMETERS
0172                *              ELSE, CALL+2 IS ASSUMED TO POINT TO A 16 WORD
0173                *              TABLE AT WHICH THE FOLLOWING PARAMETERS WILL
0174                *              BE STORED IF ENCOUNTERED DURING THE LOAD
0175                *                    LOC(BYTE) DESCRIPTION
0176                *                       0,1    PROGRAM ENTRY POINT(IF NON-ZERO
0177                *                       2,3    PROGRAM LOAD POINT
0178                *                       4      #OF CH. IN PROGRAM NAME(1-28)
0179                *                       5-(4)  PROGRAM NAME
0180                * CALLING SEQUENCE:
0181                *              BL   @ABSLDR
0182                *              DATA PARMTABLE
0183                *              JMP  ERROR
0184                *              -
0185                *              -
0186                *              *** GETBIT PARAMETERS ***
0187                       DEF  ABSLDR
0188                       DEF ABSBUT
0189                       REF  SVCALT
0190                       REF  LDCBBC
0191                       REF  LDCBA
0192                       REF  LDCC
0193                       REF  LDOPCD
0194                       REF  LDPRB
0195                       REF  LDADDR
0196        0000   CLR    EQU  0
0197        FFFF   OR     EQU  -1
0198                *
0199                *              ***733 ASR CONTROL & DATA EQUATES ***
0200                *
0201        0009   DTR    EQU  9
0202        000A   RTS    EQU  >A
0203        000B   CLRWRQ EQU  >B
0204        000C   CLRRRQ EQU  >C                 CRU OUT READ REQUEST
0205        000C   RRQ    EQU  >C                 CRU IN  READ REQUEST
0206        1100   DC1    EQU  >1100              READER "ON" COMMAND
0207        7F00   DEL    EQU  >7F00              <DELETE>
0208        0A00   LF     EQU  >0A00              <LINE FEED>
0209        0D00   CR     EQU  >0D00              <CARRIAGE RETURN>
0210                *
0211                *              ***RETURN PARM TABLE DISPLACEMENTS
0212                *
0213        0000   ENTRY  EQU  0
0214        0002   LOADAD EQU  2
0215        0004   PRGNM  EQU  4
0216                *
0217                *                 ***REGISTER ALLOCATION
0218                *
```

```
0220  0000  R0    EQU  0    TEMP(SHIFT COUNT & INDEXING)
0221  0001  R1    EQU  1    ADDRESS OF«GETBIT»
0222  0002  R2    EQU  2    GETBIT=BIT STRING ON HAND
0223  0003  R3    EQU  3    GETBIT=# OF BITS IN R2
0224  0004  R4    EQU  4    GET16 TEMP(RTRN ADDR SAVE)
0225  0005  R5    EQU  5    CURRENT LOAD ADDRESS
0226  0006  R6    EQU  6    REPEAT COUNT:NMBR CHAR
0227  0007  R7    EQU  7    «LIMIT»
0228  0008  R8    EQU  8    TEMP
0229  0009  R9    EQU  9    ENTRY POINT OF LOADED PROGRAM
0230  000A  R10   EQU  10   «VALUE»
0231  000B  R11   EQU  11   AUTO SUBR. RETURN LINKAGE
0232  000C  R12   EQU  12   CRUBASE
0233  000D  R13   EQU  13   TEMP
0234  000E  R14   EQU  14   «CKSM»
0235  000F  R15   EQU  15   ABSLDR RETURN ADDRESS
0236        *
0237        *          ***ABSOLUTE DATA FORMAT DEFINITIONS
0238        *
0239  0002  RLA   EQU  2    RELOC,ADDR(UNSUPPORTED)
0240  0003  RDW   EQU  3    RELOC,WORD(UNSUPPORTED)
0241  0004  ABWD  EQU  4    ABSOLUTE DATA WORD(16BITS)
0242  0005  ABYT  EQU  5    ABSOLUTE DATA BYTE(8 BITS)
0243  0006  ALDA  EQU  6    ABOLSUTE LOAD ADDRESS
0244  0007  XTF   EQU  7    EXTENDED TAG
0245        *
0246  0000  EOM   EQU  0    END OF MODULE
0247  0001  PGN   EQU  1    PROGRAM NAME
0248  0002  PGL   EQU  2    PROGRAM LENGTH(UNSUPPORTED)
0249  0003  AEP   EQU  3    ABSOLUTE ENTRY POINT
0250  0004  REA   EQU  4    RELOC. ENTRY ADDR(UNSUPPORTED
0251  0005  CKSM  EQU  5    CHECKSUM
0252  0006  RPT   EQU  6    REPEAT COUNT
```

```
0254                    *                              0-PROCEDURE:ROMBUT
0255          0000'     ABSBUT EQU  $
0256                    *                              1-GETBIT = A(GETBIT)
0257  0000    0201            LI    R1,GETBIT          ADDR. OF GETBIT
      0002    0136'
0258                    *                              1-CALL ABSLDR(>A0)
0259  0004    06A0            BL    @AB00
      0006    001A'
0260  0008    0000            DATA  0
0261                    *                              1-IF(ERROR)CALL SPIN(3)
0262  000A    10FF            JMP   $                  RETURNS HERE ON ABORT
0263                    *                              1-IF(PROG EP,NE,0)
0264  000C    C249            MOV   R9,R9
0265  000E    1301            JEQ   BOOT1
0266                    *                              2-THEN
0267                    *                              3-GO TO C(EP)
0268  0010    0459            B     *R9                EXECUTE PROGRAM
0269                    *                              2-ELSE
0270                    *                              3-GO TO FRONT PANEL
0271  0012    0420    BOOT1   BLWP  @8                 NO EP FOUND...TRAP TO PANEL
      0014    0008
0272                    *                              1-ENDIF
0273                    *                              0-END ROMBOOT
0274                    *                              1-SUBROUTINE ABSLDR
0275          0016'     ABSLDR EQU  $                ***ENTRY POINT
0276  0016    0201            LI    R1,GTBITM
      0018    01A8'
0277          001A'     AB00    EQU  $
0278  001A    C3CB            MOV   R11,R15            SAVE RETURN ADDR
0279  001C    04C9            CLR   R9                 CLEAR ENTRY POINT
0280  001E    04E0            CLR   @LDCBBC
      0020    0000
0281  0022    04C3            CLR   R3                 CLEAR BIT COUNT
0282  0024    04CE            CLR   R14                CLEAR CHECKSUM
0283  0026    C21F            MOV   *R15,R8
0284  0028    1302            JEQ   AB01
0285  002A    04E8            CLR   @PRGNM(R8)
      002C    0004
0286                    *                              1-DO UNTIL END OF MODULE
0287          002E'     AB01    EQU  $
0288                    *                              1-REPEAT CNT=0
0289  002E    04C6            CLR   R6
0290                    *                              1-CKSM=OLD CKSM
0291  0030    C20E            MOV   R14,R8             WILL GET CLOBBERED IF TAG
0292                    *                                  IS NOT CKSM, BUT DOESNT
0293                    *                                   MATTER IN THIS CASE
0294                    *                              2-CALL GETBIT(VALUE,3,CLR)
0295                    *                              2-DO CASE
0296                    *                              3-C1:DO VALUE.EQ.XTNDED TAG
0297  0032    0691            BL    *R1                GET A TAG IN R10
0298  0034      00            BYTE  CLR,3
      0035      03
0299  0036    028A            CI    R10,XTF            CHECK FOR EXTENDED TAG
      0038    0007
```

```
0300  003A  164C          JNE   AB13                  JUMP FNOT
0301                *                                 4=CALL GETBIT(VALUE,4,CLR)
0302  003C  0691          BL    *R1
0303  003E    00          BYTE  CLR,4
      003F    04
0304                *                                 4=DO CASE
0305                *                                 5=C1;DO VALUE.EQ.PGM NAME
0306  0040  028A          CI    R10,PGN
      0042  0001
0307  0044  1620          JNE   AB07
0308                *                                 6=IF(PARM ADDR,NE,0)
0309                *                                 7=PARM PGN=PARM ADDR+PHMDSPL
0310  0046  C21F          MOV   *R15,R8
0311  0048  1302          JEQ   AB02
0312  004A  0228          AI    R8,PRGNM
      004C  0004
0313        004E' AB02    EQU   $
0314                *                                 6=LIMIT=23
0315  004E  0207          LI    R7,23
      0050  0017
0316                *                                 6=CALL GETBIT(VALUE,7,CLR)
0317  0052  0691          BL    *R1
0318  0054    00          BYTE  CLR,7
      0055    07
0319                *                                 6=NMBR CHAR=AND(VALUE,>3F)
0320  0056  024A          ANDI  R10,>3F
      0058  003F
0321  005A  C18A          MOV   R10,R6
0322                *                                 6=DO IF PARM ADDR,NE,0
0323  005C  C208          MOV   R8,R8
0324  005E  1310          JEQ   AB05
0325                *                                 7=C(PARM PGN)=NMBR CHAR
0326  0060  81CA        C       R10,R7
0327  0062  1201          JLE   AB020
0328  0064  C287          MOV   R7,R10
0329        0066' AB020   EQU   $
0330  0066  06CA          SWPB  R10
0331  0068  DE0A          MOVB  R10,*R8+
0332                *                                 7=PARM PGN=PARM PGN+1
0333                *                                 7=DO WHILE(NMBR CHAR,GE,0),AND
0334                *                                 7=         (LIMIT,GE,0)
0335        006A' AB03    EQU   $
0336  006A  0606          DEC   R6
0337  006C  110B          JLT   AB06
0338                *                                 8=CALL GETBIT(VALUE,7,CLR)
0339  006E  0691          BL    *R1
0340  0070    00          BYTE  CLR,7
      0071    07
0341                *                                 8=C(PARM PGN)=VALUE;
0342                *                               4448=PARM PGN=PARM PGN+1;
0343  0072  06CA          SWPB  R10
0344  0074  DE0A          MOVB  R10,*R8+
0345                *                                 8=LIMIT=LIMIT-1;
0346  0076  0607          DEC   R7
0347  0078  18F8          JH    AB03
```

```
0348  007A  1202          JLE   AB05
0349                *                                  7-END DO;
0350                *                                  7-DO WHILE NMBR CHAR.GE.0
0351        007C' AB04    EQU   $
0352                *                                  8-CALL GETBIT(VALUE,7,CLR)
0353  007C  0691          BL    *R1
0354  007E    00          BYTE  CLR,7
      007F    07
0355                *                                  8-NMBR CHAR=NMBR CHAR-1
0356        0080' AB05    EQU   $
0357  0080  0606          DEC   R6
0358  0082  14FC          JHE   AB04
0359                *                                  7-END DO;
0360                *                                  6-END DO;
0361                *                                  5-END C1;
0362        0084' AB06    EQU   $
0363  0084  10D4          JMP   AB01
0364                AB07
0365                *                                  5-C2:DO VALUE.EQ.ETRY POINT
0366  0086  028A          CI    R10,AEP
      0088  0003
0367  008A  1607          JNE   AB08
0368                *                                  6-CALL GET19(VALUE)
0369  008C  06A0          BL    @GET19
      008E  0114'
0370                *                                  6-EP=VALUE
0371  0090  C24A          MOV   R10,R9
0372                *                                  6-IF(PARM ADDR,NE,0)
0373                *                                  6-   C(PARM ADDR)=EP
0374  0092  C21F          MOV   *R15,R8
0375  0094  13CC          JEQ   AB01
0376  0096  C609          MOV   R9,*R8
0377                *                                  5-END C2
0378  0098  10CA          JMP   AB01
0379        009A' AB08    EQU   $
0380                *                                  5-C3:DO VALUE.EQ.CHECKSUM
0381  009A  028A          CI    R10,CKSM
      009C  0005
0382  009E  160A          JNE   AB10
0383                *                                  6-TEMP=CKSM
0384  00A0  C20E          MOV   R14,R8
0385                *                                  6-CALL GET19,VALUE
0386  00A2  06A0          BL    @GET19
      00A4  0114'
0387                *                                  **NOTE:SINCE #OF BYTES SINCE
0388                *                                    LAST CKSM MAY BE EITHER EVEN
0389                *                                    OR ODD,(AND WE DON'T KNOW
0390                *                                    WHICH)..WE'LL SETTLE FOR A
0391                *                                    MATCH-UP ANY WAY WE CAN MAKE
0392                *                                    IT HAPPEN
0393                *                                  6-IF(NOT(TEMP.EQ.VALUE).AND.
0394                *                                  6-   NOT(SHC(TEMP,8).EQ.VALUE)
0395                *                                  6-   ERROR RETURN
0396  00A6  04CE          CLR   R14                     CLEAR CHECKSUM FOR NEXT TIME
0397  00A8  8288          C     R8,R10
```

```
0398  00AA  13C1          JEQ   AB01              CKSM OK-PROCESS NEXT DATA
0399  00AC  06C8          SWPB  R8
0400  00AE  8288          C     R8,R10            CKSM OK-PRCESS NEXT DATA
0401  00B0  13BE          JEQ   AB01
0402  00B2  1004          JMP   AB11              BAD CHECKSUM,,ERROR EXIT
0403                *                             6-END IF
0404                *                             5-END C3
0405        00B4' AB10    EQU   $
0406                *                             5-C4:DO VALUE.EQ.END OF MODULE
0407  00B4  028A          CI    R10,EOM
      00B6  0000
0408  00B8  1603          JNE   AB12
0409                *                             6-RETURN
0410  00BA  05CF          INCT  R15               SUCCESSFUL LOAD
0411        00BC' AB11    EQU   $                 HERE FOR ERROR
0412  00BC  05CF          INCT  R15
0413  00BE  045F          B     *R15
0414                *                             5-END C4
0415        00C0' AB12    EQU   $
0416                *                             5-C5:DO VALUE.EQ.REPEAT TRIPLE
0417  00C0  028A          CI    R10,RPT
      00C2  0006
0418  00C4  16F8          JNE   AB11              UNSUPPORTED EXTENDED TAG
0419                *                             6-CALL GETBIT(VALUE,7,CLR)
0420  00C6  0691          BL    *R1
0421  00C8    00          BYTE  CLR,7
      00C9    07
0422                *                             6-REPEAT COUNT-AND(VALUE,>3F)
0423  00CA  C18A          MOV   R10,R6
0424  00CC  0246          ANDI  R6,>3F
      00CE  003F
0425                *                             6-CALL GET16(VALUE)
0426                *                             6-DO WHILE REPEAT COUNT,NE,0
0427                *                             7-C(CURR ADDR)-VALUE
0428                *                             7-CURR ADDR-CURR ADD+2
0429                *                             7-REPEAT COUNT-REPEAT COUNT-1
0430                *                             6-END DO
0431                *                             5-END C5
0432                *                             7-***NOTE: IN THE INTEREST OF
0433                *                             7-***       THE CODE FOR THE AB
0434                *                             7-***       IS INCLUDED IN THE
0435                *                             7-***       PROCESSING FOR ABSO
0436                *                             7-***       DATA TRIPLETS.
0437  00D0  0691          BL    *R1
0438  00D2    00          BYTE  CLR,3
      00D3    03
0439                *                             4-END CASE
0440        00D4' AB13    EQU   $
0441                *                             3-C2:DO VALUE.EQ.ABS WORD
0442  00D4  028A          CI    R10,ABWD
      00D6  0004
0443  00D8  1606          JNE   AB15
0444                *                             4-CALL GET16(VALUE)
0445  00DA  06A0          BL    @GET16
      00DC  0122'
```

```
0446                    *                       4=DO WHILE REPEAT COUNT.GT.0
0447          00DE' AB14    EQU   $
0448                    *                       5=C(CURR ADDR)=VALUE
0449                    *                       5=CURR ADDR=CUR ADDR+2
0450  00DE    CD4A            MOV   R10,*R5+
0451                    *                       5=REPEAT COUNT=REPEAT COUNT-1
0452  00E0    0606            DEC   R6
0453  00E2    15FD            JGT   AB14
0454  00E4    10A4            JMP   AB01
0455                    *                       4=END DO
0456                    *                       3=END C2
0457          00E6' AB15    EQU   $
0458                    *                       3=C3:DO VALUE.EQ.ABS BYTE
0459  00E6    028A            CI    R10,ABYT
      00E8    0005
0460  00EA    1608            JNE   AB16
0461                    *                       4=CALL GETBIT(VALUE,5,CLR)
0462  00EC    0691            BL    *R1
0463  00EE      00            BYTE  CLR,5
      00EF      05
0464                    *                       4=VALUE=SRL(VALUE,1)
0465  00F0    091A            SRL   R10,1
0466                    *                       4=CALL GETBIT(VALUE,6,OR)
0467  00F2    0691            BL    *R1
0468  00F4      FF            BYTE  OR,6
      00F5      06
0469                    *                       4=C(CURR ADDR)=SHL(VALUE,2)
0470                    *                       4=CUR ADDR=CURR ADDR+1
0471  00F6    0A6A            SLA   R10,6        EQIV TO SL 2 + SWPB
0472  00F8    DD4A            MOVB  R10,*R5+
0473                    *                       3=END C3
0474  00FA    1099            JMP   AB01
0475          00FC' AB16    EQU   $
0476                    *                       3=C4:DO VALUE.EQ.LOAD ADDR
0477  00FC    028A            CI    R10,ALDA
      00FE    0006
0478  0100    1608            JNE   AB17
0479                    *                       4=CALL GET16(VALUE)
0480  0102    06A0            BL    @GET16
      0104    0122'
0481                    *                       4=IF(PARM ADDR.NE.0)
0482                    *                       4=CUR.ADDR=VALUE
0483  0106    C14A            MOV   R10,R5
0484                    *                       4=  C(PARM ADDR+LDPT)=VALUE
0485  0108    C21F            MOV   *R15,R8
0486  010A    1391            JEQ   AB01
0487  010C    05C8            INCT  R8
0488  010E    C60A            MOV   R10,*R8
0489                    *                       3=END C4
0490  0110    108E            JMP   AB01
0491                    *                       3=C5:DO VALUE.LT.>4
0492                    *                       4=ERROR RETURN
0493          0112' AB17    EQU   $
0494  0112    10D4            JMP   AB11         IF VALUE.GE.4, WE WOULDNT
0495                    *                          BE HERE
```

```
0496                *                    1-END ABSLDR        3
```

```
0499                    *                               1-SUBROUTINE GET16
0500          0114' GET19   EQU   $
0501   0114   C10B          MOV   R11,R4
0502   0116   0691          BL    *R1
0503   0118     00          BYTE  0,3
       0119     03
0504   011A   028A          CI    R10,ABWD
       011C   0004
0505   011E   1302          JEQ   GET19A
0506   0120   10CD          JMP   AB11
0507          0122' GET16   EQU   $                 ***ENTRY POINT***
0508   0122   C10B          MOV   R11,R4              SAVE RETURN
0509          0124' GET19A  EQU   $
0510   0124   0691          BL    *R1
0511   0126     00          BYTE  0,5
       0127     05
0512   0128   091A          SRL   R10,1               DISCARD TAG FROM NEXT CH.
0513          012A' GET12A  EQU   $
0514   012A   0691          BL    *R1                   & OR IN NEXT 7 BITS
0515   012C     FF          BYTE  -1,7
       012D     07
0516   012E   091A          SRL   R10,1               DISCARD TAG FROM 3RD BYTE
0517   0130   0691          BL    *R1                   & OR IN NEXT 6 BITS
0518   0132     FF          BYTE  -1,6
       0133     06
0519   0134   0454          B     *R4                 RETURN TO CALLER
0520                    *                             1-END GET16
```

```
0523                    *                       1=SUBROUTINE GETBIT
0524          0136'     GETBIT EQU  $           ***ENTRY POINT****
0525                    *
0526                    *                        INITIALIZE GETBIT VARIABLES
0527                    *                       1=****START INPUT STREAM***
0528                    *                       1=CRUBASE=100
0529  0136  020C              LI    R12,>100
      0138  0100
0530                    *                       1= BIT.COUNT=0
0531  013A  1D09              SBO   DTR
0532  013C  1D0A              SBO   RTS
0533  013E  1D0B              SBO   CLRWRQ
0534  0140  1D0C              SBO   CLRRRQ
0535  0142  0202              LI    R2,DC1
      0144  1100
0536  0146  3202              LDCR  R2,8
0537                    *                       1=CKSM=0
0538                    *                       1=INBITS=0
0539                    *                       1=INB=0
0540  0148  388E              MPY   R14,R2
0541                    *                       1=RE ENTRY=A(GETB2)
0542  014A  0201              LI    R1,GETBT2
      014C  0188'
0543          014E'     GB01  EQU   $
0544                    *                       1=  ***NEWRECORD***
0545                    *                       2=DO WHILE CHAR.EQ.LF
0546  014E  1E0C              SBZ   CLRRRQ         CLEAR READ REQUEST
0547          0150'     GB02  EQU   $
0548  0150  1F0C              TB    RRQ
0549  0152  16FE              JNE   GB02           WAIT FOR READY
0550                    *                       3=CHAR=IN733
0551  0154  04C0              CLR   R0
0552  0156  35C0              STCR  R0,7
0553  0158  0280              CI    R0,LF
      015A  0A00
0554  015C  13F8              JEQ   GB01
0555                    *                       3=END DO
0556                    *                       1=IF(CHAR.EQ.DEL)CHAR=IN733
0557  015E  0280              CI    R0,DEL
      0160  7F00
0558  0162  1605              JNE   GB06
0559                    *                       1=******GETCHAR*****
0560          0164'     GB04  EQU   $            HERE TO GET ANOTHER CHARACTER
0561  0164  1E0C              SBZ   CLRRRQ         CLEAR READ REQUEST
0562          0166'     GB05  EQU   $
0563  0166  1F0C              TB    RRQ
0564  0168  16FE              JNE   GB05           WAIT FOR CHARACTER
0565                    *                       1=CHAR = IN733
0566  016A  04C0              CLR   R0            GET THE CHAR.
0567  016C  35C0              STCR  R0,7            FROM THE CRU
0568                    *                       1=IF CHAR.EQ.'CR'  NEWRECORD
0569          016E'     GB06  EQU   $            CHECK FOR END OF RECORD
0570  016E  0280              CI    R0,CR        IF EOR, GO START ANOTHER
      0170  0D00
```

```
0571  0172  13ED          JEQ   GB01                 RECORD
0572                *                              1-CKSM=CKSM,XOR,CHAR
0573  0174  2B80          XOR   R0,R14
0574  0176  06CE          SWPB  R14
0575                *                              1-CBS=CNC(CBS,CHAR,CBC)
0576  0178  0A10          SLA   R0,1                 LEFT JUSTIFY THE # DATA BITS
0577  017A  C0C3          MOV   R3,R3                AVOID 16 BIT SHIFT
0578  017C  1302          JEQ   GB06A
0579  017E  A003          A     R3,R0                NON,ZERO SH COUNT IN R0
0580  0180  0900          SRL   R0,R0                SHIFT CHAR BY NO BITS ON HAND
0581        0182' GB06A   EQU   $
0582  0182  E080          SOC   R0,R2                OR IN NEW BITS
0583                *                              1-CBC=CBC+7
0584  0184  0223          AI    R3,7                 UPDATE CURR BIT COUNT
      0186  0007
0585                *                              1-****GETBT2*******
0586        0188' GETBT2  EQU   $                  ***SECONDARY ENTRY POINT
0587
0588                *                              1-IF(MODE,EQ,CLR)RBS =0
0589  0188  C01B          MOV   *R11,R0              GET PARAMETERS
0590  018A  D000          MOVB  R0,R0
0591  018C  1601          JNE   GB07                 JUMP IF R10 HAS GOOD DATA
0592  018E  04CA          CLR   R10                     ELSE, CLEAR OUT TRASH
0593
0594        0190' GB07    EQU   $
0595
0596                *                              1-IF(RBC,GT,CBC)GETCHAR
0597  0190  5000          SZCB  R0,R0                CLEAR THE MODE FLAG
0598  0192  80C0          C     R0,R3                CMPR RQST TO BITS ON HAND
0599  0194  1BE7          JH    GB04                    JUMP TO GET MORE BITS
0600
0601                *                              1-DO WHILE RBC,GT,0
0602        0196' GB08    EQU   $
0603
0604                *                              2-CALL SLA(RBS,1)
0605  0196  0A1A          SLA   R10,1
0606                *                              2-CALL SLA(CBS,1)
0607  0198  0A12          SLA   R2,1
0608                *                              2-IF(CARRY)RBS=RBS+1
0609  019A  1701          JNC   GB09
0610  019C  058A          INC   R10
0611        019E' GB09    EQU   $
0612                *                              2-CBC=CBC-1
0613  019E  0603          DEC   R3
0614                *                              2-RBC=RBC-1
0615  01A0  0600          DEC   R0                   DECR REQUEST COUNT AND
0616                *                              1-END DO
0617  01A2  1BF9          JH    GB08                    REPEAT IF NOT DONE
0618                *                              0-RETURN
0619  01A4  05CB          INCT  R11                  INCREMENT RETURN ADDRESS
0620  01A6  045B          RT
```

```
0622                    *                                   2=PROCEDURE GET BIT (MONITOR C
0623                    *                                   3=/* GET BIT PROVIDES IO SUPP
0624                    *                                   3= FOR THE ABSLOADER. THE OPEN
0625                    *                                   3=   HAS BEEN PROCESSED BY THE
0626                    *                                   3=   DRIVER.
0627                    *                                   3=*/
0628                    *
0629          01A8' GTBITM EQU   $
0630                    *           MAY USE R0,R2,R3  THIS SUBROUTINE IS
0631                    *           NON-STANDARD.
0632                    *                                   3=IF CLEAR FLAG=0 THEN
0633                    *                                   4=RETURN CALUE = 0;
0634  01A8  D03B              MOVB *R11+,R0                  CALL+1= CLEAR FLAG.
0635  01AA  1601              JNE  LDI010
0636  01AC  04CA              CLR  R10
0637          01AE' LDI010 EQU   $
0638                    *                                   3=DO WHILE REQUEST BIT COUNT>0
0639  01AE  D0FB              MOVB *R11+,R3                  CALL+2 = REQ BIT COUNT
0640  01B0  0983              SRL  R3,8
0641          01B2' LDI020 EQU   $
0642                    *                                   3=IF CURRENT BIT COUNT > 0 THE
0643  01B2  0620              DEC  @LDCBBC
      01B4  0020'
0644  01B6  110A              JLT  LDI040
0645                    *                                   4=CURRENT BIT COUNT = CURRENT
0646                    *                                   4=CALL MOVE BIT(RETURN VALUE,C
0647  01B8  C020              MOV  @LDCBA,R0
      01BA  0000
0648  01BC  0A1A              SLA  R10,1
0649  01BE  D090              MOVB *R0,R2
0650  01C0  0A12              SLA  R2,1
0651  01C2  1701              JNC  LDI030
0652  01C4  058A              INC  R10
0653          01C6' LDI030 EQU   $
0654  01C6  D402              MOVB R2,*R0
0655                    *                                   4=REQUEST BIT COUNT = REQUEST
0656  01C8  0603              DEC  R3
0657                    *                                   3=END;ELSE
0658  01CA  1021              JMP  LDI060
0659          01CC' LDI040 EQU   $
0660                    *                                   4=IF CHAR COUNT >0 THEN DO;
0661  01CC  0620              DEC  @LDCC
      01CE  0000
0662  01D0  110F              JLT  LDI050
0663                    *                                   5=CHAR COUNT=CHAR COUNT -1;
0664                    *                                   5=CURRENT BIT COUNT = 7;
0665  01D2  0200              LI   R0,7
      01D4  0007
0666  01D6  C800              MOV  R0,@LDCBBC
      01D8  01B4'
0667                    *                                   5=CALL SLA(CURR BUFF ADDR,CHAR
0668                    *                                   5=CKSM = SWPB(
0669                    *                                   5=CHECKSUM = SWPB(CHECKSUM,XOR
0670                    *                                   6=CURR BUFF ADDR,CHAR))
```

```
0671                    *                                5=CURR BUFF ADDR = CURR BUFF A
0672  01DA  05A0               INC   @LDCBA
      01DC  01BA!
0673  01DE  C020               MOV   @LDCBA,R0
      01E0  01DC!
0674  01E2  06CE               SWPB  R14
0675  01E4  D090               MOVB  *R0,R2
0676  01E6  0982               SRL   R2,8
0677  01E8  2B82               XOR   R2,R14
0678  01EA  0A92               SLA   R2,9                  DISCARD HIGH ORDER BIT
0679  01EC  D402               MOVB  R2,*R0
0680                    *                                4=END;ELSE DO;
0681  01EE  100F               JMP   LDI060
0682        01F0! LDI050 EQU   $
0683                    *                                5=LOAD PRB OPCODE = 'READ';
0684  01F0  0200             LI    R0,>0900
      01F2  0900
0685  01F4  D800               MOVB  R0,@LDOPCD
      01F6  0000
0686  01F8  C00A               MOV   R10,R0
0687                    *                                5=CALL SVC(SVC CALL BLOCK);
0688  01FA  020A               LI    R10,LDPRB
      01FC  0000
0689  01FE  0420               BLWP  @SVCALT
      0200  0000
0690  0202  C280               MOV   R0,R10
0691                    *                                5=CURR BUFF ADDR = LOADER BUFF
0692  0204  C820               MOV   @LDADDR,@LDCBA
      0206  0000
      0208  01E0!
0693  020A  0620               DEC   @LDCBA
      020C  0208!
0694                    *                                4=END;
0695                    *                                3=END;
0696        020E! LDI060 EQU   $
0697  020E  C0C3               MOV   R3,R3
0698  0210  15D0               JGT   LDI020
0699                    *                                2=END GTBITM;
0700  0212  045B               RT                                    4
0701                    END
0000 ERS
```

960 - 980 CONCORDANCE

$		0255	0262	0275	0277	0287	0313	0329	0335	0351
		0356	0362	0379	0405	0411	0415	0440	0447	0457
		0475	0493	0500	0507	0509	0513	0524	0543	0547
		0560	0562	0569	0581	0586	0594	0602	0611	0629
		0637	0641	0653	0659	0682	0696			
AB00	0277	0259								
AB01	0287	0284	0363	0375	0378	0398	0401	0454	0474	0486
		0490								
AB02	0313	0311								
AB020	0329	0327								
AB03	0335	0347								
AB04	0351	0358								
AB05	0356	0324	0348							
AB06	0362	0337								
AB07	0364	0307								
AB08	0379	0367								
AB10	0405	0382								
AB11	0411	0402	0418	0494	0506					
AB12	0415	0408								
AB13	0440	0300								
AB14	0447	0453								
AB15	0457	0443								
AB16	0475	0460								
AB17	0493	0478								
ABSBUT	0255	0188								
ABSLDR	0275	0187								
ABWD	0241	0442	0504							
ABYT	0242	0459								
AEP	0249	0366								
ALDA	0243	0477								
BOOT1	0271	0265								
CKSM	0251	0381								
CLK	0196	0298	0303	0318	0340	0354	0421	0438	0463	
CLRRRQ	0204	0534	0546	0561						
CLRWRQ	0203	0533								
CR	0209	0570								
DC1	0206	0535								
DEL	0207	0557								
DTR	0201	0531								
ENTRY	0213									
EOM	0246	0407								
GB01	0543	0554	0571							
GB02	0547	0549								
GB04	0560	0599								
GB05	0562	0564								
GB06	0569	0558								
GB06A	0581	0578								
GB07	0594	0591								
GB08	0602	0617								
GB09	0611	0609								
GET12A	0513									
GET16	0507	0445	0480							
GET19	0500	0369	0386							
GET19A	0509	0505								
GETBIT	0524	0257								
GETBT2	0586	0542								
GTBITM	0629	0276								
LDADDR		0195	0692							
LDCBA		0191	0647	0672	0673	0692	0693			
LDCBBC		0190	0280	0643	0666					

```
LDCC                0192  0661
LDI010    0637      0635
LDI020    0641      0698
LDI030    0653      0651
LDI040    0659      0644
LDI050    0682      0662
LDI060    0696      0658  0681
LDOPCD              0193  0685
LDPRB               0194  0688
LF        0208      0553
LOADAD    0214
OR        0197      0468
PGL       0248
PGN       0247      0306
PRGNM     0215      0285  0312
R0        0220      0551  0552  0553  0557  0566  0567  0570  0573  0576
                    0579  0580  0580  0582  0589  0590  0590  0597  0597
                    0598  0615  0634  0647  0649  0654  0665  0666  0673
                    0675  0679  0684  0685  0686  0690
R1        0221      0257  0276  0297  0302  0317  0339  0353  0420  0437
                    0462  0467  0502  0510  0514  0517  0542
R10       0230      0299  0306  0320  0321  0326  0328  0330  0331  0343
                    0344  0366  0371  0381  0397  0400  0407  0417  0423
                    0442  0450  0459  0465  0471  0472  0477  0483  0488
                    0504  0512  0516  0592  0605  0610  0636  0648  0652
                    0686  0688  0690
R11       0231      0278  0501  0508  0589  0619  0634  0639
R12       0232      0529
R13       0233
R14       0234      0282  0291  0384  0396  0540  0573  0574  0674  0677
R15       0235      0278  0283  0310  0374  0410  0412  0413  0485
R2        0222      0535  0536  0540  0582  0607  0649  0650  0654  0675
                    0676  0677  0678  0679
R3        0223      0281  0577  0577  0579  0584  0598  0613  0639  0640
                    0656  0697  0697
R4        0224      0501  0508  0519
R5        0225      0450  0472  0483
R6        0226      0289  0321  0336  0357  0423  0424  0452
R7        0227      0315  0326  0328  0346
R8        0228      0283  0285  0291  0310  0312  0323  0323  0331  0344
                    0374  0376  0384  0397  0399  0400  0485  0487  0488
R9        0229      0264  0264  0268  0279  0371  0376
RDW       0240
REA       0250
RLA       0239
RPT       0252      0417
RRQ       0205      0548  0563
RTB       0202      0532
SVCALT              0189  0689
XTF       0244      0299

THERE ARE   0094  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS

LTR	DESCRIPTION	DATE	APPROVED

REV
SHEET

REV STATUS OF SHEETS
REV
SHEET

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, PX9LAL, LINKING LOADER MODULE-PX990

