DI : 79656

General European Road Data Information Exchange Network

GERDIEN

DRIVE-II PROJECT NUMBER: V2044

TECHNICAL ANNEX

ES15

Implementation Concepts

Author:

TNO Institute of Applied Physics

Contract Date: jan '92
Submission date: May '92

GERDIEN PARTNERS:
TNO, RWS, TECHNOLUTION, AACHEN TH, SIEMENS, DORNIER, LEEDS, PHILIPS, CRC, AVE

Commission of the European Communities - R&D programme Telematics Systems in the area of Transport (DRIVE II)

PREFACE

ES-15 is a system, designed to perform a wide range of tasks along roads in interurban areas. It is a modular system that can be flexibly configured from a variety of building blocks. The ES-15 implementation complies with the UROP (Universal ROadside Processor) reference architecture and is based on a chosen infrastructure. For communication over the roadnetwork, the ES-15 is based on the UROP Open Road Communication Architecture and for implementation of the fieldnetwork is chosen for the IEEE 1118 standard.

GLOSSARY

AID	Automatic Incident Detection
ATT	Advanced road Transport Telematics
DNT	Drive Normalized Transport (layer)
GDF	Graphic Data File
GIS	Geographic Information System
IRTE	Integrated Road Transport Environment
MMI	Man Machine Interface
TCC	Traffic Control Center
UROP	Universal ROadside Processor
VMS	Variable Message Sign

CONTENTS

1 INTRODUCTION

This report has been prepared by TNO-TPD as a result of work being carried out in the context of the Gerdien project for the EG DRIVE programme and the ES15 project for the Dutch Ministry of Transport DVK. The main result of the work covered in this report is a technical design for a Universal ROadside Processor for use in the Dutch infrastructure for Dynamic Traffic Monitoring.

1.1 Background

To allow for better information to roadusers during all stages of their trips many systems at different hierarchical levels must be able to exchange information. They must become as it were part of a larger whole. In the DRIVE programme of the European Communities this larger whole is called the Integrated Road Transport Environment" (IRTE). It is regarded as the common outcome of the programme. The Gerdien project in the DRIVE programme covers a major part of the IRTE in interurban areas.

Within the Dutch policy context the larger whole is called "Dynamic Traffic Management". The ES15 is the universal roadside system in the Dutch infrastructure for all roadside applications in the context of Dynamic Traffic Management.

1.2 Role of Roadside Processors in the IRTE

Roadside Processors are infrastructure based components of the IRTE, they are roadside tools for Dynamic Traffic Management in interurban areas of the IRTE. Roadside Processors form as it were the senses of the IRTE, they have peripherals that can monitor the road traffic state, that can communicate with passing vehicles and that can influence the traffic flows.

Roadside Processors in an area send information about the local traffic state to a common Traffic center, where management functions are performed for smaller parts of the roadnetwork. Depending on the scope of the traffic control at this regional level, actions may be determined autonomously (lane closing, hazard warnings), or in a concerted mode by higher level Traffic centers (e.g. rerouting). At the highest (national) level management functions are performed for larger areas of the roadnetwork. For both levels Roadside Processors provide the basic data for Monitoring functions and in return they receive setpoints to influence the local traffic flows.

Because of the clear hierarchy involved the IRTE can be represented as a layered structure. Figure 1.1 gives a schematic representation of this structure.

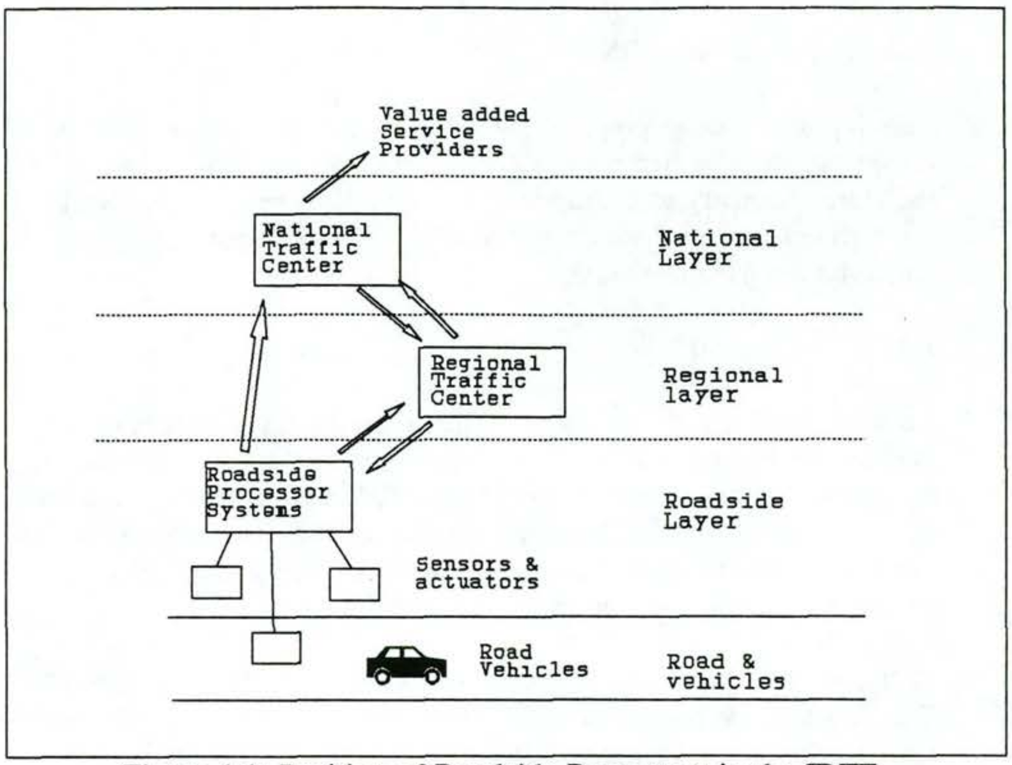


Figure 1.1: Position of Roadside Processors in the IRTE

1.3 The UROP concept

A Universal ROadside Processor (UROP) is a modular system for multiple applications along roads in interurban areas. Due to the modular design, the system can be flexibly configured out of available building blocks. The next figure shows the general system structure of a UROP.

Relatively simple systems and more complex systems can be configured from the same building blocks, making it easy to build cost effective systems depending on the needs of specific roadside locations.

Fieldnetwork

Peripherals in a UROP system structure are not regarded as passive components. Local intelligence makes it possible that peripherals can do simple preprocessing functions and quality control. Multiple peripherals are connected to higher level applications via a high speed local fieldnetwork. Data exchange in the fieldnetwork is done on a basis of low level communication functions.

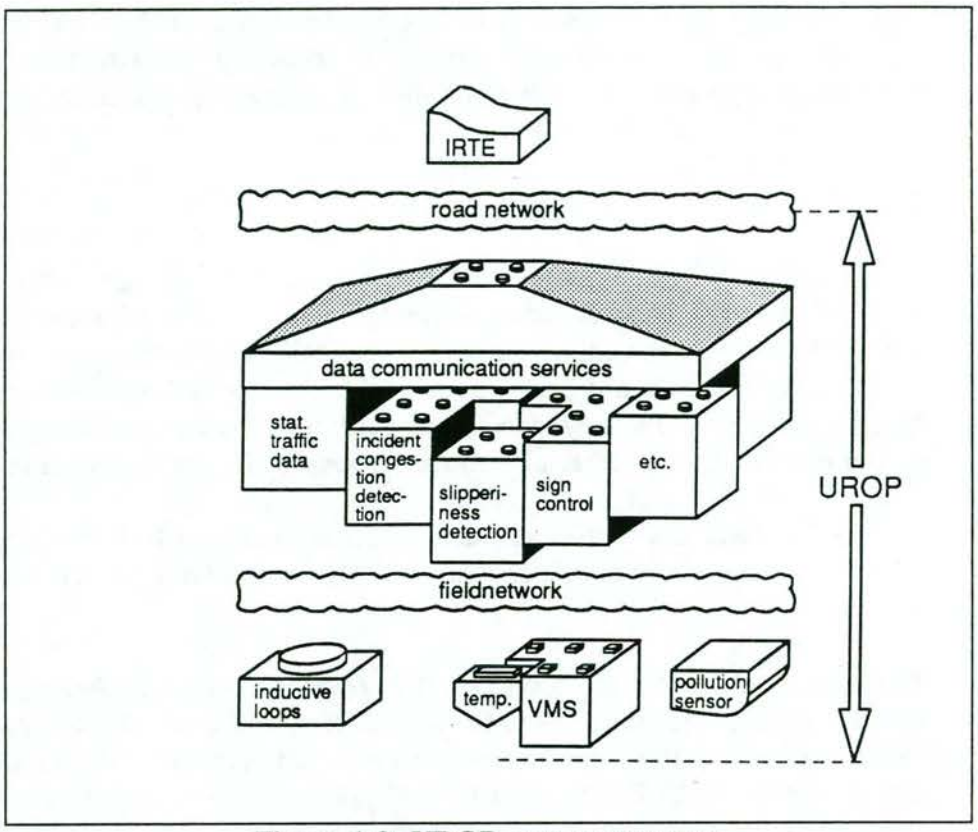


Figure 1.2: UROP system structure

The fieldnetwork between the peripherals and the central part of the UROP allow for the installation of peripherals in a geographical spread, in practice a distance of 1 km should be possible. The fieldnetwork also provides for a logical separator, that allows the UROP applications to view the peripherals as logical devices without having to be concerned of technological aspects.

Roadnetwork

The roadnetwork is the telemetry system between the roadside processors and the higher level of control. A proper set of communication services will hide all physical aspects of the roadnetwork topology to the roadside applications.

From the viewpoint of modularity the roadside applications must be independent of each other. This demands for multiple communication. The application interface to the roadnetwork protocols will provide a private transparent path for communication for multiple applications.

Roadside Applications

Applications in a UROP system interface at the one end to the road network and at the other side they can communicate to peripherals such as vehicle detectors, video camera's, variable message signs, etcetera, etcetera. A proper internal structure of the UROP allows for multiple applications to share the use of common peripherals.

1.4 Purpose of the document

The UROP concept is an implementation independent concept. In the UROP architecture model [1] possible basic building blocks are identified. Their internal functional behaviour is described and the logical interfaces between building blocks are defined. The architecture model is completely independent of any hardware and software implementation. All functionality and data in the model is described in logical terms. Consequently the architecture model can serve as a reference model for several different system implementations.

The ES-15 system is an implementation design, according to the UROP concept, for a selection of tasks in a specific chosen infrastructure. This document gives a description of the ES-15 system.

To the external users of ES15 systems it is not relevant what hardware processor or software operating system has been used for a specific installation. Market principles must allow that manufactures build systems on selected hardware and software platforms as long as they obey the ES15 interfaces between system components. In principle this document could suffice with just the physical interface descriptions. For a proper understanding however of the interfaces some understanding of the internal functional behaviour is required. Therefor this document has been titled 'Implementation concepts'.

2 SELECTED INFRASTRUCTURE

This chapter identifies the systems that are external to the ES15.

The ES-15 is designed for the infrastructure of the Dutch "Monitoring network". Within this network the ES-15 is the universal roadside system for the acquisition of all traffic and roadcondition related data and for traffic control applications. At the highest level of the national Monitoring Centres this data is used for an assessment of the traffic state at the level of the overall roadnetwork. At the regional level the data is used for road management purposes.

For the transfer of data from the roadside systems to the Monitoring centers use will be made of available cabling networks buried along the motorways. In the Netherlands this network is called the 'Road Telecommunication Network' (RTN). This cabling network consists of a collection of low frequency heavy loaded cables with a total length of approximately 2700 km. The cables in the RTN are installed around a number of centers resulting in 13 separate starshaped networks. Each of these star networks cover regions with a radius of 50 km maximum, figure 2.1 shows these networks.