SIZE A
CODE IDENT NO 96214
DRAWING NO 945390-9901
SCALE
REV.
SHEET 1 of 47

TI-0023-C

```
0003                *
0004                * TITLE:     PX9LAL
0005                *            LINKING LOADER
0006                * REVISION:  05/01/74
0007                *            ORIGINAL
0008                *            03/15/76
0009                *            MODIFIED TO BE OVERLAY COMMAND OF PX9MTR
0010                * COMPUTER:  990,990 ASSEMBLY
0011                * ABSTRACT:  PX9LAL IS A VERSION OF LAL990 MODIFIED
0012                *            TO RUN WITH PX9MTR, THE PROTOTYPING SYSTEM
0013                *            AND 733ASR PROGRAM DEVELOPMENT SYSTEM DEBUG
0014                *            MONITOR
0015                *            PX9LAL LOADS OBJECT MODULES INTO MEMORY,
0016                *            PERFORMS THE LINKING DEFINED IN THE
0017                *            PROGRAM MODULES, PERFORMS THE ADDRESS
0018                *            MODIFICATION FOR RELOCATABLE CODE, AND
0019                *            PRINTS A LOAD MAP.
0020                *
```

```
0023                * TITLE:    DATBS
0024                *           DATA BASE
0025                * REVISION: 05/01/74
0026                *           ORIGINAL
0027                *           03/15/76
0028                *           MODIFIED TO RUN WITH PX9MTR
0029                * COMPUTER: 990,990 ASSEMBLY
0030                * ABSTRACT: THIS DATA BASE CONTAINS ALL WORKSPACE
0031                *           REGISTER EQUATES, ERROR MESSAGES, AND
0032                *           MISCELLANEOUS DATA
0033                * CALLING SEQUENCE: NON-CALLABLE
0034                *
0035                      IDT  'PX9LAL'
0036                *
0037                *    EXTERNAL REFS AND DEFS
0038                *
0039                      REF  ERROR,USRPC,INIT,LDBUF,GETHEX,PRCRLF,SVCALT
0040                      DEF  LALCSR
0041                *
0042                *     MAINWP REGISTERS
0043                *
0044          0000  TEMP   EQU  0
0045          0001  RDRBF  EQU  1
0046          0002  ESTPT  EQU  2
0047          0003  SYMP   EQU  3
0048          0004  ENTPT  EQU  4
0049          0005  ENTVAL EQU  5
0050          0006  PRNTN  EQU  6
0051          0007  SPTR   EQU  7
0052          0008  TPPT   EQU  8
0053          0009  UNDPT  EQU  9
0054          000A  IOPARM EQU  10
0055          000C  SFLAGS EQU  12
0056                *
0057                * DSWP REGISTERS
0058                *
0059          0000  INPT   EQU  0
0060          0001  TGFLD  EQU  1
0061          0002  BIAS   EQU  2
0062          0003  NEWLOC EQU  3
0063          0004  LOCPT  EQU  4
0064          0005  SVPT   EQU  5
0065          0006  HLD    EQU  6
0066          0007  FLGVL  EQU  7
0067          0008  PC     EQU  8
0068          000C  LDPT   EQU  12
0069          000D  INCP   EQU  13
0070          000E  INSID  EQU  14
0071          000F  TDAT   EQU  15
0072                *
0073                *     LDWP REGISTERS
0074                *
0075          0000  RD     EQU  0
0076          0001  FST    EQU  1
```

```
0077        0002  PTR     EQU  2
0078        0003  NSMBLP  EQU  3
0079        0004  SYMB    EQU  4
0080        0005  TMP     EQU  5
0081        0006  LOC     EQU  6
0082        0007  SRSM    EQU  7
0083        0008  NMMSG   EQU  8
0084        000C  HEXP2   EQU  12
0085              *
0086              *       CMWP REGISTERS
0087              *
0088        0000  CKL     EQU  0
0089        0001  HVAL    EQU  1
0090        0002  RDRL    EQU  2
0091        0003  CUMPT   EQU  3
0092        0004  ENDOB   EQU  4
0093        0005  HEXP    EQU  5
0094        0006  RLRG    EQU  6
0095        0007  CSUM    EQU  7
0096              *
0097              *       CONVWS REGISTERS
0098              *
0099        0000  BINVAL  EQU  0
0100        0001  VALSV   EQU  1
0101        0002  MADD    EQU  2
0102        0003  SCCODE  EQU  3
0103        0007  BHCODE  EQU  7
0104        0008  HBCODE  EQU  8
0105        0009  SVCAL1  EQU  9
0106        000B  HCNT    EQU  11
0107        000C  PCNT    EQU  12
0108        000E  CHOPC   EQU  14
0109              *
0110              *       SYMBOL REGISTERS
0111              *
0112        0000  SYMPTR  EQU  0
0113        0001  SYMNO   EQU  1
0114        0002  FLAGS   EQU  2
0115        0003  SYMNM   EQU  3
0116        0004  NSYM    EQU  4
0117        0005  VALUE   EQU  5
0118        0006  SEVEN   EQU  6
0119        0007  STPTR   EQU  7
0120        0008  NWSCT   EQU  8
0121        0009  TN      EQU  9
0122        000A  NSBIT   EQU  10
0123        000C  NINE    EQU  12
0124              *
0125              *       I/O WORKSPACE EQUATES
0126              *
0127        0000  RDCODP  EQU  0
0128        0001  WRCODP  EQU  1
0129        0002  IOC     EQU  2
0130        0003  IOPLUN  EQU  3
0131        0004  FLGS    EQU  4
```

```
0132  0005  BUFADR EQU  5
0133  0006  BUFLEN EQU  6
0134  0007  CHRCNT EQU  7
0135  0008  LEN    EQU  8
0136  0009  SVCAL2 EQU  9
0137  000B  RTN    EQU  11
0138        *
0139        *      COMMON REGISTER EQUATES
0140        *
0141  0000  R0     EQU  0
0142  0001  R1     EQU  1
0143  0002  R2     EQU  2
0144  0003  R3     EQU  3
0145  0004  R4     EQU  4
0146  0005  R5     EQU  5
0147  0006  R6     EQU  6
0148  0007  R7     EQU  7
0149  0008  R8     EQU  8
0150  0009  R9     EQU  9
0151  000A  R10    EQU  10
0152  000B  R11    EQU  11
0153  000C  R12    EQU  12
0154  000D  R13    EQU  13
0155  000E  R14    EQU  14
0156  000F  R15    EQU  15
0157        *
0158        *      ERROR CODES
0159        *
0160  2301  ILSQ   EQU  >2301          ILLEGAL LOAD SEQUENCE
0161  2302  ILCD   EQU  >2302          ILLEGAL LOAD CODE
0162  2303  MSEND  EQU  >2303          MISSING END STATEMENT
0163  2304  LAER   EQU  >2304          LOAD ADDRESS ERROR
0164  2305  PRLM   EQU  >2305          PREVIOUS LOAD MODULE
0165  2306  CKSM   EQU  >2306          CHECKSUM ERROR
```

```
0167                    * !
0168                    *       MONITOR COMMAND OVERLAY SECTION
0169                    *
0170  0000    4C        TEXT 'LL'
0171  0002  00BE'       DATA LALCSR
0172  0004  0000        DATA 0
0173                    *
0174                    *
0175                    *       DATA
0176                    *
0177  0006  0A0D  LFCR  DATA >0A0D              LINE FEED/CARRIAGE RETURN
0178        0006' LF    EQU  LFCR
0179        0007' CR    EQU  LFCR+1
0180  0008    20  BLANK BYTE >20
0181        2000  EOF   EQU  >2000             END OF FILE STATUS
0182        4000  IOERR EQU  >4000             UNRECOVERABLE ERROR STATUS
0183  0009    3A  COLON BYTE ':'
0184  000A    46  F     BYTE 'F'
0185  000B    4C  LOAD  BYTE 'L'
0186  000C    45  END   BYTE 'E'
0187  000D    54  TERM  BYTE 'T'
0188                    *
0189                    *  MONITOR SUPV CALL CODES FOR I/O
0190                    *
0191        0900  RDCOD EQU  >0900             READ ASCII
0192        0B00  WRCOD EQU  >0B00             WRITE ASCII
0193                    EVEN
0194                    *
0195                    *  PERIPHERAL DEVICE DEFAULT LOGICAL UNIT NUMBERS
0196                    *
0197  000E  0000  KBLUNO DATA 0                 LOG (KEYBOARD)
0198  0010  0006  PRLUNO DATA 6                 LOG (PRINTER)
0199  0012  0007  CTLUNO DATA 7                 CS# (DEFAULT CASSETTE LUNO)
0200                    *
0201                    *      DEFAULT LOAD VALUES
0202                    *
0203        0000  DLDPT EQU  0                  LOAD POINT
0204        00A0  DLDBI EQU  >A0               LOAD BIAS
0205                    *
0206                    *      DEFINITION MESSAGE
0207                    *
0208  0014    20  DEFM  TEXT '             * '
0209  0022  0000  DEFN  DATA 0,0,0
      0024  0000
      0026  0000
0210  0028    20        TEXT '  '
0211  002A  0000  DEFL  DATA 0,0
      002C  0000
0212  002E    0D        BYTE >D
0213                    *
0214                    *      UNDEFINED SYMBOLS MESSAGE
0215                    *
0216  002F    55  UNDEF TEXT 'UNDEF'            UNDEFINED SYMBOL
0217  0034    0A        BYTE >A,>D
```

```
       0035   0D
0218   0036   20   UDS    TEXT '    '
0219   003A 0000   UDSN   DATA 0,0,0
       003C 0000
       003E 0000
0220   0040   0A          BYTE >A,>D
       0041   0D
0221                *
0222                *      MULTIPLY DEFINED SYMBOL MESSAGE
0223                *
0224   0042   20   MULDF  TEXT ' M'                MULTIPLE DEFINED SYMBOL
0225   0044   0D          BYTE >D
0226                *
0227                *      LOAD MESSAGE
0228                *
0229   0045   4C   LODMSG TEXT 'LOAD/END? '
0230   004F   0D          BYTE >D
0231                *
0232                *      NAME MESSAGE
0233                *
0234   0050   20   NAMMSG TEXT '       '
0235   0056 0000   NAME   DATA 0,0,0,'  '
       0058 0000
       005A 0000
       005C 2020
0236   005E 0000   LOCVAL DATA 0,0
       0060 0000
0237   0062   0D          BYTE >D
0238                *
0239                *      ENTRY MESSAGE
0240                *
0241   0063   20   ENTMSG TEXT ' ENTRY = '
0242   006C        LOCNUM BSS  4
0243   0070   0A          BYTE >A,>D
       0071   0D
0244                *
0245                *      END ACTION MESSAGE
0246                *
0247   0072   0A   ENDACT BYTE >A
0248   0073   54          TEXT 'TERM/CONT? '
0249   007E   0D          BYTE >D
0250                *
0251                *      LIST MESSAGE
0252                *
0253   007F   0A   LSTMSG BYTE >A
0254   0080   46          TEXT 'F/P LIST? '
0255   008A   0D          BYTE >D
0256                *
0257                *      REQUEST LOAD POINT MESSAGE
0258                *
0259   008B   0A   LPMSG  BYTE >A
0260   008C   4C          TEXT 'LD PT? '
0261   0093   0D          BYTE >D
0262                *
0263                *      REQUEST LOAD BIAS MESSAGE
```

```
0264                    *
0265  0094      0A  LBMSG   BYTE >A
0266  0095      4C          TEXT 'LD BI? '
0267  009C      0D          BYTE >D
0268                        EVEN
```

```
0271                    *  MAIN DRIVER WORKSPACE AREA
0272                    *
0273                    MAINWP
0274  009E  0000               DATA 0                          R0    TEMP
0275  00A0  0000               DATA LDBUF                      R1    RDRBF
0276  00A2  058C'              DATA ENDST                      R2    ESTPT
0277  00A4  0578'              DATA SYMBOL+SYMPTR+SYMPTR R3 SYMP
0278  00A6  0063'              DATA ENTMSG                     R4    ENTPT
0279                    ENTADD
0280  00A8  0000       ENTVL   DATA 0                          R5    ENTVAL
0281  00AA  065C'               DATA PRINTN                     R6    PRNTN
0282  00AC  2000       SYMTAB  DATA >2000                      R7    SPTR
0283  00AE  0000       TOPDAT  DATA 0                          R8    TPPT
0284  00B0  0000               DATA 0                          R9    UNDPT & I/O RTN STATUS
0285  00B2  0000               DATA 0                          R10   IOPARM
0286  00B4  0000               DATA 0                          R11   RETURN
0287  00B6  057C'              DATA SYMBOL+FLAGS+FLAGS  R12 SFLAGS
0288  00B8  0000               DATA 0                          R13
0289  00BA  0000               DATA 0                          R14
0290  00BC  0000               DATA 0                          R15
```

```
0292                    * TITLE:     PX9LAL
0293                    *            LINK AND LOAD DRIVER
0294                    * REVISION: 05/01/74
0295                    *            ORIGINAL
0296                    *            03/15/76
0297                    *            MODIFIED TO BE A COMMAND OF PX9MTR
0298                    * COMPUTER: 990, ASSEMBLY
0299                    * ABSTRACT: THE LIST OPTION IS REQUESTED. THE USER
0300                    *            SHOULD ENTER 'F' FOR FULL AND 'P' FOR
0301                    *            PARTIAL LIST.  THE QUESTION 'LOAD/END?'
0302                    *            IS ASKED. THE USER SHOULD ENTER 'L' FOR
0303                    *            LOAD OR 'E' FOR END. IF LOAD, THE DISPATCH
0304                    *            ROUTINE IS CALLED TO PROCESS THE MODULE.
0305                    *            THE LOAD/END LOOP IS CONTINUED UNTIL
0306                    *            ALL MODULES HAVE BEEN LOADED.
0307                    *            IF END, THE ENTRY POINT AND ANY UNDEFINED
0308                    *            SYMBOLS ARE PRINTED, AND THE TERM/ CONT
0309                    *            QUESTION IS ASKED. IF THERE ARE STILL
0310                    *            SOME UNDEFINED SYMBOLS, THE CONT OPTION
0311                    *            MAY BE SPECIFIED AND ADDITIONAL MODULES
0312                    *            MAY BE LOADED. ONCE ALL MODULES ARE LOADED,
0313                    *            TERM MAY BE SPECIFIED, AND CONTROL WILL
0314                    *            RETURN TO THE MONITOR. THE USER MAY THEN
0315                    *            EXECUTE OR DEBUG HIS PROGRAM WITH MONITOR
0316                    *            COMMANDS.
0317                    * CALLING SEQUENCE:
0318                    *            BL   @LALCSR
0319                    * STATISTICS: WORKSPACE = MAINWP (UNSHARED)
0320                    * ROUTINES CALLED:
0321                    *            PRINT, PRINTN, KEYIN, DSPTCH, GETSFL,
0322                    *            GETSNM, BINHEX, PRCRLF, GETHEX, OPEN
0323                    *
0324                    *       PX9LAL COMMAND INTERFACE
0325                    *
0326                    LALCSR
0327  00BE  0420                BLWP @PX9LAL              START UP LAL
      00C0  00C4'
0328  00C2  045B                RT                        RETURN TO MONITOR
0329                    *
0330                    PX9LAL
0331  00C4  009E'               DATA MAINWP,LAL           WORKSPACE,START
      00C6  00C8'
0332                    LAL
0333  00C8  02E0                LWPI MAINWP               INIT WP
      00CA  009E'
0334  00CC  06A0                BL   @PRCRLF              PRINT CR/LF
      00CE  0000
0335  00D0  1000                NOP                       IGNORE ERROR RETURN
0336                    RQLP
0337  00D2  020A                LI   IOPARM,LPMSG         REQUEST LOAD POINT
      00D4  008B'
0338  00D6  0416                BLWP *PRNTN
0339  00D8  06A0                BL   @GETHEX              INPUT HEX VALUE
      00DA  0000
```

```
0340  00DC  10FA          JMP  RQLP              IF ERROR
0341  00DE  1002          JMP STLP               GOOD INPUT VALUE
0342  00E0  020A          LI   R10,DLDPT         NO INPUT, ASSUME DEFAULT
      00E2  0000
0343                STLP
0344  00E4  C80A          MOV  R10,@LOADPT       SAVE LOAD POINT
      00E6  01EE'
0345                RQLB
0346  00E8  020A          LI   IOPARM,LBMSG      REQUEST LOAD BIAS
      00EA  0094'
0347  00EC  0416          BLWP *PRNTN
0348  00EE  06A0          BL   @GETHEX           INPUT HEX VALUE
      00F0  00DA'
0349  00F2  10FA          JMP  RQLB              IF ERROR
0350  00F4  1002          JMP  STLB              GOOD INPUT VALUE
0351  00F6  020A          LI   R10,DLDBI         NO INPUT, ASSUME DEFAULTS
      00F8  00A0
0352                STLB
0353  00FA  C80A          MOV  R10,@BIASWD       SAVE LOAD BIAS
      00FC  01DA'
0354  00FE  C14A          MOV  R10,ENTVAL        SET INITIAL ENTRY POINT
0355  0100  0207          LI   SPTR,INIT         START OF PX9MTR IS END USRMEM
      0102  0000
0356  0104  C487          MOV  SPTR,*ESTPT       END OF SYMBOL TABLE POINTER
0357  0106  0607          DEC  SPTR
0358  0108  C4C7          MOV  SPTR,*SYMP        INIT SYMBOL TABLE POINTER
0359  010A  05A0          INC  @FSTIM            SET FIRST MODULE FLAG
      010C  040A'
0360  010E  0208          LI   TPPT,>80          SET DATA LIMIT
      0110  0080
0361                *
0362                *  LIST OPTION
0363                *
0364  0112  020A          LI   IOPARM,LSTMSG     PRINT LIST OPTION MESSAGE
      0114  007F'
0365  0116  0416          BLWP *PRNTN
0366  0118  C281          MOV  RDRBF,IOPARM      WAIT FOR RESPONSE
0367  011A  0420          BLWP @KEYIN
      011C  061C'
0368  011E  04E0          CLR  @LIST             CLEAR LIST FLAG
      0120  052C'
0369  0122  9811          CB   *RDRBF,@F         IS FULL LIST DESIRED?
      0124  000A'
0370  0126  1602          JNE  ASK               NO = CONTINUE
0371  0128  05A0          INC  @LIST             YES = SET FULL LIST FLAG
      012A  052C'
0372                *
0373                *       REQUEST LOAD/END ACTION
0374                *
0375                ASK
0376  012C  02E0          LWPI MAINWP            LOAD WORKSPACE POINTER
      012F  009E'
0377  0130  06A0          BL   @PRCRLF           PRINT CR/LF
      0132  00CE'
0378  0134  1000          NOP                    IGNORE ERROR RETURN
```

```
0379                    REQLE
0380  0136  020A              LI    IOPARM,LODMSG     PRINT LOAD/END MSG
      0138  0045'
0381  013A  0416              BLWP  *PRNTN
0382  013C  C281              MOV   RDRBF,IOPARM
0383  013E  04DA              CLR   *IOPARM           INIT INPUT BUFFER TO 0
0384  0140  0420              BLWP  @KEYIN            WAIT FOR RESPONSE
      0142  061C'
0385  0144  9811              CB    *RDRBF,@END       END?
      0146  000C'
0386  0148  1314              JEQ   ENDPRO            YES
0387  014A  9811              CB    *RDRBF,@LOAD      LOAD?
      014C  000B'
0388  014E  16F3              JNE   REQLE             NO, ASK AGAIN
0389  0150  C820              MOV   @K7,@CTLUNO       DEFAULT TO CASSETTE LUNO 7
      0152  0584'
      0154  0012'
0390  0156  06D1              SWPB  *RDRBF            CASSETTE LUNO INPUT?
0391  0158  D011              MOVB  *RDRBF,TEMP
0392  015A  1304              JEQ   OPNCAS            NO, USE DEFAULT
0393  015C  0240              ANDI  TEMP,>F00         CLEAR ALL BITS EXCEPT LUNO
      015E  0F00
0394  0160  D800              MOVB  TEMP,@CTLUNO+1    SAVE INPUT LUNO
      0162  0013'
0395                    OPNCAS
0396  0164  C2A0              MOV   @CTLUNO,IOPARM    OPEN CASSETTE
      0166  0012'
0397  0168  0420              BLWP  @OPEN
      016A  05FC'
0398                    BATCH
0399  016C  0420              BLWP  @DSPTCH           PROCESS MODULE
      016E  01FC'
0400  0170  10FD              JMP   BATCH             PROCESS NEXT MODULE
0401                    *
0402                    *     PROCESS UNDEFINED SYMBOLS
0403                    *
0404                    ENDPRO
0405  0172  0709              SETO  UNDPT             SET UNDEFINED PRINT FLAG
0406  0174  C4C7              MOV   SPTR,*SYMP        POINT TO SYMBOL TABLE
0407                    CKSYMB
0408  0176  8493              C     *SYMP,*ESTPT      END OF TABLE?
0409  0178  121B              JLE   ENTB              YES = END PROCESSING
0410  017A  0420              BLWP  @GETSFL           NO = GET SYMBOL FLAGS
      017C  0598'
0411  017E  061C              DEC   *SFLAGS           UNDEFINED?
0412  0180  1114              JLT   NXSYM             NO = GET NEXT SYMBOL
0413  0182  C249              MOV   UNDPT,UNDPT       PRINT 'UNDEFINED' MESSAGE?
0414  0184  1305              JEQ   SPRNT             NO = SKIP
0415  0186  020A              LI    IOPARM,UNDEF      POINT TO MESSAGE
      0188  002F'
0416  018A  0420              BLWP  @PRINT
      018C  0658'
0417  018E  04C9              CLR   UNDPT             CLEAR PRINT FLAG
0418                    SPRNT
0419  0190  0420              BLWP  @GETSNM           POINT TO SYMBOL NAME
```

```
      0192  05AC'
0420  0194  C2A0          MOV   @SYMBOL+SYMNM+SYMNM,IOPARM
      0196  057E'
0421  0198  0200          LI    TEMP,UDSN           POINT TO NAME IN MESSAGE
      019A  003A'
0422  019C  CC3A          MOV   *IOPARM+,*TEMP+     MOVE NAME INTO MESSAGE
0423  019E  CC3A          MOV   *IOPARM+,*TEMP+
0424  01A0  C41A          MOV   *IOPARM,*TEMP
0425  01A2  020A          LI    IOPARM,UDS          POINT TO MESSAGE
      01A4  0036'
0426  01A6  0420          BLWP  @PRINT              PRINT IT
      01A8  0658'
0427                NXSYM
0428  01AA  64E0          S     @TEN,*SYMP          POINT TO NEXT ENTRY
      01AC  058A'
0429  01AE  10E3          JMP   CKSYMB              CHECK NEW SYMBOL
0430                *
0431                *  ENTRY MESSAGE AND TERMINATE OPTION
0432                *
0433                ENTB
0434  01B0  C284          MOV   ENTPT,IOPARM        PRINT ENTRY POINT MSG
0435  01B2  0420          BLWP  @BINHEX             CONVERT ENTRY LOCATION
      01B4  0540'
0436  01B6  00A8'         DATA  ENTVL               ENTRY VALUE
0437  01B8  006C'         DATA  LOCNUM              LOCATION IN MESSAGE
0438  01BA  0420          BLWP  @PRINT
      01BC  0658'
0439  01BE  020A          LI    IOPARM,ENDACT       PRINT TERM/CONT MSG
      01C0  0072'
0440  01C2  0416          BLWP  *PRNTN
0441  01C4  C281          MOV   RDRBF,IOPARM        WAIT FOR RESPONSE
0442  01C6  0420          BLWP  @KEYIN
      01C8  061C'
0443  01CA  9811          CB    *RDRBF,@TERM        TERMINATE?
      01CC  000D'
0444  01CE  16AE          JNE   ASK                 NO - GET MORE MODULES
0445                *
0446                *  END OF PX9LAL
0447                *
0448  01D0  C805          MOV   ENTVAL,@USRPC       SET UP ENTRY POINT FOR PX9MTR
      01D2  0000
0449  01D4  0380          RTWP                      RETURN
```

```
0452                    *
0453                    *  DSPTCH WORKSPACE AREA
0454                    *
0455                    DSWP
0456  01D6  0000               DATA 0              R0   INPT
0457  01D8  0000               DATA 0              R1   TGFLD
0458  01DA  00A0  BIASWD       DATA >A0            R2   BIAS
0459                    DSWP2
0460  01DC  0000  NWLOC        DATA 0              R3   NEWLOC
0461  01DE  0000               DATA 0              R4   LOCPT
0462  01E0  0000               DATA 0              R5   SVPT
0463  01E2  0000               DATA 0              R6   HLD
0464  01E4  0000               DATA 0              R7   FLGVL
0465  01E6  0000               DATA 0              R8   PC
0466  01E8  0000               DATA 0              R9   RETURN STATUS ON READ
0467  01EA  0000               DATA 0              R10  IOPARM
0468  01EC  0000               DATA 0              R11  RETURN
0469  01EE  0000  LOADPT       DATA 0              R12  LDPT
0470  01F0  03E6'              DATA INCHK          R13  INCP
0471  01F2  0000               DATA 0              R14  INSID
0472  01F4  00AE'              DATA TOPDAT         R15  TDAT
0473  01F6  0000               DATA 0              R13  OLD WP
0474  01F8  0000               DATA 0              R14  OLD PC
0475  01FA  0000               DATA 0              R15  OLD ST
```

```
0477                * TITLE:    DSPTCH
0478                *           LOAD MODULE DISPATCHER
0479                * REVISION: 05/01/74
0480                *           ORIGINAL
0481                *           03/15/76
0482                *           MODIFIED TO RUN WITH PX9MTR
0483                * COMPUTER: 990, 990 ASSEMBLY
0484                * ABSTRACT: OBJECT RECORDS ARE READ AND PROCESSED.
0485                *           EACH RECORD CONTAINS SEVERAL OBJECT ENTRIES.
0486                *           AN OBJECT ENTRY CONSISTS OF A TAG FIELD AND
0487                *           UP TO TWO ADDITIONAL FIELDS. THE TAG FIELD
0488                *           OF EACH ENTRY IS USED TO BRANCH TO THE
0489                *           APPROPRIATE PROCESSOR. A TAG FIELD OF 'F'
0490                *           WILL CAUSE A NEW RECORD TO BE READ.
0491                * CALLING SEQUENCE:
0492                *           BLWP @DSPTCH
0493                * STATISTICS:
0494                *           WORKSPACE = DSWP AND DSWP2 (UNSHARED)
0495                *           ALL SIXTEEN REGISTERS ARE USED BY OVERLAPPING
0496                *           DSWP AND DSWP2. ALL ENTRANCES ARE DEFINED
0497                *           THROUGH EXWP2 TO RETAIN THE RETURN ENVIRONMENT.
0498                *           THE FIRST INSTRUCTION IS THEN A 'LWPI DSWP'
0499                *           TO GAIN THREE ADDITIONAL REGISTERS FOR USE.
0500                *           A 'LWPI DSWP2' WILL THEN PRECEDE ANY RTWP
0501                *           INSTRUCTION.
0502                *
0503                *           THE TAG FIELDS REPRESENT THE FOLLOWING
0504                *           ENTRY TYPES:
0505                *             TAG      MEANING
0506                *              0     IDT RECORD
0507                *              1     ABSOLUTE ENTRY ADDRESS
0508                *              2     RELOCATABLE ENTRY ADDRESS
0509                *              3     EXTERNAL REFERENCE IN RELOCATABLE CODE
0510                *              4     EXTERNAL REFERENCE IN ABSOLUTE CODE
0511                *              5     RELOCATABLE EXTERNAL DEFINITION
0512                *              6     ABSOLUTE EXTERNAL DEFINITION
0513                *              7     CHECKSUM
0514                *              8     IGNORE CHECKSUM
0515                *              9     ABSOLUTE LOAD ADDRESS
0516                *              A     RELOCATABLE LOAD ADDRESS
0517                *              B     ABSOLUTE DATUM
0518                *              C     RELOCATABLE DATUM
0519                *              D     LOAD BIAS
0520                *              E     ILLEGAL TAG FIELD
0521                *              F     END OF RECORD
0522                *              G     RELOCATABLE SYMBOL (IGNORED)
0523                *              H     ABSOLUTE SYMBOL    (IGNORED)
0524                *
0525                * ROUTINES CALLED:
0526                *           BINHEX,CONVRT,CUMCHK,ERROR,GETSVL,HEXBIN,
0527                *           INCHK,LDR,PRCRLF,PRINTN,READ,SRCSYM
0528                *
0529                DSPTCH
0530  01FC  01DC'         DATA DSWP2              WORKSPACE
```

```
0531  01FE  0200'        DATA DSBG               START
0532                DSBG
0533  0200  02E0        LWPI DSWP               ADJUST WORKSPACE
      0202  01D6'
0534  0204  04CE        CLR  INSID
0535                    *
0536                    *     READ AN OBJECT RECORD (F)
0537                    *
0538                    TAGF
0539  0206  0200        LI   INPT,LDBUF         POINT TO READ BUFFER
      0208  00A0'
0540  020A  C280        MOV  INPT,IOPARM
0541  020C  0420        BLWP @READ              READ A RECORD
      020E  0612'
0542  0210  0249        ANDI R9,EOF+IOERR       EOF OR IOERR ENCOUNTERED
      0212  6000
0543  0214  1688        JNE  ASK                YES, RESTART
0544                    *
0545                    *     ADJUST BIAS AND BRANCH BY TAG CHARACTER
0546                    *
0547                    CHLP
0548  0216  0242        ANDI BIAS,>FFFE         SET BIAS TO WORD BOUNDARY
      0218  FFFE
0549  021A  04C7        CLR  FLGVL              CLEAR REF/DEF FLAG
0550  021C  04C6        CLR  HLD                CLEAR TEMP LOCN FOR OFFSET
0551  021E  D070        MOVB *INPT+,TGFLD       GET TAG FIELD
0552  0220  0221        AI   TGFLD,->3000       ADJUST FOR ASCII
      0222  D000
0553  0224  1120        JLT  TAGE               NOT IN RANGE = INV LD CD
0554  0226  0971        SRL  TGFLD,7            ADJUST TAG FOR BRANCH
0555  0228  0281        CI   TGFLD,JMPTBE-JMPTB-1    IN RANGE?
      022A  0031
0556  022C  151C        JGT  TAGE               NO = INV LD CD
0557  022E  C061        MOV  @JMPTB(TGFLD),TGFLD  BRANCH ON TAG
      0230  0234'
0558  0232  0451        B    *TGFLD
0559                    *
0560                    *     TAG CHARACTER JUMP TABLE
0561                    *
0562                    JMPTB
0563  0234  0278'        DATA TAG0               IDT RECORD
0564  0236  02BE'        DATA TAG1               ABSOLUTE ENTRY ADDRESS
0565  0238  02BC'        DATA TAG2               RELOCATABLE ENTRY ADDRESS
0566  023A  02C8'        DATA TAG3               EXTERNAL REF = RELOC CODE
0567  023C  02CA'        DATA TAG4               EXTERNAL REF = ABS CODE
0568  023E  033E'        DATA TAG5               RELOCATABLE EXTERNAL DEF
0569  0240  0340'        DATA TAG6               ABSOLUTE EXTERNAL DEF
0570  0242  03A8'        DATA TAG7               CHECKSUM
0571  0244  03AC'        DATA TAG8               IGNORE CHECKSUM
0572  0246  03B4'        DATA TAG9               ABSOLUTE LOAD ADDRESS
0573  0248  02AE'        DATA BKOUT              COLON: END OF MODUL
0574  024A  0266'        DATA TAGE               INVALID LOAD CODE
0575  024C  0266'        DATA TAGE               INVALID LOAD CODE
0576  024E  0266'        DATA TAGE               INVALID LOAD CODE
0577  0250  0266'        DATA TAGE               INVALID LOAD CODE
```

```
0578  0252  0266'        DATA TAGE              INVALID LOAD CODE
0579  0254  0266'        DATA TAGE              INVALID LOAD CODE
0580  0256  03B2'        DATA TAGA              RELOCATABLE LOAD ADDRESS
0581  0258  03C2'        DATA TAGB              ABSOLUTE DATUM
0582  025A  03C0'        DATA TAGC              RELOCATABLE DATUM
0583  025C  03DA'        DATA TAGD              LOAD BIAS
0584  025E  0266'        DATA TAGE              ERROR
0585  0260  0206'        DATA TAGF              END OF RECORD
0586  0262  0272'        DATA TAGGH             RELOCATABLE SYMBOL
0587  0264  0272'        DATA TAGGH             ABSOLUTE SYMBOL
0588        0266' JMPTBE EQU  $
0589              *
0590              *     ILLEGAL TAG CHARACTER (E)
0591              *
0592              TAGE
0593  0266  020A        LI   IOPARM,ILCD       ILLEGAL LOAD CODE
      0268  2302
0594              ERR
0595  026A  06A0        BL   @ERROR            CALL MONITOR ERROR ROUTINE
      026C  0000
0596  026E  0460        B    @LAL              FATAL ERROR = RESTART
      0270  00C8'
0597              *
0598              *     SYMBOL TABLE TAG CHARACTERS (G/H)
0599              *
0600              TAGGH
0601  0272  0220        AI   INPT,10           INCREMENT POINTER PAST ENTRY
      0274  000A
0602  0276  10CF        JMP  CHLP                TO IGNORE SYMBOL TABLE TAGS
0603              *
0604              *     PROCESS MODULE IDT (0)
0605              *
0606              TAG0
0607  0278  C802        MOV  BIAS,@LOCPTR      GET PRESENT BIAS
      027A  0414'
0608  027C  C38E        MOV  INSID,INSID       PRESENTLY INSIDE MODULE?
0609  027E  1607        JNE  TAG0B             YES = TEST IF PROPER
0610  0280  0420        BLWP @LDR              NO = PROCESS IDT
      0282  0428'
0611  0284  1018        JMP  DSPEXT            EXIT IF MODULE FLUSHED
0612  0286  0460        B    @ASK              IF ERROR, EXIT LOOP
      0288  012C'
0613  028A  058E        INC  INSID             SET INSIDE MODULE FLAG
0614  028C  100D        JMP  TAG0CN            CONTINUE
0615              TAG0B
0616  028E  06A0        BL   @CONVRT           GET CONVERTED VALUE
      0290  03F2'
0617  0292  01DC'       DATA NWLOC             STORE IT IN LOCATION TEMP
0618  0294  A803        A    NEWLOC,@NWBIAS
      0296  04D2'
0619  0298  9810        CB   *INPT,@BLANK      IS IDT NAME BLANK
      029A  0008'
0620  029C  1303        JEQ  TAG0C             YES = CONTINUE
0621              *
0622              *     MISSING END STATEMENT
```

```
0623                    *
0624  029E  020A         LI   IOPARM,MSEND      MISSING END STMT ERROR
      02A0  2303
0625  02A2  10E3         JMP  ERR
0626                    TAG0C
0627  02A4  0220         AI   INPT,-4          ADJUST POINTER PAST LENGTH
      02A6  FFFC
0628                    TAG0CN
0629  02A8  0220         AI   INPT,12          ADJUST POINTER
      02AA  000C
0630  02AC  10B4         JMP  CHLP             GET NEXT TAG
0631                    *
0632                    *     END MODULE (1)
0633                    *
0634                    BKOUT
0635  02AE  04CE         CLR  INSID            CLEAR INSIDE MODULE FLAG
0636  02B0  A0A0         A    @NWBIAS,BIAS     ADD IN NEW BIAS
      02B2  04D2'
0637  02B4  0582         INC  BIAS             ACCOUNT FOR ODD LENGTH
0638                    DSPEXT
0639  02B6  02E0         LWPI DSWP2
      02B8  01DC'
0640  02BA  0380         RTWP                  RETURN
0641                    *
0642                    *     GET RELOCATABLE ENTRY ADDRESS (2)
0643                    *
0644                    TAG2
0645  02BC  C182         MOV  BIAS,HLD         SET RELOCATABLE OFFSET
0646                    *
0647                    *     GET ABSOLUTE ENTRY ADDRESS (1)
0648                    *
0649                    TAG1
0650  02BE  069D         BL   *INCP            TEST INSIDE MODUL AND CONVERT
0651  02C0  00A8'        DATA ENTADD           PLACE IN ENTRY ADDRESS
0652  02C2  A806         A    HLD,@ENTADD      ADD IN OFFSET
      02C4  00A8'
0653  02C6  10A7         JMP  CHLP             GET NEXT TAG
0654                    *
0655                    *     GET RELOCATABLE CODE REF (3)
0656                    *
0657                    TAG3
0658  02C8  C182         MOV  BIAS,HLD         SET RELOCATABLE OFFSET
0659                    *
0660                    *     GET ABSOLUTE CODE REF (4)
0661                    *
0662                    TAG4
0663  02CA  069D         BL   *INCP            TEST INSIDE MODUL AND CONVERT
0664  02CC  01DC'        DATA NWLOC            PLACE IN NWLOC
0665  02CE  A0C6         A    HLD,NEWLOC       ADJUST WITH BIAS
0666                    *
0667                    *     PROCESS REF CHAIN LINKAGE
0668                    *
0669  02D0  C800         MOV  INPT,@NSMBL      POINT TO SYMBOL NAME
      02D2  04DA'
0670  02D4  0420         BLWP @SRCSYM          SEARCH SYMBOL TABLE
```

```
                02D6  05B0'
0671            02D8  1018          JMP   ENTERR              IF NOT FOUND, ENTER IT
0672            02DA  0220          AI    INPT,6               ADJUST POINTER
                02DC  0006
0673            02DE  C103          MOV   NEWLOC,LOCPT         POINT TO NEW END OF CHAIN
0674            02E0  139A          JEQ   CHLP                 IF ZERO, NO REF, SO END PROC
0675            02E2  0420          BLWP  @GETSVL
                02E4  05A0'
0676            02E6  C160          MOV   @SYMBOL+SYMNM+SYMNM,SVPT GET VALUE
                02E8  057E'
0677            02EA  0620          DEC   @SYMBOL+FLAGS+FLAGS
                02EC  057C'
0678            02EE  1109          JLT   FXRF                 IF DEF, SATISFY CHAIN
0679                            LPFXR
0680            02F0  A10C          A     LDPT,LOCPT           POINT TO ACTUAL MEM ADDR
0681            02F2  C194          MOV   *LOCPT,HLD           GET NEXT CHAIN ENTRY
0682            02F4  1302          JEQ   ENDLPR               IF ZERO, END OF CHAIN
0683            02F6  C106          MOV   HLD,LOCPT            CHANGE LOCATION POINTER
0684            02F8  10FB          JMP   LPFXR                CONTINUE CHAINING
0685                            ENDLPR
0686            02FA  A10C          A     LDPT,LOCPT           POINT TO ACTUAL MEM ADDR
0687            02FC  C515          MOV   *SVPT,*LOCPT         BREAK END LINK
0688            02FE  C543          MOV   NEWLOC,*SVPT         CREATE NEW LINK IN SYMBOL TAB
0689            0300  108A          JMP   CHLP                 END PROC, GET NEXT TAG
0690                            FXRF
0691            0302  C0D5          MOV   *SVPT,NEWLOC         GET VALUE
0692                            LPDF
0693            0304  A10C          A     LDPT,LOCPT           POINT TO ACTUAL MEM ADDR
0694            0306  C194          MOV   *LOCPT,HLD           POINT TO NEXT ENTRY IN CHAIN
0695            0308  C503          MOV   NEWLOC,*LOCPT        FILL IN RESOLVED VALUE
0696            030A  C106          MOV   HLD,LOCPT            CHANGE LOCATION POINTER
0697            030C  16FB          JNE   LPDF                 CONTINUE CHAINING
0698                            CHLP3
0699            030E  1083          JMP   CHLP                 END PROC, GET NEXT TAG
0700                            *
0701                            *       ENTER SYMBOL INTO TABLE
0702                            *
0703                            ENTERR
0704            0310  0587          INC   FLGVL                SET REF CODE
0705                            ENTERD
0706            0312  C160          MOV   @SYMBOL+SYMPTR+SYMPTR,SVPT
                0314  0578'
0707            0316  C547          MOV   FLGVL,*SVPT          MARK FLAG
0708            0318  0225          AI    SVPT,-7              POINT TO NAME LOCATION
                031A  FFF9
0709            031C  DD70          MOVB  *INPT+,*SVPT+
0710            031E  DD70          MOVB  *INPT+,*SVPT+        MOVE IN NAME
0711            0320  DD70          MOVB  *INPT+,*SVPT+
0712            0322  DD70          MOVB  *INPT+,*SVPT+
0713            0324  DD70          MOVB  *INPT+,*SVPT+
0714            0326  D570          MOVB  *INPT+,*SVPT
0715            0328  0225          AI    SVPT,-7              POINT TO VALUE
                032A  FFF9
0716            032C  C543          MOV   NEWLOC,*SVPT         MOVE IN VALUE
0717            032E  C805          MOV   SVPT,@ENDST          CREATE NEW END OF SYMBOL TABL
```

```
                0330  058C'
0718            0332  881F        C     *TDAT,@ENDST      IS LIMIT BELOW SYMBOL TABLE?
                0334  058C'
0719            0336  1AEB        JL    CHLP3             YES - GET NEXT TAG
0720                        *
0721                        *     LOAD ADDRESS ERROR
0722                        *
0723                        LOADER
0724            0338  020A        LI    IOPARM,LAER       POINT TO ERROR CODE
                033A  2304
0725            033C  1096        JMP   ERR
0726                        *
0727                        *     GET RELOCATABLE DEF (5)
0728                        *
0729                        TAG5
0730            033E  C182          MOV   BIAS,HLD        SET RELOCATABLE OFFSET
0731                        *
0732                        *     GET ABSOLUTE DEF (6)
0733                        *
0734                        TAG6
0735            0340  069D        BL    *INCP             TEST INSIDE MODUL AND CONVERT
0736            0342  01DC'      DATA  NWLOC             PLACE IN NWLOC
0737            0344  A0C6        A     HLD,NEWLOC        ADJUST WITH BIAS
0738                        *
0739                        *     PROCESS DEF RESOLUTION
0740                        *
0741            0346  C060        MOV   @LIST,TGFLD       IS FULL LIST DESIRED
                0348  052C'
0742            034A  1313        JEQ   PRNTOV            NO - SKIP
0743            034C  0420        BLWP  @BINHEX           YES - CONVERT LOCATION
                034E  0540'
0744            0350  01DC'      DATA  NWLOC             LOCATION VALUE
0745            0352  002A'      DATA  DEFL              LOCATION IN MESSAGE
0746            0354  0201        LI    TGFLD,DEFN        POINT TO NAME IN MESSAGE
                0356  0022'
0747                        MVNM
0748            0358  DC70        MOVB  *INPT+,*TGFLD+    MOVE IN NAME
0749            035A  0281        CI    TGFLD,DEFN+6
                035C  0028'
0750            035E  11FC        JLT   MVNM
0751            0360  0220        AI    INPT,-6           RESET POINTER
                0362  FFFA
0752            0364  06A0        BL    @PRCRLF           PRINT CR/LF
                0366  0132'
0753            0368  1000        NOP                     IGNORE ERROR RETURN
0754            036A  020A        LI    IOPARM,DEFM       PRINT DEFINITION MESSAGE
                036C  0014'
0755            036E  0420        BLWP  @PRINTN
                0370  065C'
0756                        PRNTOV
0757            0372  C800        MOV   INPT,@NSMBL       POINT TO SYMBOL NAME
                0374  04DA'
0758            0376  0420        BLWP  @SRCSYM           SEARCH SYMBOL TABLE
                0378  05B6'
0759            037A  10CB        JMP   ENTERD            NOT FOUND, ENTER IT
```

```
0760  037C  0220          AI    INPT,6              ADJUST POINTER
      037E  0006
0761  0380  0620          DEC   @SYMBOL+FLAGS+FLAGS  DEFINED?
      0382  057C'
0762  0384  1305          JEQ   SDF                 NO - SET DEFINITION VALUE
0763                *
0764                *     MULTIPLY DEFINED SYMBOL
0765                *
0766  0386  020A          LI    IOPARM,MULDF
      0388  0042'
0767  038A  0420          BLWP  @PRINTN
      038C  065C'
0768                CHLP2
0769  038E  10BF          JMP   CHLP3               CONTINUE
0770                SDF
0771  0390  0420          BLWP  @GETSVL             GET SYMBOL VALUE
      0392  05A0'
0772  0394  C160          MOV   @SYMBOL+SYMNM+SYMNM,SVPT POINT TO VALUE
      0396  057E'
0773  0398  C115          MOV   *SVPT,LOCPT         POINT TO REF CHAIN
0774  039A  C543          MOV   NEWLOC,*SVPT        CHANGE VALUE
0775  039C  C160          MOV   @SYMBOL+SYMPTR+SYMPTR,SVPT POINT TO ENTRY
      039E  0578'
0776  03A0  04D5          CLR   *SVPT               CHANGE FLAG
0777  03A2  C104          MOV   LOCPT,LOCPT         WAS ACTUAL REF USED?
0778  03A4  13F4          JEQ   CHLP2               NO - FINISH PROCESSING
0779  03A6  10AE          JMP   LPDF                PROCESS CHAIN SATISFACTION
0780                *
0781                *     GET CHECKSUM (7)
0782                *
0783                TAG7
0784  03A8  0420          BLWP  @CUMCHK             CHECK CUMULATIVE SUM
      03AA  04E2'
0785                *
0786                *     SKIP CHECKSUM (8)
0787                *
0788                TAG8
0789  03AC  0220          AI    INPT,4              SKIP CUMSUM
      03AE  0004
0790  03B0  10EE          JMP   CHLP2               GET NEXT FIELD
0791                *
0792                *     GET RELOCATABLE LOAD ADDRESS (A)
0793                *
0794                TAGA
0795  03B2  C182          MOV   BIAS,HLD            SET RELOCATABLE OFFSET
0796                *
0797                *     GET ABSOLUTE LOAD ADDRESS (9)
0798                *
0799                TAG9
0800  03B4  069D          BL    *INCP               TEST INSIDE MODUL AND CONVERT
0801  03B6  01DC'         DATA  NWLOC               PLACE IN NWLOC
0802  03B8  A0C6          A     HLD,NEWLOC          ADJUST WITH BIAS
0803  03BA  C203          MOV   NEWLOC,PC           SET LOCATION COUNTER
0804  03BC  A20C          A     LDPT,PC             ADJUST WITH ACTUAL LOAD POINT
0805  03BE  10E7          JMP   CHLP2               CONTINUE
```

```
0806                    *
0807                    *       GET RELOCATABLE DATUM (C)
0808                    *
0809                    TAGC
0810  03C0  C182               MOV   BIAS,HLD          SET RELOCATABLE OFFSET
0811                    *
0812                    *       GET ABSOLUTE DATUM (B)
0813                    *
0814                    TAGB
0815  03C2  8808               C     PC,@ENDST         IS ADDR PAST SYMBOL TABLE?
      03C4  058C'
0816  03C6  1488               JHE   LOADER            YES = ERROR
0817  03C8  821F               C     *TDAT,PC          IS OLD LIMIT ABOVE ADDRESS?
0818  03CA  1401               JHE   SETLP             YES = CONTINUE
0819  03CC  C7C8               MOV   PC,*TDAT          CREATE NEW LIMIT
0820                    SETLP
0821  03CE  C808               MOV   PC,@LOADV2        SET UP LOAD POINT
      03D0  03D4'
0822  03D2  069D               BL    *INCP             TEST INSIDE MODUL AND CONVERT
0823  03D4  03D4' LOADV2 DATA $                       PLACE IN PC
0824  03D6  AE05               A     HLD,*PC+          ADD IN OFFSET
0825  03D8  10DA               JMP   CHLP2             GET NEXT TAG
0826                    *
0827                    *       GET LOAD BIAS (D)
0828                    *
0829                    TAGD
0830  03DA  C38E               MOV   INSID,INSID       INSIDE MODUL?
0831  03DC  1606               JNE   INER              YES = ILLEGAL LD SEQUENCE
0832  03DE  06A0               BL    @CONVRT           GET CONVERTED VALUE
      03E0  03F2'
0833  03E2  01DA'              DATA BIASWD             PLACE IN BIAS
0834  03E4  10D4               JMP   CHLP2             GET NEXT TAG
0835                    *
0836                    *       TEST IF INSIDE MODUL PROCESSING
0837                    *
0838                    INCHK
0839  03E6  C38E               MOV   INSID,INSID       INSIDE MODUL?
0840  03E8  1604               JNE   CONVRT            YES = CONVERT
0841                    *
0842                    *       OUTSIDE MODULE
0843                    *
0844                    INER
0845  03EA  020A               LI    IOPARM,ILSQ       ILLEGAL LOAD SEQUENCE ERROR
      03EC  2301
0846  03EE  0460               B     @ERR
      03F0  026A'
0847                    *
0848                    *       CONVERT FIELD IN OBJECT CODE
0849                    *
0850                    CONVRT
0851  03F2  C83B               MOV   *11+,@CONLOC      MOVE IN CONVERSION LOCATION
      03F4  0400'
0852  03F6  C800               MOV   INPT,@CONVL       POINT TO BUFFER AREA
      03F8  03FE'
0853  03FA  0420               BLWP  @HEXBIN           CONVERT TO BINARY
```

```
        03FC  055C!
 0854   03FE  03FE! CONVL  DATA $          DATA FOR CONVERSION
 0855   0400  0400! CONLOC DATA $          PLACE IN PROPER LOCATION
 0856   0402  0220         AI   INPT,4     ADJUST POINTER
        0404  0004
 0857   0406  045B         RT              RETURN
```

```
0860                    *
0861                    * LDR WORKSPACE AREA
0862                    *
0863                    LDWP
0864  0408  0208'              DATA LDBUF          R0    RD
0865  040A  0000       FSTIM   DATA 0              R1    FST
0866  040C  0000               DATA 0              R2    PTR
0867  040E  04DA'               DATA NSMBL          R3    NSMBLP
0868  0410  0000               DATA 0              R4    SYMB
0869  0412  0000               DATA 0              R5    TMP
0870  0414  0000       LOCPTR  DATA 0              R6    LOC
0871  0416  0586'               DATA SRCSYM         R7    SRSM
0872  0418  0050'               DATA NAMMSG         R8    NMMSG
0873  041A  0000               DATA 0              R9
0874  041C  0000               DATA 0              R10   IOPARM
0875  041E  0000               DATA 0              R11   RTN
0876  0420  055C'               DATA HEXBIN         R12   HEXP2
0877  0422  0000               DATA 0              R13   OLD WP
0878  0424  0000               DATA 0              R14   OLD PC
0879  0426  0000               DATA 0              R15   OLD ST
```

```
0881                    * TITLE:    LDR
0882                    *           LOADER PROCESS OF IDT ENTRY
0883                    * REVISION: 05/01/74
0884                    *           ORIGINAL
0885                    *           03/15/76
0886                    *           MODIFIED TO RUN WITH PX9MTR
0887                    * COMPUTER: 990, ASSEMBLY
0888                    * ABSTRACT: AN IDT ENTRY IS CHECKED, IF THIS IS
0889                    *           THE FIRST MODULE, THE NAME WILL BE PLACED
0890                    *           IN THE SYMBOL TABLE AND CONTROL RETURNED
0891                    *           TO THE CALLING ROUTINE FOR MODULE PROCESSING.
0892                    *           IF THIS IS A LATER ENTRY WHOSE NAME IS NOT
0893                    *           IN THE SYMBOL TABLE, IT WILL BE FLUSHED.
0894                    *           IF ALREADY DEFINED, AN ERROR MESSAGE WILL
0895                    *           BE PRINTED AND THE MODULE FLUSHED. IF ONLY
0896                    *           REF'D, MODULE WILL BE PROCESSED AS FIRST
0897                    *           MODULE, AND THE NAME WILL BE MARKED AS DEF'D.
0898                    *           NO VALUE IS GIVEN TO THE NAME SO IT MAY NOT
0899                    *           BE USED TO SATISFY REFERENCES. THREE RETURNS
0900                    *           ARE PROVIDED FOR ACCEPTABLE MODULES, MODULES
0901                    *           FLUSHED BECAUSE OF ERROR, AND MODULES FLUSHED
0902                    *           BECAUSE NOT REF'D.
0903                    * CALLING SEQUENCE:
0904                    *           BLWP @LDR
0905                    *           NOT REF'D MODULE FLUSHED = 1 WD INSTR
0906                    *           ERROR = MODULE FLUSHED   = 2 WD INSTR
0907                    *           ACCEPTABLE MODULE        = 1 WD INSTR
0908                    * STATISTICS: WORKSPACE = LDWP (UNSHARED)
0909                    * ROUTINES CALLED:
0910                    *           BINHEX,ERROR,GETSVL,HEXBIN,PRCRLF,PRINTN,
0911                    *           READ,SRCSYM
0912                    LDR
0913  0428  0408'               DATA LDWP                WORKSPACE
0914  042A  042C'               DATA LDBG                START
0915                    LDBG
0916  042C  C080               MOV  RD,PTR              POINT TO READ BUFFER
0917  042E  0582               INC  PTR                 SKIP TAG
0918                    *
0919                    *     CONVERT RELOCATABLE LENGTH
0920                    *
0921  0430  C802               MOV  PTR,@PTRPT          POINT TO BUFFER
      0432  0436'
0922  0434  041C               BLWP *HEXP2              CONVERT LENGTH
0923  0436  0436' PTRPT  DATA $                         POINTER TO MODUL LENGTH
0924  0438  04D2'               DATA NWBIAS              CREATE NEW BIAS
0925  043A  0222               AI   PTR,4               ADJUST PAST LENGTH
      043C  0004
0926                    *
0927                    *     TEST FOR LOADABLE MODUL
0928                    *
0929  043E  C041               MOV  FST,FST             FIRST TIME?
0930  0440  1616               JNE  FIRST               YES = PROCESS
0931  0442  C4C2               MOV  PTR,*NSMBLP
0932  0444  0417               BLWP *SRSM               NO, SEARCH FOR NAME
```

```
0933  0446  1008          JMP   FLCL              SYMBOL NOT FOUND = FLUSH
0934  0448  0620          DEC   @SYMBOL+FLAGS+FLAGS  PREVIOUSLY DEFINED?
      044A  057C'
0935  044C  131D          JEQ   MRKDF             NO, MARK AS DEF
0936                *
0937                *     PREVIOUSLY LOADED MODULE ERROR
0938                *
0939  044E  020A          LI    IOPARM,PRLM       PREV LOAD MODULE ERROR
      0450  2305
0940  0452  06A0          BL    @ERROR            CALL MONITOR ERROR PROCESSOR
      0454  026C'
0941  0456  05CE          INCT  R14               TAKE ERROR EXIT
0942                *
0943                *     FLUSH MODULE
0944                *
0945                FLCL
0946  0458  C080          MOV   RD,PTR            POINT TO BUFFER AREA
0947                LPFL
0948  045A  C280          MOV   RD,IOPARM
0949  045C  0420          BLWP  @READ             READ A RECORD
      045E  0612'
0950  0460  0249          ANDI  R9,EOF+IOERR      EOF OR I/O ERROR?
      0462  6000
0951  0464  1603          JNE   FLRT
0952  0466  9812          CB    *PTR,@COLON       FIRST CHAR = :
      0468  0003'
0953  046A  16F7          JNE   LPFL              NO= CONTINUE
0954                FLRT
0955  046C  0380          RTWP                    RETURN
0956                *
0957                *     PROCESS FIRST MODUL
0958                *
0959                FIRST
0960  046E  04C1          CLR   FST               CLEAR FIRST MODUL FLAG
0961  0470  0205          LI    TMP,6             INIT COUNTER
      0472  0006
0962  0474  6820          S     @TEN,@ENDST       SET NEW END OF SYMBOL TABLE
      0476  058A'
      0478  058C'
0963  047A  C120          MOV   @SYMTAB,SYMB      POINT TO SYMBOL NAME LOCATION
      047C  00AC'
0964  047E  0224          AI    SYMB,-7
      0480  FFF9
0965                LP
0966  0482  DD32          MOVB  *PTR+,*SYMB+      MOVE IN MODUL NAME
0967  0484  0605          DEC   TMP
0968  0486  15FD          JGT   LP
0969                *
0970                *     PROCESS ACCEPTABLE MODUL
0971                *
0972                MRKDF
0973  0488  05C2          INCT  PTR               SKIP LAST TWO CHARACTERS
0974  048A  C120          MOV   @SYMBOL+SYMPTR+SYMPTR,SYMB
      048C  0578'
0975  048E  04D4          CLR   *SYMB             MARK AS DEF
```

```
0976  0490  0224          AI    SYMB,-7             POINT TO NAME
      0492  FFF9
0977  0494  0205          LI    TMP,NAME            MOVE NAME TO OUTPUT LINE
      0496  0056'
0978  0498  CD74          MOV   *SYMB+,*TMP+
0979  049A  CD74          MOV   *SYMB+,*TMP+
0980  049C  CD74          MOV   *SYMB+,*TMP+
0981  049E  0420          BLWP  @GETSVL             POINT TO SYMBOL VALUE
      04A0  05A0'
0982  04A2  C120          MOV   @SYMBOL+SYMNM+SYMNM,SYMB
      04A4  057E'
0983  04A6  C506          MOV   LOC,*SYMB           MOVE IN LOC VALUE
0984  04A8  0420          BLWP  @BINHEX             CONVERT LOC FOR PRINT
      04AA  0540'
0985  04AC  0414'         DATA  LOCPTR              LOCATION VALUE
0986  04AE  005E'         DATA  LOCVAL              PRINTING LOCATION
0987  04B0  06A0          BL    @PRCRLF             PRINT CR/LF
      04B2  0366'
0988  04B4  1000          NOP                       IGNORE ERROR RETURN
0989  04B6  C288          MOV   NMMSG,IOPARM        PRINT MODULE NAME MESSAGE
0990  04B8  0420          BLWP  @PRINTN
      04BA  065C'
0991  04BC  022E          AI    R14,6               TAKE ALTERNATE RETURN
      04BE  0006
0992  04C0  0380          RTWP                      RETURN
```

```
0995                    *
0996                    *       CHECKSUM CHECK WORKSPACE AREA
0997                    *
0998                    CMWP
0999  04C2  04FC'              DATA CKLOC                R0    CKL
1000  04C4  0000               DATA 0                    R1    HVAL
1001  04C6  0408'              DATA LDBUF                R2    RDRL
1002  04C8  0000               DATA 0                    R3    CUMPT
1003  04CA  01D6'              DATA DSWP+INPT+INPT       R4    ENDOB
1004  04CC  055C'              DATA HEXBIN               R5    HEXP
1005  04CE  0000   RLVL        DATA 0                    R6    RLRG
1006  04D0  0000               DATA 0                    R7    CSUM
1007  04D2  0000   NWBIAS      DATA 0                    R8
1008  04D4  0000               DATA 0                    R9
1009  04D6  0000               DATA 0                    R10   IOPARM
1010  04D8  0000               DATA 0                    R11   RETURN
1011  04DA  0000   NSMBL       DATA 0                    R12
1012  04DC  0000               DATA 0                    R13   OLD WP
1013  04DE  0000               DATA 0                    R14   OLD PC
1014  04E0  0000               DATA 0                    R15   OLD ST
```

```
1016                    * TITLE:     CUMCHK
1017                    *            CHECK CHECKSUM VALUE
1018                    * REVISION: 05/01/74
1019                    *            ORIGINAL
1020                    *            03/15/76
1021                    *            MODIFIED TO RUN WITH PX9MTR
1022                    * COMPUTER: 990, ASSEMBLY
1023                    * ABSTRACT: THE CHECKSUM IS COMPUTED AND COMPARED
1024                    *            TO THE VALUE PRESENT ON THE OBJECT
1025                    *            RECORD. IF AN ERROR OCCURS, A MESSAGE
1026                    *            IS PRINTED AND KEYBOARD RESPONSE IS
1027                    *            REQUIRED BEFORE THE NEXT INPUT RECORD
1028                    *            MAY BE READ.
1029                    * CALLING SEQUENCE:
1030                    *            BLWP @CUMCHK
1031                    * STATISTICS: WORKSPACE = CMWP (UNSHARED)
1032                    * ROUTINES CALLED:
1033                    *            ERROR,HEXBIN,KEYIN,OPEN
1034                    CUMCHK
1035  04E2  04C2'               DATA CMWP                WORKSPACE
1036  04E4  04E6'               DATA CMBG                START
1037                    *
1038                    *      COMPUTE CHECKSUM
1039                    *
1040                    CMBG
1041  04E6  04C7               CLR  CSUM                CLEAR CHECKSUM
1042  04E8  C0C2               MOV  RDRL,CUMPT          POINT TO START OF RECORD
1043                    CKCOM
1044  04EA  04C1               CLR  HVAL                CLEAR VALUE
1045  04EC  D073               MOVB *CUMPT+,HVAL        GET ASCII CHARACTER
1046  04EE  06C1               SWPB HVAL                RIGHT JUSTIFY
1047  04F0  A1C1               A    HVAL,CSUM           ADD TO CHECKSUM
1048  04F2  8503               C    CUMPT,*ENDOB        FINISHED RECORD AREA?
1049  04F4  11FA               JLT  CKCOM               NO - CONTINUE
1050  04F6  0507               NEG  CSUM                YES - NEGATE VALUE FOR CHECKS
1051                    *
1052                    *      COMPARE TO VALUE ON RECORD
1053                    *
1054  04F8  C403               MOV  CUMPT,*CKL          POINT TO VALUE ON RECORD
1055  04FA  0415               BLWP *HEXP               CONVERT TO BINARY VALUE
1056  04FC  04FC' CKLOC        DATA $                   CONVERSION LOCATION
1057  04FE  04CE'              DATA RLVL                POSITION FOR RESULT
1058  0500  81C6               C    RLRG,CSUM           COMPARE THE TWO
1059  0502  130D               JEQ  OUTCK               EQUAL - EXIT
1060                    *
1061                    *      BAD CHECKSUM PROCESSOR
1062                    *
1063  0504  020A               LI   IOPARM,CKSM         BAD CHECKSUM ERROR
      0506  2306
1064  0508  06A0               BL   @ERROR              CALL MONITOR ERROR PROCESSOR
      050A  0454'
1065  050C  C282               MOV  RDRL,IOPARM         WAIT FOR RESPONSE
1066  050E  0420               BLWP @KEYIN                FROM KEYBOARD
      0510  061C'
```

```
1067  0512  C2A0          MOV  @CTLUNO,IOPARM    OPEN CASSETTE
      0514  0012'
1068  0516  0420          BLWP @OPEN
      0518  05FC'
1069  051A  020E          LI   14,TAGF
      051C  0206'
1070                OUTCK
1071  051E  0380          RTWP
```

```
1074                    *
1075                    *       CONVERSION ROUTINE WORKSPACE AREA
1076                    *
1077                    CONVWS
1078  0520  0000  BINVL  DATA 0              R0   BINVAL  BINARY VALUE
1079  0522  0000        DATA 0              R1   VALSV   TEMP VALUE SAVE
1080  0524  0000        DATA 0              R2   MADD    LOCN HEX VALUE
1081  0526  0000  SVCBLK DATA 0             R3   SCCODE  SUPV CALL CODE
1082  0528  0000  ASCVL  DATA 0             R4           ASCII VALUE
1083  052A  0000        DATA 0              R5
1084  052C  0000  LIST   DATA 0              R6
1085  052E  0C00        DATA >C00           R7   BHCODE  BINARY/HEX SCC
1086  0530  0D00        DATA >D00           R8   HBCODE  HEX/BINARY SCC
1087  0532  0000        DATA SVCALT         R9   SVCAL1  PX9MTR SUPV CALL ROUT
1088  0534  0526'       DATA SVCBLK         R10          SUPV CALL BLOCK PTR
1089  0536  FFFC        DATA -4             R11  HCNT
1090  0538  0000        DATA 0              R12  PCNT
1091  053A  0000        DATA 0              R13          OLD WP
1092  053C  0000        DATA 0              R14  CHOPC   OLD PC
1093  053E  0000        DATA 0              R15          OLD ST
```

```
1095                    * TITLE:     BINHEX/HEXBIN
1096                    *            CONVERSION ROUTINES
1097                    * REVISION: 05/01/74
1098                    *            ORIGINAL
1099                    *            03/15/76
1100                    *            MODIFIED TO USE PX9MTR CONVERSION ROUTINES
1101                    * COMPUTER: 990, ASSEMBLY
1102                    * ABSTRACT: SETS UP SUPERVISOR CALL BLOCK FOR CALL
1103                    *            AND STORES VALUES AFTER RETURN IN USER
1104                    *            DEFINED LOCATIONS.
1105                    * CALLING SEQUENCE:
1106                    *            BINARY TO HEX ASCII
1107                    *               BLWP @BINHEX
1108                    *               DATA ADDRESS-OF-VALUE
1109                    *               DATA ADDRESS-FOR-PLACEMENT
1110                    *
1111                    *            HEX ASCII TO BINARY
1112                    *               BLWP @HEXBIN
1113                    *               DATA ADDRESS-OF-VALUE
1114                    *               DATA ADDRESS-FOR-PLACEMENT
1115                    * ROUTINES CALLED: SVCALT (EXT REF IN PX9MTR)
1116                    *
1117                    BINHEX
1118  0540  0520'               DATA CONVWS              WORKSPACE
1119  0542  0544'               DATA BIHEX               START
1120                    BIHEX
1121  0544  C03E                MOV  *CHOPC+,BINVAL      POINT TO ADDRESS OF VALUE
1122  0546  C010                MOV  *BINVAL,BINVAL      GET VALUE IN R0 FOR CALL
1123  0548  C0C7                MOV  BHCODE,SCCODE       SET UP SUPV CALL CODE
1124  054A  0419                BLWP *SVCAL1             BINARY TO HEX SUPV CALL
1125  054C  C0BE                MOV  *CHOPC+,MADD        GET PLACEMENT LOC & NEW RTN
1126  054E  0201                LI   VALSV,ASCVL         SET UP ASCII VALUE POINTER
      0550  0528'
1127  0552  C30B                MOV  HCNT,PCNT           INIT COUNT TO 4
1128  0554  DCB1  BINH          MOVB *VALSV+,*MADD+      MOVE VALUE TO PLACEMENT LOCN
1129  0556  058C                INC  PCNT
1130  0558  11FD                JLT  BINH
1131  055A  0380                RTWP
1132                    *
1133                    HEXBIN
1134  055C  0520'               DATA CONVWS              WORKSPACE
1135  055E  0560'               DATA HEXBI               START
1136                    HEXBI
1137  0560  C0BE                MOV  *CHOPC+,MADD        GET ADDRESS OF VALUE
1138  0562  C0C8                MOV  HBCODE,SCCODE       SET UP SUPERVISOR CALL CODE
1139  0564  0201                LI   VALSV,ASCVL          SET UP ASCII VALUE POINTER
      0566  0528'
1140  0568  C30B                MOV  HCNT,PCNT           INIT COUNT TO 4
1141  056A  DC72  HEXB          MOVB *MADD+,*VALSV+      MOV HEX ASCII VALUE TO SCBLK
1142  056C  058C                INC  PCNT                CONTINUE UNTIL ALL
1143  056E  11FD                JLT  HEXB                  BYTES MOVED
1144  0570  0419                BLWP *SVCAL1             HEX TO BINARY SUPV CALL
1145  0572  C07E                MOV  *CHOPC+,VALSV       GET ADDR FOR PLACEMENT LOCN
1146  0574  C440                MOV  BINVAL,*VALSV       MOVE BINARY VALUE TO LOCN
```