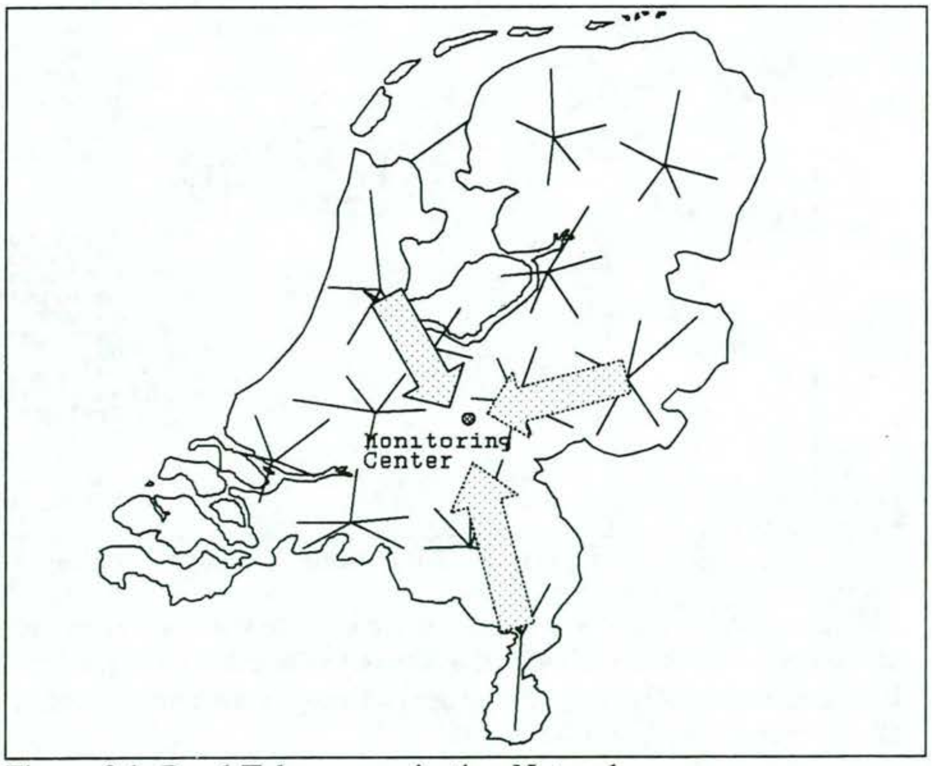


Figure 2.1: Road Telecommunication Network

The pupin coils in the cables every 3 km provide minimum attenuation within a frequency band of 300 to 3400 Hz. Outside this band the pupinisation causes large attenuation. Until a few years ago this caused a limit to the communication speed over the cables of 1200 baud. Currently, with the aid of V29 modem techniques and regeneration transmitters in the modem higher communication speed are possible. Modem tests have proved that 9600 baud over pupinized wires is feasible.

For data communication purposes the RTN cables can be used in a multidrop topology, resulting in multiple multidrop networks per region. Figure 2.2 shows the total network topology selected for the Monitoring environment.

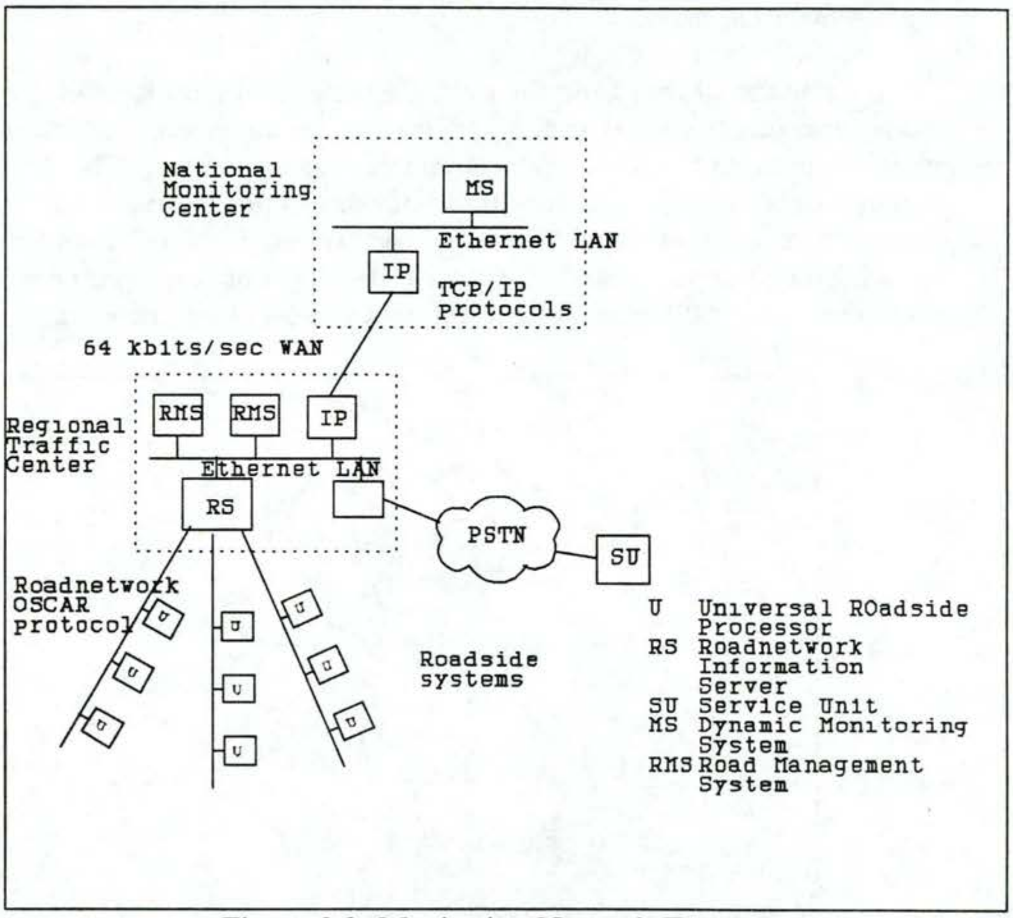

National Monitoring Center
MS
Ethernet LAN
IP
TCP/IP protocols
64 kbits/sec WAN
Regional Traffic Center
RMS
RMS
IP
Ethernet LAN
RS
PSTN
SU
Roadnetwork OSCAR protocol
U
Roadside systems
U Universal ROadside Processor
RS Roadnetwork Information Server
SU Service Unit
MS Dynamic Monitoring System
RMS Road Management System

Figure 2.2: Monitoring Network Topology

Multiple roadside stations are connected to a single wire pair. A multidrop controller function at the end of the cable divides the access to the cable over the connected ES15 systems. Each center of an RTN star cabling network may contain multiple of these controller systems, all interconnected in a LAN.

The bandwidth of the cabling network is limited with regard to the LAN. For all monitoring purposes, in which a continuous flow of data from roadside systems is requested, it would be wasteful if data is transmitted multiple times over the roadnetwork. The solution chosen in the selected infrastructure is to transmit data once to the RTN-server systems. From there the data is distributed to all 'subscribed' systems, both at the regional level and the national level.

The various systems in the network are briefly described below:

ES15 Universal ROadside Processor systems

ES15 systems are modular systems for a range of applications. Because of their modular system structure they can be flexibly configured for a mix of applications. Simple systems are possible, but also complex systems with a range of peripherals for multiple applications.

Roadnetwork Information Server systems

These are special systems that form the gateway between the roadnetwork and the higher level systems. The RTN-Server systems provide the following functions:

- a network multidrop controller function to multiple multidrop networks
- a network routing function to the PSTN
- time synchronisation of the ES15 systems
- distribution of roadside data to subscribed systems
- interface for commands to roadside applications

In general the information servers provide a transparent addressing point for all information exchange between roadside systems and systems at the higher level.

Road Management Systems

On behalf of operational road management tasks within an area, all sort of data is required. Road condition data, actual traffic data, traffic weight data, etcetera, is required for various tasks that are performed on specific RMS systems. RMS's can be regarded as workstations with a high resolution graphic screens on which a roadmap can be displayed. The traffic operator can zoom into details of the map and he can request data from selected objects on the map. Multiple RMS's will require data from common ES15 systems.

Service Unit

ES15 systems can output error messages to a CU when hardware problems are recognized. In return the CU will allow a maintenance engineer to establish a communication channel with the respective ES15. From requested maintenance data the cause of the problem will become clear and a repair action can be organized.

An other important function of the CU is that it contains a complete backup of all the location specific parameters of all ES15 systems in the network. The CU also contains specific screens for the creation of specific parameters and for the download to the respective roadside system.

National Monitoring Center

At this level a larger portion of the roadnetwork is monitored at a regional level. From correlation of the received minute messages journey time between network modes can be estimated. At this level also congestions are spotted and revolving actions are performed.

The system structure of the Monitoring Centers have great resemblance with the regional road management systems. For both system types the roadmap based presentation and storage is an important aspect.

3 GENERAL SYSTEM STRUCTURE

Figure 3.1 presents a system structure that reflects the UROP concepts. From a network point of view the system is made up of a master node and a number of slave nodes. The slave nodes can each control a number of peripherals such as Vehicle Detectors, VMS's, etcetera. All nodes are interconnected in an IEE1118 fieldnetwork.

Looking at the installation of equipment along the road an ES15 may consist of a number of units (cabinets) in a certain geographical spread. See figure 3.1 for the constituent parts of an ES-15 system.

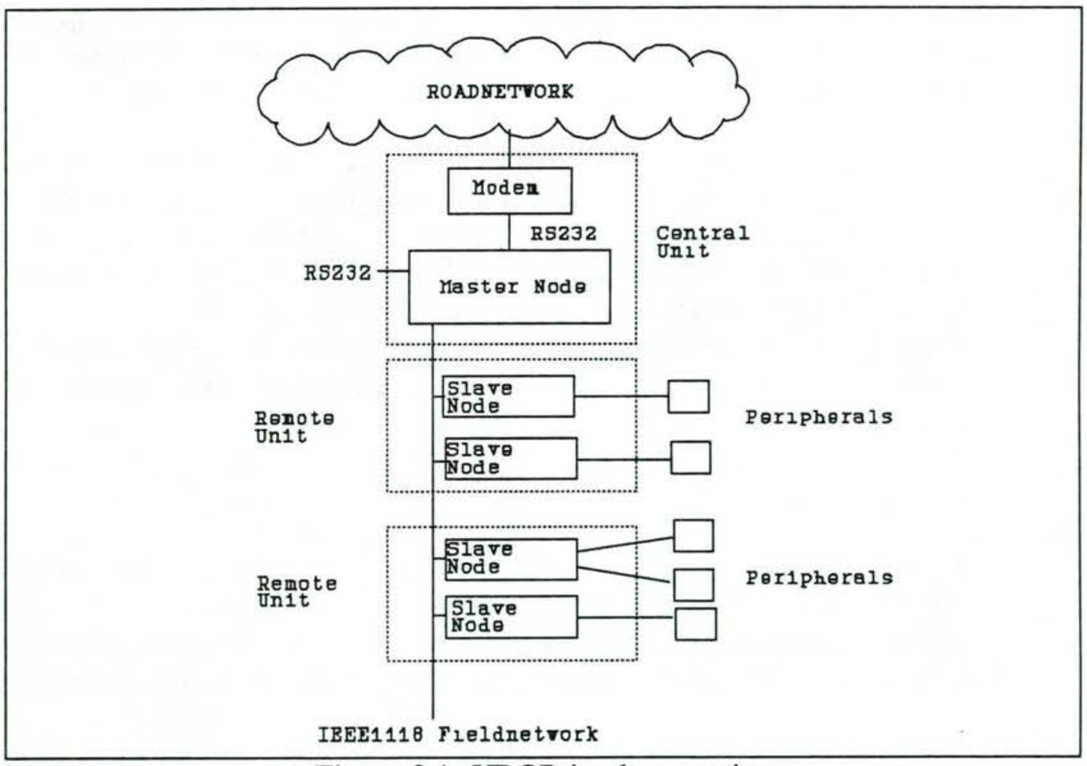


Figure 3.1: UROP implementation

An ES15 consists of a Central Unit and zero or more Remote Units. The fieldnetwork allows for a geographical spread of the remote units in an area of approximately 1 km.

Master node

The master node comprises the basic ES15 runtime environment and a number of applications. The basic environment comprises a multitasking database, the roadnetwork services, the fieldnetwork services, a service task and of course a real time multitasking operating system. The master node is designed such that the whole of the functionality can run in a single microcomputer board with prom based software. For larger systems, that require heavy processing, a number of parallel processors is possible.

The ES-15 master node contains a RS232 port for a console function and a RS232 port for a connection to the roadnetwork. Functionally both connections are equal, the same protocol is used for both. In this manner all ES-15 applications can be addressed in a uniform way, namely local as well as remote for example with a PC.

Roadnetwork

The Roadnetwork interconnects ES15 systems to a regional center. Physically the roadnetwork can be the RTN cabling network discussed in the previous chapter or the PSTN. The roadnetwork should however not be restricted to these topologies. The ES15 roadnetwork services provide a clear separation between the applications and the communication protocols, thus ensuring flexible integration of future network topologies.

The roadnetwork services provide for each connecting application a transparent communication path to a remote application. The basic concept for this is the concept of the Socket (in analogy with TCP/IP) that allows an application to establish a private path for communication with a remote application. This private path, named the "virtual circuit", allows the applications at both ends to communicate via a private handshake.
Besides services for private circuits also broadcast services are required, that allow for accurate time synchronization services of Roadside Processors in multidrop- or busnetwork topologies.