1147 0576 0380 RTWP RETURN

```
1150                    SYMBOL
1151  057A  0000                DATA 0                  WR0     SYMPTR
1152  057A  0000                DATA 0                  WR1     SYMNO
1153  057C  0000                DATA 0                  WR2     FLAGS
1154  057E  0000                DATA 0                  WR3     SYMNM
1155  0580  0000                DATA 0                  WR4     NSYM
1156  0582  0000                DATA 0                  WR5     VALUE
1157  0584  0007  K7            DATA 7                  WR6     SEVEN
1158  0586  00AC'               DATA SYMTAB             WR7     STPTR
1159  0588  0006                DATA 6                  WR8     NWSCT
1160  058A  000A  TEN           DATA 10                 WR9     TN
1161  058C  0000  ENDST         DATA 0                  WR10    NSBIT
1162  058E  0000                DATA 0                  WR11    RETURN
1163  0590  0009                DATA 9                  WR12    NINE
1164  0592  0000                DATA 0                  WR13    OLD WP
1165  0594  0000                DATA 0                  WR14    OLD PC
1166  0596  0000                DATA 0                  WR15    OLD ST
```

```
1168                    * TITLE:     SENTRY
1169                    *            SYMBOL TABLE ENTRY VALUES
1170                    * REVISION: 05/01/74
1171                    *            ORIGINAL
1172                    * COMPUTER: 990, ASSEMBLY
1173                    * ABSTRACT: THREE ROUTINES ARE USED TO PROVIDE:
1174                    *            1. SYMBOL FLAG VALUE
1175                    *            2. POINTER TO SYMBOL NAME
1176                    *            3. POINTER TO VALUE AND VALUE
1177                    * CALLING SEQUENCE:
1178                    *            BLWP @GETSFL
1179                    *            BLWP @GETSVL
1180                    *            BLWP @GETSUM
1181                    * STATISTICS: WORKSPACE = SYMBOL (SHARED WITH SRCSYM)
1182                    *            SYMBOL TABLE ENTRY APPEARS AS FOLLOWS:
1183                    *                     S   Y
1184                    *                     M   B
1185                    *                     O   L
1186                    *                         0,1
1187                    *            WHERE 0 = DEFINED, 1 = UNDEFINED
1188                    *
1189                    *     GET SYMBOL FLAG
1190                    *
1191                    GETSFL
1192  0598  0578'               DATA SYMBOL             WORKSPACE
1193  059A  059C'               DATA GTSFL              START
1194                    GTSFL
1195  059C  C090                MOV  *SYMPTR,FLAGS      GET FLAGS WORD
1196  059E  0380                RTWP                    RETURN
1197                    *
1198                    *     GET SYMBOL VALUE
1199                    *
1200                    GETSVL
1201  05A0  0578'               DATA SYMBOL             WORKSPACE
1202  05A2  05A4'               DATA GTSVL              START
1203                    GTSVL
1204  05A4  C0C0                MOV  SYMPTR,SYMNM       POINT TO START OF SYMBOL ENTR
1205  05A6  60CC                S    NINE,SYMNM         POINT TO VALUE
1206  05A8  C153                MOV  *SYMNM,VALUE       GET SYMBOL VALUE
1207  05AA  0380                RTWP                    RETURN
1208                    *
1209                    *     GET SYMBOL NAME
1210                    *
1211                    GETSNM
1212  05AC  0578'               DATA SYMBOL             WORKSPACE
1213  05AE  05B0'               DATA GTSNM              START
1214                    GTSNM
1215  05B0  C0C0                MOV  SYMPTR,SYMNM       POINT TO SYMBOL ENTRY
1216  05B2  60C6                S    SEVEN,SYMNM        POINT TO NAME
1217  05B4  0380                RTWP                    RETURN
```

```
1220                    * TITLE:    SRCSYM
1221                    *           SEARCH SYMBOL TABLE
1222                    * REVISION: 05/01/74
1223                    *           ORIGINAL
1224                    *           03/15/76
1225                    *           MODIFIED TO RUN WITH PX9MTR
1226                    * COMPUTER: 990, ASSEMBLY
1227                    * ABSTRACT: THE SYMBOL TABLE IS SEARCHED FOR A
1228                    *           GIVEN SYMBOL NAME. IF THE SYMBOL
1229                    *           IS FOUND, AN ALTERNATE EXIT IS
1230                    *           TAKEN. THE TABLE IS SEARCHED LINEARLY.
1231                    * CALLING SEQUENCE:
1232                    *           BLWP @SRCSYM
1233                    *           SYMBOL-NOT-FOUND INSTRUCTION
1234                    *           SYMBOL-FOUND INSTRUCTION
1235                    * STATISTICS: WORKSPACE = SYMBOL (SHARED WITH SENTRY)
1236                    SRCSYM
1237  05B6  0578'               DATA SYMBOL             WORKSPACE
1238  05B8  05BA'               DATA SRC                START
1239                    *
1240                    *     POINT TO BEGINNING OF SYMBOL TABLE
1241                    *
1242                    SRC
1243  05BA  C017                MOV  *STPTR,SYMPTR      POINT TO SYMBOL TABLE
1244                    *
1245                    *     COMPARE TO ENTRY
1246                    *
1247                    SYMCHK
1248  05BC  C120                MOV  @NSMBL,NSYM        POINT TO NEW SYMBOL
      05BE  04DA'
1249  05C0  8280                C    SYMPTR,NSBIT       PAST END OF TABLE?
1250  05C2  1209                JLE  EXIT               YES - EXIT
1251  05C4  C0C0                MOV  SYMPTR,SYMNM       POINT TO ENTRY
1252  05C6  60C6                S    SEVEN,SYMNM        POINT TO NAME
1253  05C8  C048                MOV  NWSCT,SYMNO        GET COUNT
1254                    COMPAR
1255  05CA  9D33                CB   *SYMNM+,*NSYM+     ARE CHARACTERS THE SAME?
1256  05CC  1605                JNE  NOTSAM             NO - GET NEXT SYMBOL
1257  05CE  0601                DEC  SYMNO              ARE CHARACTERS FINISHED
1258  05D0  16FC                JNE  COMPAR             NO, KEEP COMPARING
1259  05D2  C090                MOV  *SYMPTR,FLAGS      YES - GET FOUND SYMBOL'S FLAG
1260  05D4  05CE                INCT 14                 TAKE ALTERNATE RETURN
1261                    EXIT
1262  05D6  0380                RTWP                    RETURN
1263                    *
1264                    *     LOOK AT NEXT SYMBOL
1265                    *
1266                    NOTSAM
1267  05D8  6009                S    TN,SYMPTR          POINT TO NEXT SYMBOL
1268  05DA  10F0                JMP  SYMCHK             GET NEXT SYMBOL
```

```
1271                    *
1272                    *       WORKSPACE AREA FOR INPUT/OUTPUT ROUTINES
1273                    *
1274                    IOWKS
1275  05DC  0900                DATA RDCOD          R0      RDCODP
1276  05DE  0800                DATA WRCOD          R1      WRCODP
1277  05E0              SCBLK   BSS  12             R2-R7   I/O SUPV CALL BLK (PRB)
1278  05EC  0050                DATA 80             R8      LEN
1279  05EE  0532'               DATA SVCALT         R9      SVCAL2
1280  05F0  05E0'               DATA SCBLK          R10     POINTER TO SUPV CALL BLOCK
1281  05F2                      BSS  10             R11-R15
```

```
1283                    * TITLE:     OPEN
1284                    *            ASSIGN A DEVICE TO A TASK
1285                    * REVISION: 03/15/76
1286                    *            ORIGINAL
1287                    * COMPUTER: 990, ASSEMBLY
1288                    * ABSTRACT: SETS UP PRB AND MAKES SUPERVISOR
1289                    *            CALL TO OPEN A DEVICE
1290                    * CALLING SEQUENCE:
1291                    *            R10 = LUNO
1292                    *            BLWP @OPEN
1293                    * STATISTICS: WORKSPACE = IOWKS (SHARED WITH I/O ROUTINES)
1294                    OPEN
1295  05FC  05DC'               DATA IOWKS,$+2          TRANSFER VECTOR
      05FE  0600'
1296  0600  04C3               CLR  IOPLUN             SET UP I/O OP
1297  0602  04C2               CLR  IOC                INIT CODE FOR I/O SUPV CALL
1298  0604  C30D               MOV  R13,R12            GET CALLER'S WORKSPACE PTR
1299  0606  022C               AI   R12,R10+R10        INDEX TO CALL PARAMETER
      0608  0014
1300  060A  C31C               MOV  *R12,R12           GET LUNO
1301  060C  E0CC               SOC  R12,IOPLUN         OR LUNO INTO IOPLUN
1302  060E  0419               BLWP *SVCAL2            MAKE SUPV CALL
1303  0610  0380               RTWP                    RETURN
```

```
1305                    * TITLE:    READ
1306                    *           READ A RECORD
1307                    * REVISION: 03/15/76
1308                    *           ORIGINAL
1309                    * COMPUTER: 990, ASSEMBLY
1310                    * ABSTRACT: SETS UP PRB AND MAKES SUPV CALL
1311                    *           TO READ A RECORD
1312                    * CALLING SEQUENCE:
1313                    *           R10 = BUFFER ADDRESS
1314                    *           BLWP @READ
1315                    * RETURN PARAMETERS:
1316                    *           R10 = BUFFER ADDR+CHAR COUNT
1317                    *           R9  = FLAGS
1318                    * STATISTICS: WORKSPACE = IOWKS (SHARED WITH I/O ROUTINES)
1319                    READ
1320  0612  05DC'               DATA IOWKS,$+2          TRANSFER VECTOR
      0614  0616'
1321  0616  C0E0              MOV  @CTLUNO,IOPLUN     STORE LUNO
      0618  0012'
1322  061A  1004              JMP  KEYRD
1323                    * TITLE:    KEYIN
1324                    *           KEYBOARD INPUT ROUTINE
1325                    * REVISION: 03/15/76
1326                    *           ORIGINAL
1327                    * COMPUTER: 990, ASSEMBLY
1328                    * ABSTRACT: SETS UP PRB AND MAKES SUPV CALL
1329                    *           TO INPUT A RECORD FROM KEYBOARD
1330                    * CALLING SEQUENCE:
1331                    *           R10 = BUFFER ADDRESS
1332                    *           BLWP @KEYIN
1333                    * RETURN PARAMETERS:
1334                    *           R10 = BUFFER ADDRESS+CHAR COUNT
1335                    *           R9  = FLAGS
1336                    * STATISTICS: WORKSPACE = IOWKS (SHARED WITH I/O ROUTINES)
1337                    KEYIN
1338  061C  05DC'               DATA IOWKS,$+2          TRANSFER VECTOR
      061E  0620'
1339  0620  C0E0              MOV  @KBLUNO,IOPLUN     STORE LUNO
      0622  000E'
1340                    KEYRD
1341  0624  D0C0              MOVB RDCODP,IOPLUN      STORE I/O OP
1342  0626  C30D              MOV  R13,R12            GET CALLER'S WORKSPACE PTR
1343  0628  022C              AI   R12,R10+R10        INDEX TO CALLING PARAMETERS
      062A  0014
1344  062C  C15C              MOV  *R12,BUFADR        SET UP BUFFER ADDRESS
1345  062E  C188              MOV  LEN,BUFLEN         SET UP BUFFER LENGTH
1346  0630  04C4              CLR  FLGS               INIT FLAGS
1347  0632  04C7              CLR  CHRCNT             INIT CHAR CNT TO ZERO
1348  0634  04C2              CLR  IOC                INIT CODE FOR I/O SUPV CODE
1349  0636  0419              BLWP *SVCAL2            MAKE SUPV CALL
1350  0638  A707              A    CHRCNT,*R12        INCR BUFF PTR TO END OF DATA
1351  063A  064C              DECT R12
1352  063C  C704              MOV  FLGS,*R12          MOVE FLAGS TO CALLING WS
1353  063E  0243              ANDI IOPLUN,>00FF       IS THIS KEYBOARD READ?
```

```
                0640  00FF
1354            0642  8803          C     IOPLUN,@KBLUNO
                0644  000E'
1355            0646  1301          JEQ   KEYPLF                IF READ, RETURN
1356            0648  0380          RTWP
1357                        KEYPLF
1358            064A  C0E0          MOV   @PRLUNO,IOPLUN        SETUP UP TO PRINT LF/CR
                064C  0010'
1359            064E  D0C1          MOVB  WRCODP,IOPLUN
1360            0650  0205          LI    BUFADR,LFCR
                0652  0006'
1361            0654  04C4          CLR   FLGS
1362            0656  100E          JMP   PRTKEY
```

```
1364                    * TITLE:    PRINT/PRINTN
1365                    *           PRINT A RECORD
1366                    * REVISION: 03/15/76
1367                    *           ORIGINAL
1368                    * COMPUTER: 990, ASSEMBLY
1369                    * ABSTRACT: SETS UP PRB AND MAKES SUPV CALL
1370                    *           TO PRINT A RECORD
1371                    *           PRINT = PRINTS RECORD WITH CR
1372                    *           PRINTN= PRINTS RECORD WITHOUT CR
1373                    * CALLING SEQUENCE:
1374                    *           R10 = BUFFER ADDRESS
1375                    *           BLWP @PRINT
1376                    *               OR
1377                    *           BLWP @PRINTN
1378                    *
1379                    * STATISTICS: WORKSPACE = IOWKS (SHARED WITH I/O ROUTINES)
1380                    PRINT
1381  0658  05DC'               DATA IOWKS,PRTCR        TRANSFER VECTOR
      065A  0664'
1382                    PRINTN
1383  065C  05DC'               DATA IOWKS,PRTNCR       TRANSFER VECTOR
      065E  0660'
1384                    PRTNCR
1385  0660  0704               SETO FLGS               SET FLAG FOR PRINT WITHOUT CR
1386  0662  1001               JMP PRENT
1387                    PRTCR
1388  0664  04C4               CLR  FLGS               SET FLAG FOR PRINT WITH CR
1389                    PRENT
1390  0666  C0E0               MOV  @PRLUNO,IOPLUN     STORE LUNO
      0668  0010'
1391                    PRWRT
1392  066A  D0C1               MOVB WRCODP,IOPLUN      STORE I/O OP
1393  066C  C30D               MOV  R13,R12            GET CALLER'S WP
1394  066E  022C               AI   R12,R10+R10        INDEX TO CALLING PARAMETERS
      0670  0014
1395  0672  C15C               MOV  *R12,BUFADR        SET UP BUFFER ADDR
1396                    PRTKEY
1397  0674  06A0               BL   @GETCNT            GET CHAR COUNT
      0676  068C'
1398  0678  A1C4               A    FLGS,CHRCNT        ADJUST CHRCNT FOR CR
1399  067A  04C4               CLR  FLGS               INIT FLAGS
1400  067C  04C2               CLR  IOC                INIT CODE FOR I/O SUPV CALL
1401  067E  0419               BLWP *SVCAL2            MAKE SUPV CALL
1402  0680  A707               A    CHRCNT,*R12        INCREMENT PRINT BUFF TO END
1403  0682  C082               MOV  IOC,IOC            CHECK RETURN STATUS
1404  0684  1302               JEQ  PREXT              ZERO, EXIT
1405  0686  0460               B    @ASK               ESC CNTL RETURN OR ERROR
      0688  012C'
1406                    PREXT
1407  068A  0380               RTWP                    RETURN
```

```
 1409                   * TITLE:    GETCNT
 1410                   *           GET CHARACTER COUNT
 1411                   * REVISION: 03/15/76
 1412                   *           ORIGINAL
 1413                   * COMPUTER: 990,ASSEMBLY
 1414                   * ABSTRACT: SCANS LINE COUNTING CHARACTERS UP TO
 1415                   *           AND INCLUDING A CARRIAGE RETURN.
 1416                   * CALLING SEQUENCE:
 1417                   *           R10 = BUFFER ADDRESS
 1418                   *           BL   @GETCNT
 1419                   * RETURN PARAMETERS:
 1420                   *           CHRCNT(R7) = CHARACTER COUNT
 1421                   * STATISTICS: WORKSPACE = IOWKS
 1422                   GETCNT
 1423  068C  04C7              CLR  CHRCNT            INIT CHAR COUNT
 1424                   GCLOOP
 1425  068E  0587              INC  CHRCNT            INCR CHAR COUNT
 1426  0690  9835              CB   *BUFADR+,@CR      COMPARE CHAR TO CR
       0692  0007'
 1427  0694  1303              JEQ  GCEXT              IF EQUAL, EXIT
 1428  0696  0287              CI   CHRCNT,80         MORE THAN 80 CHARS?
       0698  0050
 1429  069A  11F9              JLT  GCLOOP             NO, CONTINUE
 1430                   GCEXT
 1431  069C  6147              S    CHRCNT,BUFADR     RESET BUFFER ADDR TO START
 1432  069E  0450              B    *RTN               RETURN
 1433                           END
0000 ERS
```

960 - 980 CONCORDANCE

```
$                0588  0823  0854  0855  0923  1056  1295  1320  1338
ASCVL   1082     1126  1139
ASK     0375     0370  0444  0543  0612  1405
BATCH   0398     0400
BHCODE  0103     1123
BIAS    0061     0548  0607  0636  0637  0645  0658  0730  0795  0810
BIASWD  0458     0353  0833
BIHEX   1120     1119
BINH    1128     1130
BINHEX  1117     0435  0743  0984
BINVAL  0099     1121  1122  1122  1146
BINVL   1078
BKOUT   0634     0573
BLANK   0180     0619
BUFADR  0132     1344  1360  1395  1426  1431
BUFLEN  0133     1345
CHLP    0547     0602  0630  0653  0674  0689  0699
CHLP2   0768     0778  0790  0805  0825  0834
CHLP3   0698     0719  0769
CHOPC   0108     1121  1125  1137  1145
CHRCNT  0134     1347  1350  1398  1402  1423  1425  1428  1431
CKCOM   1043     1049
CKL     0088     1054
CKLOC   1056     0999
CKSM    0165     1063
CKSYMB  0407     0429
CMBG    1040     1036
CMWP    0998     1035
COLON   0183     0952
COMPAR  1254     1258
CONLOC  0855     0851
CONVL   0854     0852
CONVRT  0850     0616  0832  0840
CONVWS  1077     1118  1134
CR      0179     1426
CSUM    0095     1041  1047  1050  1058
CTLUNO  0199     0389  0394  0396  1067  1321
CUMCHK  1034     0784
CUMPT   0091     1042  1045  1048  1054
DEFL    0211     0745
DEFM    0208     0754
DEFN    0209     0746  0749
DLDBI   0204     0351
DLDPT   0203     0342
DSBG    0532     0531
DSPEXT  0638     0611
DSPTCH  0529     0399
DSWP    0455     0533  1003
DSWP2   0459     0530  0639
END     0186     0385
ENDACT  0247     0439
ENDLPR  0685     0682
ENDOS   0092     1048
ENDPRO  0404     0386
ENDST   1161     0276  0717  0718  0815  0962
ENTADD  0279     0651  0652
ENTB    0433     0409
ENTERD  0705     0759
ENTERR  0703     0671
ENTMSG  0241     0278
```

```
                    0711  0712  0713  0714  0715  0716  0717  0772  0773
                    0774  0775  0776
SYMB        0079    0963  0964  0966  0974  0975  0976  0978  0979  0980
                    0982  0983
SYMBOL      1150    0277  0287  0420  0676  0677  0706  0761  0772  0775
                    0934  0974  0982  1192  1201  1212  1237
SYMCHK      1247    1268
SYMNM       0115    0420  0420  0676  0676  0772  0772  0982  0982  1204
                    1205  1206  1215  1216  1251  1252  1255
SYMNO       0113    1253  1257
SYMP        0047    0358  0406  0408  0428
SYMPTR      0112    0277  0277  0706  0706  0775  0775  0974  0974  1195
                    1204  1215  1243  1249  1251  1259  1267
SYMTAB      0282    0963  1158
TAG0        0606    0563
TAG0B       0615    0609
TAG0C       0626    0620
TAG0CN      0628    0614
TAG1        0649    0564
TAG2        0644    0565
TAG3        0657    0566
TAG4        0662    0567
TAG5        0729    0568
TAG6        0734    0569
TAG7        0783    0570
TAG8        0788    0571
TAG9        0799    0572
TAGA        0794    0580
TAGB        0814    0581
TAGC        0809    0582
TAGD        0829    0583
TAGE        0592    0553  0556  0574  0575  0576  0577  0578  0579  0584
TAGF        0538    0585  1069
TAGGH       0600    0586  0587
TDAT        0071    0718  0817  0819
TEMP        0044    0391  0393  0394  0421  0422  0423  0424
TEN         1160    0428  0962
TERM        0187    0443
TGFLD       0060    0551  0552  0554  0555  0557  0557  0558  0741  0746
                    0748  0749
TMP         0080    0961  0967  0977  0978  0979  0980
TN          0121    1267
TOPDAT      0283    0472
TPPT        0052    0360
UDS         0218    0425
UDSN        0219    0421
UNDEF       0216    0415
UNDPT       0053    0405  0413  0413  0417
USRPC               0939  0448
VALSV       0100    1126  1128  1139  1141  1145  1146
VALUE       0117    1206
WRCOD       0192    1276
WRCODP      0128    1359  1392

THERE ARE   0280  SYMBOLS
```

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																															
SHEET																															
REV STATUS OF SHEETS	REV																														
	SHEET																														

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, PROMPG, PROM PROGRAMMER, PART 1,-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945391-9901