Fieldnetwork

An IEEE1118 network is a fast local area network for the connection of simple microcontrollers. The network is based on a multidrop topology with a stable master. The network knows simple transport services that allow multiple applications in the master node to communicate transparently to multiple applications in slave nodes.

In total 250 slave controllers can be addressed in the network. The network has a wide variety of speed versus distance settings. In practice for instance a distance of approximately 1 kilometre is possible at a speed of 62.5 bits/sec without the use of repeaters.

4 ROADNETWORK SERVICES

The roadnetwork services provide communication facilities between roadside applications and applications at the higher level. The required protocols for the ES15 must be able to use existing network infrastructures (e.g. PSTN), private cabling networks and they should also be structured in multiple functional layers compatible with the ISO OSI reference model for communication. Because of this layered structure the interface between applications and the network will be independent of the physical infrastructures. This will allow for the inclusion of standard networks in the future.

The roadnetwork protocols must be able to run in simple (prom bases) systems. To the applications the protocols must provide peer to peer communication for the transfer of traffic related data. In private cabling networks the protocols must provide a broadcast mechanism for time synchronisation of connected systems. In the ES15 infrastructure the solution selected comprises a layered structure of protocol definitions named "Open Road Communication Architecture" (OSCAR). This chapter describes the characteristics of the application interface in the selected solution.

4.1 Application interface

Actual peer to peer communication between applications in computersystems starts in the OSI transport layer. The following network services are available to applications:

ORC_SOCKET
Creates a new socket in the system. This socket will behave as a full duplex communication port to the calling application

```
        CALL ORC_SOCKET (socket_name, status)
```

ORC_CONNECT
Creates a virtual circuit between two sockets.

```
        CALL ORC_SOCKET (local_socket, remote_system,
                                  remote_socket, status)
```

ORC_ACCEPT
Accepts a connect request to a local socket

```
        CALL ORC_ACCEPT (local_socket, status)
```

ORC_READ
Reads a message from a socket

```
        COUNT= ORC_READ (local_socket, buffer, status)
```

ORC_WRITE
Writes a message to a socket

CALL ORC_WRITE (local_socket, buffer, length, status)

ORC_CLOSE
Terminates the connection to a socket
CALL ORC_CLOSE (local_socket, status)

4.2 Datagram services

Datagram services allow for broadcasting of messages. In the selected infrastructure this will allow for broadcasting of time synchronisation messages from the RTN-server systems to the connected ES15 slaves.

Broadcasting in principle is an unreliable means of communication. It is not necessary to establish a connection, but messages are transmitted without further action. A special socket is required, which is constantly 'open' for datagram services.

In the ES15 specification the datagram services are used for time synchronization. Therefor the ES15 only reads datagram messages and does not transmit them. The following basic service is available for the reception of a datagram message:

OSC_DGREAD
Receive datagram message
CALL OSC_DGREAD(local_socket, buffer, status)

4.3 Network Management Services

The roadnetwork communication services maintain a number of status statistics that allow a system manager to detect the cause of a problem in case of network malfunctioning. For each layer a specific set of statistics is maintained. The following services are available to applications that access these statistics:

OSC_GET_STAT
Read layer statistics
CALL OSC_GET_STAT(layer, link, buffer, status)

OSC_CLEAR_STAT
Clear layer statistics
CALL OSC_CLEAR_STAT(layer, link, status)

For the details of all preceding service calls refer to the appropriate OSCAR documentation.

4.4 Localization of Roadside Data

In the presented infrastructure roadside data is flexibly distributed to multiple enduser systems. A condition to this organization is that all data must be uniqely referenced in the context of the whole roadnetwork.

A roadsidelocation can be uniquely described in the whole roadnetwork with "Curve linear Localization" attributes that are based on the regular distance poles along the roads. A complete localization description is a string of text with the following information:

loc_string = Road + km-point + distance + Carriagewaynumber + Lanenumber

A disadvantage of the above localization is the size of the data and the fact that multiple strings can reference the same location. In the ES15 infrastructure therefor also a unique binary code is used in continuous streams with monitoring data:

bin_loc-key = 32 bit integer code

One of the functions of the Central Service System is to manage all roadside localization codes for the whole infrastructure.

4.5 Data Formats

In order to achieve unambiguous data exchange between remote applications over the roadnetwork data types have to be defined and the manner in which they are transmitted. The following data types are used in the ES15 data descriptions according to ANSI C conventions:

Data type	Size bytes	Description
char	1	integer
unsigned short	2	integer
signed short	2	two's complement integer
unsigned long	4	integer
signed long	4	two's complement integer
float	4	single precision floating point according to IEEE754
double	8	double presicion floating point according to IEEE754

char[n]	n	variable length string of characters, with a maximum length of n-1, ended with a zero character

In the ES15 specifications a number of dataitems are named differently, their correspondence with the ANSI C notation is as follows:

Data type	Size bytes	ANSI C synonym
word	2	unsigned short

The ES15 network communication uses a number of standard records that can be regarded as datatypes specific to the ES15. They are listed below:

Data type	bytes	Description
ES15_time	4	time in seconds since 1 january 1978 00:00:00 hours
packed_string[n]	n+2	packed string, is a character string preceded by a length indicator: RECORD packed_string[n] OF char_length = n word, char_string char[n] END RECORD
Loc_string	64	location description in terms of road, km-point, etc in a packed string
bin_loc_key	4	32bit integer, representative for a particular loc_string
DB_name	max 36	packed_string[32] contains the name of an ES15 database

The packed-string allows for efficient packing of variable string names in network messages. Internally in the ES15 the datatype allows for maximum flexibility in the use of computer languages.

All datatypes are transmitted with the low order byte in front.

In order to prevent alignment problems in the receiver's software all datatypes in complex records are transmitted closely fitted.

5 FIELDNETWORK SERVICES

The ES15 fieldnetwork allows for the installation of sensors and actuators in a certain spread around the master node. A small microcontroller in front of the sensors and the actuators provide simple intelligence that allow the higher level applications to view them as logical peripherals that they can control in terms of simple communication procedures.

The general structure of a fieldnetwork node can be depicted as follows:

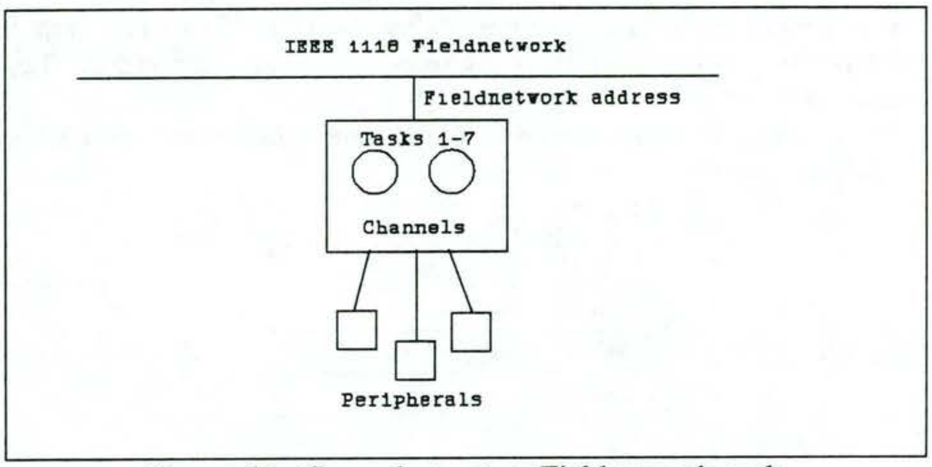


Figure 5.1: General structure Fieldnetwork node

Multiple peripherals can be connected to a single microcontroller. Each peripheral can be identified by its associated channel number. The whole controller can be uniquely identified by its address in the fieldnetwork.

From the perspective of systems sharing, the IEEE 1118 interconnect external interfacing is defined on three protocol levels, namely physical, datalink and message protocol.

Physical Level
: Twisted pair wires are used for the transmission medium, 9-pin female connectors are used for the line connection and RS485 drivers and receivers are used for the data transfer. The network can be operated in two different modes; i.e. synchronously and asynchronously, resulting in a variety of performance/distance options. For instance at a speed of 62.5 kbit/sec a distance of 1200 metres can be met without the use of repeaters.

Data link level
: The data link protocol is a subset of the SDLC protocol without the use of the reject option

Message control level
: The message transfer protocol requires that order messages may only be transmitted by

a master device. For every order message the recipient must eventually transmit back a reply message. No more than 7 order messages may be put to any slave device.

The message transfer protocol is based on a multi tasking environment. Several task in a master node can send order messages to a slave node. The different tasks in the slave node don't require to reply in the same order.

Several sources of errors can occur in the message protocol. Specific examples are irrecoverable protocol errors (e.g. due to hardware failure) or unknown tasks. If an error occurs during the delivery of an order then the complete order is immediately converted into a reply and returned to the originating task. If an irrecoverable error occurs during a reply (e.g. the slave node being reset) it is simply discarded. The originating task must account for this by setting a time out period.
The standard message format on the fieldnetwork interconnect is show in the next figure:

0	length				
1	MT	SE	DE	TR	reserved
2	node address				
3	source task				dest. task
4	command/reply				
5	DATA				

Where:

length
: Determines the number of message bytes. The header accounts for 7 bytes, including the command or reply byte

MT,SE,DE,TR
: Flags which determine the master extension status of the both the message command and the message replies:
  DE set if destination is extension
  SE set if source is extension
  MS set if message is response, else command

node address
: Unique mode address between 1 and 250

SRC/DST task
: Encodes the source and destination task numbers. The source field allows for up to 16 tasks to generate orders from the master or the master extension. The destination field allows for 16 tasks in the slave or the slave extension to generate replies.

CMD/RESP

Determines the function to be performed by the receiving task and contains the command completion status in a response message. The following codes are reserved in the response message:

00H	S$OK
80H	R$NO$TASK
91H	E$PROTOCOL$ERROR
93H	E$NO$DESTINATION$DEVICE
01H-7H	User defined

DATA Is data specific to the command or reply, up to a total of 248 bytes. Additional rules are defined in the data field on behalf of ES-15 modules

The following additional ES15 rules apply to the messages in the fieldnetwork:

1. A command from the master has the following general lay-out:

	header	
4	periph.typ	command
5	command specific data	

The command field denotes which function has to be performed by the peripheral. Code 0 is reserved for the group of maintenance commands.
The type field denotes the type of controller so that it can be checked by the fieldnetwork node:

1 =	Vehicle Detector
2 =	Variable Message Sign (VMS)
3 =	Axle Weight Detector
4 =	Wind detector
5 =	Visibility detector
6 =	Roadcondition controller
7 =	Brightness detector
8 =	Meteo (temperature + humidity) controller
9 =	Video camera

2. A reply from a slave node has the following lay-out:

	header	
4	command execution status	
5	command + type	} copied from command message
	reply specific data	

The command and the peripheral type enable the command task to verify the corresponding command to this reply.

The command execution status field gives the status from the command performed. The following common error messages exist:

condition	code
E$UNKNOWN$CMD	01H
type specific	05H-7F

3. Maintenance messages are distinguished by a "0" in the command field. The actual order is in the first field from the command specific data:

	header	
4	contr.typ	0
	maintenance comand	
	maintenance data	

A maintenance reply has the following common lay-out:

	header	
4	command ex. status	
5	contr.type	0
6	maintenance command	
	specific data	

6 ES-15 BASIC SYSTEM

The ES-15 Basic System is an empty system from the user point of view, i.e. it contains no other functionality than a run-time environment for the roadside applications. Schematically this can be represented as follows:

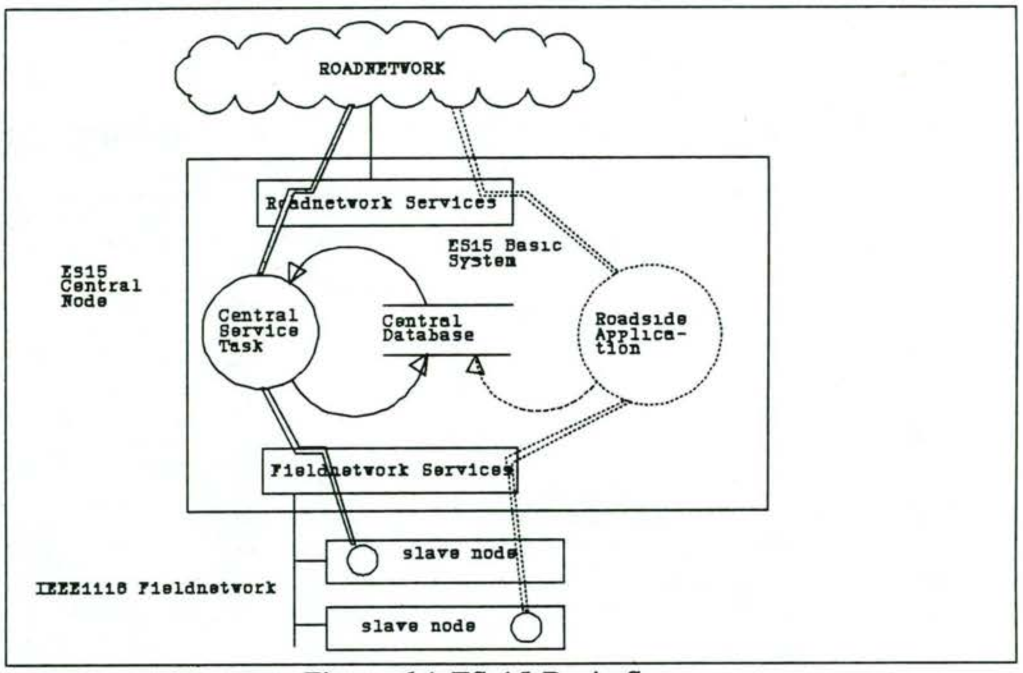


Figure 6.1 ES-15 Basic System

The Basic System contains data communication facilities to the road network, datacommunication services to the fieldnetwork, a central database and it contains a Central Service task (CS-task) for ES-15 management services.