SCALE	REV.	SHEET
		1 of 75

TI-0923-C

```
0003                    IDT  'PROMPG'
0004                *  TITLE:    PROMPG
0005                *            PROM PROGRAMMER
0006                *  REVISION: 03/15/76
0007                *            ORIGINAL
0008                *  COMPUTER: 990,ASSEMBLY
0009                *  ABSTRACT:
0010                *            INTERPRET A PROM PROGRAMMER COMMAND STRING AND
0011                *            EXECUTE COMMAND WITH ONE OF TWELVE SUBCOMMAND
0012                *            ROUTINES
0013                *
0014                *            TABLE OF CONTENTS:
0015                *                I. LEVEL ONE ROUTINES
0016                *                   A. COMMAND STRING INTERPRETORS
0017                *                      1. PROM PROGRAMMER       (PPCSI)
0018                *                      2. PROM STANDARD         (PSCSI)
0019                *                   B. SUBCOMMANDS
0020                *                      1. MEMORY BOUNDS         (BOUNDS)
0021                *                      2. ROM BOUNDS            (BOUNDS)
0022                *                      3. MEMORY IMAGE          (IMAGE)
0023                *                      4. ROM IMAGE             (IMAGE)
0024                *                      5. STRING WIDTH          (SW)
0025                *                      6. ROM CHARACTERISTICS   (RC)
0026                *                      7. ROM CRU BASE ADDRESS  (CS)
0027                *                      8. TOGGLES               (TS)
0028                *                      9. GO                    (GO)
0029                *                II. LEVEL TWO ROUTINES
0030                *                   A. SAVE STANDARD IMAGE      (SVSTIM)
0031                *                   B. DISPLAY STRING           (DISPLY)
0032                *                   C. GET ROM BURN STRING      (BROM)
0033                *                   D. GET STRING FROM ROM      (RGETFD)
0034                *                   E. CONVERT HEX TO BINARY    (CVTHXB)
0035                *               III. LEVEL THREE ROUTINES
0036                *                   A. BURN WORD                (BWORD)
0037                *                   B. CONVERT AND WRITE        (CVTWRT)
0038                *                   C. PRCRLF,BLNK,PRNTC,GETFLD (PFBPCG)
0039                *                IV. LEVEL FOUR ROUTINES
0040                *                   A. WRITE/READ               (WRTRD)
0041                *                   B. CONVERT BINARY TO HEX    (CVTBHX)
0042                *                 V. LEVEL FIVE ROUTINES
0043                *                   A. CT ROUTINE               (CTRT)
0044                *                   B. BBA ROUTINE              (BBART)
0045                *                   C. LOAD CRU                 (LOADCR)
0046                *                   D. GET STRING FROM MEMORY   (MGETFD)
0047                *                   E. ROM TO MEMORY            (RTOM)
0048                *                   F. BLANK FILL BUFFER        (BLNK)
0049                *                   G. CHECK FOR GIVEN REORD    (CHECK)
0050                *                   H. BURN CYCLE               (BNCYL)
0051                *  CALLING SEQUENCE:
0052                *            PPSCI PASSES PARAMETERS TO ALL
0053                *            SUBCOMMAND ROUTINES IN R9 THROUGH R3.
0054                *            ALL CALLS TO LEVEL TWO THROUGH LEVEL FIVE
0055                *            ROUTINES PASS PARAMETERS ACCORDING TO
0056                *            PX990 STANDARDS (R10 TOWARD R0).
```

```
0057                *
0058                *       RULES FOR SUBROUTINE REGISTER USAGE ARE:
0059                *           1. R0,R1,R14,R15 ARE VOLATILE AND MAY BE
0060                *              USED BY ANY SUBROUTINE
0061                *           2. LEVEL 1 SAVES RETURN ADDRESS IN R2
0062                *              LEVEL 2 SAVES RETURN ADDRESS IN R3
0063                *              LEVEL 3 SAVES RETURN ADDRESS IN R4
0064                *              LEVEL 4 SAVES RETURN ADDRESS IN R5
0065                *              LEVEL 5 SAVES RETURN ADDRESS IN R11
0066                *              THUS LEVEL 1 MAY USE R3-R5, LEVEL 2 R4-
0067                *              R5, AND LEVEL 3 R5 AS VOLATILE STORAGE
```

```
0069                            DEF   PPCSI
0070  0000    50                TEXT  'PP'
0071  0002  0114'               DATA  PPCSI
0072        0004' START         EQU   $
0073                            DEF   PSCSI
0074  0004    50                TEXT  'PS'
0075  0006  055E'               DATA  PSCSI
0076  0008  0000                DATA  0
0077              *                               EXTERNAL REFS AND DEFS
0078                            DEF   PGBIAS
0079                            DEF   PROGSZ
0080                            REF   LDBUF
0081                            REF   ERROR
0082                            REF   ACL
0083                            REF   RR
0084                            REF   PRCRLF
0085                            REF   PRNTC
0086                            REF   PRNTHB
0087                            REF   PRNTHN
0088                            REF   PRTPRD
0089                            REF   EQSIGN
0090                            REF   PRNTHX
0091                            REF   RETBUF
0092                            REF   SVCALT
0093                            REF   GETFLD
0094                            REF   ESC
0095                            REF   DMPSPR
0096              *                               WORKING TABLE
0097              *                               TEMPORARY MEMORY IMAGE
0098        000A' MTEMP EQU   $
0099              *                                         STRING WIDTH
0100        000A' SWT   EQU   $
0101  000A  0001                DATA  1
0102              *                                         INCREMENTS
0103        000C' IMT   EQU   $
0104  000C  0000                DATA  0
0105  000E  0000                DATA  0
0106  0010  0000                DATA  0
0107              *                                         DISPLACEMENTS
0108        0012' DMT   EQU   $
0109  0012  0000                DATA  0
0110  0014  0000                DATA  0
0111  0016  0000                DATA  0
0112              *                                         MAXIMUMS
0113        0018' MMT   EQU   $
0114  0018  0001                DATA  1
0115  001A  0001                DATA  1
0116  001C  0001                DATA  1
0117              *                               DIFFERENCE BETWEEN TWO IMAGE PARAMETERS
0118        0006  IMGDIF EQU   DMT-IMT
0119        000A  MSZ   EQU   $-MTEMP/2
0120              *                               TEMPORARY ROM IMAGE
0121        001E' RTEMP EQU   $
0122              *                                         STRING WIDTH
```

```
0123          001E' SWTR    EQU  $
0124  001E   0001          DATA 1
0125                *                              INCREMENTS
0126          0020' IRT     EQU  $
0127  0020   0000          DATA 0
0128  0022   0000          DATA 0
0129  0024   0000          DATA 0
0130                *                              DISPLACEMENTS
0131          0026' DRT     EQU  $
0132  0026   0000          DATA 0
0133  0028   0000          DATA 0
0134  002A   0000          DATA 0
0135                *                              MAXIMUMS
0136          002C' MRT     EQU  $
0137  002C   0001          DATA 1
0138  002E   0001          DATA 1
0139  0030   0001          DATA 1
0140                *                        ROM CHARACTERISTICS
0141                *                              ROM WORD WIDTH
0142          0032' WWID    EQU  $
0143  0032   0001          DATA 1
0144                *                              PROGRAM TYPE (0'S OR 1'S)
0145          0034' POOZ    EQU  $
0146  0034   0000          DATA 0
0147                *                              PULSE WIDTH
0148          0036' PW1     EQU  $
0149  0036   0001          DATA 1
0150                *                              NUMBER OF RETRIES
0151          0038' NR      EQU  $
0152  0038   0000          DATA 0
0153                *                              DUTY CYCLE
0154          003A' DC      EQU  $
0155  003A   0019          DATA 25
0156                *                              PROGRAMMABLE STRING WIDTH
0157          003C' PSTW    EQU  $
0158  003C   0001          DATA 1
0159                *                        MEMORY BOUNDS
0160          0010  RSZ     EQU  $-RTEMP/2
0161                *                              BEGINNING MEMORY ADDRESS
0162          003E' BMA     EQU  $
0163  003E   0000          DATA 0
0164                *                              ENDING MEMORY ADDRESS
0165          0040' EMA     EQU  $
0166  0040   0FFF          DATA >0FFF
0167                *                        ROM BOUNDS
0168                *                              BEGINNING ROM ADDRESS
0169          0042' BCA     EQU  $
0170  0042   0000          DATA 0
0171                *                              ENDING ROM ADDRESS
0172          0044' ECA     EQU  $
0173  0044   0FFF          DATA >0FFF
0174                *                        TOGGLES
0175                *                              COMPARE STRINGS
0176          0046' CMPR    EQU  $
0177  0046   0001          DATA 1
```

```
0178                    *
0179          0048'     DIRECT  EQU   $
0180    0048  0001              DATA  1
0181                    *
0182          004A'     DISPR   EQU   $
0183    004A  0000              DATA  0
0184                    *
0185          004C'     DISPM   EQU   $
0186    004C  0000              DATA  0
0187                    *
0188          004E'     RCRUIA  EQU   $
0189    004E  0020              DATA  >0020
0190                    *
0191    0050  0000      PGBIAS  DATA  0
0192          0052'     PROGSZ  EQU   $
0193    0052  0380              DATA  >0380
0194                    *
0195                    *
0196          3101      PPRQ    EQU   >3101
0197                    *
0198          3102      PPOB    EQU   >3102
0199                    *
0200          3103      PPNM    EQU   >3103
0201                    *
0202          3104      PPBA    EQU   >3104
0203                    *
0204          3105      PPHW    EQU   >3105
0205                    *
0206          3106      PPOL    EQU   >3106
0207                    *
0208          0000      R0      EQU   0
0209          0001      R1      EQU   1
0210          0002      R2      EQU   2
0211          0003      R3      EQU   3
0212          0004      R4      EQU   4
0213          0005      R5      EQU   5
0214          0006      R6      EQU   6
0215          0007      R7      EQU   7
0216          0008      R8      EQU   8
0217          0009      R9      EQU   9
0218          000A      R10     EQU   10
0219          000B      R11     EQU   11
0220          000C      R12     EQU   12
0221          000D      R13     EQU   13
0222          000E      R14     EQU   14
0223          000F      R15     EQU   15
0224                    *
0225          0054'     CONPRB  EQU   $
0226                    *
0227    0054  0000      CPRBOP  DATA  0
0228                    *
0229    0056    00              BYTE  0,0
        0057    00
0230    0058    00              BYTE  0,0
        0059    00
```

```
                        BURN,ROM=TO=MEMORY,SAVE IMAGE

                        DISPLAY ROM

                        DISPLAY MEMORY

ROM CRU INTERFACE ADDRESS

LOAD BIAS AND SIZE DORG PROGRAM SEGMENT

COMMAND PARAMETER ERRORS
                        REQUIRED PARAMETER MISSING

                        OUT=OF=BOUNDS

                        NO MATCH

                        BAD ADDRESS

                        HARDWARE

                        ONLINE

DEFINE REGISTERS

CONVERSION PRB DEFINITIONS

                        I/O COMMAND

                        CONVERSION BUFFER
```

```
0231                    *                               HEX ASCII TO BINARY OPCODE
0232          005A'     HXB     EQU   $
0233  005A    0D00              DATA  >0D00
0234                    *                               BINARY TO HEX ASCII OPCODE
0235          005C'     BHX     EQU   $
0236  005C    0C00              DATA  >0C00
0237                    *                         I/O ERRORS
0238          0001      MX01    EQU   >01                 UNRECOVERABLE I/O ERROR
0239                    *                         SVC/PRB EQUATES
0240          0000      OPEN    EQU   00                  OPEN FILE
0241          000D      WEOF    EQU   13                  WRITE END-OF-FILE
0242          0005      WRITA   EQU   11                  WRITE TO FILE
0243          0009      READA   EQU   9                   READ FROM FILE
0244                    *                         PRB DEFINITIONS
0245          005E'     PRB     EQU   $
0246  005E    0000      PRBSC   DATA  0                   SERVICE TYPE
0247  0060      00      PRBOP   BYTE  0                   I/O COMMAND
0248  0061      07      PRBLU   BYTE  7                   LUNO
0249  0062      00      PRBSF   BYTE  0                   SYSTEM COMPLETION FLAGS
0250  0063      00      PRBUF   BYTE  0                   USER FLAGS
0251  0064    0000      PRBBF   DATA  LDBUF               BUFFER
0252  0066    0050      PRBLN   DATA  80                  OUTPUT BUFFER LENGTH
0253  0068    0050      PRBCC   DATA  80                  CHARACTER COUNT
0254  006A    0000              DATA  0,0                 UNUSED
      006C    0000
0255                    *                         MISCELLANEUS DATA
0256                    *                               CHARACTER COUNT
0257          006E'     CHARCT  EQU   $
0258  006E    0050              DATA  >0050
0259          004D      M       EQU   'M'
0260          0052      R       EQU   'R'
0261                    *                               END-OF-FILE MASK
0262          2000      EOFMSK  EQU   >2000
0263          0070'     MEMIMG  EQU   $
0264  0070      07              BYTE  7
0265  0071      4D              TEXT  'MEM ID?'
0266          0078'     ROMIMG  EQU   $
0267  0078      07              BYTE  7
0268  0079      52              TEXT  'ROM ID?'
0269  0080      02      BLANKS  BYTE  2
0270  0081      20              TEXT  '  '
0271                            EVEN
0272          0084'     EIGHT   EQU   $
0273  0084    0008              DATA  8
0274          0086'     SEVEN   EQU   $
0275  0086    0007              DATA  7
0276                    *                         MAXIMUM DUTY CYCLE
0277          0064      MAXDC   EQU   100
0278                    *                         MAXIMUM DUTY CYCLE COUNT
0279          FDE8      MAXCT   EQU   65000
0280                    *                         MAXIMUM PULSE WIDTH
0281          0006      MAXPW1  EQU   6
0282                    *                         MAXIMUM STRING WIDTH
0283          0008      MAXWD   EQU   8
0284                    *                         MAXIMUM CRU INTERFACE ADDRESS
```

```
0285          1FFE  MAXCRU EQU  >1FFE
0286                *                     CRU INTERFACE ROM REGISTER SELECTS
0287          0000  ADDR0  EQU  >0000
0288          0100  ADDR1  EQU  >0100
0289          0300  DIN1   EQU  >0300
0290          0500  PWD0   EQU  >0500
0291  0088    0700  DOUT1  DATA >700
0292                *                     WORD TYPE=MEMORY OR ROM
0293          014D  MPRT   EQU  >100+'M'
0294          0152  RPRT   EQU  >100+'R'
0295                *                     BUFFER RETURNS
0296          008A' BUFRT1 EQU  $
0297  008A    0000         DATA 0
0298          008C' BUFRT2 EQU  $
0299  008C    0000         DATA 0
0300                *                     TOGGLE COMPARISONS
0301          0001  YES    EQU  1
0302          0001  MEMROM EQU  1
0303          0002  ROMMEM EQU  2
0304          0003  SAVIMG EQU  3
0305                *                     CRU INTERFACE BITS
0306          000F  GOBIT  EQU  15
0307          000D  LD     EQU  13
0308          000E  IE     EQU  14
0309          000E  OL     EQU  14
0310          000F  BUSY   EQU  15
0311                *                     PARAMETER DEFAULTS
0312          008E' DEFTS1 EQU  $
0313  008E    0000         DATA 0,1,0
      0090    0001
      0092    0000
0314          0094' DEFTS2 EQU  $
0315  0094    0000         DATA 0,25,1
      0096    0019
      0098    0001
0316          009A' DEFTS3 EQU  $
0317  009A    FFFF         DATA -1,-1,-1,-1
      009C    FFFF
      009E    FFFF
      00A0    FFFF
0318                *                     COMMAND TABLE FOR PROCESSING COMMAND
0319                *                           PARAMETER LIST
0320                *                     COMMAND IMPLEMENTATION ROUTINES
0321                *                     NUMBER OF PARAMETERS,PARAMETER BIT MASK
0322                *                     ADDRESSES OF DEFAULT VALUES
0323                *                     PARAMETERS PASSED TO SUBCOMMANDS
0324          00A2' CNAME  EQU  $
0325          00A2' CMND1  EQU  $
0326  00A2    4353         DATA 'CS'
0327  00A4      01         BYTE 1,>80
      00A5      80
0328  00A6    0000         DATA 0
0329  00A8    00A6         DATA CS
0330  00AA    004E'        DATA RCRUIA
0331          000A  CMDLEN EQU  $-CMND1
```

```
0332  00AC  474F          DATA 'GO'
0333  00AE    00          BYTE 0,0
      00AF    00
0334  00B0  0000          DATA 0
0335  00B2  019C'         DATA GO
0336  00B4  0000          DATA 0
0337  00B6  4D42          DATA 'MB'
0338  00B8    02          BYTE 2,>C0
      00B9    C0
0339  00BA  0000          DATA 0
0340  00BC  0038          DATA BOUNDS
0341  00BE  003E'         DATA BMA
0342  00C0  4D49          DATA 'MI'
0343  00C2    04          BYTE 4,>80
      00C3    80
0344  00C4  008E'         DATA DEFTS1
0345  00C6  0042          DATA IMAGE
0346  00C8  000C'         DATA IMT
0347  00CA  5242          DATA 'RB'
0348  00CC    02          BYTE 2,>C0
      00CD    C0
0349  00CE  0000          DATA 0
0350  00D0  0038          DATA BOUNDS
0351  00D2  0042'         DATA BCA
0352  00D4  5243          DATA 'RC'
0353  00D6    06          BYTE 6,>E0
      00D7    E0
0354  00D8  0094'         DATA DEFTS2
0355  00DA  0074          DATA RC
0356  00DC  0032'         DATA WWID
0357  00DE  5249          DATA 'RI'
0358  00E0    04          BYTE 4,>80
      00E1    80
0359  00E2  008E'         DATA DEFTS1
0360  00E4  0042          DATA IMAGE
0361  00E6  0020'         DATA IRT
0362  00E8  5357          DATA 'SW'
0363  00EA    01          BYTE 1,>80
      00EB    80
0364  00EC  0000          DATA 0
0365  00EE  0060          DATA SW
0366  00F0  00FC'         DATA SWBLK
0367  00F2  5453          DATA 'TS'
0368  00F4    04          BYTE 4,0
      00F5    00
0369  00F6  009A'         DATA DEFTS3
0370  00F8  0082          DATA TS
0371  00FA  0046'         DATA CMPR
0372        00FC' CMN2    EQU  $
0373              *                   TWO PARMS PASSED TO SW,LOCATED IN SWBLK
0374        00FC' SWBLK   EQU  $
0375  00FC  000A'         DATA SWT
0376  00FE  001E'         DATA SWTR
0377              *                   STORAGE FOR LOOP COUNTERS FOR COMPUTING
0378              *                   BIT ADDRESSES
```

```
0379          0100' CT      EQU  $
0380  0100    0000          DATA 0,0,0
      0102    0000
      0104    0000
0381                *                    BEGINNING MEMORY BYTE ADDRESS
0382          0106' BMBYA   EQU  $
0383  0106    0000          DATA 0
0384                *                    BEGINNING BIT WITHIN BYTE
0385          0108' BMBYBT  EQU  $
0386  0108      00          BYTE 0
0387          0109' BMBT2   EQU  $
0388  0109      00          BYTE 0
0389                *                    BEGINNING ROM WORD ADDRESS
0390          010A' BRWA    EQU  $
0391  010A    0000          DATA 0
0392                *                    BEGINNING BIT WITHIN WORD
0393          010C' BRWDBT  EQU  $
0394  010C      00          BYTE 0
0395          010D' BRBT2   EQU  $
0396  010D      00          BYTE 0
0397                *                    ENDING ROM WORD ADDRESS
0398          010E' ERWA    EQU  $
0399  010E    0000          DATA 0
0400                *                    MEMORY AND ROM STRINGS
0401          0110' MSTRG   EQU  $
0402  0110      00          BYTE 0
0403          0111' MSTRG2  EQU  $
0404  0111      00          BYTE 0
0405          0112' RSTRG   EQU  $
0406  0112      00          BYTE 0
0407          0113' RSTRG2  EQU  $
0408  0113      00          BYTE 0
```

```
0411                    * TITLE:     PPCSI
0412                    *            PROM PROGRAMMER COMMAND STRING INTERPRETER
0413                    * REVISION: 03/15/76
0414                    *            ORIGINAL
0415                    * COMPUTER: 990,ASSEMBLY
0416                    * ABSTRACT:
0417                    *            CHECK FOR MISSING REQUIRED PARAMETERS FOR
0418                    *            COMMAND, FIND SUBCOMMAND IN COMMAND TABLE,
0419
0420
0421                    *            OPTIONAL PARAMETER MISSING GET DEFAULT VALUE
0422                    * CALLING SEQUENCE:
0423                    *            ENTRY:
0424                    *            R10 CONTAINS ADDRESS OF COMMAND PROCESSOR LIST
0425                    *            EXIT:
0426                    *            CONTROL PASSED DIRECTLY TO SUBCOMMAND ROUTINE
0427                    *            THROUGH R11
0428                    *            R3 CONTAINS ADDRESS OF SUBCOMMAND PARAMETER
0429                    *            R9-R4 CONTAIN SUBCOMMAND PARAMETER VALUES
0430          0114' PPCSI   EQU   $
0431                    *                        SAVE RETURN ADDRESS
0432  0114    C08B            MOV   R11,R2
0433                    *                        PRESBT=LSHFT(CPL(1),9)
0434  0116    C27A            MOV   *R10+,R9
0435  0118    0A99            SLA   R9,9
0436                    *                        IF(PARM(1).EQ.' ')CALL ERR
0437  011A    1732            JNC   REQMIS
0438                    *                        WD1=LSHFT(CPL(2),8)
0439  011C    C0FA            MOV   *R10+,R3
0440  011E    D203            MOVB  R3,R8
0441                    *                        WD2=RSHFT(CPL(3),8)
0442  0120    C13A            MOV   *R10+,R4
0443                    *                        CMND=OR(WD1,WD2)
0444  0122    D0C4            MOVB  R4,R3
0445  0124    06C3            SWPB  R3
0446                    *                        INCREMENT CPL PAST EXTRA CHARS
0447  0126    0998            SRL   R8,9
0448  0128    0A18            SLA   R8,1
0449  012A    0648            DECT  R8
0450  012C    A288            A     R8,R10
0451                    *                              FIND COMMAND IN TABLE AND USE
0452                    *                              TABLE VALUES TO CHECK THAT
0453                    *                              REQUIRED PARAMETERS FOR
0454                    *                              COMMAND ARE GIVEN AND TO
0455                    *                              DEFAULT OPTIONAL PARAMETERS
0456                    *                              THAT ARE OMITTED
0457                    *                        DO 1 J=1,CTBLSZ
0458                    *                        IF(CMND.EQ.CNAME(J))GO TO 19
0459  012E    0205            LI    R5,CNAME
      0130    00A2'
0460          0132' PPFLD1  EQU   $
0461                    *                        J=J+1
0462  0132    8043            C     R3,*R5+
0463  0134    1306            JEQ   PPFLD2
```

```
0464                    *                       J=J+3
0465  0136  0225        PPFLD1  AI    R5,CMDLEN-2
      0138  0008
0466  013A  0285                CI    R5,CMN2
      013C  00FC'
0467                    *                       IF(J.NE.CTBLSZ)CONTINUE
0468  013E  16F9                JNE   PPFLD1
0469                    *                       CALL ERR
0470  0140  1025                JMP   BADADD
0471        0142' PPFLD2  EQU   $
0472                    *                       R12=CNAME(J+1)  NO. OF PARMS.
0473  0142  D335                MOVB  *R5+,R12
0474                    *                       R14=CNAME(J+2)  PARM. BIT MASK
0475  0144  D3B5                MOVB  *R5+,R14
0476                    *                       R15=CNAME(J+3)  DEFAULTS FOR PARMS.
0477  0146  C3F5                MOV   *R5+,R15
0478                    *                       R11=CNAME(J+4)  COMMAND ROUTINE
0479  0148  C2F5                MOV   *R5+,R11
0480                    *                               ALL SUBCOMMAND ROUTINES
0481                    *                               EXCEPT GO ARE IN PROGRAM
0482                    *                               EXTENSION AND ARE LOCATED BY
0483                    *                               ADDING EXTENSION PROGRAM BIAS
0484                    *                               TO SUBCOMMAND ROUTINE ADDRESS
0485  014A  028B                CI    R11,GO
      014C  019C'
0486  014E  1302                JEQ   PPFLD3
0487  0150  A2E0                A     @PGBIAS,R11
      0152  0050'
0488        0154' PPFLD3  EQU   $
0489                    *                       R3=CNAME(J+5)  SUBCOMMAND PARMS
0490  0154  C0F5                MOV   *R5+,R3
0491  0156  098C                SRL   R12,8             <R12>=NO. OF PARAMETERS
0492                    *                       R1=ADDR(R0)
0493  0158  02A1                STWP  R1
0494                    *                       R1=ADDR(R9)
0495  015A  0221                AI    R1,R9*2
      015C  0012
0496                    *                               PARAMETER VALUES FOR SUB-
0497                    *                               ROUTINES TO BE TRANSFERRED IN
0498                    *                               R9-R4
0499  015E  C009                MOV   R9,R0
0500                    *                       DO 219 L=1,NOPRM
0501        0160' GP000   EQU   $
0502                    *                               IF NO PARAMETERS TO PROCESS,
0503                    *                               SKIP AND BRANCH TO SUBROUTINE
0504  0160  060C                DEC   R12
0505  0162  1108                JLT   GP100
0506                    *                       IF (MASK.EQ.1) GO TO 229
0507                    *                               CHECK FOR REQUIRED PARAMETER
0508  0164  0A1E                SLA   R14,1
0509                    *                               IF REQUIRED JUMP TO SEE IF
0510                    *                               PRESENT
0511  0166  1804                JOC   GP010
0512                    *                               MOVE DEFAULT PARAMETER VALUE
0513                    *                               INTO TRANSFER REGISTER
```

```
0514  016A  C47F          MOV   *R15+,*R1
0515                *                       IF (PRESBT,EQ,0)GO TO 219
0516                *                                 CHECK IF PARAMETER PRESENT
0517  016A  0A10          SLA   R0,1
0518                *                                 IF PRESENT JUMP TO REPLACE
0519                *                                 DEFAULT VALUE WITH ACTUAL
0520                *                                 VALUE
0521  016C  1803          JOC   GP015
0522  016E  1003          JMP   GP020
0523                * 229                     IF(PRESBT,EQ,0)CALL ERR
0524        0170' GP010   EQU   $
0525                *                                 IF REQUIRED BUT NOT PRESENT,
0526                *                                 CALL ERROR
0527  0170  0A10          SLA   R0,1
0528  0172  1706          JNC   REQMIS
0529        0174' GP015   EQU   $
0530                *                       PARM(L)=CPL(L+3)
0531                *                                 MOVE PARAMETER VALUE INTO
0532                *                                 TRANSFER REGISTER
0533  0174  C47A          MOV   *R10+,*R1
0534                * 219                     CONTINUE
0535        0176' GP020   EQU   $
0536                *                                 GET NEXT TRANSFER REGISTER
0537  0176  0641          DECT  R1
0538  0178  10F3          JMP   GP000
0539        017A' GP100   EQU   $
0540                *                                 SET UP ERROR RETURN
0541  017A  0200          LI    R0,ERR
      017C  0196'
0542                *                       R12 IS -1; USE FOR COMPARISONS
0543                *                       IN COMMAND ROUTINES TO CHECK FOR
0544                *                       MISSING PARAMETERS
0545                *                       RETURN
0546                *                                 BRANCH TO SUBCOMMAND ROUTINE
0547  017E  0458          B     *R11
0548        0180' REQMIS  EQU   $
0549  0180  020A          LI    R10,PPRQ
      0182  3101
0550  0184  1008          JMP   ERR
0551        0186' NMATCH  EQU   $
0552  0186  020A          LI    R10,PPNM
      0188  3103
0553  018A  1005          JMP   ERR
0554        018C' BADADD  EQU   $
0555  018C  020A          LI    R10,PPBA
      018E  3104
0556  0190  1002          JMP   ERR
0557        0192' OUTBDS  EQU   $
0558  0192  020A          LI    R10,PPOB
      0194  3102
0559        0196' ERR     EQU   $
0560  0196  06A0          BL    @ERROR
      0198  0000
0561  019A  0452          B     *R2
```

```
0564                    * TITLE:     GO
0565                    *            GO SUBCOMMAND
0566                    * REVISION:  03/15/76
0567                    *            ORIGINAL
0568                    * COMPUTER:  990,ASSEMBLY
0569                    * ABSTRACT:
0570                    *            GENERATE BEGINNING BIT AND WORD ADDRESSES, TEST
0571                    *            TOGGLES AND CALL SUBROUTINES FOR TOGGLES SET.
0572                    * CALLING SEQUENCE:
0573                    *            ENTRY:
0574                    *            NONE
0575                    *            EXIT:
0576                    *            UPON COMPLETION RETURN CONTROL TO MONITOR
0577                    *            THROUGH R2
0578          019C'     GO      EQU $
0579                    *                        N=MMT(1)*MMT(2)*MMT(3)
0580  019C    0200              LI    R0,MMT
      019E    0018'
0581  01A0    C230              MOV   *R0+,R8             GET MMT(1)
0582  01A2    3A30              MPY   *R0+,R8             X=MMT(1)*MMT(2)
0583  01A4    3A50              MPY   *R0,R9              N(1)=MMT(3)*MOD(X)
0584  01A6    C1C8              MOV   R8,R7
0585  01A8    3900              MPY   *R0,R7              N(2)=MMT(3)*(X/256)
0586  01AA    A248              A     R8,R9               N(2)=N(2)+CARRY(N(1))
0587                    *                               R7,R8 CONCATENATED CONTAIN
0588                    *                               NUMBER OF LOOPS THROUGH
0589                    *                               MEMORY
0590  01AC    C1C9              MOV   R9,R7
0591  01AE    C20A              MOV   R10,R8
0592                    *                        NO=MRT(1)*MRT(2)*MRT(3)
0593  01B0    0200              LI    R0,MRT
      01B2    002C'
0594  01B4    C130              MOV   *R0+,R4             GET MRT(1)
0595  01B6    3930              MPY   *R0+,R4             X=MRT(1)*MRT(2)
0596  01B8    3950              MPY   *R0,R5              NO(1)=MRT(3)*MOD(X)
0597  01BA    C0C4              MOV   R4,R3
0598  01BC    38D0              MPY   *R0,R3              NO(2)=MRT(3)*(X/256)
0599  01BE    A144              A     R4,R5               NO(2)=NO(2)+CARRY(NO(1))
0600                    *                               R5,R6 CONCATENATED CONTAIN
0601                    *                               NUMBER OF LOOPS THROUGH ROM
0602                    *                        IF(N,NE,NO)CALL ERR
0603  01C0    8188              C     R8,R6
0604  01C2    16E1              JNE   NMATCH
0605  01C4    8147              C     R7,R5
0606  01C6    16DF              JNE   NMATCH
0607  01C8    0200              LI    R0,CMPR
      01CA    0046'
0608  01CC    C3B0              MOV   *R0+,R14            R14 = CMPR TOGGLE
0609  01CE    C270              MOV   *R0+,R9             R9 = DIRECTION TOGGLE
0610  01D0    C2B0              MOV   *R0+,R10            R10 = DISPLAY ROM
0611  01D2    C3D0              MOV   *R0,R15             R15 = DISPLAY MEMORY
0612                    *                               CHECK IF SAVE IMG IS ONLY TOG
0613  01D4    1608              JNE   GOFL0               IF MEM DISP = 0
0614  01D6    0289              CI    R9,SAVIMG           AND DIRECT = 3
```

```
                01D8  0003
0615            01DA  1605          JNE   GOFL0
0616            01DC  C00E          MOV   R14,R0              AND ROM AND MEM DISPLAY = 0
0617            01DE  A00A          A     R10,R0
0618            01E0  1602          JNE   GOFL0               THEN
0619            01E2  0460          B     @GOFL12               CALL SAVEIMG
                01E4  031A'
0620                  01E6' GOFL0   EQU   $
0621            01E6  0204          LI    R4,1                SET DISPLAY FLAG TO BE DECR
                01E8  0001
0622                        *                                 TO ZERO IN DISPLAY ROUTINE
0623                        *                                 TO INDICATE PRINT CARRIAGE
0624                        *                                 RETURN , LINE FEED BEFORE
0625                        *                                 DISPLAY
0626                        *                                 COMPUTE MEMORY AND ROM ADDRES
0627                        *                                 USING MAXIMUM COUNT
0628            01EA  C186          MOV   R6,R6
0629            01EC  1601          JNE   G01
0630            01EE  0605          DEC   R5
0631                  01F0' G01     EQU   $
0632            01F0  0606          DEC   R6
0633                        *                                 FLAG=INDICATE MAXIMUM COUNT
0634                        *                                 COMPUTATION LOOP
0635            01F2  070A          SETO  R10
0636            01F4  1003          JMP   GOBR1
0637                        *                       DO 160 KK=1,N
0638                        *                       K=KK-1
0639                  01F6' GOBK1   EQU   $
0640            01F6  04C5          CLR   R5                  BEGIN AT LOOP COUNT ZERO
0641            01F8  04C6          CLR   R6
0642                  01FA' GOFLD1  EQU   $
0643                        *                                 FLAG=MAXIMUM COUNT COMPUTA-
0644                        *                                 TION COMPLETE
0645            01FA  04CA          CLR   R10
0646                  01FC' GOBR1   EQU   $
0647                        *                                 SET UP PROGRAM BIAS TO LOCATE
0648                        *                                 SUBROUTINES IN EXTENSION
0649                        *                                 PROGRAM
0650            01FC  C320          MOV   @PGBIAS,R12
                01FE  0050'
0651                        *                                 COMPUTE CT VALUES TO BE USED
0652                        *                                 IN BBART
0653                        *                       CALL CTRT(MMT)
0654            0200  042C          BLWP  @CTR(R12)
                0202  0020
0655            0204  0018'         DATA  MMT
0656            0206  0100'         DATA  CT
0657                        *                                 COMPUTE MEMORY ADDRESS AND
0658                        *                                 STORE
0659                        *                       CALL BBART(IMT)
0660            0208  042C          BLWP  @BBAR(R12)
                020A  0024
0661            020C  000C'         DATA  IMT
0662            020E  0100'         DATA  CT
0663            0210  003E'         DATA  BMA
```

```
0664  0212  0084'        DATA EIGHT
0665  0214  0105'        DATA BMBYA
0666  0216  0108'        DATA BMBYBT
0667  0218  0040'        DATA EMA
0668  021A  0192'        DATA OUTRDS              ERROR EXIT
0669                *                            COMPUTE CT VALUES TO BE USED
0670                *                            IN BBART
0671                *                  CALL CTRT(MRT)
0672  021C  042C        BLWP @CTR(R12)
      021E  0020
0673  0220  002C'        DATA MRT
0674  0222  0100'        DATA CT
0675                *                            COMPUTE ROM ADDRESS AND STORE
0676                *                  CALL BBART(IRT)
0677  0224  042C        BLWP @BBAR(R12)
      0226  0024
0678  0228  0020'        DATA IRT
0679  022A  0100'        DATA CT
0680  022C  0042'        DATA BCA
0681  022E  0032'        DATA WWID
0682  0230  010A'        DATA BRWA
0683  0232  010C'        DATA BRWDBT
0684  0234  0044'        DATA ECA
0685  0236  0192'        DATA OUTRDS              ERROR EXIT
0686                *                            IF MAXIMUM COUNT COMPUTATION
0687                *                            LOOP, GO BACK AND BEGIN
0688                *                            LOOPING AT COUNT ZERO
0689  0238  C28A        MOV  R10,R10
0690  023A  16DD        JNE  GOBK1
0691                *                  ERWA=(BRBA+SWT-1)/WWID
0692  023C  C0E0        MOV  @SWT,R3
      023E  000A'
0693  0240  0603        DEC  R3
0694  0242  A0C3        A    R3,R1
0695  0244  1701        JNC  GBFLD1
0696  0246  0580        INC  R0
0697        0248' GBFLD1 EQU  $
0698  0248  3C20        DIV  @WWID,R0
      024A  0032'
0699  024C  C800        MOV  R0,@ERWA
      024E  010E'
0700                *                  IF(DIRECT.EQ.MEMROM)CALL MGETFD
0701                *                            R9 IS DIRECTION TOGGLE
0702  0250  0289        CI   R9,MEMROM
      0252  0001
0703  0254  1611        JNE  GOFLD2
0704                *                            GET MEMORY STRING IF TOGGLE
0705                *                            SET TO BURN
0706  0256  042C        BLWP @MGET(R12)
      0258  0028
0707  025A  0105'        DATA BMBYA
0708  025C  0108'        DATA BMBYBT
0709  025E  000A'        DATA SWT
0710  0260  0110'        DATA MSTRG
0711                *                  IF(DIRECT.EQ.MEMROM)CALL BROM
```

```
0712                    *                                   BURN STRING
0713  0262  042C               BLWP @BRM(R12)
      0264  002C
0714  0266  010C'              DATA BRWDBT
0715  0268  000A'              DATA SWT
0716  026A  0110'              DATA MSTRG
0717  026C  010A'              DATA BRWA
0718  026E  0032'              DATA WWID
0719  0270  0034'              DATA POOZ
0720  0272  0320'              DATA BWORD
0721  0274  0196'              DATA ERR                    ERROR EXIT
0722  0276  010E'              DATA ERWA
0723                    *                         IF(DISPR.EQ.YES.OR.CMPR.EQ.YES.OR.
0724                    *                            DIRECT.EQ.ROMMEM)CALL RGETFD
0725        0278' GOFLD2 EQU $
0726  0278  C2A0               MOV  @DISPR,R10
      027A  004A'
0727  027C  1605               JNE  GOFLD3
0728  027E  C38E               MOV  R14,R14                R14 CONTAINS COMPARE TOGGLE
0729  0280  1603               JNE  GOFLD3
0730                    *                                   R9 IS DIRECTION TOGGLE
0731  0282  0289               CI   R9,ROMMEM
      0284  0002
0732  0286  160B               JNE  GOFLD4
0733        0288' GOFLD3 EQU $
0734                    *                                   GET ROM STRING IF TOGGLES SET
0735                    *                                   FOR ROM-TO-MEMORY,DISPLAY ROM
0736                    *                                   OR COMPARE
0737  0288  042C               BLWP @RGET(R12)
      028A  0030
0738  028C  0032'              DATA WWID
0739  028E  010C'              DATA BRWDBT
0740  0290  004E'              DATA RCRUIA
0741  0292  000A'              DATA SWT
0742  0294  010A'              DATA BRWA
0743  0296  0196'              DATA ERR
0744  0298  0088'              DATA DOUT1
0745  029A  010E'              DATA ERWA
0746  029C  0112'              DATA RSTRG
0747                    *                         IF(DIRECT.EQ.ROMMEM)CALL RTOM
0748        029E' GOFLD4 EQU $
0749                    *                                   R9 IS DIRECTION TOGGLE
0750  029E  0289               CI   R9,ROMMEM
      02A0  0002
0751  02A2  1606               JNE  GOFLD5
0752                    *                                   ROM-TO-MEMORY
0753  02A4  042C               BLWP @RTM(R12)
      02A6  0034
0754  02A8  0106'              DATA BMBYA
0755  02AA  000A'              DATA SWT
0756  02AC  0108'              DATA BMBYBT
0757  02AE  0112'              DATA RSTRG
0758                    *                         IF(CMPR.EQ.YES)CALL COMPAR
0759                    *                         IF(DISPM.EQ.YES.OR.CMPR.EQ.YES)
0760                    *                            CALL MGETFD
```

```
0761          02B0' GOFLD5 EQU $
0762                *                                  R15 IS DISPLAY MEMORY TOGGLE
0763  02B0    C3CF        MOV  R15,R15
0764  02B2    1602        JNE  GOFLD6
0765  02B4    C38E        MOV  R14,R14                 R14 IS COMPARE TOGGLE
0766  02B6    1306        JEQ  GOFLD7
0767          02B8' GOFLD6 EQU $
0768                *                                  GET MEMORY STRING IF TOGGLES
0769                *                                  SET TO COMPARE OR DISPLAY
0770                *                                  MEMORY
0771  02B8    042C        BLWP @MGET(R12)
      02BA    0028
0772  02BC    0106'       DATA BMBYA
0773  02BE    0108'       DATA BMBYBT
0774  02C0    000A'       DATA SWT
0775  02C2    0110'       DATA MSTRG
0776          02C4' GOFLD7 EQU $
0777  02C4    04C1        CLR  R1                      CLEAR FLAG= NO COMPARE
0778  02C6    C38E        MOV  R14,R14                 R14 IS COMPARE TOGGLE
0779  02C8    1303        JEQ  GOFLD8
0780                *                        IF(MSTRG.EQ.RSTRG)CALL DISPLY(MSTRG)
0781                *                        IF(MSTRG.EQ.RSTRG)CALL DISPLY(RSTRG)
0782                *                                  IF TOGGLE SET, COMPARE
0783                *                                  STRINGS
0784  02CA    8820        C    @MSTRG,@RSTRG
      02CC    0110'
      02CE    0112'
0785  02D0    1304        JEQ  GOFLD8
0786  02D2    0701        SETO R1                      IF NOT SAME,SET FLAG SO TWO
0787                *                                  STRINGS WILL BE DISPLAYED
0788  02D4    0204        LI   R4,1                    SET DISPLAY FLAG TO BE DECR
      02D6    0001
0789                *                                  TO ZERO IN DISPLAY ROUTINE
0790                *                                  TO INDICATE PRINT CARRIAGE
0791                *                                  RETURN , LINE FEED BEFORE
0792                *                                  DISPLAY
0793  02D8    1002        JMP  GOFL8A                  JUMP TO DISPLAY
0794                *                        IF(DISPM.EQ.YES)CALL DISPLY
0795          02DA' GOFLD8 EQU $
0796  02DA    C3CF        MOV  R15,R15                 R15 IS DISPLAY MEMORY TOGGLE
0797  02DC    1307        JEQ  GOFLD9
0798                *                        CALL DISPLY(MPRT,BMWA,BMWDBT,MSTRG)
0799          02DE' GOFL8A EQU  $
0800                *                                  DISPLAY MEMORY STRING IF
0801                *                                  TOGGLE SET TO DISPLAY MEMORY
0802                *                                  OR IF COMPARED STRINGS DO NOT
0803                *                                  MATCH
0804  02DE    06A0        BL   @DISPLY
      02E0    03C6'
0805  02E2    014D        DATA MPRT
0806  02E4    0106'       DATA BMBYA
0807  02E6    0109'       DATA BMBT2
0808  02E8    0111'       DATA MSTRG2
0809  02EA    1019        JMP  GOFL10                  ERROR RETURN
0810                *                        IF(DISPR.EQ.YES)CALL DISPLY
```

```
0811          02EC' GOFLD9 EQU $
0812  02EC    C041        MOV  R1,R1              TEST FLAG- JUMP TO DISPLAY IF
0813                      *                       SET
0814  02EE    1602        JNE  GOFL9A
0815  02F0    C28A        MOV  R10,R10            R10 IS DISPLAY ROM TOGGLE
0816  02F2    1307        JEQ  GOFLD0
0817                      *                 CALL DISPLY(RPRT,BRWA,BRWDBT,RSTRG)
0818          02F4' GOFL9A EQU  $
0819                      *                       DISPLAY ROM STRING IF TOGGLE
0820                      *                       SET TO DISPLAY ROM OR IF
0821                      *                       COMPARED STRINGS DO NOT MATCH
0822  02F4    06A0        BL   @DISPLY
      02F6    03C8'
0823  02F8    0152        DATA RPRT
0824  02FA    010A'       DATA BRWA
0825  02FC    010D'       DATA BRBT2
0826  02FE    0113'       DATA RSTRG2
0827  0300    100E        JMP  GOFL10
0828                      *  160            CONTINUE
0829          0302' GOFLD0 EQU $
0830  0302    0586        INC  R6                 IF COUNT IN R6 INCREMENTED TO
0831  0304    1601        JNE  GOFL0C             VALUE GREATER THAN >FFFF ADD
0832  0306    0585        INC  R5                 CARRY TO R5
0833          0308' GOFL0C EQU $
0834  0308    8206        C    R6,R8              CHECK FOR LOOPING COMPLETED
0835  030A    1602        JNE  GOFL0D
0836  030C    8147        C    R7,R5
0837  030E    1302        JEQ  GOFL0A
0838          0310' GOFL0D EQU $
0839  0310    0460        B    @GOFLD1            CONTINUE LOOPING
      0312    01FA'
0840          0314' GOFL0A EQU $
0841                      *             IF (DIRECT.EQ.SAVIMG)CALL SVSTIM
0842                      *                       R9 IS DIRECTION TOGGLE
0843  0314    0289        CI   R9,SAVIMG
      0316    0003
0844  0318    1602        JNE  GOFL10
0845          031A' GOFL12 EQU $
0846                      *                       SAVE STANDARD IMAGE
0847                      *                       CONTROL RETURNED TO MONITOR
0848                      *                       FROM SVSTIM THROUGH R2
0849
0850  031A    0460        B    @SVSTIM
      031C    0442'
0851          031E' GOFL10 EQU $
0852                      *                 RETURN
0853  031E    0452        B    *R2
```

```
0856                    * TITLE:     BWORD
0857                    *            BURN WORD
0858                    * REVISION:  03/15/76
0859                    *            ORIGINAL
0860                    * COMPUTER:  990,ASSEMBLY
0861                    * ABSTRACT:
0862                    *            LOAD ADDRESS, PULSE WIDTH, AND SUBSTRING OF
0863                    *            PROGRAMMABLE STRING WIDTH IN CRU REGISTERS.
0864                    *            ON RETURN FROM PROGRAMMING CYCLE, LOOP FOR
0865                    *            RETRY IF NECESSARY AND REQUESTED
0866                    * CALLING SEQUENCE:
0867                    *            ENTRY:
0868                    *            R9 CONTAINS SUBSTRING OF ROM WORD WIDTH
0869                    *            R7 CONTAINS ROM WORD ADDRESS
0870                    *            R7-R10 , R13-R15 ARE NON-VOLATILE TO THIS RTNE
0871                    *            EXIT:
0872                    *            R3 CONTAINS LOOP COUNTER FOR BWORD ROUTINE AND
0873                    *            IS NON-VOLATILE TO ALL SUBROUTINES CALLED BY
0874                    *            BWORD
0875          0320'     BWORD  EQU   $
0876                    *                        MASK NUMBER OF BITS TO BE BURNED AT
0877                    *                        ONCE, THEN LOAD VALUES IN CRU REGISTERS
0878                    *                        CALL SUBROUTINE TO SET GO BIT AND TEST
0879                    *                        FOR ERRORS
0880                    *                        II=WWID/PSTW
0881  0320  C120               MOV   @WWID,R4
      0322  0032'
0882  0324  C020               MOV   @PSTW,R0
      0326  003C'