Applications in the master node interface to the roadnetwork services, to the fieldnetwork (when necessary) and to the database. Applications can exchange data via the internal database or via the roadnetwork services.

6.1 Central Database

The central database in the master node performs storage and retrieval functions for a variable number of applications in the Roadside Processor. The system's hardware preserves the database contents at powerfail conditions. The database contains data in a number of categories. Roughly the two main categories are Location Specific Data (LSP) and measurement & control data.

Location Specific Data is static data that describes the ES15 configuration and the local road geometry. Usually this data is inserted once at system installation and thereafter rarely modified.

Measurement & Control data is regularly produced time stamped data, that describes the current road and traffic conditions in the vicinity of the ES15 system.

6.1.1 Data model

The ES15 data model reflects the general system structure of the ES15, see figure 6.2.

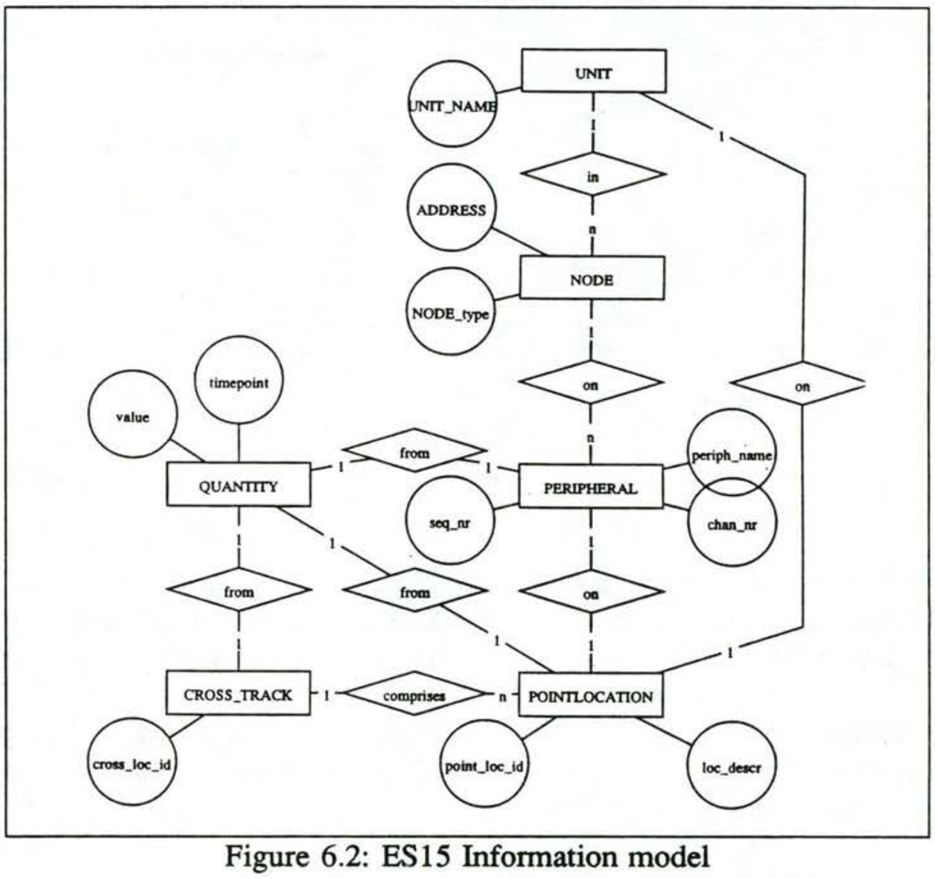


Figure 6.2: ES15 Information model

The following definitions are given to the entities in the datamodel:

Pointlocation

A pointlocation is a logical point in a single lane in a road, relative to a kilometerpoint reference point.

Cross-track

A cross-track is a logical crosssection in a road over a number of lanes with the same direction, relative to a kilometerpoint reference point. A Cross-track can be regarded as an aggregation of a number of pointlocations.

Peripheral

Peripherals are sensors or actuators that act on the traffic flows. Peripherals are installed on a pointlocation from where they measure or influence the traffic. One ore more peripherals are connected to a single node in the ES15 fieldnetwork. Multiple peripherals can be installed on a same location for backup purposes.

Node

An ES15 node is a controller in the fieldnetwork, it is either the master node or one of the slave nodes. The controller nodes perform preprocessing and datacommnunication functions for one or more peripherals

Unit

A unit is a collection of ES15 nodes in a single installation.

Quantity

Quantities are the results of a measurement process on a location during a specific time interval. Quantities are either related to a location or to a peripheral(status data).

The overall conceptual schema of the ES15 central database can be described with the following relational tables:

UNIT(Unit-name,point-loc-id) NODE(Node-address,Node-type,Unit-name) POINTLOCATION(Point-loc-id,location-description CROSS-TRACK(Cross-track-id,Point-loc-id) PERIPH(Node-address,chan-nr,periph_name, point-loc-id,seq_nr)	Managed by Central Service Task
QUANTITYXXXX(Point-loc-id,seq-nr....) QUANTITYYYYY(Node-address,chan-nr,...)	Managed by Application

Where:

Attribute	Description	Data type
Unit-name	= name of ES15 constituent part	char[32]
Node-type	= name recognizable to serviceman	char[6]
Location-description	= string describing a location	Loc_string
Node-address	= network address in fieldnetwork	word
chan-nr	= rank number of peripheral on controller	word
seq-nr	= rank number of peripheral on location (also sequence nr for quantities mea- (sured on a location)	word
Point-loc-id	= binary location ident. of a point loc.	bin_loc_key
Cross-loc_id	= binary location ident. of a cross loc.	bin_loc_key

The schema allows for multiple peripherals of the same type on a single location, resulting in multiple quantities (of the same type) from a single location.

In the ES15 design philosophy there is a solid distinction between the system configuration parameters and the applications that use them. All configuration parameters are input and controlled by the Central Service Task. During System installation they will be input by a service engineer.
Quantities result from applications that control specific peripherals. All location specific parameters for applications are input via the applications.

6.1.2 Database manager

To the applications that interface to the database the database comprises a number of logical databases, where a database is defined as a collection of record occurrences. Depending on the nature of the data the database manager distinguishes a number of databasetypes:

type identifier	description
L_DB_TYPE_LOC_TI_DATA	location time interval data
L_DB_TYPE_PER_TI_DATA	Peripheral time interval data
L_DB_TYPE_TABLE_DATA	Table data
L_DB_TYPE_EVENT_DATA	Time event data
L_DB_TYPE_LSP_DATA	Location Specific Parameter data

Each databasetype has a specific heading that allows for generic ES15 database functions.

In the following the general characteristics of the databases are explained. In the ES15 concepts it is assumed that the central database resides in memory. Therefor all datarelationships are explained in terms of memory datastructures. This does not imply that the database functions are restricted to a memory database. The dataexchange over the roadnetwork is specified such that whatever database technology can be applied.

Time interval data

Time Interval databases are used for the storage of a history of regularly generated time stamped data. The general description of the required storage structure is a circular buffer for a number of measurement points. The header of the total structure contains a number of datafields that allow the Central Service task to treat the datastructure as a collection transparent <interval_data> items.

The total storage is a two dimensional array. Each row in the datastructure contains a header that identifies the data in the row and control variables for the control of a circular buffer in the row. Because the internal allignment of the data can differ from external alignment in network communication buffers, the database contains pointers to conversion routines for the <interval_data> items.

There are two types of databases that may contain quantities either related to a location or to a peripheral. A database of the following structure can contain measurement data of a location:

```
Time_interval_database-name         (
  DB_type=L_DB_TYPE_LOC_TI_DATA                 word,       * database type              *
  DB_rows = max-entries                         word,       * number rows in database *
  DB_interval_size                              word,       * time interval size, can *
                                                            * be 1,5 or 60 minutes      *
  DB_interval_elements = history_length        word,       * number timepoints in db *
  DB_element_size = SIZE OF interval_data word,            * size of appl. spec. data*
  PTR_PACKED_TO                            unsigned long,   * conv. to internal format*
  PTR_PACKED_FROM                          unsigned long,   * conv. to external format*
  ARRAY (max-entries) OF
       RECORD row OF
           location_code                     bin_loc_key,   * location code              *
           seq.nr                               word,      * sequence nr on location *
           young-rec-time                          ES15-time, * youngest recorded time  *
           index-to-youngest                    word,       * index to youngest record*
           ARRAY (history-length) OF
             RECORD interval_data OF                        * record definition          *
               data-fields                                  * specific for application*
             END-RECORD interval_data
       END-RECORD row
  END-ARRAY
                             )
```

The general structure of a database that can store a history of status data for a range of peripherals is as follows:

```
Time_interval_database-name         (
  DB_type=L_DB_TYPE_PER_TI_DATA                 word,       * database type      *
  DB_rows = max-entries                         word,       * number rows in database *
  DB_interval_size                              word,       * time interval size, can *
                                                            * be 1,5 or 60 minutes      *
  DB_interval_elements = history_length        word,       * number timepoints in db *
  DB_element_size = SIZE OF interval_data word,            * size of appl. spec. data*
  PTR_PACKED_TO                            unsigned long,   * conv. to internal format*
  PTR_PACKED_FROM                          unsigned long,   * conv. to external format*
  ARRAY (max-entries) OF
       RECORD row OF
           Node_address                         word,       * fieldnetwork address      *
           channel_nr                           word,       * channel on node            *
           young-rec-time                       ES15-time, * youngest recorded time  *
           index-to-youngest                    word,       * index to youngest record*
           ARRAY (history-length) OF
             RECORD interval_data OF                        * record definition          *
               data-fields                                  * specific for application*
             END-RECORD interval_data
       END-RECORD row
  END-ARRAY
                             )
```

The ES15 database will contain a number of the above structured databases for the storage of specific <interval_data> items. At system startup all ES15 measurement applications allocate empty databases, thus reserving dataspace for the registration of data over a certain period.

Table data

A table is a simple collection of elements, the general database structure can be described as follows:

```
Table_database-name          (
   DB_type=L_DB_TYPE_TABLE_DATA                   word,      * database type            *
   DB_elements = elements_in_table                word,      * number timepoints in db *
   DB_element_size = SIZE OF table_element        word,      * size of appl. spec. data*
   PTR_PACKED_TO                          unsigned long,     * conv. to internal format*
   PTR_PACKED_FROM                        unsigned long,     * conv. to external format*
     ARRAY (elements_in_table) OF
        RECORD table_element
           data-fields                                       * specific for application*
        END-RECORD
                                  )
```

Time Event data

These databases are used for the storage of unregularly generated data in the system. The general description of the required database structure is a one dimensional array of records:

```
Event_database-name          (
   DB_type=L_DB_TYPE_EVENT_DATA char,                        * database type            *
   DB_nr_events = history_length                 word,       * number timepoints in db *
   DB_event_size = SIZE OF event_registration word,          * size of appl. spec. data*
   PTR_PACKED_TO                          unsigned long,     * conv. to internal format*
   PTR_PACKED_FROM                        unsigned long,     * conv. to external format*
   ARRAY (history-length) OF
      RECORD event_registration OF
         event_name                       packed_string[6],* event identification     *
         event_time                          ES15-time,      * time of event occurrence*
         event_data                            char[n],      * specific for application*
      END-RECORD event
                                          )
```

The database provides storage for a limited amount of events, that can be cyclically overwritten. The event-name field allows for a family of events that can be registered. Each event has n bytes available for parameter storage.

Location Specific Parameter data

The general database structure for the storage of parameters is a plain record with no special header:

```
LSP_database-name            (
   DB_type=L_DB_TYPE_LSP_DATA              word,             * database type             *
   LSP_version                             word,             * version number LSP-set    *
   LSP-status                              word,             * verification status       *
   DB_LSP_size = SIZE OF LSP_data          word,             * size of appl. spec. data  *
   PTR_PACKED_TO                   unsigned long,            * conv. to internal format  *
   PTR_PACKED_FROM                 unsigned long,            * conv. to external format  *
      RECORD LSP-data
         LSP-set                                             * set Location Specific
                                                             * Parameters                *
      END-RECORD
                                  )
```

The record provides storage for a set of associated parameters.

6.1.3 Database Access & Integrity

The ES15 database manager provides all access to the data in the databases. In the ES15 design concepts an application can access a database under its type and its name. To the database manager the data in the databases are transparent (apart from the headers). Therefor the database manager has no responsibility for the database integrity.

All data specific to an ES15 is stored in LSP-sets (see database parameter data). To the database manager these sets are transparent records. Only the ES15 applications that use the LSP-set 'understand' the data in it. This is the reason why all Location Specific Parameters are input via the applications that use them.

Applications may also use LSP-sets that have been input via other applications. This results in hierarchical relationships between LSP-sets. A simple example is the configuration data, input via the Central Service Task, that is used by other applications. The ES15 system distinguishes 'dependent' and 'independent' LSP-sets. Again it is the responsibility of each application to watch over the integrity of its own dependent LSP-sets. In order to do that he must be warned when other related LSP-sets are modified. In the ES15 philosophy this is accomplished with the aid of 'modification events'. Each LSP-set has an associated modification event that is raised when the set is modified. ES15 applications can subscribe to these events. After reception of such an event an application should verify the integrity of his own dependent LSP-sets.

The results of a verification pass are stored in the parameter databases in the LSP-status field. This field may contain the following values:

L_VALID	valid set LSPś
L_INVALID	inconsistent set
L_DEFAULT	default values from system startup

6.2 Location Specific Configuration Parameters

The basic system contains seven LSP-sets, that describe the system configuration data and the roadnetwork topology.

The following LSP_database describes the ES15 units:

```
LSP_database_name = LSP_CS_SYS                    * database name                  *
The LSP_data record in the database is defined as:

    RECORD LSP-CS_SYS_data OF
       CS-CU-HOST           packed_string[36],    * name central unit              *
       CS-CU-LOC               bin_loc_key,       * binary location code           *
       FAIL-REPORT-HOST     packed_string[36],    * host system for problem call   *
       T_FAIL_REPEAT           word,              * timeout count for repeat call      *
       CS_N_RS = N-RU-max      word,           * nr remote units space reserved      *
       ARRAY (N-RU-max) OF
          RECORD
             CS-RU-LOC         bin_loc_key,    * location description remote unit *
                                               * IF zero then empty               *
             CS-RU-NAME      packed_string[6]  * name remote system unit          *
          END-RECORD
     END-RECORD LSP-CS-SYS-data
```

The following parameter database contains the set of point-locations where ES15 equipment is installed:

```
LSP_database_name = LSP_CS_PLOC                  * database name                    *
The LSP_data record in the database is defined as:

     RECORD LSP-CS_PLOC_data
       nr-locations = max_locations  word,      * nr locations space reserved for  *
       ARRAY (max_locations)                                *                       *
             RECORD
                 location-id    bin_loc_key,    * location code, if zero than empty*
                 location-descr  LC-string      * location description             *
             END-RECORD
     END-RECORD LSP-CS-PLOC
```

The following database describes the composition of the Cross-tracks from the constituent point-locations:

```
LSP_database_name = LSP_CS_CLOC                  * database name                    *
The LSP_data record in the database is defined as:

     RECORD LSP_CS_CLOC_data OF
       nr-locations = max_cross_tracks  word,  * nr locations space reserved for  *
       ARRAY (max_cross_tracks)                *                                     *
          RECORD
             cross_track-id        bin_loc_key,    * location code, if zero than empty    *
             ARRAY (8) OF
                 location-id    bin_loc_key,       * location code, if zero than empty    *
          END-RECORD
```

The following database describes the collection of controllers in the ES15 fieldnetwork:

```
LSP_database_name = LSP_CS_NODES                 * database name                    *
The LSP_data record in the database is defined as:

     RECORD LSP_CS_NODES_data OF
       nr-nodes = max_nodes             word,      * nr nodes space reserved for      *
       ARRAY (max_nodes)                        *                                   *
          RECORD
             Node_address             word,     * fieldnetwork address, empty if 0 *
             Node_type       packed_string[6],    * type name of peripheral           *
             CS_RU_Name      packed_string[6],    * unit in which node is installed       *
          END-RECORD
```

The following LSP-set describes the collection of peripherals:

```
LSP_database_name = LSP_CS_PERIPH             * database name                    *
The LSP_data record in the database is defined as:

     RECORD LSP_CS_PERIPH_data OF
       nr-periph= max_periph          word,   * nr peripherals pace reserved for *
       ARRAY (max_periph)                     *                                  *
          RECORD
             Periph_name    packed_string[6],    * peripheral type identification     *
             Node_address            word,    * address in fieldnetwork          *
             chan_nr                  word,   * channel nr on fieldnetwork node  *
             periph_loc           bin_loc_key         * peripheral location      *
             loc_seq_nr               word    * rank nr of this peripheral on the  *
                                              * same location                    *
          END-RECORD
```

The following parameter database contains the network data:

```
LSP_database_name = LSP_CS_NHT                * database name                    *
The LSP_data record in the database is defined as:

     RECORD LSP_CS_NHT_data OF
       NHT_ENTRIES = L_MAX_NODES      word,   * nr entries space reserved for    *
       ARRAY (L_MAX_NODES)                    *                                  *
         RECORD
           node_type                 word,    * PSTN or cabling, if zero than empty
                                                        *
           node_address   packed_string[34],  * destination node name            *
           data_link_address packed_string[34],* telephone number if PSTN        *
         END-RECORD
```

6.3 Time Management

The ES15 uses an internal clock for the activation of all time related actions, such as calculation and registration of timeinterval data. The internal ES15 clock can be synchronized with a centralized external clock through broadcasted regular time messages or through ad hoc time commands.
When the newly reveived times differ from the internal system time the ES15 will perform 'Time Difference Actions'. In general the ES15 distinguishes two types of time settings:

Asynchronous (external) time setting event

These are unexpected time settings. The resultings time differences are recognized at the reception of a new time command from an external system, ordering a new systemtime that differs significant from the current systemtime.

Ad hoc time commands can be ordered to the ES15 Central Cervice Task via it's remote control socket. Regular time settings can be ordered also to the ES15 via it's datagram input socket for the reception of time broadcast messages.

Synchronous (internal) time setting event

These are time settings handled by the ES15 itself on preprogrammed timepoints. The

ES15 can be ordered to handle synchronous time differences via the remote control socket of the Central Service Task.

The consequences to the registration of data in the database to both types are briefly discussed in the following:

Asynchronous time differences

In the case of a positive time difference the new time will, depending on the size of the time difference, have passed a number of 1 minute-, 5 minute- or hourly timepoints in the database. For each of these passed intervals the associated time actions will be shortly processed, leading to 'empty' dataitems in the database for the youngest timeintervals. In the case of a negative time difference, again depending on the size of the time gap, a number of 1 minute-, 5 minute or hourly timepoints into the past have been skipped. For each completed time interval the data in the database is wiped, leading to 'empty' dataitems for the oldest timeintervals in the database.

Synchronous time diferences

Only one synchronous timedifference is required; the one that deals with the transition of wintertime to summertime. On that transition the clock is set back for 1 hour on a clearly defined timepoint. In order not to throw away data for the skipped time interval a special integration time is processed. The ES15 is notified in advance that at a certain time the internal clock must be set back for 1 hour.

6.4 Central Service Task

The master node contains a specific application for control and maintenance of the whole system. This Central Service Task performs the following functionality:
- Input and checking of a basic LSP set, comprising all sort of configuration parameters
- Backup and restore of all LSP's in the database
- Periodic checking of connected parts in the system and logging of failure messages in case of problems
- Relay of (transparent) maintenance commands to peripherals in the fieldnetwork
- Time settings
- Event handling for synchronization between applications

In the next paragraph the operating procedures for remote control are explained. These procedures are common to all remote IO between ES15 applications. In next paragraphs the specific commands for the input of parameters and for database access are explained.

6.4.1 Operating procedures

The concept of the datacommunication socket allows peer to peer communication between two applications. The central service task is reachable for command handling via the socket name:

"CS_SERVICE"

The general operating procedures distinct a clear master slave relationship between the two applications. The master outputs command messages and the slave outputs reply messages. The information exchange is handled in a half duplex handshake in which one command must be followed by a single answer from the slave. Both command and answer however can comprise multiple network messages. The following diagrams illustrate the handshake for a single-buffer command and a multi-buffer reply:

```
MASTER application
CONNECT to remote socket
    SEND command message
    RECEIVE reply message
        Y \  last message?  / N
            | RECEIVE REPLY message
    UNTIL final message received
UNTIL session ended
DISCONNECT from socket
```

```
SLAVE application
DO FOR EVER
    WAIT for connect request
    ACCEPT request
        RECEIVE command message
        SEND first reply message
            N \send more replies? / Y
            %  | SEND next reply
        UNTIL all replies transmitt.
    UNTIL disconnected
```

6.4.2 Command Dispatching

By default all commands from the master application are given in ASCII coded buffers. The ES-15 reply is by default binary coded, due to reasons of networkload and the ease of processing in the master application.

Command format

A command to a ES-15 application consists of a string of ASCII characters in a special form or a binary coded buffer starting with a zero field:

```
| CC  arg1, arg2, ..., argn |   or:  | 0 | command + arguments |
```

ASCII command

If the first byte contains a non-zero value then the buffer contains ASCII characters that have to be interpreted as follows:

CC	The first characters until a separation sign will be considered as a command. A command is terminated by a space or end-of-buffer character
arg	Is an argument for the command and can be a string, a NULL argument or a decimal number: . decimal number = 'ddd.dd.' eventually preceded by '+' or '-' sign . string ' 'ssssssss' . NULL = '*' or empty

Arguments are separated by comma characters. The end of an argument is indicated by a comma or the message end. The meaning of an argument is determined by it's specific location within the command. Each command expects it's arguments on a prescribed place. A NULL argument has no related value. In the command syntax this is represented by square brackets as follows:

```
CC  arg1,[arg2],arg3          or:   CC arg1,[arg2],[arg3]
```

Binary command

The general format of a binary command has the following layout:

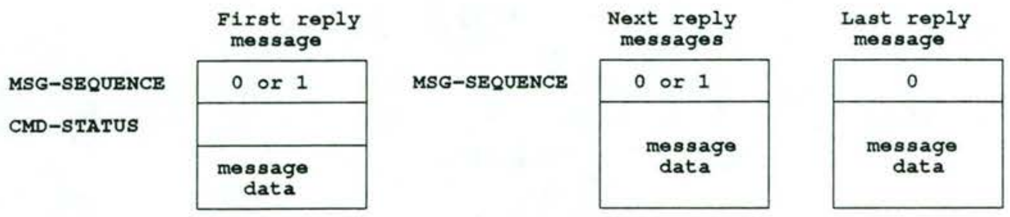
fieldname	contents	datatype	description
bin_command	L_BIN_COMMAND	char	identifier binary command
bin_command_id		char	command identification
cmd_sequence		char	buffer sequence control
	data		command parameters

If the first byte in a command has a zero value (L_BIN_COMMAND) then the remaining part of the buffer has to be considered as a binary command. The second byte in the buffer identifies the command. The third byte in the buffer identifies the number of messages in the command:

```
CMD-SEQUENCE      = 0 = L_LAST_BLOCK        final message, no more to come
                  = 1 = L_MORE_BLOCKS       more messages will follow
```

Generic form of an application reply

A reply from a ES-15 application can consists of a number of buffers of the following general format:



The first byte in every reply message contains information on the buffer sequence:

MSG-SEQUENCE	= 0 = L_LAST_BLOCK	final message, no more to come
	= 1 = L_MORE_BLOCKS	more messages will follow

The second byte of the first message contains the status of the executed command. The following codes are used in the second byte of the first buffer:

CMS-STATUS	L_S_OK	command successfully executed
	L_E_UNKNOWN_CMD	error, unknown command

6.4.3 READ database data

The Central service task has a number of commands available for the retrieval of database data in a generic format. The commands make use of the type and name identifiers in the databases. The commands are both available in an ASCII and in a binary format.