0883                    *                              COMPUTE HOW MANY TIMES
0884                    *                              THROUGH PROGRAMMING LOOP TO
0885                    *                              PROGRAM STRING OF ROM WORD
0886                    *                              WIDTH IF PROGRAMMING SUB-
0887                    *                              STRING OF PROGRAMMABLE STRING
0888                    *                              WIDTH
0889  0328  3CC0               DIV   R0,R3
0890                    *                        IF(II*PSTW.NE.WWID)II=II+1
0891  032A  C104               MOV   R4,R4
0892  032C  1301               JEQ   BWFLD1
0893  032E  0583               INC   R3
0894          0330'     BWFLD1 EQU   $
0895                    *                        MASK=-1
0896  0330  0705               SETO  R5
0897                    *                              MASK IS RIGHT JUSTIFIED
0898                    *                              STRING OF ZEROES OF PROGRAM-
0899                    *                              MABLE STRING WIDTH
0900  0332  0A05               SLA   R5,0
0901                    *                        DO 250 I=1,II
0902          0334'     BWFLD3 EQU   $
0903                    *                        IF(POOZ.EQ.1)STR=AND(STR,XOR(MASK,-1))
0904                    *                              GET NEW WORKSPACE
0905  0334  0420               BLWP  @ACL
      0336  0000
0906  0338  C0AD               MOV   @R2*2(R13),R2
```

```
                033A  0004
0907            033C  C120          MOV   @PGBIAS,R4
                033E  0050'
0908                        *                                   R9 FROM PREVIOUS WORKSPACE
0909                        *                                   CONTAINS STRING OF ROM WORD
0910                        *                                   WIDTH
0911                        *                                   CHECK IF PROGRAMMING ONES OR
0912                        *                                   ZEROES. MASK ALL BITS BUT
0913                        *                                   STRING OF PROGRAMMABLE STRING
0914                        *                                   WIDTH
0915            0340  C041          MOV   R1,R1
0916            0342  1302          JEQ   BWFLD5
0917            0344  4245          SZC   R5,R9
0918            0346  1001          JMP   BWFLD6
0919                        *                       IF(POOZ.EQ.0)STR=OR(STR,MASK)
0920                0348' BWFLD5 EQU   $
0921            0348  E245          SOC   R5,R9
0922                        *                       <CRURG0>=ADDR/256
0923                0348' BWFLD6 EQU   $
0924                        *                                   SET CRU BASE ADDRESS
0925            034A  C320          MOV   @RCRUIA,R12
                034C  004E'
0926            034E  C047          MOV   R7,R1
0927                        *                                   GET MSB OF ROM WORD ADDRESS
0928            0350  0981          SRL   R1,8
0929                        *                                   LOAD IN ROM REGISTER 0
0930            0352  06A4          BL    @LOADCR(R4)
                0354  0226
0931            0356  0000          DATA  ADDR0
0932                        *                       <CRURG1>=MOD(ADDR/256)
0933                        *                                   GET LSB OF ROM WORD ADDRESS
0934            0358  C047          MOV   R7,R1
0935                        *                                   LOAD IN ROM REGISTER 1
0936            035A  06A4          BL    @LOADCR(R4)
                035C  0226
0937            035E  0100          DATA  ADDR1
0938                        *                       <CRURG3>=MOD(STR,256)
0939                        *                                   GET MASKED DATA
0940            0360  C049          MOV   R9,R1
0941                        *                                   LOAD IN ROM REGISTER 3
0942            0362  06A4          BL    @LOADCR(R4)
                0364  0226
0943            0366  0300          DATA  DIN1
0944                        *                       <CRURG5>=PW1
0945                        *                                   GET PULSE WIDTH PARAMETER
0946            0368  C020          MOV   @PW1,R0
                036A  0036'
0947                        *                                   DETERMINE HARDWARE PULSE
0948                        *                                   WIDTH INDICATOR AS 2**(PW1-1)
0949            036C  0201          LI    R1,1
                036E  0001
0950            0370  0600          DEC   R0
0951            0372  1301          JEQ   BWFL6A
0952            0374  0A01          SLA   R1,R0
0953                0376' BWFL6A EQU   $
```

```
0954                    *                               LOAD PULSE WIDTH INDICATOR IN
0955                    *                               ROM REGISTER 5
0956  0376  06A4               BL    @LOADCR(R4)
      0378  0226
0957  037A  0500               DATA  PWD0
0958                    *                         DO 120 J=1,RETRY+1
0959                    *                               GET NUMBER OF RETRIES
0960  037C  C1A0               MOV   @NR,R6            <R6>=J
      037E  0038'
0961                    *                               GO THROUGH PROGRAMMING LOOP
0962                    *                               ONCE IF NO RETRIES
0963  0380  0586               INC   R6
0964                    *                         CALL GORT
0965        0382' BWFLD7  EQU   $
0966                    *                               CALL PROGRAMMING SUBROUTINE
0967  0382  06A4               BL    @BNCYL(R4)
      0384  0232
0968  0386  101A               JMP   ERRET             JMP TO PROCESS ERROR RETURN
0969  0388  003A'              DATA  DC
0970                    *                               LOAD CRU WITH ROM REGISTER 7
0971                    *                               AND READ BACK FROM ROM
0972  038A  32E0               LDCR  @DOUT1,11
      038C  0088'
0973  038E  3601               STCR  R1,8
0974  0390  0981               SRL   R1,8
0975                    *                               CLEAR ROM READBACK REGISTER
0976  0392  04C0               CLR   R0
0977  0394  32C0               LDCR  R0,11
0978                    *                         IF(AND(RSTRG,XOR(MASK,-1)),EQ,AND(STR,
0979                    *                            XOR(MASK,-1)))GO TO 250
0980                    *                               GET DATA WHICH WAS TO BE
0981                    *                               PROGRAMMED,MASK ALL BUT
0982                    *                               PROGRAMMABLE STRING WIDTH OF
0983                    *                               TWO WORDS AND COMPARE
0984  0396  C02D               MOV   @R9*2(R13),R0
      0398  0012
0985  039A  4045               SZC   R5,R1
0986  039C  4005               SZC   R5,R0
0987  039E  8001               C     R1,R0
0988                    *                               IF PROGRAMMING SUCCESSFUL,
0989                    *                               CONTINUE, OTHERWISE CHECK FOR
0990                    *                               RETRIES
0991  03A0  1302               JEQ   BWFLD8
0992                    *  120                    CONTINUE
0993  03A2  0606               DEC   R6
0994  03A4  15EE               JGT   BWFLD7
0995                    *  250                    CONTINUE
0996        03A6' BWFLD8  EQU   $
0997                    *                               RETURN WORKSPACE
0998  03A6  0420               BLWP  @RR
      03A8  0000
0999                    *                         SHIFT R5 TO MASK NEXT PSTW
1000  03AA  0200               LI    R0,16
      03AC  0010
1001  03AE  6020               S     @PSTW,R0
```

```
                03B0  003C'
1002  03B2  0B05         SRC   R5,0
1003                *
1004                *                                  WIDTH HAS BEEN PROGRAMMED.
1005  03B4  0603         DEC   R3
1006  03B6  16BE         JNE   BWFLD3
1007  03B8  05C8         INCT  R11                      SKIP ERROR RETURN
1008                *                          RETURN
1009        03BA' BWFLD9   EQU   $
1010  03BA  045B         B     *R11
1011                *                                  ERROR RETURN
1012        03BC' ERRET    EQU   $
1013  03BC  0420         BLWP  @RR                      RETURN WKSP
      03BE  03A8'
1014  03C0  CB4A         MOV   R10,@R10*2(R13)          SAVE ERROR MESSAGE
      03C2  0014
1015  03C4  10FA         JMP   BWFLD9                   TAKE ERROR EXIT
```

```
1018                    * TITLE:     DISPLY
1019                    *            DISPLAY STRING
1020                    * REVISION: 03/15/76
1021                    *            ORIGINAL
1022                    * COMPUTER: 990,ASSEMBLY
1023                    * ABSTRACT:
1024                    *            DISPLAY STRING AS BYTE ADDR,BIT ADDR= STRING
1025                    * CALLING SEQUENCE:
1026                    *            ENTRY:
1027                    *            R11 CONTAINS ADDRESS OF DATA PASSED BY CALLING
1028                    *            ROUTINE
1029                    *            EXIT:
1030                    *            NONE
1031          03C6'     DISPLY EQU $
1032                    *                        SAVE RETURN ADDRESS
1033    03C6  C0CB              MOV  R11,R3
1034    03C8  0228              AI   R11,8              SET RETURN ADDRESS
      03CA  0008
1035    03CC  0420              BLWP @ACL               GET NEW WORKSPACE
      03CE  0336'
1036                    *                        IF(COUNT.EQ.0)CALL PRCRLF
1037                    *                                CHECK FOR NEW LINE
1038    03D0  0604              DEC  R4
1039    03D2  160D              JNE  DISP1
1040                    *                        CARRIAGE RETURN, LINE FEED
1041    03D4  06A0              BL   @PRCRLF
      03D6  0000
1042    03D8  1031              JMP  DSEXIT
1043    03DA  0204              LI   R4,4               SET FLAG TO DISPLAY REST
      03DC  0004
1044                    *                                OF LINE WITHOUT CR LF
1045    03DE  C041              MOV  R1,R1              CHECK FOR COMPARE ERROR
1046    03E0  1306              JEQ  DISP1              IF NOT, SKIP ERROR PRINT
1047    03E2  020A              LI   R10,DMPSPR         GET ERROR INDICATOR
      03E4  0000
1048    03E6  06A0              BL   @PRNTC             AND DISPLAY
      03E8  0000
1049    03EA  1028              JMP  DSEXIT             ABORT IF ERROR IN PRINT CALL
1050    03EC  0644              DECT R4                 SET FLAG TO DISPLAY SECOND
1051                    *                                DISPLAY STRING AND THEN START
1052                    *                                A NEW LINE
1053          03EE'     DISP1  EQU  $
1054                    *                        COUNT=COUNT+1
1055                    *                                SAVE VALUE IN R4 OF PREVIOUS
1056                    *                                WORKSPACE
1057    03EE  C844              MOV  R4,@R4*2(R13)
      03F0  0008
1058    03F2  C283              MOV  R3,R10
1059    03F4  05C3              INCT R3
1060                    *                        CALL PRINT(TYPE)
1061                    *                                R10 CONTAINS ADDRESS OF 'M'
1062                    *                                OR 'R' TO BE PRINTED
1063    03F6  06A0              BL   @PRNTC
      03F8  03E8'
```

```
1064  03FA  1020          JMP   DSEXIT
1065  03FC  C073          MOV   *R3+,R1
1066  03FE  C2B1          MOV   *R1+,R10
1067                *                       CALL PRINT(BBA)
1068                *                            R10 CONTAINS STRING ADDRESS
1069  0400  06A0          BL    @PRNTHN
      0402  0000
1070  0404  101B          JMP   DSEXIT
1071                *                       CALL PRINT(.)
1072  0406  020A          LI    R10,PRTPRD
      0408  0000
1073                *                            R10 CONTAINS ADDRESS OF
1074                *                            PERIOD
1075  040A  06A0          BL    @PRNTC
      040C  03F8'
1076  040E  1016          JMP   DSEXIT
1077  0410  C073          MOV   *R3+,R1
1078  0412  D291          MOVB  *R1,R10
1079                *                       CALL PRINT(BWDBIT)
1080                *                            R10 CONTAINS BIT WITHIN BYTE
1081                *                            OR WORD
1082  0414  06A0          BL    @PRNTHB
      0416  0000
1083  0418  1011          JMP   DSEXIT
1084                *                       CALL PRINT(=)
1085  041A  020A          LI    R10,EQSIGN
      041C  0000
1086                *                            R10 CONTAINS EQUAL SIGN
1087  041E  06A0          BL    @PRNTC
      0420  040C'
1088  0422  100C          JMP   DSEXIT
1089  0424  C073          MOV   *R3+,R1
1090  0426  D291          MOVB  *R1,R10
1091                *                       CALL PRINT(STRING)
1092                *                            R10 CONTAINS STRING
1093  0428  06A0          BL    @PRNTHB
      042A  0416'
1094  042C  1007          JMP   DSEXIT
1095                *                       CALL PRINT(  )
1096  042E  020A          LI    R10,BLANKS
      0430  0060'
1097  0432  06A0          BL    @PRNTC
      0434  0420'
1098  0436  1002          JMP   DSEXIT                ERROR RETURN
1099  0438  05ED          INCT  @R11*2(R13)
      043A  0016
1100        043C' DSEXIT  EQU   $
1101  043C  0420          BLWP  @RR                   RETURN WORKSPACE
      043E  038E'
1102  0440  045B          B     *R11
```

```
1105                    * TITLE:     SVSTIM
1106                    *            SAVE STANDARD IMAGE
1107                    * REVISION: 03/15/76
1108                    *            ORIGINAL
1109                    * COMPUTER: 990,ASSEMBLY
1110                    * ABSTRACT:
1111                    *            TAKE IMAGES IN WORKING TABLE AND STORE THEM
1112                    *            WITH A USER-PROVIDED ID CHARACTER STRING IN
1113                    *            HEX ASCII FORMAT ON CASSETTE OR REPLACE RECORDS
1114                    *            WHICH HAVE ID CHARACTER STRING MATCHING USER
1115                    *            INPUTTED STRING
1116                    * CALLING SEQUENCE:
1117                    *            ENTRY:
1118                    *            NONE
1119                    *            EXIT:
1120                    *            UPON COMPLETION RETURN CONTROL TO MONITOR
1121                    *            THROUGH R2
1122          0442'     SVSTIM EQU   $
1123                    *                                  SET UP PROGRAM BIAS TO LOCATE
1124                    *                                  SUBROUTINES IN EXTENSION
1125                    *                                  PROGRAM
1126    0442  C320              MOV   @PGBIAS,R12
      0444  0050'
1127                    *                                  PRINT QUESTION AND GET
1128                    *                                  RESPONSE; SAVE BUFFER RETURN,
1129                    *                                  NUMBER OF CHARS, IN MEMORY ID
1130                    *                                  STRING, ADDRESS OF MEMORY ID
1131                    *                                  STRING
1132                    *100                    CALL PFBPCG
1133    0446  06A0              BL    @PFBPCG
      0448  0608'
1134    044A  0070'             DATA  MEMIMG
1135    044C  0452              B     *R2                  ESCAPE
1136                    *                       SAVE R10
1137    044E  C80A              MOV   R10,@BUFRT1
      0450  008A'
1138                    *                       SAVE ADDRESS AND # OF CHARS
1139    0452  C13A              MOV   *R10+,R4
1140    0454  0984              SRL   R4,8
1141    0456  C204              MOV   R4,R8                NUMBER OF CHARS, IN MEMORY ID
1142                    *                                  STRING
1143    0458  C1CA              MOV   R10,R7               ADDR OF MEMORY ID STRING
1144                    *                                  PRINT QUESTION AND GET
1145                    *                                  RESPONSE; SAVE BUFFER RETURN,
1146                    *                                  NUMBER OF CHARS, IN ROM ID
1147                    *                                  STRING, ADDR OF ROM ID STRING
1148                    *                       CALL PFBPCG
1149    045A  06A0              BL    @PFBPCG
      045C  0608'
1150    045E  0078'             DATA  ROMIMG
1151    0460  0452              B     *R2                  ESCAPE
1152                    *                       SAVE R10
1153    0462  C80A              MOV   R10,@BUFRT2
      0464  008C'
```

```
1154                    *                       SAVE ADDRESS AND # OF CHARS
1155  0466  C13A         MOV   *R10+,R4
1156  0468  0904         SRL   R4,8
1157                    *                               R4 CONTAINS NUMBER OF CHARS
1158                    *                               IN ROM ID STRING
1159  046A  C14A         MOV   R10,R5                    ADDR OF ROM ID STRING
1160                    *                               OPEN LUNOS 7 AND 8 FOR
1161                    *                               READING AND WRITING
1162                    *                       CALL OPEN
1163  046C  0201         LI    R1,OPEN*256
      046E  0000
1164  0470  06A0         BL    @WRTRD
      0472  0664'
1165  0474  161E         JNE   SV6
1166  0476  06A0         BL    @WRT1
      0478  0672'
1167  047A  1618         JNE   SV6
1168        047C' SV1     EQU   $
1169                    *1                      CONTINUE
1170                    *                               READ A RECORD
1171  047C  0201         LI    R1,READA*256
      047E  0900
1172  0480  06A0         BL    @WRT1
      0482  0672'
1173                    *                               ERROR RETURN
1174                    *                       IF(PRBSF.NE.0) GO TO 40
1175  0484  162A         JNE   SV13
1176                    *                       IF(COUNT(1).EQ.0) GO TO 10
1177                    *                               CHECK IF USER MEMORY ID
1178                    *                               STRING NOT ENTERED OR ALREADY
1179                    *                               MATCHED TO ID STRING OF A
1180                    *                               PREVIOUS RECORD
1181  0486  C008         MOV   R8,R0
1182  0488  1307         JEQ   SV3
1183                    *                               IF NOT, GET ADDR OF USER
1184                    *                               MEMORY ID STRING
1185  048A  C047         MOV   R7,R1
1186                    *                               CHECK IF STRING MATCHES ID
1187                    *                               STRING OF PRESENT RECORD
1188                    *                       CALL CHECK
1189  048C  06AC         BL    @CHECK(R12)
      048E  030A
1190  0490  0064'        DATA LDBUF
1191                    *                       CHECK NOT FOUND
1192                    *                       GO TO 10
1193  0492  1002         JMP   SV3
1194                    *                       CHECK FOUND
1195                    *                       COUNT(1)=0
1196  0494  04C8         CLR   R8                        INDICATE MATCH FOUND BY
1197                    *                               CLEARING CHAR COUNT
1198                    *                       GO TO 20
1199  0496  100F         JMP   SV7
1200        0498' SV3     EQU   $
1201                    *10                     IF(COUNT(2).EQ.0) GO TO 15
1202                    *                               CHECK IF USER ROM ID STRING
```

```
1203                    *                                   NOT ENTERED OR ALREADY
1204                    *                                   MATCHED TO ID STRING OF A
1205                    *                                   PREVIOUS RECORD
1206  049F  C004               MOV   R4,R0
1207                    *                                   IF NOT, GET ADDR OF USER ROM
1208                    *                                   ID STRING
1209  04A0  C045               MOV   R5,R1
1210  04A2  1306               JEQ   SV5
1211                    *                                   CHECK IF STRING MATCHES ID
1212                    *                                   STRING OF PRESENT RECORD
1213                    *                         CALL CHECK
1214  04A4  06AC               BL    @CHECK(R12)
      04A6  030A
1215  04A8  0490'              DATA  LOBUF
1216                    *                         CHECK NOT FOUND
1217                    *                         GO TO 15
1218  04AA  1002               JMP   SV5
1219                    *                         CHECK FOUND
1220                    *                         COUNT(2)=0
1221  04AC  04C4               CLR   R4                    INDICATE MATCH FOUND BY
1222                    *                                   CLEARING CHAR COUNT
1223                    *                         GO TO 20
1224  04AE  1006               JMP   SV7
1225        04B0' SV5     EQU   $
1226                    *                                   COPY RECORD IF NOT IDENTIFIED
1227                    *                                   BY ROM OR MEMORY ID STRINGS
1228  04B0  0201               LI    R1,WRITA*256
      04B2  0B00
1229                    *                         CALL WRITE (BFR)
1230  04B4  06A0               BL    @WRTRD
      04B6  0664'
1231        04B8' SV6     EQU   $
1232  04B8  1653               JNE   ERRER
1233                    *                                   RETURN TO PROCESS NEXT RECORD
1234                    *                         GO TO 1
1235  04BA  10E3               JMP   SV1
1236                    *                                   IF ID STRING MATCHES RECORD
1237                    *                                   ID, CHECK TAG ON RECORD ID TO
1238                    *                                   DETERMINE IF RECORD CONTAINS
1239                    *                                   MEMORY OR ROM CONTROL INFO
1240                    *                                   REPLACE RECORD WITH CONTROL
1241                    *                                   INFO FROM WORKING TABLE
1242                    *20                       IF(TYPE.EQ.'M')GO TO 30
1243        04BC' SV7     EQU   $
1244  04BC  028F               CI    R15,'M'
      04BE  004D
1245  04C0  1305               JEQ   SV9
1246                    *               WKTBL=RTEMP
1247  04C2  020E               LI    R14,RTEMP
      04C4  0A1E'
1248                    *               COUNT=16
1249  04C6  0209               LI    R9,RSZ
      04C8  0010
1250                    *                         GO TO 35
1251  04CA  1004               JMP   SV11
```

```
1252          04C6' SV9     EQU  $
1253                *30                    WKTBL=MTEMP
1254  04C6    020E          LI   R14,MTEMP
      04C8    00A'
1255                *                       COUNT=10
1256  04CA    0209          LI   R9,MS7
      04CC    000A
1257                *                             CONVERT CONTROL INFORMATION
1258                *                             FROM BINARY TO HEX ASCII AND
1259                *                             WRITE TO TAPE REPLACING
1260                *                             PRESENT RECORD
1261                *35                    CALL CVTWRT
1262          04CE' SV11    EQU  $
1263  04CE    06AC          BL   @CVTW10(R12)
      04D0    0326
1264  04D2    063E'         DATA CVTBHX
1265  04D4    0664'         DATA WRTRD
1266  04D6    1041          JMP ERRER
1267                *                       GO TO 1
1268  04D8    10D1          JMP  SV1
1269                *                       CONTINUE
1270          04DA' SV13    EQU  $
1271                *                             IF ERROR OTHER THAN END-OF-
1272                *                             FILE, ABORT
1273                *                       R0=ANDI(R0,EOFMSK)
1274  04DA    0240          ANDI R0,EOFMSK
      04DC    2000
1275                *                       IF(R0,EQ,0)CALL ERRER
1276  04DE    133D          JEQ  ERRER
1277  04E0    0206          LI   R6,LDBUF
      04E2    04A2'
1278  04E4    C046          MOV  R6,R1
1279  04E6    0221          AI   R1,80
      04E8    0050
1280                *                       IF(COUNT(1),EQ,0) GO TO 50
1281                *                             IF USER MEMORY ID STRING NOT
1282                *                             MATCHED BY END OF FILE, ADD
1283                *                             ID STRING AND PRESENT MEMORY
1284                *                             CONTROL INFORMATION TO FILE
1285  04EA    C208          MOV  R8,R8            CHECK CHAR COUNT FOR ZERO
1286                *                             INDICATING MATCHED ID STRING
1287  04EC    130E          JEQ  SV19
1288  04EE    06AC          BL   @BLNK(R12)
      04F0    035A
1289                *                       (BUFFER)=TYPE
1290                *                       BUFFER=BUFFER+1
1291  04E2    0200          LI   R0,M
      04F4    0040
1292  04F6    CD80          MOV  R0,*R6+
1293                *                       WKTBL=MTEMP
1294  04F8    020E          LI   R14,MTEMP
      04FA    00A'
1295                *                       COUNT=10
1296  04FC    0209          LI   R9,MS7
      04FE    000A
```

```
1297                    *                       CALL CVTWRT
1298  0500  06AC               BL    @CVTWRT(R12)
      0502  031E
1299  0504  063E'              DATA CVTBHX
1300  0506  0664'              DATA WRTRD
1301  0508  1028               JMP ERRER
1302                    *50                     IF(COUNT(2),EQ,0) GO TO 60
1303        050A' SV19  EQU   $
1304                    *                               IF USER ROM ID STRING NOT
1305                    *                               MATCHED BY END OF FILE, ADD
1306                    *                               ID STRING AND PRESENT ROM
1307                    *                               CONTROL INFORMATION TO FILE
1308  050A  C204               MOV   R4,R8             CHECK CHAR COUNT FOR ZERO
1309                    *                               INDICATING MATCHED ID STRING
1310  050C  1314               JEQ   SV25
1311  050E  0206               LI    R6,LDBUF
      0510  04E2'
1312  0512  C046               MOV   R6,R1
1313  0514  0221               AI    R1,80
      0516  0050
1314                    *                       CALL BLNK
1315  0518  06AC               BL    @BLNK(R12)
      051A  035A
1316  051C  C1C5               MOV   R5,R7             GET ADDR OF ROM ID STRING
1317                    *                       (BUFFER)=TYPE
1318                    *                       BUFFER=BUFFER+1
1319  051E  0200               LI    R0,R
      0520  0052
1320  0522  CD80               MOV   R0,*R6+
1321                    *                       WDTBL=RTEMP
1322  0524  020E               LI    R14,RTEMP
      0526  001E'
1323                    *                       COUNT=16
1324  0528  0209               LI    R9,RSZ
      052A  0010
1325                    *                       CALL CVTWRT
1326  052C  06AC               BL    @CVTWRT(R12)
      052E  031E
1327  0530  063E'              DATA CVTBHX
1328  0532  0664'              DATA WRTRD
1329  0534  1012               JMP ERRER
1330        0536' SV25  EQU   $
1331                    *60                     CALL WRITE(WEOF)
1332  0536  0201               LI    R1,WEOF*256
      0538  0D00
1333  053A  06A0               BL    @WRTRD
      053C  0664'
1334  053E  160D               JNE   ERRER
1335                    *                       RETURN
1336  0540  C2A0               MOV   @BUFRT1,R10
      0542  008A'
1337  0544  0420               BLWP @RETBUF
      0546  0000
1338  0548  C2A0               MOV   @BUFRT2,R10
      054A  008C'
```

```
1339  054C  0420          BLWP  @RETBUF
      054E  0546'
1340  0550  0452          B     *R2
1341                 *                      R10=EOFMSG
1342        0552' ERTYP1 EQU   $
1343                 *                      CALL ERRER
1344  0552  0460          B     @NMATCH
      0554  0186'
1345        0556' ERTYP2 EQU   $
1346  0556  0460          B     @BADADD
      0558  018C'
1347        055A' ERRER  EQU   $
1348  055A  0460          B     @ERR
      055C  0196'
```

```
1351                    * TITLE:     PSCSI
1352                    *            PROM STANDARD COMMAND STRING PROCESSOR
1353                    * REVISION: 03/15/76
1354                    *            ORIGINAL
1355                    * COMPUTER: 990,ASSEMBLY
1356                    * ABSTRACT:
1357                    *            FIND IMAGES ON CASSETTE WITH TEXT CHARACTER
1358                    *            STRINGS THAT MATCH COMMAND PARAMETERS AND
1359                    *            TRANSFER IMAGES TO WORKING TABLE
1360                    * CALLING SEQUENCE:
1361                    *            ENTRY:
1362                    *            R10 CONTAINS ADDRESS OF COMMAND PROCESSOR LIST
1363                    *            EXIT:
1364                    *            NONE
1365          055E'     PSCSI  EQU  $
1366                    *                        SAVE RETURN ADDRESS
1367    055E  C08B              MOV  R11,R2
1368                    *                        PRESBT=LSHFT(CPL(1),9)
1369    0560  C27A              MOV  *R10+,R9
1370    0562  0A99              SLA  R9,9
1371                    *                        IF(PARM(1).EQ.' ')CALL ERR
1372    0564  1802              JOC  PSCSI1
1373    0566  0460              B    @REQMIS
      0568  0180'
1374          056A'     PSCSI1 EQU  $
1375                    *                        COUNT(1)=CPL(2)/256
1376    056A  D23A              MOVB *R10+,R8
1377    056C  0988              SRL  R8,8
1378                    *                        SAVE PARM(1) AND COUNT (1)
1379                    *                             R8 CONTAINS NUMBER OF CHARS
1380                    *                             IN FIRST CHAR STRING
1381    056E  C1CA              MOV  R10,R7           ADDR OF FIRST CHAR STRING
1382    0570  C148              MOV  R8,R5            SAVE CHAR COUNT
1383                    *                             GET ADDR OF SECOND COMMAND
1384                    *                             PARAMETER
1385    0572  A14A              A    R10,R5
1386    0574  0585              INC  R5
1387    0576  0915              SRL  R5,1
1388    0578  0A15              SLA  R5,1
1389                    *                        PRESBT=LSHFT(PRESBT,1)
1390                    *                        IF(PARM(2).EQ.' ')GO TO 10
1391    057A  0A19              SLA  R9,1
1392    057C  1703              JNC  PS1
1393                    *                        COUNT(2)=CPL/256
1394    057E  D135              MOVB *R5+,R4
1395    0580  0984              SRL  R4,8
1396                    *                        GO TO 15
1397    0582  1001              JMP  PS2
1398                    *10                      COUNT(2)=0
1399          0584'     PS1    EQU  $
1400                    *                             IF SECOND PARAMETER MISSING,
1401                    *                             CLEAR CHAR COUNT
1402    0584  04C4              CLR  R4
1403                    *15                      SAVE PARM(2) AND COUNT(2)
```

```
1404          0586' PS2     EQU  $
1405                *                              R4 CONTAINS CHAR COUNT OF
1406                *                              SECOND CHARACTER STRING
1407                *                              R5 CONTAINS ADDR OF SECOND
1408                *                              CHARACTER STRING
1409                *                              OPEN LUNO 7 FOR READING
1410                *                    CALL OPEN
1411  0586  0201          LI   R1,OPEN*256
      0588  0000
1412  058A  06A0          BL   @WRT1
      058C  0672'
1413  058E  16E5          JNE  ERRER
1414                *                              SET UP PROGRAM BIAS TO LOCATE
1415                *                              SUBROUTINES IN EXTENSION
1416                *                              PROGRAM
1417  0590  C320          MOV  @PGBIAS,R12
      0592  0050'
1418                *20                  COUNINUE
1419          0594' PS3     EQU  $
1420                *                              READ A RECORD
1421  0594  0201          LI   R1,READA*256
      0596  0900
1422  0598  06A0          BL   @WRT1
      059A  0672'
1423  059C  1620          JNE  PS8
1424                *                    IF(COUNT(1),EQ,0)GO TO 30
1425  059E  C008          MOV  R8,R0               GET CHAR COUNT
1426  05A0  1307          JEQ  PS4
1427  05A2  C047          MOV  R7,R1               GET ADDR OF FIRST CHAR STRING
1428                *                              COMPARE FIRST CHAR STRING TO
1429                *                              RECORD ID STRING
1430                *                    CALL CHECK
1431  05A4  06AC          BL   @CHECK(R12)
      05A6  030A
1432  05A8  0510'         DATA LDBUF
1433                *                    CHECK NOT FOUND
1434                *                    GO TO 30
1435  05AA  1002          JMP  PS4
1436                *                    CHECK FOUND
1437                *                    COUNT(1)=0
1438  05AC  04C8          CLR  R8                  INDICATE MATCH FOUND BY
1439                *                              CLEARING CHAR COUNT
1440                *                    GO TO 40
1441  05AE  1005          JMP  PS5
1442          05B0' PS4     EQU  $
1443                *30                  IF(COUNT(2),EQ,0)GO TO 20
1444  05B0  C004          MOV  R4,R0               GET CHAR COUNT
1445  05B2  1603          JNE  PS4A
1446  05B4  C008          MOV  R8,R0               IF BOTH ID STRINGS MATCHED
1447  05B6  1323          JEQ  PS9                 JUMP TO TERMINATE
1448  05B8  10E0          JMP  PS3                 RETURN TO READ OTHERWISE
1449          05BA' PS4A    EQU  $
1450  05BA  C045          MOV  R5,R1               GET ADDR OF SECOND CHAR STR
1451                *                              IF FIRST CHAR STRING DOES NOT
1452                *                              MATCH RECORD ID, COMPARE
```

```
1453                    *                               SECOND CHAR STRING TO RECORD
1454                    *                               ID STRING
1455                    *                       CALL CHECK
1456  05BC  06AC        BL    @CHECK(R12)
      05BE  03AA
1457  05C0  05A8'       DATA LDBUF
1458                    *                       CHECK NOT FOUND
1459                    *                       GO TO 20
1460  05C2  10E8        JMP   PS3
1461                    *                       CHECK FOUND
1462                    *                       COUNT(2)=0
1463  05C4  04C4        CLR   R4                        INDICATE MATCH FOUND BY
1464                    *                               CLEARING CHAR COUNT
1465        05C6' PS5     EQU   $
1466                    *40                     IF(TYPE.EQ.'M')GO TO 50
1467                    *                               CHECK TAG ON RECORD ID TO
1468                    *                               DETERMINE IF PRESENT RECORD
1469                    *                               CONTAINS MEMORY OR ROM
1470                    *                               CONTROL INFORMATION AND MOVE
1471                    *                               INFORMATION TO WORKING TABLE
1472                    *                               AFTER CONVERTING TO BINARY
1473  05C6  028F        CI    R15,'M'
      05C8  004D
1474  05CA  1305        JEQ   PS6
1475                    *                       WKTBL=RTEMP
1476  05CC  020E        LI    R14,RTEMP
      05CE  041E'
1477                    *                       COUNT=16
1478  05D0  0209        LI    R9,RSZ
      05D2  0010
1479                    *                       GO TO 55
1480  05D4  1004        JMP   PS7
1481                    *50                     WKTBL=MTEMP
1482        05D6' PS6     EQU   $
1483  05D6  020E        LI    R14,MTEMP
      05D8  040A'
1484                    *                       COUNT=10
1485  05DA  0209        LI    R9,MSZ
      05DC  000A
1486        05DE' PS7     EQU   $
1487                    *55                     DO 60 I=1,COUNT
1488                    *                       CALL CVTHXB
1489  05DE  06AC        BL    @CVTHXB(R12)
      05E0  0342
1490  05E2  0054'       DATA CONPRB
1491  05E4  005A'       DATA HXB
1492  05E6  0000        DATA SVCALT
1493                    *                       (WKTBL)=R0
1494  05E8  CF80        MOV   R0,*R14+
1495                    *60                     CONTINUE
1496  05EA  0609        DEC   R9
1497  05EC  16F8        JNE   PS7
1498                    *                               RETURN TO PROCESS NEXT RECORD
1499                    *                       GO TO 20
1500  05EE  10D2        JMP   PS3
```

```
1501           05F0' PS8    EQU  $
1502                 *                              IF ERROR OTHER THAN END-OF-
1503                 *                              FILE, ABORT
1504                 *                        IF(AND(R0,EOFMSK).EQ.0)CALL ERTYPE
1505   05F0    0240          ANDI R0,EOFMSK
       05F2    2000
1506   05F4    1382          JEQ  ERREN
1507                 *                              IF FIRST CHAR STRING NOT
1508                 *                              MATCHED TO ID STRING OF ANY
1509                 *                              RECORD, ABORT
1510                 *                        IF(COUNT(1).NE.0)CALL ERTYP2
1511   05F6    C208          MOV  R8,R8             CHECK CHAR COUNT=0
1512   05F8    16AE          JNE  ERTYP2
1513                 *                              IF SECOND CHAR STRING NOT
1514                 *                              MATCHED TO ID STRING OF ANY
1515                 *                              RECORD, ABORT
1516                 *                        IF(COUNT(2).NE.0)CALL ERTYP2
1517   05FA    C104          MOV  R4,R4             CHECK CHAR COUNT=0
1518   05FC    16AC          JNE  ERTYP2
1519                 *                              IF MEMORY STRING WIDTH IN
1520                 *                              WORKING TABLE DOES NOT MATCH
1521                 *                              ROM STRING WIDTH IN WORKING
1522                 *                              TABLE, ABORT
1523                 *                        IF(SWT.NE.SWTR)CALL ERR
1524           05FE' PS9    EQU  $
1525   05FE    8820          C    @SWT,@SWTR
       0600    000A'
       0602    001E'
1526   0604    16A5          JNE  ERTYP1
1527                 *                              RETURN
1528   0606    0452          B    *R2
```

```
1531                    * TITLE:     PFBPCG
1532                    *            PRCRLF,BLNK,PRNTC,GETBUF
1533                    * REVISION: 03/15/76
1534                    *            ORIGINAL
1535                    * COMPUTER: 990,ASSEMBLY
1536                    * ABSTRACT:
1537                    *            PRINT QUESTION AND RECEIVE ANSWER FROM
1538                    *            KEYBOARD INPUT
1539                    * CALLING SEQUENCE:
1540                    *            ENTRY:
1541                    *            R10 CONTAINS QUESTION TO BE PRINTED
1542                    *            EXIT:
1543                    *            R10 CONTAINS ADDRESS OF RETURNED BLOCK FROM
1544                    *            GETFLD
1545                    *            *R10 CONTAIN NUMBER OF CHARACTERS
1546                    *            *R10(1) CONTAIN TERMINATING CHARACTER
1547                    *            *R10(2)-*R10(N) CONTAIN BYTES OF CHARACTERS
1548          0608' PFBPCG EQU   $
1549                    *                        SAVE RETURN ADDRESS
1550  0608  C10E        MOV   R11,R4
1551                    *                                 SAVE ADDR OF QUESTION
1552  060A  C174        MOV   *R4+,R5
1553  060C  06A0        BL    @PRCRLF
      060E  03D6'
1554  0610  100F        JMP   PFEXIT                      ERROR RETURN
1555                    *                                 GET BUFFER
1556  0612  0205        LI    R6,LDBUF
      0614  05C0'
1557  0616  C046        MOV   R6,R1
1558                    *                                 GET END OF BUFFER
1559  0618  0221        AI    R1,80
      061A  0050
1560                    *                                 BLANK FILL BUFFER
1561                    *                        CALL BLNK
1562  061C  05AC        BL    @BLNK(R12)
      061E  035A
1563                    *                                 GET ADDRESS OF QUESTION
1564  0620  C285        MOV   R5,R10
1565                    *                                 PRINT QUESTION
1566                    *                        CALL PRNTC
1567  0622  06A0        BL    @PRNTC
      0624  0434'
1568  0626  1004        JMP   PFEXIT                      ERROR RETURN
1569                    *                                 GET RESPONSE FROM KEYBOARD
1570                    *                        CALL GETFLD
1571  0628  06A0        BL    @GETFLD
      062A  0000
1572  062C  1002        JMP   PFERR
1573          062E' PFBEX  EQU   $
1574  062E  05C4        INCT  R4
1575          0630' PFEXIT EQU   $
1576  0630  0454        B     *R4
1577                    *                                 IF TERMINATING CHAR IS ESCAPE
1578                    *                                 CHAR TAKE ERROR EXIT
```

```
1579          0632' PFERR   EQU   $
1580  0632    C05A          MOV   *R10,R1
1581  0634    0A81          SLA   R1,8
1582  0636    9801          CB    R1,@ESC
      0638    0000
1583  063A    13FA          JEQ   PFEXIT
1584  063C    10F8          JMP   PFBEX
```

```
1587                  * TITLE:    CVTBHX
1588                  *           CONVERT BINARY TO HEX ASCII
1589                  * REVISION: 03/15/76
1590                  *           ORIGINAL
1591                  * COMPUTER: 990,ASSEMBLY
1592                  * ABSTRACT:
1593                  *           CONVERT BINARY VALUE IN R0 TO FOUR HEX
1594                  *           CHARACTERS AND MOVE THEM INTO BUFFER
1595                  * CALLING SEQUENCE:
1596                  *           ENTRY:
1597                  *           R6 CONTAINS CURRENT BUFFER ADDRESS
1598                  *           R0 CONTAINS BINARY WORD TO BE CONVERTED
1599                  *           R4,R5,R7,R8 NON-VOLATILE TO THIS ROUTINE
1600                  *           EXIT:
1601                  *           R6 CONTAINS INCREMENTED BUFFER ADDRESS
1602        063E' CVTBHX EQU   $
1603  063E  0420         BLWP  @ACL
      0640  03CE'
1604                  *                       <CPRBOP>=C
1605                  *                                 GET PRB FOR CONVERSION
1606  0642  020A         LI    R10,CONPRB
      0644  0054'
1607  0646  C38A         MOV   R10,R14
1608                  *                                 OPCODE FOR BINARY TO HEX
1609                  *                                 ASCII CONVERSION
1610  0648  CFA0         MOV   @BHX,*R14+
      064A  005C'
1611                  *                                 DO CONVERSION
1612                  *                       CONPRB=HEXASC(R0)
1613  064C  0420         BLWP  @SVCALT
      064E  05E6'
1614                  *                       DO 300 I=1,4
1615  0650  0203         LI    R3,4
      0652  0004
1616                  *                       BUFFER(I)=CONPRB(I)
1617        0654' CBH1   EQU   $
1618                  *                                 MOVE CHARACTERS INTO BUFFER
1619  0654  DDBE         MOVB  *R14+,*R6+
1620                  *300                    CONTINUE
1621  0656  0603         DEC   R3
1622  0658  16FD         JNE   CBH1
1623                  *                                 SAVE BUFFER ADDR IN PREV WKSP
1624  065A  C846         MOV   R6,@R6*2(13)
      065C  000C
1625  065E  0420         BLWP  @RR
      0660  043E'
1626                  *                       RETURN
1627  0662  0458         B     *R11
```

```
1630                    * TITLE:     WRTRD
1631                    *            WRITE/READ
1632                    * REVISION:  03/15/76
1633                    *            ORIGINAL
1634                    * COMPUTER:  990,ASSEMBLY
1635                    * ABSTRACT:
1636                    *            SELECT LUNO ACCORDING TO OPCODE, THEN PERFORM
1637                    *            I/O OPERATION
1638                    * CALLING SEQUENCE:
1639                    *            ENTRY:
1640                    *            R1 CONTAINS OPCODE
1641                    *            R4-R8 NON-VOLATILE TO THIS ROUTINE
1642                    *            EXIT:
1643                    *            R0 CONTAINS SYSTEM FLAG
1644                    *            R10 CONTAINS ERROR MESSAGE
1645          0664'     WRTRD  EQU   $
1646                    *                    IF READ REQUESTED SET LUNO TO 7
1647                    *                    OTHERWISE SET LUNO TO 8
1648    0664  C820              MOV   @EIGHT,@PRBLU
        0666  0084'
        0668  0061'
1649                    *                                  SET CHARACTER COUNT
1650    066A  C820              MOV   @CHARCT,@PRBCC
        066C  006E'
        066E  0068'
1651    0670  1003              JMP   WRT2
1652          0672'     WRT1   EQU   $                     ENTRY POINT
1653    0672  C820              MOV   @SEVEN,@PRBLU
        0674  0086'
        0676  0061'
1654                    *                    WRITE OR READ
1655          0678'     WRT2   EQU   $
1656                    *                                  SAVE RETURN ADDRESS
1657    0678  C38B              MOV   R11,R14
1658                    *                                  LOAD OPCODE
1659    067A  D801              MOVB  R1,@PRBOP
        067C  0060'
1660                    *                                  GET SVC PRB
1661    067E  020A              LI    R10,PRB
        0680  005E'
1662                    *                                  DO I/O
1663    0682  0420              BLWP  @SVCALT
        0684  064E'
1664    0686  020A              LI    R10,MX01             GET I/O ABRT MSG(JUST IN CASE
        0688  0001
1665                    *                                  SYSTEM FLAGS
1666    068A  C020              MOV   @PRBSF,R0
        068C  0062'
1667    068E  045E              B     *R14
1668    0000                    DORG  0
```

```
1671                    *
1672                    *
1673  0000              WKSPBF BSS  32
1674                    *
1675                    *
1676                    *
1677                    *
1678                    *
1679                    *
1680          0020      CTR    EQU  $
1681  0020    0000             DATA WKSPBF
1682  0022    0104             DATA CTRT
1683          0024      BBAR   EQU  $
1684  0024    0000             DATA WKSPBF
1685  0026    0124             DATA BBART
1686          0028      MGET   EQU  $
1687  0028    0000             DATA WKSPBF
1688  002A    017C             DATA MGETFD
1689          002C      BRM    EQU  $
1690  002C    0000             DATA WKSPBF
1691  002E    01DA             DATA BROM
1692          0030      RGET   EQU  $
1693  0030    0000             DATA WKSPBF
1694  0032    0298             DATA RGETFD
1695          0034      RTM    EQU  $
1696  0034    0000             DATA WKSPBF
1697  0036    01A8             DATA RTOM
```