The data is provided in transparent records, meaning that the external user must be familiar with the internal formats of the records.

The following commands are available:

- Read_db_loc_interval_data	reads a history of time interval data items for a location
- Read_db_per_interval_data	reads a history of time interval data items for a peripheral
- Read_db_event_data	reads a series of events from the database
- Read_db_table_data	reads a table from the database
- Read_db_LSP_set	reads a LSP-set from the database

Read_db_loc_interval_data **LI Db-name,[location],[seqnr],[timepoint],history**

Reads a history of location <data-items> from a named time interval database, where:

DB-name	Contains the name of a database
Location	Is a decimal identifier for a location. A default value is interpreted as all locations in the database.
Seqnr	Is a decimal sequence nr on a location. A default value is interpreted as a 1.
timepoint	specifies date and time in the following format:

	dd.mmm.yy hh:mm or hh:mm The default day is today and the default time is closest to current time
history	is the number of time points, while the interval between the time points depend on the parameter

The binary variant of the command has the following layout:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_CMD_R_L_IVD	char	command identification
cmd_sequence	L_LAST_BLOCK	char	buffer sequence
Database_name		DB_name	name LSP-set
location		bin_loc_key	location ident
Seq_nr		word	sequence nr on location
DB_time		ES15_time	first timepoint
history_length		word	number requested timepoints

The answer to this command wil be given in a series of reply messages that reflect the structure of a time interval database. Every message can contain a series of <interval_data> records, each preceded by its time. The header of the message contains the identifier and the length of the <interval_data> record:

fieldname	contents	datatype	description
MSG-SEQUENCE	0 or 1	char	message sequence control
CMD_STATUS	L_S_OK	char	command completion status, first message
DB_element_size	elementsize=n	word,	length of interval data item
location		bin_loc_key	location ident
Seq_nr		word	sequence nr on location
	timepoint	ES15_time	first timepoint in history
	data record	char[n]	data on timepoint
	timepoint	ES15_time	last timepoint in history
	data record	char[n]	data on timepoint

Every message contains a data history of a single identifier. When data of multiple identifiers was requested multiple messages of the above structure will be returned.
The number of reply messages will reflect the number of identifiers requested.

Read_db_per_interval_data PI Db-name,[address],[chan],[timepoint],history

Reads a history of peripheral status <data-items> from a named time interval database, where:

Address	Is a decimal identifier for an adress in the fieldnetwork. A default value is interpreted as all nodes in the database.
chan	Is a decimal channel nr on a node. A default value is interpreted as a 1.

Read_db_table_data RT Db_name[,element]

Reads elements from a named table database, where:

element	Is a decimal ranknumber in the table, a default value is interpreted as all elements.

The binary variant of the command has the following layout:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_CMD_READ-TD	char	command identification
cmd_sequence	L_LAST_BLOCK	char	buffer sequence
Database_name		DB_name	name table
element	n	word	nth element in table

The reply is given in a single reply message that reflects the structure of a time table database:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_BLOCK	char	message sequence control
CMD_STATUS	L_S_OK	char	command completion status
DB_element_size		word,	size of table element
	element data	char[n]	table element data
	element data	char[n]	

Read_db_event_data RE Db_name,[event],[timepoint-from],[timepoint-until]

Reads events from a named event database. Where:

Eevent	name of event, default is all events
Timepoint-from	Specifies a begin timepoint in the database. A default value means the begin timepoint in the database.
Timepoint-unit	Specifies an end timepoint in the database. A default value means the last timepoint in the database.

The binary variant of the command has the following layout:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_CMD_READ-ED	char	command identification
cmd_sequence	L_LAST_BLOCK	char	buffer sequence
Database_name		DB_name	name LSP-set
Event_name		packed_string[6]	event ident, zero is default all
DB_time_from		ES15_time	first timepoint
DB_time_until		ES15_time	last timepoint

The reply is given in a single reply messages that reflects the structure of a time event database. The reply message contains one or more events:

fieldname	contents	datatype	description
MSG-SEQUENCE	0 or 1	char	message sequence control
CMD_STATUS	L_S_OK	char	command completion status
DB_event_size	eventsize = n	word,	length of interval data item
	timepoint	ES15_time	timepoint first event
	event data	char[n]	event data
	=======================		
	timepoint	ES15_time	timepoint last event
	event data	char[n]	

READ-LSP-Set RS db_name

This command reads a named LSP-set from the central database. The binary layout of the command message is:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_CMD_READ-LSP	char	command identification
cmd_sequence	L_LAST_BLOCK	char	buffer sequence
LSP_name		DB_name	name LSP-set

The reply comes in a single buffer of the following layout:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_MORE_BLOCKS	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status
LSP_VERSION		word	version number LSP-set
LSP_size		word	Size of LSP-set record
LSP_STATUS		word	LSP verification status
LSP_data		char[n]	LSP set data

Notice that the reply of this command also gives the LSP verification status.

Error Messages

The above commands can give the following errorcodes:

L_S_OK	command successfully executed
L_E_DB_UNKNOWN	database not known
L_E_IDENT_UNKNOWN	requested ident not in the database
L_E_TIME	time outside the database limits

6.4.4 Input of Location Specific Parameters

The following command is available for the installation of a LSP-set:

Change-LSP-Set

This binary command loads or changes a LSP-set in the ES15 central database. Before the LSP is actually loaded he is verified first. The layout of the command message is as follows:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_CMD_CH_LSP	char	command identification
cmd_sequence	L_LAST_BLOCK	char	buffer sequence
LSP_name		DB_name	name LSP-set
LSP_VERSION		word	version number LSP-set
LSP_size		word	size of LSP dataset
LSP_data		char[n]	data LSP-set

The reply to this command is a simple message with only the return status of the command, indicating the success or failure of the command.

The command can return the following error codes:

L_S_OK	Parameters successfully loaded
L_E_CONS_ERROR	inconsistent set of parameters
L_E_INTEGR_ERROR	set not consistent with other parameters

6.4.5 Database Backup and Restore

In the design philosophy of the ES15 the applications are responsible for the input of location specific data and for the integrity of the Central Database. To allow for simple system management the ES15 Central Service task can make a full backup of all Location Specific Parameters in a transparent manner. In this way a remote Service Unit, at a central site, can maintain a full backup of all Location Specific Parameters of all ES15 systems in the network.

Via the same central Service Unit all LSP parameters can be restored in an ES15 system, for instance after replacement of processor equipment. The ES15 restore procedure replaces a transparent data set into the databases. In order to avoid an inconsistent database in the ES15 all applications are warned to verify their LSP-sets before actually using them.

First the available commands are briefly introduced, lateron they are described in detail:

Backup_parameters

Query of all LSP sets in the system. The parameters are collected from the database and transferred in a series of reply messages.

Restore_parameters

Start of a IO sequence for restore of the complete collection of LSP sets. All LSP sets from the backup are restored in the central ES15 database. The Central Service Task performs a handshake for each LSP set from the master application.

After the restore operation a 'Restore Event' is raised in the system. This causes all applications to read their LSP's from the database and to verify their consistency.

Read_verification

Reads the verification results from the various LSP service tasks in the system.

In the following the details of the above commands are described.

Backup-parameters

The layout of the backup command is:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_CMD_BACKUP	char	command identification
cmd_sequence	L_LAST_BLOCK	char	buffer sequence

The first reply message returns the command status and gives administrative backup information:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_MORE_BLOCKS	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status
BP_VERSION		char	software version of the backup
TIME		ES15-time	current ES15 time
CS_CU_HOST		char[33]	ES15 name

The following error messages may be returned to this command:
.... tbs.......

After transmission of a first OK message the next reply messages each contain a single LSP set until all LSP databases have been copied to the host application. Each of those messages will have the following general layout:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_MORE_BLOCKS	char	identifier binary command
LSP_NAME	L_S_OK	DB_name	name of the LSP-section
LSP_VERSION		word	version number LSP-set
LSP_data		char[n]	LSP set data

Restore Parameters

The Restore Command starts a reading cycle in the central service task in which a series of LSP databases is read from the network. The layout of the command is:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_CMD_RESTORE	char	command identification
cmd_sequence	L_LAST_BLOCK	char	buffer sequence
BP_VERSION		char	version of the service task that created the backup

The following error messages may be returned to this command:

L_E_RESTORE_BUSY	restore already busy

The OK reply to the restore command is of the following contents:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_MORE_BLOCKS	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status

After transmission of the OK reply the ES15 initializes internal buffer space and waits for messages with LSP data. The command allows for the transmission of a burst of LSP-set. Each of the sets is of the following general format:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_RESTORE_NEXT	char	command identification
cmd_sequence		char	buffer sequence
LSP_NAME		DB_name	name of the LSP-set
LSP_VERSION		word	version number LSP-set
LSP_data		char[n]	LSP set data

The reception of the burst of LSP-data message will be acknowledged by the Central Service task in the following reply message:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_DATA	char	identifier binary command
RESTORE_STATUS		char	command reception status
LSP_name	name	DB_name	name of LSP_set

The RESTORE_STATUS field can have the following codes:

L_S_OK	data successfully received
L_E_LENGTH_ERROR	length mismatch
L_E_VERSION_ERROR	version mismatch
L_E_INVALID	unknown LSP-set
L_E_NO_REST_ACT	restore is not active

When the ES15 has reported an error after reception of an LSP-set the command sequence is ended. The LSP_name field in the reply contains the name for which the error was encountered. No changes have been made in the central database than. When an OK status is returned it means that the ES15 is awaiting a next LSP-set.

After the last LSP-set has been transmitted from the host system a final message is ordered to activate all transmitted LSP-sets:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_RESTORE_ACT	char	command identification

After reception of this command the ES15 Central Service Task will actually replace the contents of the database with the newly received LSP-sets and will notify all service applications of the database restore. The final reply message to the host system will be:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_DATA	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status

The following error status can be reported in the final reply:
..... tbs ......

Read LSP verification

After a database restore, all applications in the system read their LSP-sets from the database and check their consistency. The verification status in the LSP-database shows the result of this verification.

A verification overview can be obtained after a restore operation (the status of the restore operation only replies successful restore). The format of the command message is as follows:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_CMD_VERIFY	char	command identification
cmd_sequence	L_LAST_BLOCK	char	buffer sequence

The reply buffer is filled with status information of the various LSP databases:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_BLOCK	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status
LSP_NAME		DB_name	name LSP-set
LSP_STATUS		word	LSP verification status
		DB_name	name LSP-set
		word	LSP verification status

The following errorcodes can be returned to this command:
.... tbs......

6.4.6 Time Settings

The following two commands are available to read and change the current system time:

Read_date_time

Retrieval of date and time. The service application returns the ES-15 system time. The command is of the following format:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_READ_TIM	char	command identification

The final reply message to the host system will be:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_DATA	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status
TIME	L_S_OK	ES15_tim	current ES15 time

Set_date_time

Changes ES-15 system time. All other applications are notified by issuing a time change event. The comand is of the following format:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_SET_TIM	char	command identification
time		ES15_time	new time

The reply to the host system will be:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_DATA	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status

The following commands are available for the programming of a synchronous time setting event:

Programme_time_setting

Orders the ES15 to change its time setting at the specified date & time over a period of <time_difference> seconds. The command is of the following format:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_PROG_TIM	char	command identification
time		Es15_time	timepoint
time_diff		word	time difference

The reply message to the host system will be:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_DATA	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status

Cancel_programmed_time_setting

Cancels the programmed time setting, the command is of the following format:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_CAN_PROG_TIM	char	command identification

The reply message to the host system will be:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_DATA	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status

6.4.7 Maintenance Commands

By means of the next command, it is possible to request a summary of the ES-15 configuration:

Configuration_overview

This command gives an overview of the system configuration and the available applications. The command is of the following format:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_CON_OVERVIEW	char	command identification

The reply message to the host system will be:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_DATA	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status
socket_name		packed_string[32]	socket name
application		packed_string[32]	aplication name
socket_name		packed_string[32]	socket name
application		packed_string[32]	aplication name

Software_version

This command identifies the version of the ES15 basic system. The command is of the following format:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_SOFTW_VER	char	command identification

The reply message to the host system will be:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_DATA	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status
version		word	software version

Transparent Maintenance commands to Fieldnetwork Peripherals

Using this command a service engineer can request service information from a selected peripheral in the fieldnetwork. The format of the command message is as follows:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_CMD_FIELD_MC	char	command identification
cmd_sequence	L_LAST_BLOCK	char	buffer sequence
node_address	address	char	controller address in fieldnetwork
node_type	type	char	controller type
	data	char[n]	transparent maintenance command

The data fields of the message must contain the maintenance data of the command as described in chapter 5 (exclusive the bytes 0 to 8). The reply buffer contains the full reply buffer of the selected fieldnetwork controller (bytes 5 to end):

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_BLOCK	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status
	data	char[n]	maintenance reply

The following codes can be returned by this command:

Erase log table

The system uses internal a logtable for the output of problem messages. The table prevents problem messages to be output for problems already notified. The above maintenance command can be used to wipe the entire table after a repair action. All problems that still exist in the system will be immediately reported. The command is of the following format:

fieldname	contents	datatype	description
bin_command	0	char	identifier binary command
bin_command_id	L_ERASE_ERRLOG	char	command identification

The reply message to the host system will be:

fieldname	contents	datatype	description
MSG-SEQUENCE	L_LAST_DATA	char	identifier binary command
CMD_STATUS	L_S_OK	char	command completion status

6.5 Network Broadcasts

Network broadcasting is used for time synchronization purposes of all ES15 systems connected to a common cabling network. The mechanism is open ended so it can be used in the future to other purposes as well.