NEW WORKSPACE BUFFER FOR ALL SUBROUTINES FOLLOWING DORG

BLWP COMMAND TABLE
DATA NEW WORKSPACE BUFFER
DATA ADDRESS OF ROUTINE TO BE EXECUTED
GET NEW WORKSPACE AND POINT PC TO ROUTINE

```
1700                    * TITLE:     BOUNDS
1701                    *            IMAGE BOUNDARIES
1702                    * REVISION: 03/15/76
1703                    *            ORIGINAL
1704                    * COMPUTER: 990,ASSEMBLY
1705                    * ABSTRACT:
1706                    *            USE PARAMETERS TO OBTAIN BEGINNING AND ENDING
1707                    *            IMAGE ADDRESSES.  STORE ADDRESSES IN WORKING
1708                    *            TABLE, CHECK FOR INVALID ENDING ADDRESS.
1709                    * CALLING SEQUENCE:
1710                    *            ENTRY:
1711                    *            R3 CONTAINS ADDRESS OF BOUNDS PARAMETERS
1712                    *            R9 CONTAINS BEGINNING ADDRESS
1713                    *            R8 CONTAINS ENDING ADDRESS
1714                    *            EXIT:
1715                    *            UPON COMPLETION RETURN CONTROL TO MONITOR
1716                    *            THROUGH R2
1717                    *                     MEMORY BOUNDS SUBCOMMAND
1718                    *                     ROM BOUNDS SUBCOMMAND
1719          0038      BOUNDS EQU $
1720                    *   25                IF(EA.LT.BA)CALL ERR
1721    0038  8248              C     R8,R9
1722    003A  1A61              JL    BADAD
1723                    *                     BEGINNING ADDRESS
1724    003C  CCC9              MOV   R9,*R3+
1725                    *                     ENDING ADDRESS
1726    003E  C4C8              MOV   R8,*R3
1727                    *                     RETURN
1728    0040  0452              B     *R2
```

```
1731                    * TITLE:     IMAGE
1732                    *            NON-STANDARD IMAGE
1733                    * REVISION:  03/15/76
1734                    *            ORIGINAL
1735                    * COMPUTER:  990,ASSEMBLY
1736                    * ABSTRACT:
1737                    *            USE PARAMETERS TO OBTAIN LEVEL NUMBER AND
1738                    *            NON-STANDARD IMAGE MAPPING VALUES, STORE
1739                    *            VALUES IN WORKING TABLE, CHECK FOR INVALID
1740                    *            LEVEL NUMBER
1741                    * CALLING SEQUENCE:
1742                    *            ENTRY:
1743                    *            R3 CONTAINS ADDRESS OF IMAGE PARAMETERS
1744                    *            R9 CONTAINS LEVEL
1745                    *            R8 CONTAINS INCREMENT
1746                    *            R7 CONTAINS MAXIMUM LOOP COUNT
1747                    *            R6 CONTAINS DISPLACEMENT
1748                    *            EXIT:
1749                    *            UPON COMPLETION RETURN CONTROL TO MONITOR
1750                    *            THROUGH R2
1751                    *                     MEMORY IMAGE SUBCOMMAND
1752                    *                     ROM IMAGE SUBCOMMAND
1753          0042      IMAGE  EQU $
1754    0042  0201             LI   R1,IMGDIF-2
      0044  0004
1755                    *                     IF(LEVEL.LT.1.OR.LEVEL.GT.3)CALL ERROR
1756    0046  0289             CI   R9,3
      0048  0003
1757    004A  1B56             JH   OUTBD
1758                    *                     LEVEL=(LEVEL-1)*2
1759    004C  0609             DEC  R9
1760    004E  1154             JLT  OUTBD
1761    0050  0A19             SLA  R9,1
1762                    *                     RETURN
1763    0052  A243             A    R3,R9
1764                    *                     INCREMENT FOR LEVEL
1765    0054  CE48             MOV  R8,*R9+
1766    0056  A241             A    R1,R9
1767                    *                     DISPLACEMENT FOR LEVEL
1768    0058  CE46             MOV  R6,*R9+
1769    005A  A241             A    R1,R9
1770                    *                     MAXIMUM LOOP COUNT FOR LEVEL
1771    005C  CE47             MOV  R7,*R9+
1772    005E  0452             B    *R2
```

```
1775                    * TITLE:     SW
1776                    *            STRING WIDTH SUBCOMMAND
1777                    * REVISION: 03/15/76
1778                    *            ORIGINAL
1779                    * COMPUTER: 990,ASSEMBLY
1780                    * ABSTRACT:
1781                    *            CHECK FOR INVALID STRING WIDTH, STORE STRING
1782                    *            WIDTH IN WORKING TABLE.
1783                    * CALLING SEQUENCE:
1784                    *            ENTRY:
1785                    *            R3 CONTAINS ADDRESS OF SW PARAMETERS
1786                    *            R9 CONTAINS STRING WIDTH
1787                    *            EXIT:
1788                    *            UPON COMPLETION RETURN CONTROL TO MONITOR
1789                    *            THROUGH R2
1790           0060     SW      EQU $
1791                    *                        IF(WIDTH.LT.1.OR.WIDTH.GT.>8) CALL ERR
1792    0060   C249             MOV   R9,R9
1793    0062   134A             JEQ   OUTBD
1794    0064   0289             CI    R9,MAXWD
        0066   0008
1795    0068   1847             JH    OUTBD
1796                    *                              MEMORY STRING WIDTH
1797                    *                        SWT=WIDTH
1798    006A   C133             MOV   *R3+,R4
1799    006C   C509             MOV   R9,*R4
1800                    *                              ROM STRING WIDTH
1801                    *                        SWTR=WIDTH
1802    006E   C133             MOV   *R3+,R4
1803    0070   C509             MOV   R9,*R4
1804                    *                        RETURN
1805    0072   0452             B     *R2
```

```
1808                    * TITLE:    RC
1809                    *           ROM CHARACTERISTICS SUBCOMMAND
1810                    * REVISION: 03/15/76
1811                    *           ORIGINAL
1812                    * COMPUTER: 990,ASSEMBLY
1813                    * ABSTRACT:
1814                    *           STORE PARAMETERS IN WORKING TABLE
1815                    *           CHECK FOR INVALID STRING WIDTH, PULSE WIDTH,
1816                    *           PROGRAMMABLE STRING WIDTH, PROGRAMMING TYPE
1817                    *           AND DUTY CYCLE
1818                    * CALLING SEQUENCE:
1819                    *           ENTRY:
1820                    *           R3 CONTAINS ADDRESS OF RC PARAMETERS
1821                    *           R9 CONTAINS ROM WORD WIDTH
1822                    *           R8 CONTAINS PROGRAM TYPE(1'S OR 0'S)
1823                    *           R7 CONTAINS PULSE WIDTH
1824                    *           R6 CONTAINS NUMBER OF RETRIES
1825                    *           R5 CONTAINS DUTY CYCLE
1826                    *           R4 CONTAINS PROGRAMMABLE STRING WIDTH
1827                    *           EXIT:
1828                    *           UPON COMPLETION RETURN CONTROL TO MONITOR
1829                    *           THROUGH R2
1830          0074      RC        EQU $
1831                    *                       IF(WIDTH.LT.1.OR.WIDTH.GT.>8) CALL ERR
1832  0074    C249              MOV   R9,R9
1833  0076    1340              JEQ   OUTBD
1834  0078    0289              CI    R9,MAXWD
      007A    0008
1835  007C    1B3D              JH    OUTBD
1836                    *                       IF(PSTW.GT.WIDTH)CALL ERR
1837  007E    8244              C     R4,R9
1838  0080    1B3B              JH    OUTBD
1839                    *                       IF(POOZ.NE.0.OR.POOZ.NE.1) CALL ERR
1840  0082    0288              CI    R8,YES
      0084    0001
1841  0086    1B38              JH    OUTBD
1842                    *                       IF(PWID1.LT.1.OR.PWID1.GT.MAXPW1)
1843                    *                          CALL ERR
1844  0088    C1C7              MOV   R7,R7
1845  008A    1336              JEQ   OUTBD
1846  008C    0287              CI    R7,MAXPW1
      008E    0006
1847  0090    1B33              JH    OUTBD
1848                    *                       IF(DUCY.LT.0.OR.DUCY.LT.100)CALL ERR
1849  0092    0285              CI    R5,MAXDC
      0094    0064
1850  0096    1B30              JH    OUTBD
1851                    *4                      RETURN
1852                    *                       STORAGE LOCATIONS FOR ROM CHARACTERISTI
1853                    *                       PARAMETERS ARE CONTIGUOUS IN MEMORY,
1854                    *                       ALLOWS LOADING FIRST LOCATION IN R0 AND
1855                    *                       STORING BY AUTOINCREMENTING
1856                    *                       ROM WORD WIDTH
1857  0098    CCC9              MOV   R9,*R3+
```

```
1858                    *                       PROGRAM ONES OR ZEROS
1859  009A  CCC8               MOV   R8,*R3+
1860                    *                       NORMAL PULSE WIDTH
1861  009C  CCC7               MOV   R7,*R3+
1862                    *                       NUMBER OF RETRIES
1863  009E  CCC6               MOV   R6,*R3+
1864                    *                       DUTY CYCLE
1865  00A0  CCC5               MOV   R5,*R3+
1866                    *                       PROGRAMMABLE STRING WIDTH
1867  00A2  CCC4               MOV   R4,*R3+
1868  00A4  0452               B     *R2
```

```
1871                    * TITLE:     CS
1872                    *            SET CRU ROM ADDRESS SUBCOMMAND
1873                    * REVISION:  03/15/76
1874                    *            ORIGINAL
1875                    * COMPUTER:  990,ASSEMBLY
1876                    * ABSTRACT:
1877                    *            STORE CRU ROM ADDRESS IN WORKING TABLE
1878                    *            CHECK FOR INVALID CRU ROM ADDRESS
1879                    * CALLING SEQUENCE:
1880                    *            ENTRY:
1881                    *            R3 CONTAINS ADDRESS OF CS PARAMETER
1882                    *            R9 CONTAINS CRU ROM INTERFACE BASE ADDRESS
1883                    *            EXIT:
1884                    *            UPON COMPLETION RETURN CONTROL TO MONITOR
1885                    *            THROUGH R2.
1886          00A6      CS      EQU  $
1887                    *                      IF(RCRUIA.GT.MAXCRU)CALL ERR
1888    00A6  C249              MOV  R9,R9
1889    00A8  0289              CI   R9,MAXCRU
        00AA  1FFE
1890    00AC  1825              JH   OUTBD
1891                    *                                  CRU INTERFACE ROM BASE ADDRES
1892    00AE  C4C9              MOV  R9,*R3
1893                    *                      RETURN
1894    00B0  0452              B    *R2
```

```
1897                    * TITLE:     TS
1898                    *            SET TOGGLES SUBCOMMAND
1899                    * REVISION: 03/15/76
1900                    *            ORIGINAL
1901                    * COMPUTER: 990,ASSEMBLY
1902                    * ABSTRACT:
1903                    *            CHECK FOR INVALID TOGGLE VALUES, SET TOGGLES IN
1904                    *            WORKING TABLE.
1905                    * CALLING SEQUENCE:
1906                    *            ENTRY:
1907                    *            R3 CONTAINS ADDRESS OF TS PARAMETERS
1908                    *            R9 CONTAINS DISPLAY MEMORY TOGGLE
1909                    *            R8 CONTAINS DISPLAY ROM TOGGLE
1910                    *            R7 CONTAINS DIRECTION TOGGLE
1911                    *            R6 CONTAINS COMPARE TOGGLE
1912                    *            R12 CONTAINS -1, TO BE USED FOR COMPARISON
1913                    *            EXIT:
1914                    *            UPON COMPLETION RETURN CONTROL TO MONITOR
1915                    *            THROUGH R2
1916          00B2      TS      EQU $
1917                    *                       SET LOOP COUNTER, FIRST STORAGE
1918                    *                       LOCATION FOR TOGGLES, ADDRESS OF WR6
1919                    *                       DO 140 I=1,4
1920    00B2  C103              MOV   R3,R4
1921                    *                                 SET COUNTER TO CHECK FOR FOUR
1922                    *                                 TOGGLES
1923    00B4  020F              LI    R15,4
        00B6  0004
1924    00B8  02A1              STWP  R1
1925                    *                                 TOGGLES ARE IN R6-R9
1926    00BA  0221              AI    R1,R6*2           R1=ADDR R6
        00BC  000C
1927    00BE  C141              MOV   R1,R5
1928          00C0  TSFLD1 EQU  $
1929    00C0  060F              DEC   R15
1930                    *                       IF(PARM.EQ.' ')GO TO 140
1931                    *                                 R12=-1; DEFAULT FOR MISSING
1932                    *                                 TOGGLE PARAMETER
1933    00C2  8311              C     *R1,R12
1934    00C4  130E              JEQ   TSFLD2
1935                    *                       IF(PARM.EQ.0.OR.PARM.EQ.1)GO TO 145
1936    00C6  C2D1              MOV   *R1,R11
1937                    *                                 ZERO AND ONE VALID VALUES FOR
1938                    *                                 ALL TOGGLES
1939    00C8  028B              CI    R11,YES
        00CA  0001
1940    00CC  1205              JLE   TSFLD3
1941                    *                                 IF NOT DIRECTION TOGGLE AND
1942                    *                                 VALUE GREATER THAN 1, ABORT
1943                    *                       IF(PARM.NE.DIRECT)GO TO 150
1944    00CE  028F              CI    R15,2
        00D0  0002
1945    00D2  1606              JNE   TSTERR
1946                    *                       IF(PARM.EQ.ROMMEM)GO TO 145
```

```
1947  00D4  0280          CI    R11,ROMMEM
      00D6  0002
1948  00D8  1305          JEQ   TSFLD3
1949                *                                  IF DIRECTION TOGGLE VALUE
1950                *                                  GREATER THAN 3, ABORT
1951                *           IF (PARM.EQ.SAVIMG)GO TO 145
1952  00DA  0280          CI    R11,SAVIMG
      00DC  0003
1953  00DE  1302          JEQ   TSFLD3
1954                *150                    CALL ERR
1955        00E0  TSTERR  EQU   $
1956  00E0  1008          JMP   OUTBD
1957        00E2  TSFLD2  EQU   $
1958                *                                  RETAIN PREVIOUS TOGGLE VALUE
1959  00E2  C453          MOV   *R3,*R1
1960                * 45                     TOGGLE=PARM
1961        00E4  TSFLD3  EQU   $
1962                *                                  GET ADDR OF NEXT TOGGLE
1963  00E4  05C1          INCT  R1
1964                *                                  GET ADDR OF NEXT PREVIOUS
1965                *                                  TOGGLE
1966  00E6  05C3          INCT  R3
1967                *140                    CONTINUE
1968                *                                  CHECK IF ALL TOGGLES
1969                *                                  PROCESSED
1970  00E8  C3CF          MOV   R15,R15
1971  00EA  16EA          JNE   TSFLD1
1972        00EC  TSFLD4  EQU   $
1973                *                       STORAGE LOCATIONS FOR TOGGLES PARAMETER
1974                *                       ARE CONTIGUOUS IN MEMORY
1975                *                       ALLOWS LOADING FIRST LOCATION IN R0 AND
1976                *                       STORING BY AUTOINCREMENTING
1977                *                                  R4 CONTAINS ADDRESS OF FIRST
1978                *                                  STORAGE LOCATION FOR TOGGLES.
1979                *                                  R5 CONTAINS ADDRESS OF FIRST
1980                *                                  TOGGLE. STORE FOUR TOGGLE
1981                *                                  VALUES.
1982  00EC  C035          MOV   *R5+,*R4+
1983  00EE  058F          INC   R15
1984  00F0  028F          CI    R15,4
      00F2  0004
1985  00F4  16FB          JNE   TSFLD4
1986                *  155                  RETURN
1987  00F6  0452          B     *R2
1988        00F8  OUTBD   EQU   $
1989  00F8  020A          LI    R10,PP08
      00FA  3102
1990  00FC  1002          JMP   ERRE
1991        00FE  BADAD   EQU   $
1992  00FE  020A          LI    R10,PPBA
      0100  3104
1993        0102  ERRE    EQU   $
1994  0102  0450          B     *R0
```

```
1997                    * TITLE:     CTRT
1998                    *            COUNT ROUTINE
1999                    * REVISION:  03/15/76
2000                    *            ORIGINAL
2001                    * COMPUTER:  990,ASSEMBLY
2002                    * ABSTRACT:
2003                    *            COMPUTE COUNTERS TO BE USED TO CALCULATE
2004                    *            BEGINNING MEMORY AND ROM BIT ADDRESSES.
2005                    * CALLING SEQUENCE:
2006                    *            ENTRY:
2007                    *            R14 CONTAINS ADDRESS OF DATA LOCATIONS PASSED
2008                    *            BY CALLING PROGRAM
2009                    *            R13-R15 CONTAIN CONTROL VALUES FOR PREVIOUS
2010                    *            WORKSPACE, AND ARE NON-VOLATILE
2011                    *            R5,R6 CONCATENATED CONTAIN K(ITERATION COUNT)
2012                    *            EXIT:
2013                    *            NONE
2014          0104  CTRT    EQU $
2015                    *                        COMPUTE CT(1), CT(2), CT(3)
2016  0104    C27E            MOV   *R14+,R9          GET ADDRESS MT(1)
2017  0106    C23E            MOV   *R14+,R8
2018  0108    C16D            MOV   @R5*2(R13),R5     GET LOOP COUNTERS FROM OLD WK
      010A    000A
2019  010C    C1AD            MOV   @R6*2(R13),R6
      010E    000C
2020  0110    04C4            CLR   R4
2021  0112    8645            C     R5,*R9            IF K1>MT1 DO DOUBLE
2022  0114    1A01            JL    CTR1              REGISTER DIVIDE
2023  0116    3D39            DIV   *R9+,R4
2024          0118  CTR1    EQU $
2025  0118    3D79            DIV   *R9+,R5           X=K/MT1
2026  011A    CE06            MOV   R6,*R8+           CT1=MOD(X)
2027  011C    3D39            DIV   *R9+,R4           Y=(INT(X))/MT2
2028  011E    CE05            MOV   R5,*R8+           CT2=MOD(Y)
2029  0120    CE04            MOV   R4,*R8+           CT3=INT(Y)
2030  0122    0380            RTWP
```

```
2033                    * TITLE:    BBART
2034                    *           BEGINNING BIT ADDRESS ROUTINE
2035                    * REVISION: 03/15/76
2036                    *           ORIGINAL
2037                    * COMPUTER: 990,ASSEMBLY
2038                    * ABSTRACT:
2039                    *           COMPUTE BEGINNING BIT, BYTE, WORD ADDRESSES AND
2040                    *           BEGINNING BIT WITHIN BYTE OR WORD
2041                    * CALLING SEQUENCE:
2042                    *           ENTRY:
2043                    *           R14 CONTAINS ADDRESS OF DATA PASSED BY CALLING
2044                    *           ROUTINE
2045                    *           R13-R15 CONTAIN CONTROL VALUES FOR PREVIOUS
2046                    *           WORKSPACE, AND ARE NON-VOLATILE
2047                    *           EXIT:
2048                    *           R0 AND R1 CONCATENATED CONTAIN BEGINNING BIT
2049                    *           ADDRESS
2050          0124      BBART   EQU $
2051                    *                       R0=DMT(I)
2052  0124    0200              LI    R0,IMGDIF         (DMT=IMT=DRT=IRT)
      0126    0006
2053  0128    C27E              MOV   *R14+,R9           GET ADDRESS OF IMT OR IRT
2054  012A    A009              A     R9,R0
2055  012C    04C1              CLR   R1
2056                    *                       (R7,R8)=0
2057  012E    39C1              MPY   R1,R7             CLR R7,R8
2058  0130    C17E              MOV   *R14+,R5           GET ADDRESS OF CT
2059          0132      BBART2  EQU   $
2060                    *                       (R7,R8)=(R7,R8)+DMT(I)
2061  0132    A230              A     *R0+,R8
2062  0134    1701              JNC   BBFLD1
2063  0136    0587              INC   R7
2064          0138      BBFLD1  EQU   $
2065                    *                       DO 3 I=1,3
2066                    *                       (R7,R8)=(R7,R8)+(T(I)*IT(I)
2067  0138    C0B5              MOV   *R5+,R2
2068                    *                             MULTIPLY IT(I)*CT(I)
2069  013A    3889              MPY   *R9+,R2
2070                    *                             ADD PRODUCT TO TOTAL
2071  013C    A203              A     R3,R8
2072  013E    1701              JNC   BBFLD2
2073  0140    0587              INC   R7
2074          0142      BBFLD2  EQU   $
2075  0142    A1C2              A     R2,R7
2076                    *3                      CONTINUE
2077  0144    05C1              INCT  R1
2078  0146    0281              CI    R1,3*2
      0148    0006
2079  014A    16F3              JNE   BBART2
2080                    *                       BBA=BA*WIDTH+(R7,R8)
2081  014C    C03E              MOV   *R14+,R0
2082  014E    C090              MOV   *R0,R2              GET BEGINNING ADDR
2083  0150    C07E              MOV   *R14+,R1
2084                    *                             MULTIPLY ADDRESS*WIDTH
```

```
2085  0152  3891          MPY   *R1,R2
2086                *                                ADD PRODUCT TO TOTAL
2087  0154  A203          A     R3,R8
2088  0156  1701          JNC   BBFLD3
2089  0158  0587          INC   R7
2090        015A  BBFLD3   EQU   $
2091  015A  A1C2          A     R2,R7
2092                *                    BA=BBA/WIDTH
2093                *                    BBT=MOD(BBA/WIDTH)
2094  015C  CB47          MOV   R7,@R0*2(R13)      SAVE BIT ADDRESS IN R0 AND R1
      015E  0000
2095  0160  CB48          MOV   R8,@R1*2(R13)
      0162  0002
2096                *                                R1 CONTAIN WIDTH
2097  0164  3DD1          DIV   *R1,R7
2098  0166  C03E          MOV   *R14+,R0
2099                *                                GET ADDR OF BRWA OR BMBYA
2100  0168  C407          MOV   R7,*R0
2101  016A  C03E          MOV   *R14+,R0
2102                *                                GET ADDR OF BRWDBT OR BMBYBT
2103  016C  C408          MOV   R8,*R0
2104  016E  C07E          MOV   *R14+,R1           CHECK THAT LOOPING DOES NOT
2105  0170  8447          C     R7,*R1             PASS INDICATED ENDING MEMORY
2106  0172  1802          JH    BBERR              OR ROM ADDRESS
2107  0174  05CE          INCT  R14                SKIP ERROR EXIT
2108        0176  BBFLD4   EQU   $
2109  0176  0380          RTWP
2110        0178  BBERR    EQU   $
2111  0178  C39E          MOV   *R14,R14           MOVE ERROR EXIT TO PC FOR RET
2112  017A  10FD          JMP   BBFLD4
```

```
2115                    * TITLE:     MGETFD
2116                    *            GET STRING FROM MEMORY
2117                    * REVISION: 03/15/76
2118                    *            ORIGINAL
2119                    * COMPUTER: 990,ASSEMBLY
2120                    * ABSTRACT:
2121                    *            GET STRING FROM MEMORY AND STORE TEMPORARILY IN
2122                    *            MEMORY LOCATION MSTRG
2123                    * CALLING SEQUENCE:
2124                    *            ENTRY:
2125                    *            R14 CONTAINS ADDRESS OF DATA PASSED BY CALLING
2126                    *            ROUTINE
2127                    *            R13-R15 CONTAIN CONTROL VALUES FOR PREVIOUS
2128                    *            WORKSPACE, AND ARE NON-VOLATILE
2129                    *            EXIT:
2130                    *            NONE
2131          017C      MGETFD EQU $
2132                    *                              MSTRG=<BMBYA>*256+<BMBYA+1>
2133  017C    C03E              MOV   *R14+,R0
2134  017E    C050              MOV   *R0,R1
2135  0180    D271              MOVB  *R1+,R9
2136  0182    0989              SRL   R9,8
2137  0184    D271              MOVB  *R1+,R9
2138                    *                                    R9 CONTAINS BEGINNING MEMORY
2139                    *                                    BYTE + FOLLOWING BYTE
2140  0186    0B89              SRC   R9,8
2141                    *                              SHIFT=BMBYBT
2142  0188    C07E              MOV   *R14+,R1
2143  018A    C011              MOV   *R1,R0
2144                    *                              MSTRG=ALSHFT(MSTRG,SHIFT)
2145  018C    1301              JEQ   MGFLD1
2146                    *                                    LEFT JUSTIFY DATA CONTAINING
2147                    *                                    STRING
2148  018E    0A09              SLA   R9,0
2149          0190      MGFLD1 EQU  $
2150  0190    06C9              SWPB  R9                     WORK WITH VALUE IN LSB R9
2151                    *                              MASK=-1
2152  0192    070A              SETO  R10
2153                    *                              SHIFT=SWT
2154  0194    C07E              MOV   *R14+,R1
2155  0196    C011              MOV   *R1,R0
2156                    *                              MASK=LRSHFT(MASK,SHIFT)
2157                    *                                    MASK IS LEFT-JUSTIFIED STRING
2158                    *                                    OF ZEROES OF STRING WIDTH
2159  0198    090A              SRL   R10,0
2160  019A    06CA              SWPB  R10                    WORK WITH MASK IN LSB R10
2161                    *                              MSTRG=AND(XOR(MASK,-1)MSTRG)
2162                    *                                    MASK OUT GARBAGE AT RIGHT OF
2163                    *                                    STRING AND IN MSB
2164  019C    424A              SZC   R10,R9
2165  019E    0A09              SLA   R9,0                   RIGHT JUSTIFY VALUE IN MSB R9
2166  01A0    06C9              SWPB  R9                     MSTRG RIGHT JUSTIFIED IN R9
2167  01A2    C07E              MOV   *R14+,R1
2168                    *                                    STORE STRING IN MEMORY
```

```
2169  01A4  C449          MOV   R9,*R1
2170                 *                        RETURN
2171  01A6  0380          RTWP
```

```
2174                    * TITLE:     RTOM
2175                    *            ROM TO MEMORY
2176                    * REVISION:  03/15/76
2177                    *            ORIGINAL
2178                    * COMPUTER:  990,ASSEMBLY
2179                    * ABSTRACT:
2180                    *            STORE STRING FROM ROM INTO LOCATION IN MEMORY
2181                    * CALLING SEQUENCE:
2182                    *            ENTRY:
2183                    *            R14 CONTAINS ADDRESS OF DATA PASSED BY CALLING
2184                    *            ROUTINE
2185                    *            R13-R15 CONTAIN CONTROL VALUES FOR PREVIOUS
2186                    *            WORKSPACE, AND ARE NON-VOLATILE
2187                    *            EXIT:
2188                    *            NONE
2189          01A8      RTOM   EQU  $
2190                    *                        STEMP=<BMYA>*256+<BMBYA+2>
2191    01A8  C03E              MOV  *R14+,R0
2192    01AA  C050              MOV  *R0,R1
2193    01AC  D1F1              MOVB *R1+,R7
2194    01AE  0987              SRL  R7,8
2195    01B0  D1D1              MOVB *R1,R7
2196                    *                                  R7 CONTAINS BEGINNING MEMORY
2197                    *                                  BYTE + FOLLOWING BYTE
2198    01B2  0887              SRC  R7,8
2199                    *                        MASK=-1
2200    01B4  070A              SETO R10
2201                    *                        SHIFT=SWT
2202    01B6  C1BE              MOV  *R14+,R6
2203    01B8  C016              MOV  *R6,R0              STRING WIDTH
2204                    *                        MASK=LRSHFT(MASK,SHIFT)
2205                    *                                  MASK IS LEFT-JUSTIFIED STRING
2206                    *                                  OF ZEROES OF BIT STRING WIDTH
2207    01BA  090A              SRL  R10,R0
2208                    *                        SHIFT=BMBYBT
2209    01BC  C17E              MOV  *R14+,R5
2210    01BE  C015              MOV  *R5,R0
2211                    *                        MASK=CRSHFT(MASK,SHIFT)
2212                    *                                  SHIFT MASK SO STRING OF
2213                    *                                  ZEROES CORRESPONDS TO DATA
2214                    *                                  STRING IN MEMORY BYTE
2215    01C0  0B0A              SRC  R10,0
2216    01C2  054A              INV  R10
2217                    *                        STEMP=AND(XOR(MASK,-1),STEMP)
2218                    *                                  MASK OUT BIT STRING IN MEMORY
2219                    *                                  WORD
2220    01C4  41CA              SZC  R10,R7
2221    01C6  C17E              MOV  *R14+,R5
2222                    *                                  RSTRG RIGHT-JUSTIFIED, LEFT
2223                    *                                  FILLED WITH ZEROES
2224    01C8  C255              MOV  *R5,R9
2225                    *                        SHIFT=BMBYBT+SWT
2226    01CA  A016              A    *R6,R0
2227                    *                                  LEFT JUSTIFY STRING;
```

```
2228                    *                       SHIFT SO ROM BIT STRING
2229                    *                       CORRESPONDS TO MASKED OUT
2230                    *                       DATA LOCATION IN MEMORY WORD
2231                    *                 RSTRG=CRSHFT(RSTRG,SHIFT)
2232  01CC  0B09               SRC   R9,R0
2233                    *                 RSTRG=OR(RSTRG,STEMP)
2234                    *                       R7 CONTAINS ROM BIT STRING IN
2235                    *                       MEMORY WORD
2236  01CE  E1C9               SOC   R9,R7
2237                    *                 <BMBYA>=RSTRG/256
2238                    *                 <BMBYA+2>=MOD(RSTRG/256)
2239  01D0  0601               DEC   R1              <R1>=BMBYA
2240                    *                       STORE BACK INTO MEMORY
2241  01D2  DC47               MOVB  R7,*R1+
2242  01D4  06C7               SWPB  R7
2243  01D6  D447               MOVB  R7,*R1
2244  01D8  0380               RTWP
```

```
2247                    * TITLE:     BROM
2248                    *            GET ROM WORD STRING
2249                    * REVISION:  03/15/76
2250                    *            ORIGINAL
2251                    * COMPUTER:  990,ASSEMBLY
2252                    * ABSTRACT:
2253                    *            PASS MEMORY STRING AND ROM WORD ADDRESS TO
2254                    *            'BURN WORD' SUBROUTINE ONE WORD AT A TIME
2255                    * CALLING SEQUENCE:
2256                    *            ENTRY:
2257                    *            R14 CONTAINS ADDRESS OF DATA PASSED BY CALLING
2258                    *            ROUTINE
2259                    *            R13-R15 CONTAIN CONTROL VALUES FOR PREVIOUS
2260                    *            WORKSPACE, AND ARE NON-VOLATILE
2261                    *            EXIT:
2262                    *            NONE
2263          01DA      BROM    EQU $
2264  01DA    C07E              MOV   *R14+,R1
2265                    *                                  BEGINNING BIT WITHIN ROM WORD
2266  01DC    C001              MOV   *R1,R3
2267                    *                           MASK=-1
2268  01DE    0708              SETO R8
2269                    *                           SHIFT=SWT
2270  01E0    C07E              MOV   *R14+,R1
2271  01E2    C011              MOV   *R1,R0
2272                    *                           MASK=LRSHFT(MASK,SHIFT)
2273                    *                                  MASK IS LEFT-JUSTIFIED STRING
2274                    *                                  OF ZEROES OF STRING WIDTH
2275  01E4    0908              SRL   R8,0
2276  01E6    C07E              MOV   *R14+,R1
2277                    *                                  MEMORY STRING RIGHT-JUSTIFIED
2278                    *                                  LEFT ZERO FILLED
2279  01E8    C291              MOV   *R1,R10
2280  01EA    0B0A              SRC   R10,0                LEFT JUSTIFY STRING
2281                    *                           DO 200 ADDR=BRWA,ERWA
2282  01EC    C07E              MOV   *R14+,R1
2283                    *                                  BEGINNING ROM WORD ADDRESS
2284  01EE    C1D1              MOV   *R1,R7
2285                    *                                  BEGIN LOOP TO SELECT SUB-
2286                    *                                  STRING TO BURN INTO PROM
2287          01F0      BRFLD1  EQU   $
2288                    *                           SHIFT=16-WWID+BRWDBT
2289  01F0    0200              LI    R0,16
        01F2    0010
2290  01F4    C07E              MOV   *R14+,R1
2291  01F6    6011              S     *R1,R0
2292  01F8    A003              A     R3,R0
2293                    *                           MASK=CRSHFT(MASK,SHIFT)
2294                    *                                  RIGHT JUSTIFY A SUBSTRING OF
2295                    *                                  ROM WORD WIDTH OF BOTH MASK
2296                    *                                  AND MEMORY STRING
2297  01FA    0B08              SRC   R8,0
2298                    *                           MSTRG=CRSHFT(MSTRG,SHIFT)
2299                    *                                  FIRST SUBSTRING IS SHIFTED TO
```

```
2300                    *                                   BEGIN AT BEGINNING ROM WORD
2301                    *                                   BIT
2302  01FC  0B0A               SRC   R10,0
2303                    *                         BRWDBT=0
2304                    *                                   SUBSEQUENT SUBSTRINGS BEGIN
2305                    *                                   AT BIT ZERO
2306  01FE  0403               CLR   R3
2307                    *                         SAVE MSTRG
2308  0200  C24A               MOV   R10,R9
2309                    *                         IF(POOZ.EQ.0)STEMP=OR(MASK,MSTRG)
2310  0202  C03E               MOV   *R14+,R0
2311  0204  C050               MOV   *R0,R1
2312  0206  1601               JNE   BRFLD2
2313                    *                                   IF PROGRAMMING ZEROES MASK
2314                    *                                   BITS LEFT OF SUBSTRING TO
2315                    *                                   ONES
2316  0208  E248               SOC   R8,R9                 RIGHT JUSTIFIED SUBSTRING
2317        020A      BRFLD2   EQU   $
2318  020A  C03E               MOV   *R14+,R0
2319                    *                                   BURN SUBSTRING
2320  020C  0690               BL    *R0
2321  020E  1009               JMP   RETERR                JUMP TO PROCESS ERROR RETURN
2322                    *                         CONTINUE
2323                    *                                   INCREMENT ROM WORD ADDRESS
2324  0210  05CE               INCT  R14                   SKIP ERROR RETURN
2325  0212  0587               INC   R7
2326  0214  C07E               MOV   *R14+,R1
2327                    *                                   IF NOT LAST WORD LOOP BACK TO
2328                    *                                   CONTINUE SELECTING SUBSTRINGS
2329  0216  8447               C     R7,*R1
2330  0218  1B03               JH    BRFLD3
2331  021A  022E               AI    R14,-10
      021C  FFF6
2332  021E  10E8               JMP   BRFLD1
2333        0220      BRFLD3   EQU   $
2334                    *                         RETURN
2335  0220  0380               RTWP
2336        0222      RETERR   EQU   $
2337  0222  C39E               MOV   *R14,R14              MOVE ERROR EXIT TO PC FOR RET
2338  0224  10FD               JMP   BRFLD3
```

```
2341                    * TITLE:     LOADCR
2342                    *            LOAD CRU
2343                    * REVISION:  03/15/76
2344                    *            ORIGINAL
2345                    * COMPUTER:  990,ASSEMBLY
2346                    * ABSTRACT:
2347                    *            LOAD CRU REGISTER
2348                    * CALLING SEQUENCE:
2349                    *            ENTRY:
2350                    *            R11 CONTAINS ADDRESS OF DATA PASSED BY CALLING
2351                    *            ROUTINE
2352                    *            LEAST SIGNIFICANT BYTE OF R1 CONTAINS DATA FOR
2353                    *            CRU INTERFACE
2354                    *            R4,R5,R7,R9 NON-VOLATILE TO THIS ROUTINE
2355                    *            EXIT:
2356                    *            NONE
2357          0226      LOADCR EQU  $
2358                    *                                   MOVE ROM REGISTER SELECT TO
2359                    *                                   MSB R1
2360  0226  D07B              MOVB  *R11+,R1
2361  0228  058B              INC   R11
2362                    *                        LOAD CRU WITH <R1>
2363                    *                                   REGISTER SELECT AND DATA
2364  022A  33C1              LDCR  R1,15
2365                    *                                   MOVE DATA TO SELECTED REG.
2366  022C  1D0D              SBO   LD
2367                    *                                   RESET LOAD BIT
2368  022E  1E0D              SBZ   LD
2369                    *                        RETURN
2370  0230  045B              B     *R11
```

```
2373                    * TITLE:     BNCYL
2374                    *            BURN CYCLE
2375                    * REVISION: 03/15/76
2376                    *            ORIGINAL
2377                    * COMPUTER: 990,ASSEMBLY
2378                    * ABSTRACT:
2379                    *            SET GO BIT ON CRU INTERFACE, TIME PROGRAMMING
2380                    *            CYCLE, AND USE DUTY CYCLE TO CAUSE PAUSE BEFORE
2381                    *            NEXT CYCLE
2382                    * CALLING SEQUENCE:
2383                    *            ENTRY:
2384                    *            R11 CONTAINS ADDRESS OF DATA PASSED BY CALLING
2385                    *            ROUTINE
2386                    *            R3 CONTAINS ERROR EXIT
2387                    *            R4,R5,R6 NON-VOLATILE TO THIS ROUTINE
2388                    *            EXIT:
2389                    *            NONE
2390          0232      BNCYL  EQU   $
2391                    *                              <IF(ONLNOT.EQ.1)CALL ERR
2392                    *                                   CLEAR DUTY CYCLE COUNTER
2393  0232    04CE              CLR   R14
2394  0234    1F0E              TB    0L
2395  0236    1328              JEQ   ONLERR
2396                    *                              DISABLE INTERRUPT
2397  0238    1E0E              SBZ   IE
2398                    *                                   RESET GO BIT
2399  023A    1E0F              SBZ   GOBIT
2400                    *                                   SET GO BIT
2401  023C    1D0F              SBO   GOBIT
2402                    *                              IF(BUSY.NE.1)GO TO 500
2403          023E      BNC1   EQU   $
2404                    *                                   LOOP UNTIL BUSY BIT GOES HIGH
2405                    *                                   BEFORE BEGINNING DC COUNT
2406  023E    1F0F              TB    BUSY
2407  0240    16FE              JNE   BNC1
2408          0242      BNC2   EQU   $
2409                    *                              IF(DUCYCT.GT.MAXCT)CALL ERR
2410                    *                                   COMPUTE DUTY CYCLE COUNT
2411  0242    058E              INC   R14
2412                    *                                   IF PROGRAMMING DOES NOT
2413                    *                                   TERMINATE IN REASONABLE TIME
2414                    *                                   CALL ERROR
2415  0244    028E              CI    R14,MAXCT
      0246    FDE8
2416  0248    1B22              JH    HWERR
2417                    *                                   LOOP WHILE PROGRAMMING
2418  024A    1F0F              TB    BUSY
2419  024C    1601              JNE   CONTIN
2420  024E    10F9              JMP   BNC2
2421                    *  500                         WAITCT=((100-DC)/DC)*DUCYCT
2422          0250      CONTIN EQU   $
2423                    *                                   RESET GO BIT
2424  0250    1E0F              SBZ   GOBIT
2425                    *                              IF(VCC.EQ.1.OR.VC.EQ.1)CALL ERR
```

```
2426  0252  0200          LI   R0,100
      0254  0064
2427                *                                   GET DUTY CYCLE
2428  0256  05CB          INCT R11                      SKIP ERROR RETURN
2429  0258  C3FB          MOV  *R11+,R15
2430  025A  601F          S    *R15,R0
2431  025C  C040          MOV  R0,R1
2432  025E  04C0          CLR  R0                       PREPARE FOR DIVISION
2433  0260  3C3F          DIV  *R15+,R0
2434  0262  C041          MOV  R1,R1
2435  0264  1301          JEQ  GOON
2436  0266  0580          INC  R0
2437        0268  GOON    EQU  $
2438                *                                   R0,R1 CONCATENATED CONTAIN
2439                *                                   WAIT COUNT
2440  0268  380E          MPY  R14,R0
2441                *                                   CHECK FOR ZERO COUNT
2442  026A  C041          MOV  R1,R1
2443  026C  1308          JEQ  OUTLP
2444                *                         DO 510 L=1,WAITCT
2445        026E  NEXTLP  EQU  $
2446                *                                   TIMING IN COUNT DOWN LOOP
2447                *                                   APPROXIMATELY EQUAL COUNT UP
2448                *                                   IN PROGRAMMING LOOP
2449  026E  1000          NOP
2450  0270  1000          NOP
2451  0272  1000          NOP
2452  0274  1000          NOP
2453  0276  1000          NOP
2454  0278  1000          NOP
2455  027A  0601          DEC  R1
2456  027C  1BF8          JH   NEXTLP
2457        027E  OUTLP   EQU  $
2458                *                                   CHECK FOR END OF COUNT DOWN
2459  027E  C000          MOV  R0,R0
2460  0280  1302          JEQ  OUT
2461  0282  0600          DEC  R0
2462  0284  10F4          JMP  NEXTLP
2463                *                               RETURN
2464        0286  OUT     EQU  $
2465  0286  045B          B    *R11
2466        0288  ONLERR  EQU  $
2467  0288  020A          LI   R10,PPOL
      028A  3105
2468  028C  1002          JMP  ERJMP2
2469        028E  HWERR   EQU  $
2470  028F  020A          LI   R10,PPHW
      0290  3105
2471        0292  ERJMP2  EQU  $
2472                *                                   ERROR RETURN
2473  0292  CB4A          MOV  R10,@R10*2(R13)          SAVE ERROR MESSAGE
      0294  0014
2474  0296  045B          B    *R11
```

```
2477                    * TITLE:     RGETFD
2478                    *            GET STRING FROM ROM
2479                    * REVISION: 03/15/76
2480                    *            ORIGINAL
2481                    * COMPUTER: 990,ASSEMBLY
2482                    * ABSTRACT:
2483                    *            GET STRING FROM ROM AND STORE TEMPORARILY IN
2484                    *            ROM LOCATION RSTRG
2485                    * CALLING SEQUENCE:
2486                    *            ENTRY:
2487                    *            R14 CONTAINS ADDRESS OF DATA PASSED BY CALLING
2488                    *            ROUTINE
2489                    *            R13-R15 CONTAIN CONTROL VALUES FOR PREVIOUS
2490                    *            WORKSPACE, AND ARE NON-VOLATILE
2491                    *            EXIT:
2492                    *            NONE
2493          0298      RGETFD EQU  $
2494                    *                        SHIFT=WWID
2495  0298    C0FE              MOV   *R14+,R3
2496  029A    C293              MOV   *R3,R10
2497                    *                        COUNT=SHIFT-BRWDBT
2498  029C    C07E              MOV   *R14+,R1
2499  029E    6291              S     *R1,R10
2500                    *                        WDMASK:=-1
2501  02A0    0708              SETO  R8
2502  02A2    C013              MOV   *R3,R0
2503                    *                        WDMASK=ALSHFT(WDMASK,SHIFT)
2504                    *                                 MASK IS RIGHT-JUSTIFIED
2505                    *                                 STRING OF ZEROES OF ROM WORD
2506                    *                                 WIDTH
2507  02A4    0A08              SLA   R8,R0
2508  02A6    C07E              MOV   *R14+,R1
2509                    *                                 MOVE CRU BASE ADDRESS TO R12
2510  02A8    C311              MOV   *R1,R12
2511                    *                        SHIFT=SWT
2512  02AA    C1FE              MOV   *R14+,R7
2513  02AC    C017              MOV   *R7,R0
2514                    *                        STMASK:=-1
2515  02AE    0709              SETO  R9
2516                    *                        STMASK=LRSHFT(STMASK,SHIFT)
2517                    *                                 STMASK IS LEFT-JUSTIFIED
2518                    *                                 STRING OF ZEROES OF BIT
2519                    *                                 STRING WIDTH
2520  02B0    0909              SRL   R9,R0
2521                    *                        DO 210 ADDR=BRWA,ERWA,1
2522  02B2    C07E              MOV   *R14+,R1
2523                    *                                 GET BRWA
2524  02B4    C151              MOV   *R1,R5
2525                    *                                 PROCESS ERROR IF PROM
2526                    *                                 PROGRAMMER OFF LINE
2527  02B6    1F0E              TB    0L
2528  02B8    1322              JEQ   OFFLN
2529                    *                                 SKIP ERROR RETURN
2530  02BA    05CE              INCT  R14
```

```
2531          02BC  RGFLD1 EQU  $
2532                *                        <CRURG0>=ADDR/256
2533  02BC  C045           MOV  R5,R1
2534                *                        GET MSB OF ROM WORD ADDRESS
2535  02BE  0981           SRL  R1,8
2536                *                        STORE IN ROM INTERFACE
2537                *                        REGISTER 0
2538  02C0  06A0           BL   @LOADCR
      02C2  0226
2539  02C4  0000           DATA ADDR0
2540                *                        <CRURG1>=MOD(ADDR/256)
2541                *                        GET LSB OF ROM WORD ADDRESS
2542  02C6  C045           MOV  R5,R1
2543                *                        STORE IN ROM INTERFACE
2544                *                        REGISTER 1
2545  02C8  06A0           BL   @LOADCR
      02CA  0226
2546  02CC  0100           DATA ADDR1
2547                *                   RSTRG=<ADDR>
2548  02CE  C07E           MOV  *R14+,R1
2549                *                        SELECT ROM INTERFACE REGISTER
2550                *                        7 FOR READOUT
2551  02D0  32D1           LDCR *R1,11
2552                *                        GET DATA OF ROM WORD RIGHT=
2553                *                        JUSTIFIED IN MSB R1
2554  02D2  3601           STCR R1,8
2555                *                   RSTRG=CRSHFT(RSTRG,8)
2556                *                        MOVE TO LSB R1
2557  02D4  06C1           SWPB R1
2558                *                   MASK ALL BUT RIGHT JUSTIFIED SUBSTRING
2559                *                   OF ROM WORD WIDTH TO ZEROS
2560  02D6  4048           SZC  R8,R1
2561                *                   SHIFT=COUNT
2562  02D8  C004           MOV  R10,R0
2563                *                   RSTRG=CRSHFT(RSTRG,SHIFT)
2564                *                        SHIFT DATA SO BEGINNING
2565                *                        STRING BIT WITHIN SUBSTRING
2566                *                        OF ROM WORD WIDTH IS LEFT
2567                *                        JUSTIFIED.
2568  02DA  0801           SRC  R1,R0
2569                *                   MASK ALL BITS OTHER THAN THOSE IN BOTH
2570                *                   SUBSTRING AND STRING
2571  02DC  4049           SZC  R9,R1
2572                *                        SAVE PRESENT MASKED DATA IN
2573                *                        R9
2574  02DE  E241           SOC  R1,R9
2575                *                   COUNT=COUNT+WWID
2576  02E0  A293           A    *R3,R10
2577                *  210              CONTINUE
2578                *                        GET NEXT ADDRESS AND CHECK IF
2579                *                        GREATER THAN ERWA
2580  02E2  0585           INC  R5
2581  02E4  C07E           MOV  *R14+,R1
2582  02E6  8151           C    *R1,R5
2583  02E8  1A03           JL   RGFLD2
```

```
2584                    *                                   IF NOT, RESET R3 TO POINT TO
2585                    *                                   CORRECT DATA WORD IN CALLING
2586                    *                                   PROGRAM
2587  02EA  022E                AI    R14,-4
      02EC  FFFC
2588  02EE  10E5                JMP   RGFLD1
2589        02F0  RGFLD2  EQU   $
2590                    *                         RSTRG RIGHT JUSTIFIED
2591                    *                                   LEFT BITS ZEROED OUT
2592  02F0  0200                LI    R0,16
      02F2  0010
2593  02F4  6017                S     *R7,R0
2594  02F6  0909                SRL   R9,0
2595  02F8  C07E                MOV   *R14+,R1
2596                    *                                   STORE IN RSTRG
2597  02FA  C449                MOV   R9,*R1
2598                    *                         RETURN
2599        02FC  RGFLD3  EQU   $
2600  02FC  0380                RTWP
2601                    *                                   GET ERROR MESSAGE AND ERROR
2602                    *                                   RETURN
2603        02FE  OFFLN   EQU   $
2604  02FE  020A                LI    R10,PPOL
      0300  3106
2605  0302  CB4A                MOV   R10,@R10*2(R13)
      0304  0014
2606  0306  C39E                MOV   *R14,R14
2607  0308  10F9                JMP   RGFLD3
```

```
2610                * TITLE:    CHECK
2611                *           CHECK FOR GIVEN RECORD ON TAPE
2612                * REVISION: 03/15/76
2613
2614                *           ORIGINAL
2615                * COMPUTER: 990,ASSEMBLY
2616                * ABSTRACT:
2617                *           COMPARE BUFFER TO INPUTTED NAME AND INDICATE
2618                *           ON RETURN IF MATCH FOUND
2619                * CALLING SEQUENCE:
2620                *           ENTRY:
2621                *           R11 CONTAINS ADDRESS OF DATA PASSED BY CALLING
2622                *           ROUTINE
2623                *           R1 CONTAINS ADDRESS OF FIRST BYTE OF NAME
2624                *           R0 CONTAINS CHARACTER COUNT OF NAME
2625                *           R4,R5,R7,R8 NON-VOLATILE TO THIS ROUTINE
2626                *           EXIT:
2627                *           R6 CONTAINS INCREMENTED BUFFER ADDRESS
2628         030A   CHECK  EQU  $
2629                *                        R6=RFR
2630   030A  C1BB          MOV  *R11+,R6
2631                *                        TYPE=(BUFFER)
2632                *                        BUFFER=BUFFER+1
2633   030C  D3F6          MOVB *R6+,R15
2634   030E  098F          SRL  R15,8
2635                *                        DO 100 I=1,COUNT
2636                *                        IF(<BUFFER>,NE,<ADDR>)GO TO  110
2637         0310   CH1    EQU  $
2638                *                              COMPARE USER INPUTTED NAME TO
2639                *                              NAME IN BUFFER
2640   0310  9C76          CB   *R6+,*R1+
2641   0312  1604          JNE  CH3
2642                *100                    CONTINUE
2643   0314  0600          DEC  R0
2644   0316  16FC          JNE  CH1
2645   0318  0586          INC  R6                SKIP BLANK
2646                *                        INDICATE CHECK FOUND
2647   031A  05CB          INCT R11
2648                *110                    RETURN
2649         031C   CH3    EQU  $
2650   031C  045B          B    *R11
```

```
2653                    * TITLE:     CVTWRT
2654                    *            CONVERT AND WRITE
2655                    * REVISION:  03/15/76
2656                    *            ORIGINAL
2657                    * COMPUTER:  990,ASSEMBLY
2658                    * ABSTRACT:
2659                    *            TRANSFER  WORKING TABLE TO BUFFER AND WRITE
2660                    *            BUFFER
2661                    * CALLING SEQUENCE:
2662                    *            ENTRY:
2663                    *            R11 CONTAINS ADDRESS OF DATA PASSED BY CALLING
2664                    *            ROUTINE
2665                    *            R7 CONTAINS ADDRESS OF FIRST BYTE OF NAME
2666                    *            R6 CONTAINS PRESENT BUFFER ADDRESS
2667                    *            R9 CONTAINS TABLE SIZE
2668                    *            R14 CONTAINS ADDRESS OF FIRST WORD OF TABLE
2669                    *            R8 CONTAINS CHARACTER COUNT OF NAME
2670                    *            R4,R5 NON-VOLATILE TO THIS ROUTINE
2671                    *            EXIT:
2672                    *            R6 CONTAINS INCREMENTED BUFFER ADDRESS
2673          031E      CVTWRT EQU   $
2674          031E      CVTW1  EQU   $
2675                    *                        DO 200 I=1,COUNT
2676                    *                        <BUFFER>=<ADDR>
2677                    *                                 MOVE NAME TO BUFFER
2678    031E  0DB7              MOVB *R7+,*R6+
2679                    *                        CONTINUE
2680    0320  0608              DEC  R8
2681    0322  16FD              JNE  CVTW1
2682    0324  0586              INC  R6                   MOVE BLANK TO BUFFER
2683          0326      CVTW10 EQU   $                    ENTRY POINT
2684                    *                        SAVE RETURN ADDRESS
2685    0326  C0CB              MOV  R11,R3
2686          0328      CVTW20 EQU   $
2687                    *                        DO 210 I=1,COUNT
2688                    *                        R0=<WKTBL>
2689                    *                                 MOVE BINARY DATA TO R0
2690    0328  C03E              MOV  *R14+,R0
2691                    *                        CALL CVTBHX
2692    032A  C053              MOV  *R3,R1
2693                    *                                 CONVERT DATA TO HEX AND MOVE
2694                    *                                 INTO BUFFER
2695    032C  0691              BL   *R1
2696                    *210                     CONTINUE
2697    032E  0609              DEC  R9
2698    0330  16F8              JNE  CVTW20
2699    0332  0201              LI   R1,WRITA*256
      0334  0800
2700                    *                        CALL WRITE (BFR)
2701    0336  05C3              INCT R3
2702    0338  C033              MOV  *R3+,R0
2703                    *                                 WRITE BUFFER
2704    033A  0690              BL   *R0
2705    033C  1601              JNE  ERTYP
```

```
2706                    *                                   NO ERROR RETURN
2707  033E  05C3                INCT R3
2708        0340  ERTYP  EQU   $
2709  0340  0453                B     *R3
```

```
2712                    * TITLE:     CVTHXB
2713                    *            CONVERT HEX ASCII TO BINARY
2714                    * REVISION:  03/15/76
2715                    *            ORIGINAL
2716                    * COMPUTER:  990,ASSEMBLY
2717                    * ABSTRACT:
2718                    *            MOVE FOUR BYTES OF BUFFER TO CONVERSION BLOCK
2719                    *            AND USE SUPERVISOR CALL TO DO CONVERSION.
2720                    *            R0 CONTAINS RESULT OF CONVERSION.
2721                    * CALLING SEQUENCE:
2722                    *            ENTRY:
2723                    *            R11 CONTAINS ADDRESS OF DATA PASSED BY CALLING
2724                    *            ROUTINE
2725                    *            R6 CONTAINS PRESENT BUFFER ADDRESS
2726                    *            R4,R5,R7,R8 NON-VOLATILE TO THIS ROUTINE
2727                    *            EXIT:
2728                    *            R6 CONTAINS INCREMENTED BUFFER ADDRESS
2729                    *            R0 CONTAINS BINARY DATA
2730           0342     CVTHXB EQU   $
2731                    *                        <CPRB0P>=0
2732                    *                                 GET PRB FOR CONVERSION
2733    0342   C28B             MOV   *R11+,R10
2734    0344   C0CA             MOV   R10,R3
2735    0346   C079             MOV   *R11+,R1
2736                    *                                 OPCODE FOR HEX TO BINARY
2737                    *                                 CONVERSION
2738    0348   CCD1             MOV   *R1,*R3+
2739                    *                        DO 200 I=1,4
2740    034A   0200             LI    R0,4
        034C   0004
2741                    *                        CONPRB(I)=BUFFER(I)
2742           034E     CHB1   EQU   $
2743                    *                                 MOVE CHARACTERS FROM I/O
2744                    *                                 BUFFER TO CONVERSION BUFFER
2745    034E   DCF6             MOVB  *R6+,*R3+
2746                    *200                     CONTINUE
2747    0350   0600             DEC   R0
2748    0352   16FD             JNE   CHB1
2749                    *                        R0=BINARY(CONPRB)
2750    0354   C079             MOV   *R11+,R1
2751                    *                                 DO CONVERSION
2752    0356   0411             BLWP  *R1
2753                    *                        RETURN
2754    0358   045B             B     *R11
```

```
2757                    * TITLE     :BLNK
2758                    *            BLANK FILL BUFFER
2759                    * REVISION: 03/15/76
2760                    *            ORIGINAL
2761                    * COMPUTER: 990,ASSEMBLY
2762                    * ABSTRACT :
2763                    *            FILL THE BUFFER WITH BLANKS
2764                    * CALLING SEQUENCE:
2765                    *            ENTRY:
2766                    *            R6 CONTAINS BEGINNING BUFFER ADDRESS
2767                    *            R1 CONTAINS ADDRESS OF LAST BUFFER BYTE TO BE
2768                    *            FILLED
2769                    *            R4,R5,R7,R8 NON-VOLATILE TO THIS ROUTINE
2770                    *            EXIT:
2771                    *            R6 CONTAINS BEGINNING BUFFER ADDRESS
2772                    *                 BUFFER IS BLANK FILLED
2773                    *
2774          035A      BLNK   EQU  $
2775                    *
2776  035A    0200            LI   R0,'  '            INITIALIZATION VALUE
      035C    2020
2777                    *                            SAVE BEGINNING BUFFER ADDRESS
2778  035E    C386            MOV  R6,R14
2779          0360      BLNK1  EQU  $
2780  0360    CD80            MOV  R0,*R6+            STORE INIT VALUE IN BFR
2781  0362    8046            C    R6,R1              CHECK FOR END OF BUFFER AND
2782  0364    11FD            JLT  BLNK1                REPEAT IF NOT
2783  0366    13FC            JEQ  BLNK1
2784  0368    C18E            MOV  R14,R6
2785  036A    045B            B    *R11
2786                          END
0000 ERS
```

960 - 980 CONCORDANCE

```
BUSY    0310    2406  2418
BWFL6A  0953    0951
BWFL01  0894    0892
BWFL03  0902    1006
BWFL05  0920    0916
BWFL06  0923    0918
BWFL07  0965    0994
BWFL08  0996    0991
BWFL09  1009    1015
BWORD   0875    0720
CBM1    1617    1622
CH1     2637    2644
CH3     2649    2641
CHARCT  0257    1650
CHB1    2742    2748
CHECK   2628    1189  1214  1431  1456
CMDLEN  0331    0465
CMN2    0372    0466
CMND1   0325    0331
CMPR    0176    0371  0607
CNAME   0324    0459
CONPRB  0225    1490  1606
CONTIN  2422    2419
CPRBOP  0227
CS      1886    0329
CT      0379    0656  0662  0674  0679
CTR     1680    0654  0672
CTR1    2024    2022
CTRT    2014    1682
CVTBHX  1602    1264  1299  1327
CVTHXB  2730    1489
CVTW1   2674    2681
CVTW10  2683    1263
CVTW20  2686    2698
CVTWRT  2673    1298  1326
DC      0154    0969
DEFTS1  0312    0344  0359
DEFTS2  0314    0354
DEFTS3  0316    0369
DIN1    0289    0943
DIRECT  0179
DISP1   1053    1039  1046
DISPLY  1031    0804  0822
DISPM   0185
DISPR   0182    0726
DMPSPR          0095  1047
DMT     0108    0118
DOUT1   0291    0744  0972
DRT     0131
DSEXIT  1100    1042  1049  1064  1070  1076  1083  1088  1094  1098
ECA     0172    0684
EIGHT   0272    0664  1648
EMA     0165    0667
EOFMSK  0262    1274  1505
EQSIGN          0089  1085
ERJMP2  2471    2468
ERR     0559    0541  0550  0553  0556  0721  0743  1348
ERRE    1993    1990
ERRER   1347    1232  1266  1276  1301  1329  1334  1413  1506
ERRET   1012    0968
ERROR           0081  0560
ERTYP   2708    2705
```

R0	0208	0499	0517	0527	0541	0580	0581	0582	0583	0585
		0593	0594	0595	0596	0598	0607	0608	0609	0610
		0611	0616	0617	0696	0698	0699	0882	0889	0946
		0950	0952	0976	0977	0984	0986	0987	1000	1001
		1181	1206	1274	1291	1292	1319	1320	1425	1444
		1446	1494	1505	1566	1994	2052	2054	2061	2081
		2082	2094	2098	2100	2101	2103	2133	2134	2143
		2155	2191	2192	2203	2207	2210	2226	2232	2271
		2289	2291	2292	2310	2311	2318	2320	2426	2430
		2431	2432	2433	2436	2440	2459	2459	2461	2502
		2507	2513	2520	2562	2568	2592	2593	2643	2690
		2702	2704	2740	2747	2776	2780			
R1	0209	0493	0495	0514	0533	0537	0694	0777	0786	0812
		0812	0915	0915	0926	0928	0934	0940	0949	0952
		0973	0974	0985	0987	1045	1045	1065	1066	1077
		1078	1089	1090	1163	1171	1185	1209	1228	1278
		1279	1312	1313	1332	1411	1421	1427	1450	1557
		1559	1580	1581	1582	1659	1754	1766	1769	1924
		1926	1927	1933	1936	1959	1963	2055	2057	2077
		2078	2083	2085	2095	2097	2104	2105	2134	2135
		2137	2142	2143	2154	2155	2167	2169	2192	2193
		2195	2239	2241	2243	2264	2266	2270	2271	2276
		2279	2282	2284	2290	2291	2311	2326	2329	2360
		2364	2431	2434	2434	2442	2442	2455	2498	2499
		2508	2510	2522	2524	2533	2535	2542	2548	2551
		2554	2557	2560	2568	2571	2574	2581	2582	2595
		2597	2640	2692	2695	2699	2735	2738	2750	2752
		2781								
R10	0218	0434	0439	0442	0450	0533	0549	0552	0555	0558
		0591	0610	0617	0635	0645	0689	0689	0726	0815
		0815	1014	1014	1047	1058	1066	1072	1078	1085
		1090	1096	1137	1139	1143	1153	1155	1159	1336
		1338	1360	1376	1381	1385	1564	1580	1606	1607
		1661	1664	1989	1992	2152	2159	2160	2164	2200
		2207	2215	2216	2220	2279	2280	2302	2308	2467
		2470	2473	2473	2496	2499	2562	2576	2604	2605
		2605	2733	2734						
R11	0219	0432	0479	0485	0487	0547	1007	1010	1033	1034
		1099	1102	1367	1550	1627	1657	1936	1939	1947
		1952	2360	2361	2370	2428	2429	2465	2474	2630
		2647	2650	2685	2733	2735	2750	2754	2785	
R12	0220	0473	0491	0504	0650	0654	0660	0672	0677	0706
		0713	0737	0753	0771	0925	1126	1189	1214	1263
		1288	1298	1315	1326	1417	1431	1456	1489	1562
		1933	2510							
R13	0221	0906	0984	1014	1057	1099	2018	2019	2094	2095
		2473	2605							
R14	0222	0475	0508	0608	0616	0728	0728	0765	0765	0778
		0778	1247	1254	1294	1322	1476	1483	1494	1607
		1610	1619	1657	1667	2016	2017	2053	2058	2081
		2083	2098	2101	2104	2107	2111	2111	2133	2142
		2154	2167	2191	2202	2209	2221	2264	2270	2276
		2282	2290	2310	2318	2324	2326	2331	2337	2337
		2393	2411	2415	2440	2495	2498	2508	2512	2522
		2530	2548	2581	2587	2595	2606	2606	2690	2778
		2784								
R15	0223	0477	0514	0611	0763	0763	0796	0796	1244	1473
		1923	1929	1944	1970	1970	1983	1984	2429	2430
		2433	2633	2634						
R2	0210	0432	0561	0853	0906	0906	1135	1151	1340	1367
		1528	1728	1772	1805	1868	1894	1987	2067	2069
		2075	2082	2085	2091					

```
RSTRG2  0407    0826
RSZ     0160    1249  1324  1478
RTEMP   0121    0160  1247  1322  1476
RTM     1695    0753
RTOM    0189    1697
SAVIMG  0304    0614  0843  1952
SEVEN   0274    1653
START   0072
SV1     1168    1235  1268
SV11    1262    1251
SV13    1270    1175
SV19    1303    1287
SV25    1330    1310
SV3     1200    1182  1193
SV5     1225    1210  1218
SV6     1231    1165  1167
SV7     1243    1199  1224
SV9     1252    1245
SVCALT          0092  1492  1613  1663
SVSTIM  1122    0850
SW      1790    0365
SWBLK   0374    0366
SWT     0100    0375  0692  0709  0715  0741  0755  0774  1525
SWTR    0123    0376  1525
TS      1916    0370
TSFLD1  1928    1971
TSFLD2  1957    1934
TSFLD3  1961    1940  1948  1953
TSFLD4  1972    1985
TSTERR  1955    1945
WEOF    0241    1332
WKSPBF  1573    1681  1684  1687  1690  1693  1696
WRITA   0242    1228  2699
WRT1    1652    1166  1172  1412  1422
WRT2    1555    1651
WRTRO   1545    1164  1230  1265  1300  1328  1333
WWIO    0142    0356  0681  0698  0718  0738  0881
YES     0301    1840  1939

THERE ARE    0292   SYMBOLS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																
SHEET																															
REV STATUS OF SHEETS	REV																															
	SHEET																															

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DWN	DATE
	CHK	
	ENGR	
	QA	
	APVD	
	CONTR NO	
	DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, DMBNPF, BNPF DUMP-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945397-9901

SCALE	REV.	SHEET 1 of 23

TI-0923-C

```
0003                    IDT  'DMBNPF'
0004                * TITLE: DMBNPF
0005                *        DMBNPF FORMAT AND DESCRIPTION
0006                * REVISION: 03/15/76
0007                *        ORIGINAL
0008                * ABSTRACT:
0009                *        DUMP FROM 0 TO 8192 BYTES OF MEMORY TO CASSETTE
0010                *        TAPE IN BNPF FORMAT, COMPARE MEMORY TO TAPE, OR
0011                *        LOAD DATA IN BINARY FORMAT INTO MEMORY.
0012                * COMPUTER: 990,ASSEMBLY
0013                *** EXTERNAL REFERENCES
0014                *
0015                        DEF  DMBNPF
0016                        REF  ERROR
0017                        REF  RR
0018                        REF  ACL
0019                        REF  PRCRLF
0020                        REF  PRNTC
0021                        REF  PRNTHN
0022                        REF  EQSIGN
0023                        REF  PRNTHX
0024                        REF  SVCALT
0025                *
0026  0000    44            TEXT 'DB'
0027  0002  0090'           DATA DMBNPF
0028  0004  0000            DATA 0
0029                *
0030                *** ERROR MESSAGE EQUATES
0031                *
0032        0205  MS05   EQU  >0205          REQUIRED PARM MISSING
0033        0100  MP00   EQU  >0100          ILLEGAL TOGGLE
0034        1103  DP03   EQU  >1103          DUMP GT 8192 BYTES
0035        1113  DP13   EQU  >1113          INVALID RANGE
0036        0001  MX01   EQU  >0001          UNRECOVERABLE I/O ERROR
0037                *
0038                *** SVC/PRB EQUATES
0039                *
0040        000B  WRITA  EQU  11             WRITE ASCII
0041        0009  READA  EQU  9              READ ASCII
0042        0000  OPEN   EQU  00             OPEN FILE
0043        000D  WEOF   EQU  13             WRITE END-OF-FILE
0044                *
0045                *** PRB.DEFINITIONS
0046                *
0047        0006' PRB    EQU  $
0048  0006  0000  PRBSC  DATA 0              SERVICE TYPE
0049  0008    00  PRBOP  BYTE 0              I/O COMMAND
0050  0009    07  PRBLU  BYTE 7
0051  000A    00  PRBSF  BYTE 0              SYSTEM COMPLETION FLAGS
0052  000B    00  PRBUF  BYTE 0              USER FLAGS
0053  000C  003A' PRBBF  DATA BFR            OUTPUT BUFFER ADDRESS
0054  000E  0056  PRBLN  DATA 86             OUTPUT BUFFER LENGTH
0055  0010  0050  PRBCC  DATA 80             CHARACTER COUNT
0056  0012  0000         DATA 0,0            UNUSED
```

```
       0014  0000
0057          0016' CONPRB EQU  $
0058   0016   0000  CPRBOP DATA 0                    CONVERT COMMAND
0059   0018     00         BYTE 0                    SIGN
0060   0019     00         BYTE 0,0                  VALUE TO BE CONVERTED
       001A     00
0061   001B     00         BYTE 0,0
       001C     00
0062   001D     00         BYTE 0
0063          001E' ENDPRB EQU  $
0064                *
0065                *** MISCELLANEOUS DATA
0066                *
0067          4000  IOMSK  EQU  >4000
0068          001E' INIT   EQU  $
0069   001E     09         BYTE 9
0070   001F     42         TEXT 'BEG ADDR='
0071          0028' LAST   EQU  $
0072   0028     09         BYTE 9
0073   0029     45         TEXT 'END ADDR='
0074          0032' DCB    EQU  $
0075   0032   0B00         DATA >0B00
0076          0034' CHARCT EQU  $
0077   0034   0050         DATA >0050                CHARACTER COUNT FOR WRITE
0078          014D  MPRT   EQU  >100+'M'             MEMORY WORD TYPE
0079          0154  TPRT   EQU  >100+'T'             TAPE WORD TYPE
0080          0020  BLANK  EQU  ' '                  BLANK=' '
0081   0036   4246  ABF    DATA 'BF'                 BNPF DATA BEGIN/FINISH MARKER
0082   0038   4E50  ANP    DATA 'NP'                 BNPF NEG/POS DATA BIT INDCATR
0083   003A   0000  BFR    DATA 0,0,0,0,0,0,0,0,0,0   EIGHTY=
       003C   0000
       003E   0000
       0040   0000
       0042   0000
       0044   0000
       0046   0000
       0048   0000
       004A   0000
       004C   0000
0084   004E   0000         DATA 0,0,0,0,0,0,0,0,0,0      SIX
       0050   0000
       0052   0000
       0054   0000
       0056   0000
       0058   0000
       005A   0000
       005C   0000
       005E   0000
       0060   0000
0085   0062   0000         DATA 0,0,0,0,0,0,0,0,0,0        BYTES
       0064   0000
       0066   0000
       0068   0000
       006A   0000
       006C   0000
```

```
                006E  0000
                0070  0000
                0072  0000
                0074  0000
0086            0076  0000          DATA 0,0,0,0,0,0,0,0,0,0          OF
                0078  0000
                007A  0000
                007C  0000
                007E  0000
                0080  0000
                0082  0000
                0084  0000
                0086  0000
                0088  0000
0087            008A  0000          DATA 0,0,0                        ZEROES
                008C  0000
                008E  0000
0088                        *** REGISTER DEFINITIONS ***
```

```
0090                    *       ALL  REG. REFERENCES IN BNPF DUMP ARE VIA SYMBOLIC
0091                    *       ACRONYMS INDICATIVE OF THE CURRENT USE OF THE REGIS-
0092                    *       TER.  THESE ACRONYMS ARE DEFINED BELOW.  ANY TIME
0093                    *       A REGISTER FREES UP, OR IS RE-ASSIGNED, ONE OF THE
0094                    *       FOLLOWING COMMENT CARDS WILL APPEAR IN THE LISTING
0095                    *
0096                    *** REGISTER REASSIGNMENT: OLD WPX='PTR1', NEW WPX='FREE'
0097                    *** REGISTER REASSIGNMENT: OLD WPX='PTR1', NEW WPX='PTR2'
0098                    *
0099          0000  R0      EQU  0                 SCRATCH
0100          0001  R1      EQU  1                 SCRATCH
0101          0002  CPRM    EQU  2                 PTR TO COMMAND PARM LIST
0102          0003  HDR     EQU  3                 HEADER FROM CMD. PARM. LIST
0103          0004  ACT     EQU  4                 DUMP OR COMPARE TOGGLE
0104          0005  STRT    EQU  5                 DUMP OR CMPR START ADDRESS
0105          0005  MEM     EQU  5                 DUMP OR CMPR STRT+CURRENT IND
0106          0006  END     EQU  6                 DUMP OR CMPR END ADDRESS
0107          0007  R7      EQU  7                 SCRATCH
0108          0008  BFRA    EQU  8                 OUT BFR BASE + CURR INDEX
0109          0009  BFRE    EQU  9                 PTR TO MX DATA ADDR IN BUFFER
0110          000A  PRBA    EQU  10                ADDRESS OF PRB
0111          000A  R10     EQU  10                SCRATCH
0112          000B  R11     EQU  11                RETURN ADDRESS
0113          000D  R13     EQU  13                PTR TO PREVIOUS WORKSPACE
0114          000E  R14     EQU  14                SCRATCH
0115          000F  R15     EQU  15                SCRATCH
```

```
0117                    *1= BEGIN DMBNPF
0118          0090'     DMBNPF EQU  $
0119                           DEF  DMBNPF
0120  0090    C08A             MOV  R10,CPRM          SAVE PARM ADDR,===GET AND
0121  0092    0420             BLWP @ACL                LINK A NEW WORKSPACE
      0094    0000
0122  0096    04C0             CLR  R0
0123  0098    C0F2             MOV  *CPRM+,HDR          HEADER WORD
0124  009A    D032             MOVB *CPRM+,R0         GET CHARACTER COUNT
0125  009C    D132             MOVB *CPRM+,ACT        DUMP/COMPARE TOGGLE
0126  009E    0984             SRL  ACT,8
0127  00A0    0990             SRL  R0,9
0128  00A2    A080             A    R0,CPRM           SKIP OTHER CHARACTERS
0129  00A4    C172             MOV  *CPRM+,STRT         DUMP STARTING ADDR
0130  00A6    C1B2             MOV  *CPRM+,END          DUMP END ADDRESS
0131                    *** PARAMETER VALIDATION
0132                    *1   IF(STRTB,AND,ENDB,EQ,0)CALL ABORT(MS05)
0133  00A8    0A93             SLA  HDR,9             SHIFT TOGGLE PRESENT BIT INTO
0134  00AA    1707             JNC  DMB01               CARRY
0135  00AC    0284             CI   ACT,'D'           IF NO DUMP IGNORE ADDRESS
      00AE    0044
0136  00B0    1614             JNE  DMB01B            PARAMETERS
0137  00B2    0A13             SLA  HDR,1             TOGGLE OK, CHECK FOR START
0138  00B4    1702             JNC  DMB01             JUMP IF OK
0139  00B6    0A13             SLA  HDR,1             START OK, CHECK FOR END
0140  00B8    1805             JOC  DMB01A              JUMP IF OK
0141          00BA'     DMB01  EQU  $
0142  00BA    020A             LI   R10,MS05          ABORT=REQUIRED PARM MISSING
      00BC    0205
0143          00BE'     ABORT  EQU  $
0144  00BE    06A0             BL   @ERROR
      00C0    0000
0145  00C2    1054             JMP  DMB03B
0146                    *
0147          00C4'     DMB01A EQU  $
0148                    *1   X=END-START
0149  00C4    C005             MOV  END,R0
0150  00C6    020A             LI   R10,DP13          JUMP IF START IS GREATER
      00C8    1113
0151  00CA    6005             S    STRT,R0
0152                    *1   IF(X,LE,0)CALL ABORT(MP06)
0153  00CC    12F8             JLE  ABORT               THAN END ADDRESS
0154                    *1   IF(X,GT,8192)CALL ABORT(MP05)
0155  00CE    020A             LI   R10,DP03          JUMP IF LENGTH GREATER THAN
      00D0    1103
0156  00D2    0280             CI   R0,8192             8192 BYTES
      00D4    2000
0157  00D6    18F3             JH   ABORT
0158  00D8    1008             JMP  DMB02
0159          00DA'     DMB01B EQU  $
0160  00DA    020A             LI   R10,MP00
      00DC    0100
0161  00DE    0284             CI   ACT,'C'           IF TOGGLE NOT 'C'
      00E0    0043
```

```
0162  00E2  1303          JEQ   DMB02
0163  00E4  0284          CI    ACT,'L'             AND TOGGLE NOT 'L'
      00E6  004C
0164  00E8  16EA          JNE   ABORT               ABORT
0165                *
0166                ** PARAMETER VALIDATION COMPLETE
0167                * **                          OLD R2=CPRM, NEW R2=FREE
0168                * **                          OLD R3=HDR   NEW R3=FREE
0169                *
0170          0002  R2      EQU   2
0171          0003  R3      EQU   3
0172                *
0173                *1= CALL OPEN(PRBA)
0174                *       (IN LINE EXPANSION)
0175                *
0176          00EA' DMB02   EQU   $
0177  00EA  0201          LI    R1,OPEN*256+WRITA OPEN LUNO 7
      00EC  000B
0178  00EE  C820          MOV   @CHARCT,@PRBCC    SET CHAR. CT. FOR WRITE
      00F0  0034'
      00F2  0010'
0179  00F4  0284          CI    ACT,'D'           IF TOGGLE NOT SET TO DUMP
      00F6  0044
0180  00F8  1302          JEQ   DMB02A            PREPARE TO READ
0181  00FA  0201          LI    R1,OPEN*256+READA OPEN LUNO 7
      00FC  0009
0182          00FE' DMB02A  EQU   $
0183  00FE  D801          MOVB  R1,@PRBOP
      0100  0008'
0184  0102  020A          LI    PRBA,PRB
      0104  0006'
0185  0106  0420          BLWP  @SVCALT
      0108  0000
0186  010A  06C1          SWPB  R1                SET THE PRB TO FUNCTION
0187  010C  D801          MOVB  R1,@PRBOP           CODE TO WRITE/READ ASCII
      010E  0008'
0188  0110  0209          LI    BFRE,BFR+80-13    SET UP END OF I/O BUFFER
      0112  007D'
0189  0114  020F          LI    R15,INIT
      0116  001E'
0190  0118  04C7          CLR   R7
0191  011A  0284          CI    ACT,'L'           IF TOGGLE SET TO LOAD
      011C  004C
0192  011E  1327          JEQ   DMB04             JUMP TO LOAD SEQUENCE
0193  0120  0284          CI    ACT,'C'           IF TOGGLE SET TO COMPARE
      0122  0043
0194  0124  134E          JEQ   DMB07             BRANCH TO COMPARE SEQUENCE
0195                *1= MEM=STRT
0196                * **REGISTER REASSIGNMENT:OLD R5=STRT, NW R5=MEM
0197                *1= DO WHILE(MEM,LT,END)
0198                *2  CALL BLNK(BFRA,BFRE)
0199          0126' DMB03   EQU   $
0200  0126  0208          LI    BFRA,BFR          SET UP START OF OUTPUT BUFFER
      0128  003A'
0201  012A  0209          LI    BFRE,BFR+80       SET BUFFER END FOR BLANK FILL
```

```
      012C  008A'
0202  012E  06A0          BL   @BLNK2              BLANK-FILL THE BUFFER
      0130  0360'
0203                      *
0204                      *1- CALL FILL(BFRA,E,MEM,END)
0205  0132  0209          LI   BFRE,BFR+80-13      SET UP END OF I/O BUFFER
      0134  007D'
0206  0136  06A0          BL   @FILL                 FILL BFR W/BNPF DATA
      0138  024C'
0207                      *2  CALL WRTRD(PRB)
0208  013A  06A0          BL   @WRTRD
      013C  0370'
0209  013E  16BF          JNE  ABORT
0210                      *1  CONTINUE
0211  0140  8185          C    MEM,END
0212  0142  12F1          JLE  DMB03
0213  0144  0208          LI   BFRA,BFR
      0146  003A'
0214                      *1  CALL BLNK(BFRA,BFRE)
0215  0148  0209          LI   BFRE,BFR+80         SET BUFFER END FOR BLANK FILL
      014A  008A'
0216  014C  06A0          BL   @BLNK2              BLANK-FILL THE BUFFER
      014E  0360'
0217                      *1  BFR(0)='$'
0218  0150  0200          LI   R0,'$'*256
      0152  2400
0219  0154  D800          MOVB R0,@BFR
      0156  003A'
0220                      *1- CALL WRTRD(PRBA)
0221  0158  06A0          BL   @WRTRD                WRITE EOF RECORD(BNPF)
      015A  0370'
0222  015C  16B0          JNE  ABORT               ABORT IF WRITE FAILED
0223  015E  0200          LI   R0,WEOF*256         WRITE AN END-OF-FILE
      0160  0D00
0224  0162  D800          MOVB R0,@PRBOP             SET OPCODE
      0164  0008'
0225                      *1 CALL WRTRD(WEOF)
0226  0166  06A0          BL   @WRTRD
      0168  0370'
0227        016A' DMB03A  EQU  $
0228  016A  16A9          JNE  ABORT                 ABORT IF WEOF FAILED
0229        016C' DMB03B  EQU  $
0230  016C  1068          JMP  DMBTRM
0231        016E' DMB04   EQU  $
0232                      * **REGISTER REASSIGNMENT: OLD R4=ACT, NEW R4=FREE
0233        0004  R4      EQU  4
0234  016E  0208          LI   BFRA,BFR            SET UP STRT OF OUTPUT BUFFER
      0170  003A'
0235  0172  0202          LI   R2,CONPRB
      0174  0016'
0236  0176  0203          LI   R3,ENDPRB
      0178  001E'
0237  017A  06A0          BL   @BLNK               BLANK-FILL BUFFER
      017C  0356'
0238                      *1-CALL WRTRD(PRB)
```

```
0239  017E  06A0          BL    @WRTRD            READ A RECORD
      0180  0370'
0240  0182  165C          JNE   ERTYPE            IF ERROR JUMP TO ABORT OR END
0241        0184' DMB05   EQU   $
0242  0184  C018          MOV   *BFRA,R0          IF FIRST CHARACTER OF BUFFER
0243  0186  0280          CI    R0,'$ '           IS '$' READ NEXT RECORD
      0188  2420
0244  018A  13F1          JEQ   DMB04             FOR END OF FILE
0245  018C  06A0          BL    @CVTDCB           CONVERT ASCII DEC ADDR TO BIN
      018E  02A8'
0246  0190  05C8          INCT  BFRA              SKIP FIRST BLANK AND 'B'
0247  0192  028F          CI    R15,INIT          IS RECORD FIR0T
      0194  001E'
0248  0196  1604          JNE   DMB06             DISPLAY ADDRESS IF SO
0249  0198  06A0          BL    @PRNRTN           DISPLAY BEGINNING FILE ADDR
      019A  02FA'
0250  019C  1053          JMP   DMBTRM            ESC RETURN
0251  019E  04CF          CLR   R15
0252        01A0' DMB06   EQU   $
0253  01A0  06A0          BL    @GETBTE           GET BYTE FROM BUFFER
      01A2  02DC'
0254  01A4  DD44          MOVB  R4,*MEM+          STORE BYTE IN MEMORY
0255  01A6  8248          C     BFRA,BFRE         CHECK END OF BFR
0256  01A8  14E2          JHE   DMB04             AND READ NEW RECORD IF SO
0257  01AA  05C8          INCT  BFRA              SKIP BLANK AND 'B'
0258  01AC  0200          LI    R0,'  '           R0 CONTAINS BLANKS
      01AE  2020
0259  01B0  9018          CB    *BFRA,R0          IF NOT LAST BYTE IN FILE
0260  01B2  16F6          JNE   DMB06             JUMP BACK TO GET ANOTHER BYTE
0261  01B4  0605          DEC   MEM               GET LAST BYTE ADDRESS
0262  01B6  020F          LI    R15,LAST
      01B8  0028'
0263  01BA  06A0          BL    @PRNRTN           DISPLAY LAST FILE ADDR
      01BC  02FA'
0264  01BE  1042          JMP   DMBTRM            ESC RETURN
0265  01C0  10D6          JMP   DMB04             CONYINUE READING TO E=O=F
0266        01C2' DMB07   EQU   $
0267  01C2  0208          LI    BFRA,BFR          SET UP START OF I/O BUFFER
      01C4  003A'
0268  01C6  0202          LI    R2,CONPRB
      01C8  0016'
0269  01CA  0203          LI    R3,ENDPRB
      01CC  001E'
0270  01CE  06A0          BL    @BLNK             BLANK=FILL BUFFER
      01D0  0356'
0271  01D2  06A0          BL    @WRTRD            READ A RECORD
      01D4  0370'
0272  01D6  1632          JNE   ERTYPE            IF ERROR JUMP TO CHECK TYPE
0273  01D8  C018          MOV   *BFRA,R0          IF FIRST CHARACTER OF BUFFER
0274  01DA  0280          CI    R0,'$ '           IS '$' READ NEXT RECORD
      01DC  2420
0275  01DE  13F1          JEQ   DMB07             FOR END OF FILE
0276  01E0  06A0          BL    @CVTDCB           CONVERT ASCII DEC ADDR TO BIN
      01E2  02A8'
0277  01E4  05C8          INCT  BFRA              SKIP FIRST BLANK AND 'B'
```

```
0278  01E6  028F          CI    R15,INIT          IS RECORD FIRST
      01E8  001E'
0279  01EA  1604          JNE   DMB08             DISPLAY ADDRESS IF SO
0280  01EC  06A0          BL    @PRNRTN           DISPLAY BEGINNING FILE ADDR
      01EE  02FA'
0281  01F0  1029          JMP   DMBTRM            ESC RETURN
0282  01F2  04CF          CLR   R15
0283        01F4' DMB08   EQU   $
0284  01F4  0287          CI    R7,4
      01F6  0004
0285  01F8  1601          JNE   DMB08A
0286  01FA  04C7          CLR   R7
0287        01FC' DMB08A  EQU   $
0288  01FC  06A0          BL    @GETBTE           GET BYTE FROM BUFFER
      01FE  02DC'
0289  0200  9544          CB    R4,*MEM           COMPARE BUFFER AND MEMORY
0290  0202  1309          JEQ   DMB09             IF NOT EQUAL
0291  0204  06A0          BL    @DISPLY           DISPLAY BYTE FROM TAPE
      0206  031A'
0292  0208  0154          DATA  TPRT
0293  020A  101C          JMP   DMBTRM            ESC RETURN
0294  020C  D115          MOVB  *MEM,R4           GET MEMORY BYTE
0295  020E  06A0          BL    @DISPLY           DISPLAY BYTE FROM MEMORY
      0210  031A'
0296  0212  014D          DATA  MPRT
0297  0214  1017          JMP   DMBTRM            ESC RETURN
0298        0216' DMB09   EQU   $
0299  0216  0585          INC   MEM               GET NEXT MEMORY BYTE ADDRESS
0300  0218  8248          C     BFRA,BFRE         CHECK END OF BFR
0301  021A  14D3          JHE   DMB07             AND READ NEW RECORD IF SO
0302  021C  05C8          INCT  BFRA              SKIP BLANK AND 'B'
0303  021E  0200          LI    R0,'  '           R0 CONTAINS BLANKS
      0220  2020
0304  0222  0287          CI    R7,4
      0224  0004
0305  0226  1601          JNE   DMB09A
0306  0228  04C7          CLR   R7
0307        022A' DMB09A  EQU   $
0308  022A  9018          CB    *BFRA,R0          IF NOT LAST BYTE IN FILE
0309  022C  16E3          JNE   DMB08             JUMP BACK TO GET ANOTHER BYTE
0310        022E' DMB010  EQU   $
0311  022E  0605          DEC   MEM               GET LAST BYTE ADDRESS
0312  0230  020F          LI    R15,LAST
      0232  0028'
0313  0234  06A0          BL    @PRNRTN           DISPLAY LAST FILE ADDR
      0236  02FA'
0314  0238  1005          JMP   DMBTRM            ESC RETURN
0315  023A  10C3          JMP   DMB07             CONYINUE READING TO E-O-F
0316        023C' ERTYPE  EQU   $
0317  023C  0240          ANDI  R0,IOMSK          IF END-OF-FILE ENCOUNTERED
      023E  4000
0318  0240  1301          JEQ   DMBTRM            LOAD COMPLETE
0319  0242  1093          JMP   DMB03A            OTHERWISE ABORT
0320              *1  END DMBNPF
0321        0244' DMBTRM  EQU   $
```

```
0322  0244  0420        BLWP @RR
      0246  0000
0323  0248  C282        MOV  CPRM,R10
0324  024A  045B        RT
```

```
0327                *1= SUBROUTINE FILL(BFRA,BFRE,MEM,END)
0328                * TITLE    :FILL
0329                *           FILL OUTPUT BUFFER
0330                *
0331                * REVISION: 03/15/76
0332                *           ORIGINAL
0333                * ABSTRACT :
0334                *           GENERATE A'BNPF FORMATTED DATA RECORD
0335                *            CC1=N   : DECIMAL ASCII DATA ADDRESS=-1 TO 5
0336                *                      CHARACTERS W/NO LEADING BLANKS OR '0'
0337                *             CCN+1  : ' '  FIELD SEPARATOR
0338                *            CCN+2   :  B   BEGINNING OF DATA BITS
0339                *            CCN+3=
0340                *             TO N+10: 'P' OR 'N', REPRESENTING A HIGH OR
0341                *                      LOW STATE OF A BIT OF MEMORY DATA.
0342                *            CCN+11  : 'F'  FINISH OF DATA BITS
0343                *            CCN+12
0344                *             TO N+66: REPITITION OF CCN+1 TO CCN+11 UP TO
0345                *                      SIX TIMES
0346                * COMPUTER: 990,ASSEMBLY
0347                * CALLING SEQUENCE:
0348                *           ENTRY:
0349                *             R5=MEM =ADDRESS OF MEMORY DATA
0350                *             R8=BFRA=ADDRESS OF OUTPUT BUFFER
0351                *             R9=BFRE=ADDR. OF END OF OUTPUT BUFFER
0352                *             R6=END =ENDING ADDR. OF MEMORY DATA
0353                *
0354                *           EXIT:
0355                *             R0=>R4
0356                *             MEM =ADDRESS OF NEXT BYTE TO BE CONVERTED
0357                *             BFRA=ADDR+1 OF LAST BYTE IN OUTPUT BUFFER
0358                *             THE BNPF RECORD IS FORMATTED IN 'BFR'.
0359         024C'  FILL   EQU  $              ****ENTRY POINT****
0360                *
0361                *1  CALL BINDEC(MEM,BFRA)
0362                *       IN LINE EXPANSION
0363   024C  04C0          CLR  R0
0364   024E  C045          MOV  MEM,R1             INIT: VALUE TO BE CONVERTED
0365   0250  0204          LI   R4,10                    CONVERSION BASE
      0252  000A
0366   0254  0203          LI   R3,10000                 MAX # OF DGTS IN CNVTD
      0256  2710
0367         0258' FLL1    EQU  $                  HERE TO NXT LBL ELIMINATES
0368                *                            LEADING ZEROES
0369   0258  3C03          DIV  R3,R0                FIND
0370   025A  04C2          CLR  R2                     FIRST
0371   025C  8080          C    R0,R2                    NON=ZERO
0372   025E  1603          JNE  FLL2                       DIGIT(JUMP WHEN FOUND)
0373   0260  3C84          DIV  R4,R2                NOT FOUND, REDUCE MAX # OF
0374   0262  A0C2          A    R2,R3                  DIGITS BY 1 AND TRY AGAIN
0375   0264  16F9          JNE  FLL1                   IF ORIG. # WAS NOT ZERO
0376         0266' FLL2    EQU  $
0377   0266  0220          AI   R0,>30             CONVERT THE DIGIT TO DECIMAL
      0268  0030
```

```
0378  026A  06C0          SWPB  R0          ASCII AND
0379  026C  DE00          MOVB  R0,*BFRA+   STORE IN THE OUTPUT BFR.
0380  026E  04C2          CLR   R2          REDUCE NUMBER OF
0381  0270  3C84          DIV   R4,R2       DIGITS REMAINING BY 1, AND
0382  0272  A0C2          A     R2,R3       WHEN # OF DIGITS REMAINING IS
0383  0274  C082          MOV   R2,R2
0384  0276  1303          JEQ   FLL3        ZERO, JUMP OUT OF LOOP
0385  0278  04C0          CLR   R0          ELSE, COMPUTE
0386  027A  3C03          DIV   R3,R0       NEXT DIGIT,
0387  027C  10F4          JMP   FLL2        GO CONVERT TO ASCII.
0388        027E' FLL3    EQU   $
0389  027E  0588          INC   BFRA        PUTS A BLANK IN 1ST CH.
0390  0280  0201          LI    R1,ABF      ADDR. OF 'BF' IN R1
      0282  0036'
0391  0284  0200          LI    R0,8        BIT COUNTER IN R0
      0286  0008
0392  0288  DE31          MOVB  *R1+,*BFRA+ 'B' FOR BEGIN FIELD
0393  028A  D0F5          MOVB  *MEM+,R3    MEMORY BYTE IN R3
0394        028C' FLL4    EQU   $
0395  028C  0202          LI    R2,ANP      ADDR. OF 'NP' IN R2
      028E  0038'
0396  0290  0A13          SLA   R3,1        BIT TO BE STORED IN ARRY STAT
0397  0292  1701          JNC   FLL5        JUMP FOR NEGATIVE BIT
0398  0294  0582          INC   R2            INC TO 'P' FOR POS. BIT
0399  0296  DE12  FLL5    MOVB  *R2,*BFRA+  STORE 'N' OR 'P'
0400  0298  0600          DEC   R0          DECREMENT BIT COUNT, AND
0401  029A  16F8          JNE   FLL4          LOOP BACK IF BYTE NOT DONE
0402  029C  DE11          MOVB  *R1,*BFRA+    'F' FOR FINISH FIELD
0403  029E  8248          C     BFRA,BFRE   BYTE DONE, CHECK END OF BFR,
0404  02A0  1402          JHE   FLL6          AND GET OUT IF SO
0405  02A2  8185          C     MEM,END     CHECK FOR END-OF-DUMP AND
0406  02A4  12EC          JLE   FLL3          LOOP BACK IF NOT FOR NXT BY
0407        02A6' FLL6    EQU   $
0408  02A6  045B          RT
```

```
0411                    * TITLE:    CVTDCB
0412                    *           CONVERT ASCII DECIMAL TO BINARY
0413                    * REVISION: 03/15/76
0414                    *           ORIGINAL
0415                    * ABSTRACT:
0416                    *           CONVERT ASCII DECIMAL ADDRESS TO BINARY ADDRESS
0417                    * COMPUTER: 990,ASSEMBLY
0418                    * CALLING SEQUENCE:
0419                    *           ENTRY:
0420                    *            R8=BFRA=BEGINNING BUFFER ADDRESS
0421                    *            R5=MEM =MEMORY ADDRESS
0422                    *           EXIT:
0423                    *            R0,R1,R10,R14 CLOBBERED
0424          02A8' CVTDCB EQU  $               ****ENTRY POINT****
0425  02A8    C38B         MOV  R11,R14          SAVE RETURN ADDRESS
0426  02AA    020A         LI   R10,CONPRB       PREPARE TO DO CONVERSION
      02AC    0016'
0427  02AE    C04A         MOV  R10,R1
0428  02B0    0200         LI   R0,'  '          R0 CONTAINS BLANK
      02B2    2020
0429  02B4    CC60         MOV  @DCB,*R1+        SET FNCT CODE TO DEC TO BIN
      02B6    0032'
0430  02B8    0221         AI   R1,5             GET END OF CONPRB
      02BA    0005
0431          02BC' CVT1   EQU  $
0432  02BC    0588         INC  BFRA
0433  02BE    9018         CB   *BFRA,R0         CHECK FOR END OF ADDRESS
0434  02C0    16FD         JNE  CVT1             IF NOT JUMP BACK FOR NXT BYTE
0435  02C2    C088         MOV  BFRA,R2          SAVE END OF ADDRESS
0436  02C4    0608         DEC  BFRA             GET LAST BYTE
0437          02C6' CVT2   EQU  $
0438  02C6    D458         MOVB *BFRA,*R1        MOVE ADDR BYTE TO CONPRB BFR
0439  02C8    0608         DEC  BFRA
0440  02CA    0601         DEC  R1
0441  02CC    0288         CI   BFRA,BFR         IF BEGINNING BYTE NOT
      02CE    003A'
0442  02D0    14FA         JHE  CVT2             TRANSFERRED, JUMP BACK
0443  02D2    0420         BLWP @SVCALT          DO CONVERSION
      02D4    0108'
0444  02D6    C140         MOV  R0,MEM           SAVE MEMORY ADDRESS
0445  02D8    C202         MOV  R2,BFRA
0446  02DA    045E         B    *R14
```

```
0449                    * TITLE:    GETBTF
0450                    *           GET BYTE
0451                    * REVISION: 03/15/76
0452                    *           ORIGINAL
0453                    * ABSTRACT:
0454                    *           GET BYTE FROM BUFFER AND CONVERT TO BINARY
0455                    * COMPUTER: 990,ASSEMBLY
0456                    * CALLING SEQUENCE:
0457                    *           ENTRY:
0458                    *           R8=BFRA=PRESENT BUFFER ADDRESS
0459                    *           EXIT:
0460                    *           R4=BINARY BYTE VALUE
0461                    *           R1,R2,R4 CLOBBERED
0462          02DC'     GETBTE EQU  $               ****ENTRY POINT****
0463  02DC    0201              LI   R1,'F '            R1 CONTAINS 'F '
      02DE    4620
0464          02E0'     GTBT1  EQU  $
0465  02E0    0704              SETO R4                 BUFFER BYTE VALUE TO BE IN R4
0466  02E2    0202              LI   R2,ANP             R2 CONTAINS ADDRESS OF 'NP'
      02E4    0038'
0467          02E6'     GTBT2  EQU  $
0468  02E6    9488              CB   *BFRA+,*R2         IF BUFFER CONTAINS 'P',
0469  02E8    1602              JNE  GTBT3              JUMP TO STORE A 1
0470  02EA    0A14              SLA  R4,1               SHIFT 0 INTO RIGHTMOST BIT R4
0471  02EC    1001              JMP  GTBT4
0472          02EE'     GTBT3  EQU  $
0473  02EE    0BF4              SRC  R4,15              SHIFT 1 INTO RIGHTMOST BIT R4
0474          02F0'     GTBT4  EQU  $
0475  02F0    9058              CB   *BFRA,R1           IF NOT END OF BYTE,
0476  02F2    16F9              JNE  GTBT2              CONTINUE CONVERSION
0477  02F4    0588              INC  BFRA               SKIP 'F'
0478  02F6    0A84              SLA  R4,8               GET BYTE VALUE
0479  02F8    0459              RT
```

```
0482                    * TITLE:    PRNRTN
0483                    *           PRINT ROUTINE
0484                    * REVISION: 03/15/76
0485                    *           ORIGINAL
0486                    * ABSTRACT:
0487                    *           PRINT CARRIAGE RETURN,LINE FEED, ADDRESS DESCR,
0488                    *           AND VALUE
0489                    * COMPUTER: 990,ASSEMBLY
0490                    * CALLING SEQUENCE:
0491                    *           ENTRY
0492                    *           R1=ADDRESS OF DESCRIPTION MESSAGE
0493                    *           R5=MEM=MEMORY ADDRESS
0494                    *           EXIT:
0495                    *           R0,R10 CLOBBERED
0496          02FA' PRNRTN EQU  $                ****ENTRY POINT****
0497   02FA   C08B         MOV  R11,R2             SAVE RETURN ADDRESS
0498   02FC   C389         MOV  BFRE,R14           SAVE END OF BUFFER
0499   02FE   06A0         BL   @PRCRLF
       0300   0000
0500   0302   100A         JMP  PRNEXT             ESC RETURN
0501   0304   C28F         MOV  R15,R10
0502   0306   06A0         BL   @PRNTC             PRINT ADDRESS DESCR
       0308   0000
0503   030A   1006         JMP  PRNEXT             ESC RETURN
0504   030C   C285         MOV  MEM,R10
0505   030E   06A0         BL   @PRNTHX            PRINT ADDRESS VALUE
       0310   0000
0506   0312   1002         JMP  PRNEXT             ESC RETURN
0507   0314   C24E         MOV  R14,BFRE           RESTORE END OF BUFFER
0508   0316   05C2         INCT R2                 NORMAL RETURN
0509          0318' PRNEXT EQU  $
0510   0318   0452         B    *R2
```

```
0513                    * TITLE:    DISPLY
0514                    *           DISPLAY STRING
0515                    * REVISION: 03/15/76
0516                    *           ORIGINAL
0517                    * ABSTRACT:
0518                    *           DISPLAY STRING AS BYTE ADDRESS=STRING
0519                    * COMPUTER: 990,ASSEMBLY
0520                    * CALLING SEQUENCE:
0521                    *           ENTRY:
0522                    *             MEM CONTAINS ADDRESS
0523                    *             R14 CONTAINS STRING
0524                    *           EXIT:
0525                    *              R1,R10,R0 DESTROYED
0526                    *             BFRE SAVED
0527          031A'     DISPLY EQU  $              ****ENTRY POINT****
0528                    *  SAVE RETURN ADDRESS AND BFRE
0529    031A  C08B            MOV  R11,R2
0530    031C  C389            MOV  BFRE,R14
0531    031E  C1C7            MOV  R7,R7
0532    0320  1603            JNE  DISP1
0533                    *  CARRIAGE RETURN, LINE FEED
0534    0322  06A0            BL   @PRCRLF
        0324  0300'
0535    0326  1015            JMP  DSEXIT             ESC RETURN
0536          0328'     DISP1  EQU  $
0537    0328  0587            INC  R7
0538    032A  C282            MOV  R2,R10
0539    032C  05C2            INCT R2
0540                    *  CALL PRINT(TYPE)
0541    032E  06A0            BL   @PRNTC
        0330  0308'
0542    0332  100F            JMP  DSEXIT             ESC RETURN
0543    0334  C285            MOV  MEM,R10
0544                    *  CALL PRINT(ADDR)
0545    0336  06A0            BL   @PRNTHN
        0338  0000
0546    033A  1009            JMP  DSEXIT             ESC RETURN
0547    033C  020A            LI   R10,EQSIGN
        033E  0000
0548                    *  CALL PRINT(=)
0549    0340  06A0            BL   @PRNTC
        0342  0330'
0550    0344  1006            JMP  DSEXIT             ESC RETURN
0551    0346  04CA            CLR  R10
0552    0348  D284            MOVB R4,R10
0553                    *  CALL PRINT(STRING)
0554    034A  06A0            BL   @PRNTHX
        034C  0310'
0555    034E  1001            JMP  DSEXIT             ESC RETURN
0556    0350  05C2            INCT R2                 NORMAL RETURN TO CALLER
0557          0352'     DSEXIT EQU  $
0558                    *  RESTORE BFRE
0559    0352  C24E            MOV  R14,BFRE
0560    0354  0452            B    *R2
```

```
0563                    * TITLE     :BLNK
0564                    *            BLANK FILL BUFFER
0565                    * REVISION: 03/15/76
0566                    *            ORIGINAL
0567                    * ABSTRACT :
0568                    *            SIMPLE ROUTNE TO BLANK FILL THE BUFFER
0569                    * COMPUTER: 990,ASSEMBLY
0570                    * CALLING SEQUENCE:
0571                    *            ENTRY:
0572                    *                 R8=BFRA=ADDRESS OF BFR
0573                    *                 R9=BFRE=ADDRESS OF END OF BUFFER
0574                    *            EXIT :
0575                    *                 R0 CLOBBERED
0576                    *                 ALL OTHER REGS INTACT
0577                    *                 BUFFER IS BLANK FILLED
0578                    *
0579          0356'     BLNK      EQU   $              ****ENTRY POINT****
0580                    *
0581    0356  0200                LI    R0,'  '            INITIALIZATION VALUE
        0358  2020
0582          035A'     BLNK1     EQU   $
0583    035A  CC80                MOV   R0,*R2+            STORE INIT VALUE IN CONBFR
0584    035C  80C2                C     R2,R3              CHECK FOR END OF BUFFER AND
0585    035E  1AFD                JL    BLNK1              REPEAT IF NOT
0586          0360'     BLNK2     EQU   $
0587    0360  0200                LI    R0,'  '            INITIALIZATION VALUE
        0362  2020
0588          0364'     BLNK3     EQU   $
0589    0364  CE00                MOV   R0,*BFRA+          STORE INIT VALUE IN BFR
0590    0366  8248                C     BFRA,BFRE          CHECK FOR END OF BUFFER AND
0591    0368  12FD                JLE   BLNK3              REPEAT IF NOT
0592    036A  0208                LI    BFRA,BFR           RESET BFRA TO START OF BUFFER
        036C  003A'
0593    036E  045B                RT                       RETURN TO CALLER
```

```
 0596                    * TITLE     :WRTRD
 0597                    *            WRITE/READ
 0598                    * REVISION: 03/15/76
 0599                    *            ORIGINAL
 0600                    * ABSTRACT:
 0601                    *            FLUSH THE BUFFER AS DESCRIBED BY THE PRB
 0602                    *            SET AU SATAUS .EQ. FOR GOOD COMPLETION
 0603                    *                           .NE. FOR I/O ABORT
 0604                    *            R10 WILL CONTAIN THEAPPROPRIATE ERROR MESSAGE
 0605                    *            ID IN CASE OF AN ABORT.
 0606                    * COMPUTER: 990,ASSEMBLY
 0607                    *CALLING SEQUENCE:
 0608                    *            ENTRY:
 0609                    *                 R10=PRBA=ADDRESS OF PRB
 0610                    *            EXIT:
 0611                    *                 R0,R10 CLOBBERED
 0612                    *                 ALL OTHER REGS INTACT
 0613          0370'     WRTRD  EQU  $              ****ENTRY POINT****
 0614   0370   020A             LI   PRBA,PRB
        0372   0006'
 0615   0374   0420             BLWP @SVCALT
        0376   02D4'
 0616   0378   020A             LI   R10,MX01           GET I/O ABRT MSG(JUST IN CASE
        037A   0001
 0617   037C   D020             MOVB @PRBSF,R0          SET AU STATUS WITH SYSTEM FLG
        037E   000A'
 0618   0380   045B             RT                      RETURN TO CALLER
 0619                           END
0000 ERS
```

960 - 980 CONCORDANCE

$		0047	0057	0063	0068	0071	0074	0076	0118	0141
		0143	0147	0159	0176	0182	0199	0227	0229	0231
		0241	0252	0266	0283	0287	0298	0307	0310	0316
		0321	0359	0367	0376	0388	0394	0407	0424	0431
		0437	0462	0464	0467	0472	0474	0496	0509	0527
		0536	0557	0579	0582	0586	0588	0613		
ABF	0081	0390								
ABORT	0143	0153	0157	0164	0209	0222	0228			
ACL		0018	0121							
ACT	0103	0125	0126	0135	0161	0163	0179	0191	0193	
ANP	0082	0395	0466							
BFR	0083	0053	0188	0200	0201	0205	0213	0215	0219	0234
		0267	0441	0592						
BFRA	0108	0200	0213	0234	0242	0246	0255	0257	0259	0267
		0273	0277	0300	0302	0308	0379	0389	0392	0399
		0402	0403	0432	0433	0435	0436	0438	0439	0441
		0445	0468	0475	0477	0589	0590	0592		
BFRE	0109	0188	0201	0205	0215	0255	0300	0403	0498	0507
		0530	0559	0590						
BLANK	0080									
BLNK	0579	0237	0270							
BLNK1	0582	0585								
BLNK2	0586	0202	0216							
BLNK3	0588	0591								
CHARCT	0076	0178								
CONPRB	0057	0235	0268	0426						
CPRBOP	0058									
CPRM	0101	0120	0123	0124	0125	0128	0129	0130	0323	
CVT1	0431	0434								
CVT2	0437	0442								
CVTDCB	0424	0245	0276							
DCB	0074	0429								
DISP1	0536	0532								
DISPLY	0527	0291	0295							
DMB01	0141	0134	0138							
DMB010	0310									
DMB01A	0147	0140								
DMB01B	0159	0136								
DMB02	0176	0158	0162							
DMB02A	0182	0180								
DMB03	0199	0212								
DMB03A	0227	0319								
DMB03B	0229	0145								
DMB04	0231	0192	0244	0256	0265					
DMB05	0241									
DMB06	0252	0248	0260							
DMB07	0266	0194	0275	0301	0315					
DMB08	0283	0279	0309							
DMB08A	0287	0285								
DMB09	0298	0290								
DMB09A	0307	0305								
DMBNPF	0118	0015	0027	0119						
DMBTRM	0321	0230	0250	0264	0281	0293	0297	0314	0318	
DP03	0034	0155								
DP13	0035	0150								
DSEXIT	0557	0535	0542	0546	0550	0555				
END	0106	0130	0149	0211	0405					
ENDPRB	0063	0236	0269							
EQSIGN		0022	0547							
ERROR		0016	0144							

```
READA   0041        0181
RR                  0017   0322
STRT    0104        0129   0151
SVCALT              0024   0185   0443   0615
TPRT    0079        0292
WEOF    0043        0223
WRITA   0040        0177
WRTRD   0613        0208   0221   0226   0239   0271

THERE ARE   0108  SYMBOLS
```

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																
SHEET																															
REV STATUS OF SHEETS	REV																															
	SHEET																															

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE

CHK

ENGR

QA

APVD

CONTR NO

DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

AL, PROMLD, LOAD PROM PROGRAMMER-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945398-9901

SCALE | REV. | SHEET 1 of 4

TI-0023-C

```
0003                        IDT  'PROMLD'
0004                 * TITLE:   PROMLD
0005                 *          PROM LOAD
0006                 * REVISION:
0007                 *          ORIGINAL
0008                 * COMPUTER: 990,ASM
0009                 * ABSTRACT: THIS PROGRAM WILL LOAD THE PROM PROGRAMMER INTO
0010                 *          MEMORY. ONE PART IS LOADED IN THE OVERLAY AREA
0011                 *          BY A CALL TO THE OVERLAY DRIVER. THE SECOND PAR
0012                 *          IS LOADED INTO THE USER AREA BY CALLING THE LOA
0013                 *          DRIVER AND SIMULATING AN OVERLAY CALL.
0014                 * CALLING SEQUENCE:
0015                 *          CALLED FROM COMMAND STRING PROCESSOR
0016                 *            ENTRY - R10 POINTS TO COMMAND PARM LIST
0017                 *          DESTROYS R1,R8,R9,R15,R14
0018                 *
0019                        DEF  PL
0020                        REF  INIT,OVLY,LOADOV,PGBIAS
0021                        REF  PROGSZ
0022                 *
0023        0001     R1     EQU  1
0024        0008     R8     EQU  8
0025        0009     R9     EQU  9
0026        000A     R10    EQU  10
0027        000B     R11    EQU  11
0028        000F     R15    EQU  15
0029                 *
0030                 *          -1-PROCEDURE PROM LOAD(CPL POINTER)
0031        0000'    PL     EQU  $
0032  0000  C3CB           MOV  R11,R15          SAVE RETURN ADDRESS
0033                 *          -2-CALL OVLY(CPL POINTER)
0034  0002  06A0           BL   @OVLY
      0004  0000
0035                 *          -2-DFBIAS = LOC(INIT) - PROG SIZE
0036  0006  0201           LI   R1,INIT
      0008  0000
0037  000A  C220           MOV  @PROGSZ,R8
      000C  0000
0038  000E  6048           S    R8,R1
0039                 *          -2-POINTER = CPL POINTER
0040  0010  C20A           MOV  R10,R8
0041  0012  C278           MOV  *R8+,R9
0042  0014  0A99           SLA  R9,9
0043                 *          -2-IF PARM(1) .NE. NULL THEN
0044  0016  1701           JNC  PL1
0045                 *          -3-POINTER = POINTER + 2
0046  0018  05C8           INCT R8
0047        001A'    PL1    EQU  $
0048  001A  0A19           SLA  R9,1
0049                 *          -2-IF PARM(2) .NE. NULL THEN
0050  001C  1701           JNC  PL2
0051                 *          -3- BIAS = LOC(POINTER)
0052  001E  C058           MOV  *R8,R1
0053        0020'    PL2    EQU  $
```

```
 0054  0020  C801          MOV  R1,@PGBIAS          STORE PROG BIAS IN OVERLAY
       0022  0000
 0055                *           -2-CALL LOADOV(CPL POINTER,BIAS)
 0056  0024  06A0          BL   @LOADOV
       0026  0000
 0057                *           -1-END PROM LOAD
 0058  0028  045F          B    *R15
 0059                      END
0000 ERS
```

960 - 980 CONCORDANCE

$		0031	0047	0053			
INIT		0020	0036				
LOADOV		0020	0056				
OVLY		0020	0034				
PGBIAS		0020	0054				
PL	0031	0019					
PL1	0047	0044					
PL2	0053	0050					
PROGSZ		0021	0037				
R1	0023	0036	0038	0052	0054		
R10	0026	0040					
R11	0027	0032					
R15	0028	0032	0058				
R8	0024	0037	0038	0040	0041	0046	0052
R9	0025	0041	0042	0048			

THERE ARE 0015 SYMBOLS

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV | SHEET

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, UPFLD, UP FRONT LAODER-PX990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	945392-9901

SCALE | REV. | SHEET 1 of 16

TI-0023-C

```
0003                *   PROCEDURE LOADER:
0004                *      THIS LOADER MAY BE USED TO PERFORM BOOT
0005                *      STRAP LOAD.
0006                *      CALLING PROGRAMS MUST SPECIFY THE ADDRESS
0007                *      OF A <GETBIT> ROUTINE TO PERFORM INPUT FOR
0008                *      THE LOAD PROCESS.
0009                *   ***
0010                *      STAND-ALONE OPERATING PROCEDURE
0011                *
0012                *      MOUNT AND POSITION THE CASSETTE TO THE FILE
0013                *      TO BE LOADED, AND PLACE THE DRIVE IN PLAY-
0014                *      BACK MODE.  PRESS THE LOAD BUTTON ON THE
0015                *      FRONT PANEL.  IF NO ENTRY POINT IS ENCOUNTERD
0016                *      DURING THE LOAD, CONTROL WILL BE RETURNED TO
0017                *      THE FRONT PANEL.  ELSE, CONTROL IS PASSED TO
0018                *      THE LOADED PROGRAM AT THE SPECIFIED EP.
0019                *
0020                *   ***
0021                *
0022                *      GETBIT  EQU  $       GETBIT ENTRY POINT
0023                *                           (SEE <GETBIT> FOR PROG.
0024                *                            DESCRIPTION)
0025                *   PROCEDURE:ROMBUT
0026                *   CALL ABSLDR(>A0)
0027                *   IF(ERROR)CALL SPIN(S)
0028                *   IF(PROG EP.NE.0)
0029                *      THEN
0030                *         GO TO C(EP)
0031                *      ELSE
0032                *         GO TO FRONT PANEL
0033                *   ENDIF
0034                *   END ROMBOOT
0035                *   SUBROUTINE ABSLDR
0036                *   GETBIT = A(GETBIT)
0037                *   DO UNTIL END OF MODULE
0038                *   REPEAT CNT=0
0039                *   CKSM=OLD CKSM
0040                *      CALL GETBIT(VALUE,3,CLR)
0041                *      DO CASE
0042                *         C1:DO VALUE.EQ.XTNDED TAG
0043                *            CALL GETBIT(VALUE,4,CLR)
0044                *            DO CASE
0045                *               C1:DO VALUE.EQ.PGM NAME
0046                *                  CALL GETBIT(VALUE,7,CLR)
0047                *                  NMBR CHAR=AND(VALUE,>3F)
0048                *                     END DO;
0049                *                     DO WHILE NMBR CHAR.GE.0
0050                *                        CALL GETBIT(VALUE,7,CLR)
0051                *                        NMBR CHAR=NMBR CHAR-1
0052                *                     END DO;
0053                *               END C1:
0054                *               C2:DO VALUE.EQ.ETRY POINT
0055                *                  CALL GET19(VALUE)
0056                *                  EP=VALUE
```

```
0057                *                  IF(PARM ADDR.NE.0)
0058                *                    C(PARM ADDR)=EP
0059                *                END C2
0060                *                C3:DO VALUE.EQ.CHECKSUM
0061                *                   TEMP=CKSM
0062                *                   CALL GET19,VALUE
0063                *                   IF(NOT(TEMP.EQ.VALUE),AND.
0064                *                      NOT(SHC(TEMP,8).EQ.VALUE)
0065                *                      ERROR RETURN
0066                *                   END IF
0067                *                END C3
0068                *                C4:DO VALUE.EQ.END OF MODULE
0069                *                   RETURN
0070                *                END C4
0071                *                C5:DO VALUE.EQ.REPEAT TRIPLET
0072                *                   CALL GETBIT(VALUE,7,CLR)
0073                *                   REPEAT COUNT=AND(VALUE,>3F)
0074                *                   CALL GET16(VALUE)
0075                *                   DO WHILE REPEAT COUNT.NE.0
0076                *                      C(CURR ADDR)=VALUE
0077                *                      CURR ADDR=CURR ADD+2
0078                *                      REPEAT COUNT=REPEAT COUNT-1
0079                *                   END DO
0080                *                END C5
0081                *                      ***NOTE: IN THE INTEREST OF SPACE,
0082                *                      ***       THE CODE FOR THE ABOVE
0083                *                      ***       IS INCLUDED IN THE
0084                *                      ***       PROCESSING FOR ABSOLUTE
0085                *                      ***       DATA TRIPLETS.
0086                *             END CASE
0087                *          C2:DO VALUE.EQ.ABS WORD
0088                *             CALL GET16(VALUE)
0089                *             DO WHILE REPEAT COUNT.GT.0
0090                *                C(CURR ADDR)=VALUE
0091                *                CURR ADDR=CUR ADDR+2
0092                *                REPEAT COUNT=REPEAT COUNT-1
0093                *             END DO
0094                *          END C2
0095                *          C3:DO VALUE.EQ.ABS BYTE
0096                *             CALL GETBIT(VALUE,5,CLR)
0097                *             VALUE=SRL(VALUE,1)
0098                *             CALL GETBIT(VALUE,6,OR)
0099                *             C(CURR ADDR)=SHL(VALUE,2)
0100                *             CUR ADDR=CURR ADDR+1
0101                *          END C3
0102                *          C4:DO VALUE.EQ.LOAD ADDR
0103                *             CALL GET16(VALUE)
0104                *             IF(PARM ADDR.NE.0)
0105                *             CUR.ADDR=VALUE
0106                *               C(PARM ADDR+LDPT)=VALUE
0107                *          END C4
0108                *          C5:DO VALUE.LT.>4
0109                *             ERROR RETURN
0110                *    END ABSLDR
0111                *    SUBROUTINE GET16
```

```
0112                *   END GET16
0113                *   SUBROUTINE GETBIT
0114                *   ***START INPUT STREAM***
0115                *   CRUBASE=0
0116                *    BIT.COUNT=0
0117                *   CKSM=0
0118                *   INBITS=0
0119                *   INB=0
0120                *   RE ENTRY=A(GETB2)
0121                *     ***NEWRECORD***
0122                *      DO WHILE CHAR.EQ.LF
0123                *         CHAR=IN733
0124                *         END DO
0125                *   IF(CHAR.EQ.DEL)CHAR=IN733
0126                *   *****GETCHAR*****
0127                *   CHAR = IN733
0128                *   IF CHAR.EQ.'CR'  NEWRECORD
0129                *   CKSM=CKSM.XOR.CHAR
0130                *   CBS=CNC(CBS,CHAR,CBC)
0131                *   CBC=CBC+7
0132                *   ****GETBT2*******
0133                *   IF(MODE.EQ.CLR)RBS =0
0134                *   IF(RBC.GT.CBC)GETCHAR
0135                *   DO WHILE RBC.GT.0
0136                *      CALL SLA(RBS,1)
0137                *      CALL SLA(CBS,1)
0138                *      IF(CARRY)RBS=RBS+1
0139                *      CBC=CBC-1
0140                *      RBC=RBC-1
0141                *   END DO
0142                *   RETURN
0143                       IDT  'UPFLD'
0144                *                              0=PROCEDURE LOADER:
0145                *                              2=THIS LOADER MAY BE USED TO P
0146                *                              2=STRAP LOAD.
0147                *                              2=CALLING PROGRAMS MUST SPECIF
0148                *                              2=OF A <GETBIT> ROUTINE TO PER
0149                *                              2=THE LOAD PROCESS.
0150                *                              1=***
0151                *                              2=STAND-ALONE OPERATING PROCED
0152                *                              1=
0153                *                              2=MOUNT AND POSITION THE CASSE
0154                *                              2=TO BE LOADED, AND PLACE THE
0155                *                              2=BACK MODE.  PRESS THE LOAD B
0156                *                              2=FRONT PANEL.  IF NO ENTRY PO
0157                *                              2=DURING THE LOAD, CONTROL WIL
0158                *                              2=THE FRONT PANEL.  ELSE, CONT
0159                *                              2=THE LOADED PROGRAM AT THE SP
0160                *                              2=
0161                *                              1=***
0162                *                              2=
0163                *                              2=GETBIT  EQU  $      GETBIT E
0164                *                              2=                    (SEE <GE
0165                *                              2=                     DESCRIP
0166                *          *** GETBIT PARAMETERS ***
```

```
0167  0000  CLR     EQU  0
0168  FFFF  OR      EQU  -1
0169        *
0170        *            ***733 ASR CONTROL & DATA EQUATES ***
0171        *
0172  0009  DTR     EQU  9
0173  000A  RTS     EQU  >A
0174  000B  CLRWRQ  EQU  >B
0175  000C  CLRRRQ  EQU  >C                 CRU OUT READ REQUEST
0176  000C  RRQ     EQU  >C                 CRU IN  READ REQUEST
0177  1100  DC1     EQU  >1100              READER "ON" COMMAND
0178  7F00  DEL     EQU  >7F00              <DELETE>
0179  0A00  LF      EQU  >0A00              <LINE FEED>
0180  0D00  CR      EQU  >0D00              <CARRIAGE RETURN>
0181        *
0182        *
0183        *               ***REGISTER ALLOCATION
0184        *
```