All datagram messages that are broadcasted over the Roadnetwork have the following general layout:

fieldname	contents	datatype	description
Message-id		char	message identification
	message data		

The time synchronisation message is of the following specific form:

fieldname	contents	datatype	description
Message-id	L_DG_TIME_SYNC	char	identification time sync message
Current_time	time	ES15-time	current time

6.6 Problem Calls

A problem call is issued by the ES-15 to a remote system with the name FAIL_REPORT_HOST (subset of the basic LSP-set). First a temporary communication link is build up with the socket FAIL_REPORT_SOCKET in a remote system that contains a server application. Afterwards the link is closed again. Failure messages can be repeated after a preset time.

The format of a failure message is as follows:

fieldname	contents	datatype	description
CS_CU_HOST		char[33]	ES15 name
CU_LOC		bin_loc_key	locationcode central unit
TIME		ES15-time	call time
RU_name		packed_string[6]	unit name, empty for central unit
RU_LOC		bin_loc_key	locationcode remote unit
Node_address		word	node address in fieldnetwork, 0 for central unit
Chan_id		word	channel on node
FAIL_CODE		word	reported problem code
T_FIRST_FAIL		ES15-time	Time that failure occurred for the first time
T_LAST_FAIL		ES15-time	Time that failure occurred for the last time
N_FAIL		word	Number of failures since last message was send

7 ROADSIDE APPLICATIONS

7.1 Statistical Row Traffic Datacollection

Row Traffic Data collection is based on detection of single vehicles, with the aid of vehicle detectors in a lane. This paragraph describes the ES15 internal databases that contain statistical traffic data. The data in these databases can be requested by external systems with the aid of the generic database commands in the ES15 Central Service Task.

Three different databases, each of the type Time_Interval_data, with names TID_1_min_traffic, TID_5_min_traffic and TID_60_minute_traffic contain respectively lane traffic data over 1 minute intervals, 5 minute intervals and 60 minute intervals for a range of locations.

The data record in the 1 minute database is of the following structure:

```
Name:      TID_1_min_Traffic                      % database name                      %
Type:      L_DB_TYPE_LOC_TI_data                  % database type                      %

RECORD 1_minute_traffic_interval_data
  interval_length           word,                  % length of observation interval  %
  duration_I_per            word,                 % duration of observ. period for flow %
  duration_V_per            word,
  N_OCC_MESS                word,                 % number detector failures           %
  SUM_T_OCC                 word,                 % number unclassified vehicles       %
  N_REV_DIR                 word,                 % number vehicles opposite direction  %
  occ_time                  word,                 % occupancy time                     %
vehicle_category_data:
  nr_cat                    char = ncat,          % nr categories                      %
  ARRAY ncat OF
    RECORD category_traffic_data OF
           vehicle_cat      char,                 % vehicle category                   %
           flow_in_cat      dword,                % flow                               %
           sum_speed        dword,                % sum speed in category              %
           sum_sq_speed     dword,                % sum square speed                   %
           V_mean           word,                 % avaraged speed                     %
           V_sdev           word,                 % standard deviation speed           %
           Time-gap         word,                 % averaged time between vehicles      %
END RECORD
```

The records for the 5 minute- and 60 minute databases are of the same structure.

7.2 Statistical Roadcondition Datacollection

Statististical Roadcondition Datacollection is based on the measurement of various meteorological- and roadsurface conditions in and along the road. The measurement results are stored in a number of different databases from where they can be requested with the aid of the generic database commands in the Central Service Task.

A database of the following characteristics contains statistical wind information over 5 minute intervals for a single windspeed/winddirection sensor over a period of 24 hours:

```
Name:      TID_5_min_wind                      % database name                      %
Type:      L_DB_TYPE_LOC_TI_data               % database type                      %

RECORD 5_minute_wind_interval_data OF
  interval_length            word,              % length of observation interval   %
  duration_w_speed           word,             % duration of observation period    %
  duration_w_direction       word,
  W_direction                word,             % averaged wind direction           %
  W_speed                    word,             % avaraged wind speed               %
  duration_w_gust            word,             % duration of wind gust             %
  W_gust                     word              % max 2-sec gust within period      %
END_RECORD
```

A database of the following characteristics contains visibility data over 5 minute intervals for a single visibility sensor:

```
Name:      TID_5_min_visibility                % database name                      %
Type:      L_DB_TYPE_LOC_TI_data               % database type                      %

RECORD 5_minute_wind_interval_data OF
  interval_length            word,              % length of observation interval   %
  duration_visibility        word,             % duration of observation period    %
  5_min_visibility           word,             % 5 minute averaged visibility      %
  minimum_visibility         word,             % minimum 12 sec visibility in 5 min  %
                                                 % period                          %
END RECORD
```

The following database contains temperature information over 5 minute intervals for a range of temperature sensors:

```
Name:      TID_5_min_temperature               % database name                      %
Type:      L_DB_TYPE_LOC_TI_data               % database type                      %

RECORD 5_minute_temp_interval_data OF
  interval_length            word,              % length of observation interval   %
  duration_temp              word,             % duration of observation period    %
  5_min_temperature          word,             % averaged temperature              %
END_RECORD
```

The following database contains roadcondition information over 5 minute intervals for a range of roadcondition detectors:

```
Name:      TID_5_min_roadcondition             % database name                      %
Type:      L_DB_TYPE_LOC_TI_data               % database type                      %

RECORD 5_minute_temp_interval_data OF
  interval_length            word,              % length of observation interval   %
  duration_roadcond          word,             % duration of observation period    %
  duration_dry                 word,           % period dry surface                %
  duration_wet               word,             % period wet surface                %
  duration_icy               word,             % period icy surface                %
  duration_salty             word,             % period salty surface               %
  delta_freezing_temp        word,             % delta in freezing temperature     %
  duration_freezing_temp     word              % duration observation period fr.temp.%
END RECORD
```

The following database contains precipitation information over 5 minute intervals for a single precipitation detector and a single relative humidity sensor:

```
Name:       TID_5_min_precipitation               % database name                          %
Type:       L_DB_TYPE_LOC_TI_data                 % database type                          %

RECORD 5_minute_precipitation_interval_data OF
  interval_length           word,                  % length of observation interval  %
  duration_precipitation    word,                 % duration of observation period  %
  5_min_precipitation       word                  % 5-min avaraged precipitation       %
  duration_humidity         word,                 % duration of observation period  %
  5_min_rel_humidity        word                  % 5-min avaraged humidity        %
END_RECORD
```

The following optional database contains radiation information over 5 minute intervals for a single radiation detector:

```
Name:       TID_5_min_radiation                   % database name                          %
Type:       L_DB_type_one_TI_data                 % database type                          %

RECORD 5_minute_radiation_interval_data OF
  interval_length           word,                  % length of observation interval  %
  duration_radiation        word,                 % duration of observation period  %
  5_min_radiation           word                  % 5-min avaraged radiation           %
END_RECORD
```

7.3 Roadside Monitoring

7.3.1 General Principles

Because of its generic nature the Monitoring application can be viewed as a generic part of the basic system. Other ES15 applications can interface to it for the transport of data to higher order systems. However in practice Monitoring is limited to ES15 systems that can maintain a permanent virtual circuit for communication.

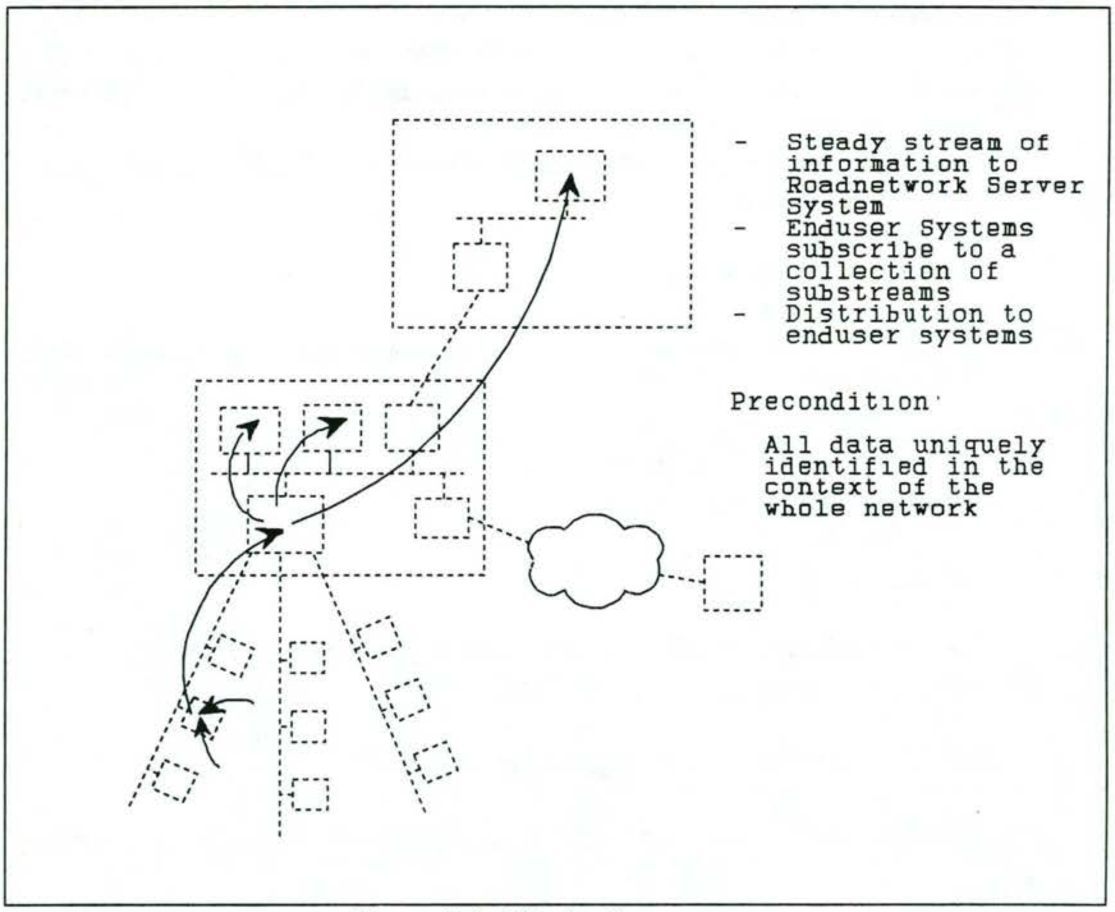

- Steady stream of information to Roadnetwork Server System
- Enduser Systems subscribe to a collection of substreams
- Distribution to enduser systems
Precondition:
All data uniquely identified in the context of the whole network

Figure 7.1: Monitoring concept

From a network point of view ES15 Monitoring is based on a contineous stream of unsolicited information through a permanent virtual circuit. At system startup and each time after a network failure the "Central ES15 Monitoring" will establish a virtual circuit to a remote "Monitoring Server" system. This permanent link is used as a common pipeline for all ES15 applications that interface to the Monitoring network. A standard header in front of the data identifies the type and nature of the data. This header information enables a distributor function in the Monitoring Server system to copy subsets of the data into multiple output streams to subscribed enduser systems. Figure 7.1 illustrates the Monitoring concept.

Besides the contineous datastream also an other datastream is maintained in Monitoring. In general two types of data transfer can be distinguished, that each use a separate private circuit for communication:

- State data messages
  These datamessages describe the current state. Regular measurement data can be transmitted, but also ad hoc status data.

- alert messages
  These are special messages that are transmitted at ad hoc timepoints. The transmission of these special messages has priority over all other data transmission.

In the following paragraphs the general principles are further explained and the data messages are detailed described.