```
0186  0000          WSP     BSS   32                  WORKSPACE FOR BOOTLOADER
0187                *
0188          0000  R0      EQU   0                   TEMP(SHIFT COUNT & INDEXING)
0189          0001  R1      EQU   1                   ADDRESS OF«GETBIT»
0190          0002  R2      EQU   2                   GETBIT=BIT STRING ON HAND
0191          0003  R3      EQU   3                   GETBIT=# OF BITS IN R2
0192          0004  R4      EQU   4                   GET16 TEMP(RTRN ADDR SAVE)
0193          0005  R5      EQU   5                   CURRENT LOAD ADDRESS
0194          0006  R6      EQU   6                   REPEAT COUNT:NMBR CHAR
0195          0007  R7      EQU   7                   «LIMIT»
0196          0008  R8      EQU   8                   TEMP
0197          0009  R9      EQU   9                   ENTRY POINT OF LOADED PROGRAM
0198          000A  R10     EQU   10                  «VALUE»
0199          000B  R11     EQU   11                  AUTO SUBR, RETURN LINKAGE
0200          000C  R12     EQU   12                  CRUBASE
0201          000D  R13     EQU   13                  TEMP
0202          000E  R14     EQU   14                  «CKSM»
0203          000F  R15     EQU   15                  ABSLDR RETURN ADDRESS
0204                *
0205                *               ***ABSOLUTE DATA FORMAT DEFINITIONS
0206                *
0207          0002  RLA     EQU   2                   RELOC,ADDR(UNSUPPORTED)
0208          0003  RDW     EQU   3                   RELOC,WORD(UNSUPPORTED)
0209          0004  ABWD    EQU   4                   ABSOLUTE DATA WORD(16BITS)
0210          0005  ABYT    EQU   5                   ABSOLUTE DATA BYTE(8 BITS)
0211          0006  ALDA    EQU   6                   ABOLSUTE LOAD ADDRESS
0212          0007  XTF     EQU   7                   EXTENDED TAG
0213                *
0214          0000  EOM     EQU   0                   END OF MODULE
0215          0001  PGN     EQU   1                   PROGRAM NAME
0216          0002  PGL     EQU   2                   PROGRAM LENGTH(UNSUPPORTED)
0217          0003  AEP     EQU   3                   ABSOLUTE ENTRY POINT
0218          0004  REA     EQU   4                   RELOC, ENTRY ADDR(UNSUPPORTED
0219          0005  CKSM    EQU   5                   CHECKSUM
0220          0006  RPT     EQU   6                   REPEAT COUNT
```

```
0222                    *                                  0-PROCEDURE:ROMBUT
0223          0020'     ABSBUT  EQU   $
0224    0020  1001              JMP   ABS01
0225    0022  100A              JMP   ABSLDR                MONITOR CONTROL ENTRY POINT
0226          0024'     ABS01   EQU   $
0227    0024  02E0              LWPI  WSP                   WORKSPACE FOR BOOTLOADER
        0026  0000'
0228                    *                                  1-CALL ABSLDR(>A0)
0229    0028  06A0              BL    @ABSLDR
        002A  0038'
0230                    *                                  1-IF(ERROR)CALL SPIN($)
0231    002C  10FF              JMP   $                     RETURNS HERE ON ABORT
0232                    *                                  1-IF(PROG EP,NE,0)
0233    002E  C249              MOV   R9,R9
0234    0030  1301              JEQ   BOOT1
0235                    *                                  2-THEN
0236                    *                                  3-GO TO C(EP)
0237    0032  0459              B     *R9                   EXECUTE PROGRAM
0238                    *                                  2-ELSE
0239                    *                                  3-GO TO FRONT PANEL
0240    0034  0420      BOOT1   BLWP  @8                    NO EP FOUND...TRAP TO PANEL
        0036  0008
0241                    *                                  1-ENDIF
0242                    *                                  0-END ROMBOOT
0243                    *                                  1-SUBROUTINE ABSLDR
0244          0038'     ABSLDR  EQU   $                 ***ENTRY POINT
0245    0038  C3CB              MOV   R11,R15               SAVE RETURN ADDR
0246                    *                                  1-GETBIT = A(GETBIT)
0247    003A  0201              LI    R1,GETBIT             ADDR. OF GETBIT
        003C  0118'
0248    003E  04C9              CLR   R9                    CLEAR ENTRY POINT
0249    0040  04C3              CLR   R3                    CLEAR BIT COUNT
0250    0042  04CE              CLR   R14                   CLEAR CHECKSUM
0251                    *                                  1-DO UNTIL END OF MODULE
0252          0044'     AB01    EQU   $
0253                    *                                  1-REPEAT CNT=0
0254    0044  04C6              CLR   R6
0255                    *                                  1-CKSM=OLD CKSM
0256    0046  C20E              MOV   R14,R8                WILL GET CLOBBERED IF TAG
0257                    *                                    IS NOT CKSM, BUT DOESNT
0258                    *                                     MATTER IN THIS CASE
0259                    *                                  2-CALL GETBIT(VALUE,3,CLR)
0260                    *                                  2-DO CASE
0261                    *                                  3-C1:DO VALUE,EQ,XTNDED TAG
0262    0048  0691              BL    *R1                   GET A TAG IN R10
0263    004A    00              BYTE  CLR,3
        004B    03
0264    004C  028A              CI    R10,XTF               CHECK FOR EXTENDED TAG
        004E  0007
0265    0050  1632              JNE   AB13                    JUMP FNOT
0266                    *                                  4-CALL GETBIT(VALUE,4,CLR)
0267    0052  0691              BL    *R1
0268    0054    00              BYTE  CLR,4
        0055    04
```

```
                     *                              4-DO CASE
                     *                              5-C1:DO VALUE.EQ.PGM NAME
      0056  028A           CI   R10,PGN
      0058  0001
      005A  160A           JNE  AB07
                     *                              6-CALL GETBIT(VALUE,7,CLR)
      005C  0691           BL   *R1
      005E    00           BYTE CLR,7
      005F    07
                     *                              6-NMBR CHAR=AND(VALUE,>3F)
      0060  024A           ANDI R10,>3F
      0062  003F
      0064  C18A           MOV  R10,R6
                     *                              7-END DO;
                     *                              7-DO WHILE NMBR CHAR.GE.0
            0066'  AB04    EQU  $
                     *                              8-CALL GETBIT(VALUE,7,CLR)
      0066  0691           BL   *R1
      0068    00           BYTE CLR,7
      0069    07
                     *                              8-NMBR CHAR=NMBR CHAR-1
            006A'  AB05    EQU  $
      006A  0606           DEC  R6
      006C  1BFC           JH   AB04
                     *                              7-END DO;
                     *                              5-END C1:
      006E  10EA           JMP  AB01
                   AB07
                     *                              5-C2:DO VALUE.EQ.ETRY POINT
      0070  028A           CI   R10,AEP
      0072  0003
      0074  1604           JNE  AB08
                     *                              6-CALL GET19(VALUE)
      0076  06A0           BL   @GET19
      0078  00F6'
                     *                              6-EP=VALUE
      007A  C24A           MOV  R10,R9
                     *                              6-IF(PARM ADDR.NE.0)
                     *                              6-  C(PARM ADDR)=EP
                     *                              5-END C2
      007C  10E3           JMP  AB01
            007E'  AB08    EQU  $
                     *                              5-C3:DO VALUE.EQ.CHECKSUM
      007E  028A           CI   R10,CKSM
      0080  0005
      0082  160A           JNE  AB10
                     *                              6-TEMP=CKSM
      0084  C20E           MOV  R14,R8
                     *                              6-CALL GET19,VALUE
      0086  06A0           BL   @GET19
      0088  00F6'
                     *                            **NOTE:SINCE #OF BYTES SINCE
                     *                              LAST CKSM MAY BE EITHER EVEN
                     *                              OR ODD,(AND WE DON'T KNOW
                     *                              WHICH)..WE'LL SETTLE FOR A
```

```
0316                    *                       MATCH-UP ANY WAY WE CAN MAKE
0317                    *                       IT HAPPEN
0318                    *                     6-IF(NOT(TEMP,EQ,VALUE),AND,
0319                    *                     6-   NOT(SHC(TEMP,8),EQ,VALUE)
0320                    *                     6-   ERROR RETURN
0321  008A  04CE               CLR   R14       CLEAR CHECKSUM FOR NEXT TIME
0322  008C  8288               C     R8,R10
0323  008E  13DA               JEQ   AB01      CKSM OK-PROCESS NEXT DATA
0324  0090  06C8               SWPB  R8
0325  0092  8288               C     R8,R10    CKSM OK-PRCESS NEXT DATA
0326  0094  13D7               JEQ   AB01
0327  0096  1004               JMP   AB11      BAD CHECKSUM..ERROR EXIT
0328                    *                     6-END IF
0329                    *                     5-END C3
0330        0098' AB10   EQU   $
0331                    *                     5-C4:DO VALUE.EQ.END OF MODULE
0332  0098  028A               CI    R10,EOM
      009A  0000
0333  009C  1602               JNE   AB12
0334                    *                     6-RETURN
0335  009E  05CF               INCT  R15       SUCCESSFUL LOAD
0336        00A0' AB11   EQU   $               HERE FOR ERROR
0337  00A0  045F               B     *R15
0338                    *                     5-END C4
0339        00A2' AB12   EQU   $
0340                    *                     5-C5:DO VALUE.EQ.REPEAT TRIPLE
0341  00A2  028A               CI    R10,RPT
      00A4  0006
0342  00A6  16FC               JNE   AB11      UNSUPPORTED EXTENDED TAG
0343                    *                     6-CALL GETBIT(VALUE,7,CLR)
0344  00A8  0691               BL    *R1
0345  00AA    00               BYTE  CLR,7
      00AB    07
0346                    *                     6-REPEAT COUNT=AND(VALUE,>3F)
0347  00AC  C18A               MOV   R10,R6
0348  00AE  0246               ANDI  R6,>3F
      00B0  003F
0349                    *                     6-CALL GET16(VALUE)
0350                    *                     6-DO WHILE REPEAT COUNT.NE.0
0351                    *                     7-C(CURR ADDR)=VALUE
0352                    *                     7-CURR ADDR=CURR ADD+2
0353                    *                     7-REPEAT COUNT=REPEAT COUNT-1
0354                    *                     6-END DO
0355                    *                     5-END C5
0356                    *                     7-***NOTE: IN THE INTEREST OF
0357                    *                     7-***      THE CODE FOR THE AB
0358                    *                     7-***      IS INCLUDED IN THE
0359                    *                     7-***      PROCESSING FOR ABSO
0360                    *                     7-***      DATA TRIPLETS.
0361  00B2  0691               BL    *R1
0362  00B4    00               BYTE  CLR,3
      00B5    03
0363                    *                     4-END CASE
0364        00B6' AB13   EQU   $
0365                    *                     3-C2:DO VALUE.EQ.ABS WORD
```

```
0366  00B6  028A          CI    R10,ABWD
      00B8  0004
0367  00BA  1606          JNE   AB15
0368                *
0369  00BC  06A0          BL    @GET16                  4=CALL GET16(VALUE)
      00BE  0104'
0370                *                                   4=DO WHILE REPEAT COUNT.GT.0
0371        00C0' AB14    EQU   $
0372                *                                   5=C(CURR ADDR)=VALUE
0373                *                                   5=CURR ADDR=CUR ADDR+2
0374  00C0  CD4A          MOV   R10,*R5+
0375                *                                   5=REPEAT COUNT=REPEAT COUNT-1
0376  00C2  0606          DEC   R6
0377  00C4  15FD          JGT   AB14
0378  00C6  10BE          JMP   AB01
0379                *                                   4=END DO
0380                *                                   3=END C2
0381        00C8' AB15    EQU   $
0382                *                                   3=C3:DO VALUE.EQ.ABS BYTE
0383  00C8  028A          CI    R10,ABYT
      00CA  0005
0384  00CC  1608          JNE   AB16
0385                *                                   4=CALL GETBIT(VALUE,5,CLR)
0386  00CE  0691          BL    *R1
0387  00D0    00          BYTE  CLR,5
      00D1    05
0388                *                                   4=VALUE=SRL(VALUE,1)
0389  00D2  091A          SRL   R10,1
0390                *                                   4=CALL GETBIT(VALUE,6,OR)
0391  00D4  0691          BL    *R1
0392  00D6    FF          BYTE  OR,6
      00D7    06
0393                *                                   4=C(CURR ADDR)=SHL(VALUE,2)
0394                *                                   4=CUR ADDR=CURR ADDR+1
0395  00D8  0A6A          SLA   R10,6                    EQIV TO SL 2 + SWPB
0396  00DA  DD4A          MOVB  R10,*R5+
0397                *                                   3=END C3
0398  00DC  10B3          JMP   AB01
0399        00DE' AB16    EQU   $
0400                *                                   3=C4:DO VALUE.EQ.LOAD ADDR
0401  00DE  028A          CI    R10,ALDA
      00E0  0006
0402  00E2  1608          JNE   AB17
0403                *                                   4=CALL GET16(VALUE)
0404  00E4  06A0          BL    @GET16
      00E6  0104'
0405                *                                   4=IF(PARM ADDR.NE.0)
0406                *                                   4=CUR.ADDR=VALUE
0407  00E8  C14A          MOV   R10,R5
0408                *                                   4=  C(PARM ADDR+LDPT)=VALUE
0409  00EA  C21F          MOV   *R15,R8
0410  00EC  13A8          JEQ   AB01
0411  00EE  05C8          INCT  R8
0412  00F0  C60A          MOV   R10,*R8
0413                *                                   3=END C4
```

```
0414  00F2  10A8          JMP   AB01
0415                *
0416                *                          3=C5!DO VALUE.LT.>4
0417        00F4' AB17    EQU   $              4=ERROR RETURN
0418  00F4  10D5          JMP   AB11
0419                *                           IF VALUE.GE.4, WE WOULDNT
0420                *                             BE HERE
                                                1=END ABSLDR
```

```
0423                    *                               1*SUBROUTINE GET16
0424          00F6' GET19   EQU   $
0425    00F6  C10B          MOV   R11,R4
0426    00F8  0691          BL    *R1
0427    00FA    00          BYTE  0,3
        00FB    03
0428    00FC  028A          CI    R10,ABWD
        00FE  0004
0429    0100  1302          JEQ   GET19A
0430    0102  10CE          JMP   AB11
0431          0104' GET16   EQU   $                 ***ENTRY POINT***
0432    0104  C10B          MOV   R11,R4              SAVE RETURN
0433          0106' GET19A  EQU   $
0434    0106  0691          BL    *R1
0435    0108    00          BYTE  0,5
        0109    05
0436    010A  091A          SRL   R10,1               DISCARD TAG FROM NEXT CH.
0437          010C' GET12A  EQU   $
0438    010C  0691          BL    *R1                   & OR IN NEXT 7 BITS
0439    010E    FF          BYTE  -1,7
        010F    07
0440    0110  091A          SRL   R10,1               DISCARD TAG FROM 3RD BYTE
0441    0112  0691          BL    *R1                   & OR IN NEXT 6 BITS
0442    0114    FF          BYTE  -1,6
        0115    06
0443    0116  0454          B     *R4                 RETURN TO CALLER
0444                    *                             1*END GET16
```

```
0447                    *                          1=SUBROUTINE GETBIT
0448          0118'  GETBIT EQU  $                ***ENTRY POINT****
0449                    *
0450                    *                           INITIALIZE GETBIT VARIABLES
0451                    *                          1=****START INPUT STREAM***
0452                    *                          1=CRUBASE=0
0453  0118  020C              LI   R12,>0
      011A  0000
0454                    *                          1= BIT.COUNT=0
0455  011C  1D09              SBO  DTR
0456  011E  1D0A              SBO  RTS
0457  0120  1D0B              SBO  CLRWRQ
0458  0122  1D0C              SBO  CLRRRQ
0459  0124  0202              LI   R2,DC1
      0126  1100
0460  0128  3202              LDCR R2,8
0461                    *                          1=CKSM=0
0462                    *                          1=INBITS=0
0463                    *                          1=INB=0
0464  012A  388E              MPY  R14,R2
0465                    *                          1=RE ENTRY=A(GETB2)
0466  012C  0201              LI   R1,GETBT2
      012E  016A'
0467        0130'  GB01   EQU  $
0468                    *                          1=  ***NEWRECORD***
0469                    *                          2=DO WHILE CHAR.EQ.LF
0470  0130  1E0C              SBZ  CLRRRQ             CLEAR READ REQUEST
0471        0132'  GB02   EQU  $
0472  0132  1F0C              TB   RRQ
0473  0134  16FE              JNE  GB02               WAIT FOR READY
0474                    *                          3=CHAR=IN733
0475  0136  04C0              CLR  R0
0476  0138  35C0              STCR R0,7
0477  013A  0280              CI   R0,LF
      013C  0A00
0478  013E  13F8              JEQ  GB01
0479                    *                          3=END DO
0480                    *                          1=IF(CHAR.EQ.DEL)CHAR=IN733
0481  0140  0280              CI   R0,DEL
      0142  7F00
0482  0144  1605              JNE  GB06
0483                    *                          1=*****GETCHAR*****
0484        0146'  GB04   EQU  $                    HERE TO GET ANOTHER CHARACTER
0485  0146  1E0C              SBZ  CLRRRQ             CLEAR READ REQUEST
0486        0148'  GB05   EQU  $
0487  0148  1F0C              TB   RRQ
0488  014A  16FE              JNE  GB05               WAIT FOR CHARACTER
0489                    *                          1=CHAR = IN733
0490  014C  04C0              CLR  R0                GET THE CHAR.
0491  014E  35C0              STCR R0,7                FROM THE CRU
0492                    *                          1=IF CHAR.EQ.'CR'  NEWRECORD
0493        0150'  GB06   EQU  $                    CHECK FOR END OF RECORD
0494  0150  0280              CI   R0,CR             IF EOR, GO START ANOTHER
      0152  0D00
```

```
0495  0154  13ED          JEQ   GB01                  RECORD
0496                *                                1=CKSM=CKSM.XOR.CHAR
0497  0156  2880          XOR   R0,R14
0498  0158  06CE          SWPB  R14
0499                *                                1=CBS=CNC(CBS,CHAR,CBC)
0500  015A  0A10          SLA   R0,1                  LEFT JUSTIFY THE > DATA BITS
0501  015C  C0C3          MOV   R3,R3                  AVOID 16 BIT SHIFT
0502  015E  1302          JEQ   GB06A
0503  0160  A003          A     R3,R0                  NON.ZERO SH COUNT IN R0
0504  0162  0900          SRL   R0,R0                  SHIFT CHAR BY NO BITS ON HAND
0505        0164' GB06A   EQU   $
0506  0164  E080          SOC   R0,R2                  OR IN NEW BITS
0507                *                                1=CBC=CBC+7
0508  0166  0223          AI    R3,7                   UPDATE CURR BIT COUNT
      0168  0007
0509                *                                1=*****GETBT2*******
0510        016A' GETBT2  EQU   $                   ***SECONDARY ENTRY POINT
0511
0512                *                                1=IF(MODE.EQ.CLR)RBS =0
0513  016A  C01B          MOV   *R11,R0                GET PARAMETERS
0514  016C  D000          MOVB  R0,R0
0515  016E  1601          JNE   GB07                   JUMP IF R10 HAS GOOD DATA
0516  0170  04CA          CLR   R10                       ELSE, CLEAR OUT TRASH
0517
0518        0172' GB07    EQU   $
0519
0520                *                                1=IF(RBC.GT.CBC)GETCHAR
0521  0172  5000          SZCB  R0,R0                  CLEAR THE MODE FLAG
0522  0174  80C0          C     R0,R3                  CMPR RQST TO BITS ON HAND
0523  0176  1BE7          JH    GB04                      JUMP TO GET MORE BITS
0524
0525                *                                1=DO WHILE RBC.GT.0
0526        0178' GB08    EQU   $
0527
0528                *                                2=CALL SLA(RBS,1)
0529  0178  0A1A          SLA   R10,1
0530                *                                2=CALL SLA(CBS,1)
0531  017A  0A12          SLA   R2,1
0532                *                                2=IF(CARRY)RBS=RBS+1
0533  017C  1701          JNC   GB09
0534  017E  058A          INC   R10
0535        0180' GB09    EQU   $
0536                *                                2=CBC=CBC-1
0537  0180  0603          DEC   R3
0538                *                                2=RBC=RBC-1
0539  0182  0600          DEC   R0                     DECR REQUEST COUNT AND
0540                *                                1=END DO
0541  0184  1BF9          JH    GB08                      REPEAT IF NOT DONE
0542                *                                0=RETURN
0543  0186  05CB          INCT  R11                    INCREMENT RETURN ADDRESS
0544  0188  045B          RT
0545                      END   ABSBUT
0000 ERS
```

960 - 980 CONCORDANCE

```
                    0401   0407   0412   0428   0436   0440   0516   0529   0534
R11       0199      0245   0425   0432   0513   0543
R12       0200      0453
R13       0201
R14       0202      0250   0256   0309   0321   0464   0497   0498
R15       0203      0245   0335   0337   0409
R2        0190      0459   0460   0464   0506   0531
R3        0191      0249   0501   0501   0503   0508   0522   0537
R4        0192      0425   0432   0443
R5        0193      0374   0396   0407
R6        0194      0254   0278   0287   0347   0348   0376
R7        0195
R8        0196      0256   0309   0322   0324   0325   0409   0411   0412
R9        0197      0233   0233   0237   0248   0299
RDW       0208
REA       0218
RLA       0207
RPT       0220      0341
RRQ       0176      0472   0487
RTS       0173      0456
WSP       0186      0227
XTF       0212      0264

THERE ARE   0074  SYMBOLS
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV

SHEET

REV STATUS OF SHEETS

REV

SHEET

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN DATE

CHK

ENGR

QA

APVD

CONTR NO

DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

AL, CCLOAD, STANDARD LOADER, PROTOTYPE-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	943344-9903

SCALE REV. A SHEET 1 of 14

TI-0923-C

```
0003                    IDT  'CCLOAD'
0004              *     TITLE=CCLOAD = CARD/CASSETTE LOADER.
0005              *     AUTHOR= J. ALLEN.
0006              *     COMPUTER= 990 / ASSEMBLY LANGUAGE.
0007              *     ABSTRACT=  THIS MODULE CONTIANS TWO ROUTINES: THE
0008              *     *    FRONT PANEL SUPPORT ROUTINE AND THE LOADER
0009              *     *    ROUTINE.
0010              *     STATISTICS=  THE FRONT PANEL ROUTINE CHECKS ITS
0011              *     *    R1 FOR A VALUE OF >9900. IF THIS VALUE IS
0012              *     *    FOUND, RUN IS ENABLED AND A BRANCH IS TAKEN
0013              *     *    TO THE ADDRESS SPECIFIED IN R0. THIS ALLOWS
0014              *     *    AN EXTERNAL PROCESSOR TO GAIN CONTROL OF THE
0015              *     *    RESTART TRAP.  THE LOADER SOFTWARE CONSISTS
0016              *     *    OF TWO CO=ROUTINES: ONE FOR THE LOAD FUNCTION
0017              *     *    AND ONE FOR CHARACTER I/O PROCESSING. AN
0018              *     *    EXTERNAL PROCESS COULD DO CHARACTER I/O
0019              *     *    PROCESSING FROM ANOTHER PERIPHERIAL DEVICE
0020              *     *    AND USE THE ROM=RESIDENT LOADER CO=ROUTINE
0021              *     *    FOR THE LOAD FUNCTION. LOCATION >FFFA CONTAINS
0022              *     *    THE ENTRY TO THE LOADER CO=ROUTINE.
```

```
0024  FE00                    AORG >FE00
0025                  *  TITLE = PANEL - 990 FRONT PANEL PROGRAM
0026                  *  REVISION = 24JUN75
0027                  *                INCORPORATE NEW H/W FEATURES
0028                  *        *     10/01/75 - INCORPORATE INTO CASSETTE LOADER
0029                  *        *     -  STNADARD REGISTER NAMES
0030                  *
0031                  *  COMPUTER = 990,MIRA
0032                  *
0033                  *  ABSTRACT = THIS ROUTINE DRIVES THE FRONT PANEL FOR THE
0034                  *                990 COMPUTER.
0035                  *
0036                  *
0037                  *
0038          0000    R0      EQU  0
0039          0001    R1      EQU  1
0040          0002    R2      EQU  2
0041          0003    R3      EQU  3
0042          0004    R4      EQU  4
0043          0005    R5      EQU  5
0044          0006    R6      EQU  6
0045          0007    R7      EQU  7
0046          0008    R8      EQU  8
0047          0009    R9      EQU  9
0048          000A    R10     EQU  10
0049          000B    R11     EQU  11
0050          000C    R12     EQU  12
0051          000D    R13     EQU  13
0052          000E    R14     EQU  14
0053          000F    R15     EQU  15
0054                  *
0055                  *
0056                  *
0057                  *       PANEL CRU BIT ASSIGNMENTS
0058                  *
0059          1FE0    CRUOFF  EQU  >1FE0
0060          0008    SCAN    EQU  8
0061          0008    TYPE    EQU  8
0062          0009    SWPB    EQU  9
0063          0009    CLEAR   EQU  9
0064          000A    RUNLGT  EQU  10
0065          000A    TIMER   EQU  10
0066          000B    FPBIT   EQU  11
0067          000D    STIME   EQU  13
0068          000E    MANTPL  EQU  14                  MAINTANCE PANEL PRESENT-
0069          000E    SIEFLG  EQU  14
0070                  *
0071          FE00    FPSTRT  EQU  $
0072                  *
0073                  *       INITALIZE VALUES
0074                  *
0075  FE00    020C    START   LI   R12,CRUOFF   LOAD CRU OFFSET
      FE02    1FE0
0076                  *
```

```
0077                    *         DETERMINE H/W CONFIGURATION
0078                    *
0079  FE04  0221               AI    R1,=>9900          EXTERNAL SIE?
      FE06  6700
0080  FE08  1602               JNE   FP                 NO =
0081  FE0A  1D0A               SBO   RUNLGT
0082  FE0C  0450               B     *R0
0083        FE0E  FP           EQU   $
0084  FE0E  0209               LI    R9,>A0             LOAD BIAS
      FE10  00A0
0085  FE12  04C0               CLR   R0              LOADER SELECT SWITCH
0086  FE14  1F08               TB    FPBIT              TEST FOR FP PRESENT
0087  FE16  13F2               JEQ   LOADFN             JUMP TO LOADER FUNCTION
0088  FE18  C08E               MOV   R14,R2             MOVE PC TO DATA WORD
0089                    *
0090                    *         MAIN LOOP TO SCAN SWITCHES FOR KEY DEPRESSION
0091                    *
0092  FE1A  3202        LOOP   LDCR  R2,8               OUTPUT MSH LIGHTS
0093  FE1C  3220               LDCR  @FPWP+5,8          OUTPUT LSH LIGHTS
      FE1E  F805
0094  FE20  0205               LI    R5,WAIT
      FE22  FE7C
0095        FE24  OFF          EQU   $
0096  FE24  0695               BL    *R5
0097  FE26  16FE               JNE   OFF
0098  FE28  0695               BL    *R5
0099  FE2A  16FC               JNE   OFF
0100  FE2C  04C1               CLR   R1
0101        FE2E  LOOK         EQU   $
0102  FE2E  1D08               SBO   SCAN
0103  FE30  0695               BL    *R5
0104  FE32  13FD               JEQ   LOOK
0105  FE34  0695               BL    *R5
0106  FE36  13FB               JEQ   LOOK
0107  FE38  1F09               TB    SWPB               CHECK FOR INPUT OF LSH OF DAT
0108  FE3A  1601               JNE   OVER               IF NOT, SKIP SWAP
0109  FE3C  06C1               SWPB  R1                 SWAP DATA TO LSH OF WORD
0110  FE3E  1F08  OVER   TB    TYPE CHECK FOR DATA OR FUNCTION KEY
0111  FE40  1302               JEQ   FUNC               JUMP TO FUNCTION
0112                    *                                    PROCESSING IF SET
0113  FE42  2881               XOR   R1,R2              TOGGLE DATA
0114  FE44  10EA               JMP   LOOP               PROCESS NEXT KEY
0115  FE46  04C5  FUNC   CLR   R5                       CLEAR INDEX REG
0116  FE48  05C5  CHEK   INCT  R5                       INCREMENT INDEX R
0117  FE4A  0A11               SLA   R1,1               SHIFT DATA TILL DETECT
0118  FE4C  17FD               JNC   CHEK               CHECK NEXT BIT
0119                    *                                    IF NONE FOUND
0120  FE4E  04A5  EXEC   X     @TABLE-2(R5)             EXECUTE FUNCTION
      FE50  FE52
0121  FE52  10E3               JMP   LOOP               PROCESS NEXT KEY
0122        FE54  TABLE        EQU   $
0123  FE54  1013               JMP   SIE+$-EXEC-1       PROCESS SIE
0124  FE56  1011               JMP   RUN+$-EXEC-1       PROCESS RUN
0125  FE58  0360               RSET                     RESET
0126  FE5A  10D5               JMP   LOADFN+$-EXEC-1    PERFORM LOAD
```

```
0127                    *
0128  FE5C  C08D               MOV   R13,R2          WP ==> DATA
0129  FE5E  C08E               MOV   R14,R2          PC ==> DATA
0130  FE60  C08F               MOV   R15,R2          ST ==> DATA
0131  FE62  C087               MOV   R7,R2           MA ==> DATA
0132                    *
0133  FE64  C342               MOV   R2,R13          DATA ==> WP
0134  FE66  C382               MOV   R2,R14          DATA ==> PC
0135  FE68  C3C2               MOV   R2,R15          DATA ==> ST
0136  FE6A  C1C2               MOV   R2,R7           DATA ==> MA
0137                    *
0138  FE6C  C097               MOV   *R7,R2          (MA) ==> DATA
0139  FE6E  05C7               INCT  R7              INCREMENT MA
0140  FE70  C5C2               MOV   R2,*R7          DATA ==> (MA)
0141  FE72  04C2               CLR   R2              ZERO ==> DATA
0142                    *
0143  FE74  1D0A  RUN          SBO   RUNLGT          ENABLE RUN (RESTART)
0144  FE76  0380               RTWP                  RETURN TO USER
0145  FE78  1D0E  SIE          SBO   SIEFLG          ENABLE SIE
0146  FE7A  0380               RTWP
0147        FE7C  WAIT         EQU   $
0148  FE7C  1D0D               SBO   STIME
0149  FE7E  1F0A  WAIT2        TB    TIMER           WAIT FOR 10 MSEC
0150  FE80  16FE               JNE   WAIT2
0151  FE82  3601               STCR  R1,8            CHECK SWITCHES
0152  FE84  045B               RT
```

```
0154                    *       *     TITLE * CARD/CASSETTEE LOADR
0155                    *  AUTHOR* JAY ALLEN
0156                    *  COMPUTER*990,MIRA
0157                    *  ABSTRACT*
0158                    *  *  ACCEPTS 990 RELOCATABLE AND ABSOLUTE CODE
0159                    *  *  ACCEPTS ONE OR MODULES
0160                    *  *  WILL NOT PROCESS EXTERNAL REFS AND DEFS
0161                    *  *  EXECUTION BEGINS AT LAST ENTRY VECTOR
0162                    *  STATISTICS*
0163                    *  *  OBJECT FORMAT
0164                    *  *  TAG   MEANING       ACTION
0165                    *  *   0    MODULE ID     UPDATE LOAD BIAS
0166                    *  *   1     ABS ENTRY     SAVE ENTRY POINT
0167                    *  *   2     REL ENTRY     SAVE ENTRY POINT
0168                    *  *   3     RELREF        NONE
0169                    *  *   4     ABSREF        NONE
0170                    *  *   5     RELDEF        NONE
0171                    *  *   6    ABSDEF        NONE
0172                    *  *   7     CHECKSUM      TEST CHECKSUM
0173                    *  *   8     NO CHECKSUM GET NEW RECORD
0174                    *  *   9     ABS LOAD      UPDATE LOAD ADDRESS
0175                    *  *   A     REL LOAD      UPDATE LOAD ADDRESS
0176                    *  *   B     ABS DATA      LOAD DATA
0177                    *  *   C     REL DATA      LOAD DATA
0178                    *  *   D     SET BIAS      UPDATE LOAD BIAS FOR NEXT MODULE
0179                    *  *   E     UNUSED        ERROR
0180                    *  *   F     TERMINATOR    GET NEW RECORD
0181                    *  *   G    REL SYSBOL         SKIP
0182                    *  *   H    ABS SYMBOL         SKIP
0183                    *  *   I    OLD IDT NAME       SKIP
0184                    *  *   J    RSL TYPE           SKIP
0185                    *  *   !     EOF           START PROGRAM EXECUTION
0186          0009  DTR     EQU  9
0187          000A  RTS     EQU  10
0188          000B  WRQ     EQU  11
0189          000C  RRQ     EQU  12
0190          000D  NSF     EQU  13
0191          0007  PRSNCE  EQU  7
0192          0001  NREADY  EQU  1
0193          000F  COLIND  EQU  15
0194          000F  READ    EQU  15
0195                *       REGISTER INITALIZATION
0196          FE86  LOADER  EQU   $
0197  FE86    0360          RSET
0198  FE88    0203          LI   R3,START1          DEFAULT END VECTOR
      FE8A    FFCA
0199  FE8C    020B          LI   R11,GETENT      ENTRY POINT TO GET CHAR
      FE8E    FFC8
0200  FE90    0205          LI   R5,>F7FE
      FE92    F7FE
0201          FE94  MEMCLR  EQU  $
0202  FE94    C555          MOV  *R5,*R5
0203  FE96    0645          DECT R5
0204  FE98    18FD          JOC  MEMCLR
```

```
0205  FE9A  1D0A          SBO   RUNLGT            ENABLE RESTART
0206  FE9C  1F0E          TB    MANTPL            MAINTANCE PANEL PRESENT?
0207  FE9E  1613          JNE   LDR1              YES USE PANELS LOADER
0208  FEA0  020C          LI    R12,>40           CR CRU BASE
      FEA2  0040
0209  FEA4  C000          MOV   R0,R0           LOADER SELECT
0210  FEA6  1605          JNE   CRINIT
0211  FEA8  04CC          CLR   R12                 ASR CRU BASE
0212  FEAA  1D09          SBO   DTR
0213  FEAC  1D0A          SBO   RTS
0214  FEAE  3220          LDCR  #PBON,8
      FEB0  FFC0
0215        FEB2  CRINIT  EQU   $
0216              *  GET  NEW   RECORD
0217        FEB2  NEWREC  EQU   $
0218  FEB2  C000          MOV   R0,R0           LOADER SELECT
0219  FEB4  1308          JEQ   NEWRC1
0220  FEB6  1F07          TB    PRSNCE
0221  FEB8  13FE          JEQ   $-2
0222  FEBA  1F01          TB    NREADY
0223  FEBC  13FE          JEQ   $-2
0224  FEBE  1E0F          SBZ   READ                READER ON
0225  FEC0  1F07          TB    PRSNCE
0226  FEC2  16FE          JNE   $-2
0227  FEC4  1D0F          SBO   READ
0228        FEC6  LDR1    EQU   $
0229              NEWRC1
0230  FEC6  04C7          CLR   R7                  CHECKSUM
0231              *  GET  NEW   FIELD
0232        FEC8  GETFEL  EQU   $
0233        FEC8  RSL     EQU   $
0234  FEC8  069B          BL    *R11
0235  FECA  D0AA          MOVB  #OP(R10),R2     JUMP DILPLACMENT
      FECC  FFE4
0236  FECE  1327          JEQ   FTAG
0237  FED0  0882          SRA   R2,8
0238  FED2  C107          MOV   R7,R4               SAVE CHECKSUM
0239  FED4  0201          LI    R1,4                GET NEXT 4 CHARS
      FED6  0004
0240        FED8  REP     EQU   $
0241  FED8  069B          BL    *R11
0242  FEDA  0A46          SLA   R6,4                 ACCUMULATOR
0243  FEDC  A18A          A     R10,R6
0244  FEDE  0601          DEC   R1
0245  FEE0  16FB          JNE   REP
0246        FEE2  JMP     EQU   $
0247  FEE2  0462          B     #JMP(R2)        GO TO TAG PROCESSOR
      FEE4  FEE2
0248        FEE6  RELSYM  EQU   $
0249        FEE6  ABSSYM  EQU   $
0250        FEE6  RLDEF   EQU   $
0251        FEE6  ABDEF   EQU   $
0252        FEE6  RLREF   EQU   $
0253        FEE6  ABREF   EQU   $
0254  FEE6  0201          LI    R1,6
```

```
      FEE8  0006
0255  FEEA  1015          JMP   SKIP              SKIP  REF/DEF
0256  FEEC  A189  RELOAD  A     R9,R6             RELOCATE
0257  FEEE  C146  ABLOAD  MOV   R6,R5             GET NEW LOAD ADDRESS
0258  FEF0  10EB          JMP   GETFEL
0259  FEF2  A189  REDATA  A     R9,R6             RELOAATE
0260  FEF4  DD46  ABDATA  MOVB  R6,*R5+           OAD DATA
0261  FEF6  06C6          SWPB  R6
0262  FEF8  DD46          MOVB  R6,*R5+
0263  FEFA  10E6          JMP   GETFEL
0264  FEFC  A106  CHECK   A     R6,R4             CHECKSUM
0265  FEFE  13E4          JEQ   GETFEL
0266        FF00  EREXIT  EQU   $
0267  FF00  046B          B     @2(R11)
      FF02  0002
0268  FF04  C246  SBIAS   MOV   R6,R9             NEW RELOC LOAD BIAS,
0269  FF06  0249          ANDI  R9,>FFFE
      FF08  FFFE
0270  FF0A  10DE          JMP   GETFEL
0271  FF0C  A189  RLENT   A     R9,R6             RELOACTE
0272  FF0E  C0C6  ABENT   MOV   R6,R3             ENTRY VECTOR
0273  FF10  10DB          JMP   GETFEL
0274        FF12  LENG    EQU   $
0275  FF12  0201  OLDIDT  LI    R1,8
      FF14  0008
0276        FF16  SKIP    EQU   $
0277  FF16  069B          BL    *R11
0278  FF18  0601          DEC   R1
0279  FF1A  16FD          JNE   SKIP
0280  FF1C  10D5          JMP   GETFEL
0281              *       STATUS WIL BE EQUAL ON END OF RECORD
0282        FF1E  FTAG    EQU   $
0283  FF1E  069B          BL    *R11
0284  FF20  16FE          JNE   FTAG
0285  FF22  10C7          JMP   NEWREC
```

```
0287                    *
0288                    *       GET CHAR CO-ROUTINE
0289                    *       CALL FOR GET CHAR
0290                    *       BL    *R11
0291                    *
0292                    *       CALL TO ERROR EXIT
0293                    *       B     @2(R11)
0294                    *
0295                    *       R11 INITIAL VALUE IS AT END OF ROUTINE
0296                    *             E. G.
0297                    *       GETRT   BL    *R11
0298                    *       ENTRY   JMP   GET
0299                    *
0300          FF24      GET EQU $
0301    FF24  C000              MOV   R0,R0           LOADER SLELECT
0302    FF26  1338              JEQ   GTASR
0303                    *
0304                    *       DECODE AND TRANSLATE CARD PUNCHES
0305                    *
0306    FF28  1F07              TB    PRSNCE
0307    FF2A  1634              JNE   EORSET            GO SET END OF RECORD
0308    FF2C  1F0F              TB    COLIND
0309    FF2E  16FC              JNE   $-6
0310    FF30  340A              STCR  R10,0           GET CARD PUNCHES
0311                    *                                ! 12!4567! PMZ!98   !
0312                    *
0313    FF32  04C8              CLR   R8
0314    FF34  D20A              MOVB  R10,R8             ! 12!4567!
0315    FF36  092A              SRL   R10,2
0316    FF38  0A8A              SLA   R10,11             !PMZ9!8    !
0317                    *
0318    FF3A  058A              INC   R10
0319    FF3C  0A18              SLA   R8,1
0320    FF3E  16FD              JNE   $-4
0321                    *
0322    FF40  060A              DEC   R10                !PMZ9!8    !     ! 0-7!
0323                    *
0324    FF42  C20A              MOV   R10,R8
0325    FF44  06C8              SWPB  R8                 !     ! 07-!PMZ9!8    !
0326    FF46  E288              SOC   R8,R10
0327    FF48  C20A              MOV   R10,R8             !PMZ9!0--8!PMZ9!0--8!
0328                    *
0329    FF4A  024A              ANDI  R10,>F
        FF4C  000F
0330    FF4E  022A              AI    R10,>30            !     !     !     ! 11!0--8!
        FF50  0030
0331                    *
0332    FF52  0A18              SLA   R8,1               P?
0333    FF54  1702              JNC   $+6
0334    FF56  022A              AI    R10,>10
        FF58  0010
0335                    *
0336    FF5A  0A18              SLA   R8,1               M?
0337    FF5C  1702              JNC   $+6
```

```
0338  FF5E  022A          AI    R10,>19
      FF60  0019
0339                *
0340  FF62  0A18          SLA   R8,1              Z?
0341  FF64  1702          JNC   $+6
0342  FF66  022A          AI    R10,>23
      FF68  0023
0343                *
0344  FF6A  0A18          SLA   R8,1              9?
0345  FF6C  1702          JNC   $+6
0346  FF6E  022A          AI    R10,>9
      FF70  0009
0347                *
0348  FF72  028A          CI    R10,>30            BLANK?
      FF74  0030
0349  FF76  1602          JNE   $+6
0350  FF78  020A          LI    R10,>20
      FF7A  0020
0351                *
0352  FF7C  028A          CI    R10,>53            Z?
      FF7E  0053
0353  FF80  1602          JNE   $+6
0354  FF82  020A          LI    R10,>30
      FF84  0030
0355                *
0356  FF86  028A          CI    R10,>55            Z,2-9
      FF88  0055
0357  FF8A  1101          JLT   $+4
0358  FF8C  064A          DECT  R10
0359  FF8E  1F0F          TB    COLIND
0360  FF90  13FE          JEQ   $-2
0361  FF92  1010          JMP   GET1
0362        FF94  EORSET  EQU   $
0363  FF94  B28A          C     R10,R10          SET EQUAL STATUS
0364  FF96  1017          JMP   GETRT
0365        FF98  GTASR   EQU   $
0366  FF98  1F0C          TB    RRQ
0367  FF9A  16FE          JNE   $-2
0368  FF9C  1E0C          SBZ   RRQ
0369  FF9E  35CA          STCR  R10,7
0370  FFA0  098A          SRL   R10,8
0371  FFA2  028A          CI    R10,>D           END OF RECORD?
      FFA4  000D
0372  FFA6  130F          JEQ   GETRT
0373  FFA8  028A          CI    R10,>5A
      FFAA  005A
0374  FFAC  15F5          JGT   GTASR
0375  FFAE  028A          CI    R10,>20
      FFB0  0020
0376  FFB2  11F2          JLT   GTASR
0377        FFB4  GET1    EQU   $
0378  FFB4  A1CA          A     R10,R7             UPDATE CHECKSUM
0379  FFB6  022A          AI    R10,->30       CONVERT
      FFB8  FFD0
0380  FFBA  028A          CI    R10,>A
```

```
       FFBC  000A
0381   FFBE  1306          JEQ   EXIT            END OF LOA D MODULE
0382         FFC0  PBON     EQU   $
0383   FFC0  1102          JLT   GETRT
0384   FFC2  022A          AI    R10,-7
       FFC4  FFF9
0385         FFC8  GETRT    EQU   $
0386   FFC6  069B          BL    *R11
0387         FFC8  GETENT   EQU   $
0388   FFC8  10AD          JMP   GET
0389               *        HERE ON ERROR EXIT
0390         FFCA  START1   EQU   $                 LOAD ERROR OR NO VECTOR
0391   FFCA  0340          IDLE                    INDICATE ERROR WITH IDLE LIGH
0392         FFCC  EXIT     EQU   $
0393   FFCC  C000          MOV   R0,R0             LOADER SELECT
0394   FFCE  1603          JNE   RESET
0395   FFD0  1E08          SBZ   WRQ
0396   FFD2  1E0C          SBZ   RRQ
0397   FFD4  0581          INC   R1
0398   FFD6  16FE          JNE   $-2
0399   FFD8  1F0C          TB    RRQ
0400   FFDA  13F8          JEQ   EXIT
0401   FFDC  1E09          SBZ   DTR
0402   FFDE  1E0D          SBZ   NSF
0403         FFE0  RESET    EQU   $
0404   FFE0  0360          RSET
0405   FFE2  0453          B     *R3       ENTRY VECTOR
```

```
 0407                    ****************************
 0408                    *  JUMP TABLE
 0409                    ****************************
 0410  FFE4     30   OP      BYTE   LENG=JMP
 0411  FFE5     2C           BYTE  ABENT=JMP
 0412  FFE6     2A           BYTE  RLENT=JMP
 0413  FFE7     04           BYTE  RLREF=JMP
 0414  FFE8     04           BYTE  ABREF=JMP
 0415  FFE9     04           BYTE  RLDEF=JMP
 0416  FFEA     04           BYTE  ABDEF=JMP
 0417  FFEB     1A           BYTE  CHECK=JMP
 0418  FFEC     E6           BYTE GETFEL=JMP
 0419  FFED     0C           BYTE ABLOAD=JMP
 0420  FFEE     0A           BYTE RELOAD=JMP
 0421  FFEF     12           BYTE ABDATA=JMP
 0422  FFF0     10           BYTE REDATA=JMP
 0423  FFF1     22           BYTE  SBIAS=JMP
 0424  FFF2     1E           BYTE EREXIT=JMP
 0425  FFF3     00           BYTE  0
 0426  FFF4     04           BYTE RELSYM=JMP
 0427  FFF5     04           BYTE ABSSYM=JMP
 0428  FFF6     30           BYTE OLDIDT=JMP
 0429  FFF7     E6           BYTE    RSL=JMP
 0430  FFF8   FE86           DATA LOADER
 0431  FFFA   FEC6           DATA LDR1
 0432  FFFC   F800           DATA FPWP
 0433  FFFE   FE00           DATA START
 0434                  *      LOADFN EQU LOADER      W/O SELF=TEST
 0435                  *      LOADFN EQU  >FDFC      FOR SELF=TEST
 0436         FDFC     LOADFN EQU  >FDFC
 0437                  *      FPWP   EQU  >F800      FOR PROTOTYPING SYSTEM
 0438                  *      FPWP   EQU  >80        OTHERWISE
 0439         F800     FPWP   EQU  >F800
 0440                           END
0000 ERS
```

960 = 980 CONCORDANCE

```
R10      0048      0235  0243  0310  0314  0315  0316  0318  0322  0324
                   0326  0327  0329  0330  0334  0338  0342  0346  0348
                   0350  0352  0354  0356  0358  0363  0363  0369  0370
                   0371  0373  0375  0378  0379  0380  0384
R11      0049      0199  0234  0241  0267  0277  0283  0386
R12      0050      0075  0208  0211
R13      0051      0128  0133
R14      0052      0088  0129  0134
R15      0053      0130  0135
R2       0040      0088  0092  0113  0128  0129  0130  0131  0133  0134
                   0135  0136  0138  0140  0141  0235  0237  0247
R3       0041      0198  0272  0405
R4       0042      0238  0264
R5       0043      0094  0096  0098  0103  0105  0115  0116  0120  0200
                   0202  0202  0203  0257  0260  0262
R6       0044      0242  0243  0256  0257  0259  0260  0261  0262  0264
                   0268  0271  0272
R7       0045      0131  0136  0138  0139  0140  0230  0238  0378
R8       0046      0313  0314  0319  0324  0325  0326  0327  0332  0336
                   0340  0344
R9       0047      0084  0256  0259  0268  0269  0271
READ     0194      0224  0227
REDATA   0259      0422
RELOAD   0256      0420
RELSYM   0248      0426
REP      0240      0245
RESET    0403      0394
RLDEF    0250      0415
RLENT    0271      0412
RLREF    0252      0413
RRQ      0189      0366  0368  0396  0399
RSL      0233      0429
RTS      0187      0213
RUN      0143      0124
RUNLGT   0064      0081  0143  0205
SBIAS    0268      0423
SCAN     0060      0102
SIE      0145      0123
SIEFLG   0069      0145
SKIP     0276      0255  0279
START    0075      0433
START1   0390      0198
STIME    0067      0148
SWPB     0062      0107
TABLE    0122      0120
TIMER    0065      0149
TYPE     0061      0110
WAIT     0147      0094
WAIT2    0149      0150
WRQ      0188      0395

THERE ARE   0094  SYMBOLS
```