7.3.2 Location Specific Parameters

The following LSP_database contains the location Specific Parameters for the ES15 Monitoring functionality:

```
LSP_database_name = LSP_MO                        * database name                     *
The LSP_data record in the database is defined as:

    RECORD LSP-MO_data OF
       MO-SERVER             packed_string[36], * name distribution host            *
       MO-MASK,                         word    * mask for suppression of data      *
                                                * types                             *
     END-RECORD LSP-MO-data
```

The above described database identifies the remote Information Server system to which a permanent virtual circuit is maintained.

7.3.3 Remote Operating Procedures

The ES15 Monitoring functionality can be remote controlled via the socket name:
"MO_SERVICE"

After establishment of a virtual circuit the following commands are available for the installation and verification of Location Specific Parameters for the Monitoring application:
- Change_LSP_Set
- Read_LSP_Set
See the description of the ES15 basic system for a detailed description of these commands.

7.3.4 State data messages

For the transfer of this data a permanent virtual cicuit is maintained to a remote MO_SERVER system. This link is used for the transmission of a steady stream of data messages from multiple applications in the ES15.

Roughly the Monitoring State Data messages can be discriminated in regular measurement data and ad hoc information messages. Each application can have several regular messages and ad hoc messages. A Message-id field in front of the data identifies the application category and the data in the message:

```
fieldname        contents        datatype        description

Message-id    |             |    char          message identification
              |-------------|
              | message data|
```

The contents of the Message-id field is organized as follows:

```
|------|---------|----------|
 \________/\______________/
    |        |____________ specific message
    |_____________________ type of data:
                           0 = row traffic monitoring data
                           1 = road condition monitoring data
                           2-6 spare
                           7 = location id descriptions
```

The above definition allows for flexible subscription to a mix of data by enduser systems at the higher level. It also allows for easy masking off of data messages in the ES15. See the relevant application descriptions for detailed information on the state data messages.

Table 7.1 gives an overview of the specific ES15 Monitoring Sate data messages:

Type of Data		Specific Message		nature
0	Row Traffic Monitoring data	1	Lane_traffic_state_data	each minute
		2	Crosstrack_traffic_state_data	each minute
1	Road Condition Monitoring data			
7	Location ID descriptions	1	Point Location description	
		2	Cross-track description	

Table 7.1: Monitoring Sate Data Messages

7.3.4.1 Location description messages

In the monitoring network all measurement data will be regularly transmitted with the short location codes as location reference identifications, for reasons of efficiency. As a rule each application will generate its own messages with location descriptions. In practice this will result in multiple transmissions of location-id messages when peripherals are installed on same locations. The prime reason is however that it prevents the transmission of general messages that can be confusing to end-user systems.

7.3.4.2 Row Traffic Monitoring data

Row trafic Monitoring is based on the detection of single vehicles in a lane.

Lane_traffic_state_data

Lane_traffic_state_data messages describe the traffic state in a lane. The format is the message is as follows:

fieldname	contents	datatype	description
Message_id	L_RTM_LTSD	char	message identification
location_code		bin_loc_key	locationcode measurement point
date_&_time		ES15-time	date&time observation period
period_length		word	length of observation interval in seconds
Statusdata:			
duration_I_per		word	duration of obervation period for flow Q in seconds
duration_V_per		word	duration of observation period for for V in seconds
det_failures		word	count of detector alarms
trafficdata:			
flow_unclass		word	count unclassified vehicles
occ_time		word	occupancy time
vehicle_category_data:			
nr_cat	ncat	char	number vehicle categories
cat_data		L_LTSD_CD[ncat]	records with category data

The L_LTSD_CD records with category data are of the following format:

fieldname	contents	datatype	description
vehicle_cat		char	category identification
flow_in_cat		word	count of vehicles in category
sum_speed		signed long	sum speed in category
sum_sq_speed		signed long	sum square speed in category

Crosstrack_traffic_state_data

The crosstrack_traffic_state_data messages are of the following format:

fieldname	contents	datatype	description
Message_id	L_RTM_CTSD	char	message identification
location_code		bin_loc_key	locationcode crosstrack
date_&_time		ES15-time	date&time observation period
period_length		word	length of observation period in seconds
Statusdata:			
duration_per		word	duration of obervation period in seconds
det_failures		word	count of detector alarms
traffic_state_data:			
capacity		word	road capacity
cong_status		byte	congestion status
unrest_est		word	unrest estimator

7.3.4.3 Alert Data Messages

..... to be specified ....

8 FIELDNETWORK PERIPHERALS

This chapter gives a detailed description of the datacommunication over the fieldnetwork. For each fieldnetwork node we give a general description of the functionality and a detailed description of the communication involved.

8.1 Variable Message Sign

The function of a Variable Message Sign (VMS) is to display readable information to the road user. Several VMS units are linked to a VMS controller, which itself is linked to the fieldnetwork. In a schematic way this can be represented as follows:

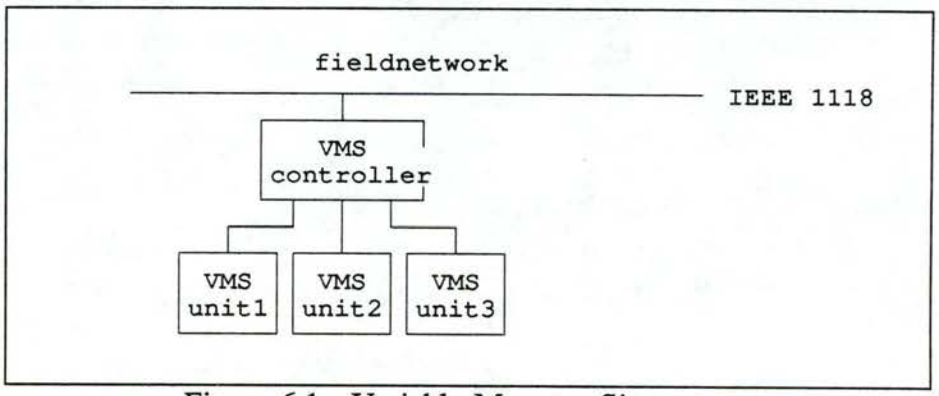


Figure 6.1 : Variable Message Sign set-up

A VMS controller has 3 tasks. Each one is able to handle one or more commands via the network:

1. A control task continuously checks the function of he connected VMS units. If a variation occurs a status message is queued to the network.
2. A service task can perform maintenance commands and reply to requests for current status information
3. A display task can perform commands to display a certain pictogram on a VMS unit

The communication with the fieldnetwork consists the following messages for a VMS controller:

id.	command	task	response
0	Maintenance Command	2	Maintenance reply
1	Display Pictogram	3	Display Status reply
2	Set lightness	3	lightness status reply
3	Pictogram Status request	2	Pictogram Status reply
4	VMS status request	1	Asynchronous status reply
			Dummy Status Reply
5	Request Configuration	2	Configuration Information

The three VMS controller tasks can each simultaneously handle one specific command. In normal operation mode, the control task continuously has one VMS Status Request active. As soon as a status change occurs in one of the VMS-units, it enables the controller to report this event to the master task via an Asynchronous Status Reply. This reply is issued via a FIFO buffer. If on that moment no VMS Status Request is handled, then the reply stays in the FIFO until a next request is received. If on that moment a status request is handled, then the reply is issued directly via the FIFO to the network. If the control task is handling a VMS Status Request and within 1 minute no status change occurs, then it sends back a Dummy Status Reply. This enables the master process to check the proper working of the controller via a time-out mechanism.

The service task and the display task send a reply as soon as a command has been completed. In incidental cases some delay can occur, e.g. if the status is requested while at the same time a new pictogram is displayed.

A VMS unit can be constructed out of specific technology. The distinctions occurring are solely visible in the set of maintenance commands.

In the next paragraph each of the commands will be described in detail. At the end the maintenance command will be described for a fibre optic VMS.

Display Pictogram command

In the Display Pictogram command the master orders the VMS display task to set a specific pictogram on one or more VMS units. If for one of the VMS units the display cannot be executed the whole command must be rejected.

3		task = 3
4	Type=2	Command=1
5	VMS-number (0, or 1..255)	
6	pictogram (0-255)	
7	VMS-number (1..255)	
8	pictogram (0-255)	
	etcetera	

Display Pictogram order message

where:

VMS-number = 1..255 for specific VMS
= 0 for complete group

pictogram = 0 VMS off (blink light included)
= 1 not defined
= 2 end of all prohibitions
= 3 speed limit 30 km/h
= 4 speed limit 40 km/h
= 5 speed limit 50 km/h
= 6 speed limit 60 km/h
= 7 speed limit 70 km/h
= 8 speed limit 80 km/h
= 9 speed limit 90 km/h
= 10 speed limit 100 km/h
= 11 speed limit 110 km/h
= 12 speed limit 120 km/h
= 13 - 19 not defined
= 20 warning sign
= 21 road works
= 22 no overtaking
= 23 end of no overtaking
= 24 no overtaking (trucks)
= 25 end of no overtaking (trucks)
= 26 - 29 = not defined
= 30 congestion
= 31 accident
= 32 fog
= 33 ice
= 34 keep lane (arrow down)
= 35 keep lane (arrow up)
= 36 switch lanes (arrow pointing to the left)
= 37 switch lanes (arrow pointing to the right)
= 38 lane blocked (cross)

= 39 - 49 not defined
= 50 blink light off
= 51 blink light on

Display status reply:

The display task reports the status of an executed Display Pictogram command in a Display Status Reply. Before the display task sets the pictograms it performs a command check, based on the internal status. The command Status field contains the general status of the full command. The data fieldd from the reply contains the same set of VMS-unit/pictogram combinations as the command buffer, each succeeded by a specific status denotation for the combination.

3		task = 3
4	Command status	
5	Type = 2	Command = 1
6	VMS-number	
7	Sign acknowledge	
8	VMS-number	
9	Sign acknowledge	
	etcetera	

```
Display Status
   reply message
```

where:

Command status	= S$OK
	= E$WARNING
	= E$ERROR

S$OK uses backup facilities; the command is executed.
E$ERROR denotes a fatal error in one of the display settings; the command could not be executed.

Sign acknowledge	= E$UNKNOWN$VMS$NR
	= E$UNKNOWN$PICT
	= E$PICT$ERROR
	= W$PICT$WARNING

Set Lightness

By means of the Set Light Command it is possible to set the lightness of a VMS to a requested value. With a setting 0 for the first VMS unit, all VMS units on the controller can be switched to the requested value in only one single command.

3		task = 3
4	Type = 2	Command = 2
5	VMS-number (0, or 1..255)	
6	light indication	
7	VMS-number	
8	light indication	
	etcetera	

Set Lightness
order buffer

where:

light indication = 0 (Automatic by VMS-group)
= 1 (Day)
= 2 (dimmed stage 1)
= 3 (dimmed stage 2)
= 4 (night)

Lightness Status reply

In a Lightness Status reply buffer the display task reports the status of an executed Set Lightness command. The command status field contains the common result of the command. The lightness acknowledge field gives the result for the specific unit.

3		task = 3
4	Command status	
5	Type = 2	Command = 2
6	VMS-number	
7	lightness acknowledge	
	etcetera	

Where:

Command status	= S$OK
	= E$ERROR
	= E$WARNING
Lightness acknowledge	= E$UNKNOWN$VMS$NR
	= E$UNSUPP$LIGHTN

Pictogram Status request

In this command the master task requests for the status of one or more pictograms on a VMS unit. VMS number equals 0 in the first VMS-number field is a fast way to find out if one of the VMS units has a problem for a specific pictogram.

3		task = 2
4	Type = 2	Command = 3
5	VMS-number (0, or 1..255)	
6	pictogram (0-255)	
7	VMS-number (1..255)	
8	pictogram (0-255)	
	etcetera	

Pictogram Status Request order message

Pictogram Status reply

In a pictogram status reply message the service task reports the result of a pictogram status request. For each VMS/pictogram combination the sign status field reports if the pictogram could be displayed properly.
The command status field shows the overall result. If the VMS code 0 was selected in the first VMS-number field then in the succeeding sign status field a worst condition is reported.

3		task = 3
4	Command status	
5	Type = 2	Command = 3
6	VMS-number	
7	Sign status	
8	VMS-number	
9	Sign status	
	etcetera	

Pictogram Status reply message

Where:

Command status	= S$OK
	= E$ERROR
	= E$WARNING

Sign status	= S$OK
	= E$UNKNOWN$VMS$NR
	= E$UNKNOWN$PICT
	= E$PICT$ERROR
	= W$PICT$WARNING

VMS Status Request

In a VMS Status Request, the master enables the control task in the VMS to report a status change as soon as this occurs via its output FIFO buffer. In the FIFO-reset field can be specified if the FIFO should be first reset (e.g. during master start up).

3		task = 1
4	Type = 2	Command = 4
5	FIFO-reset (0 of 1)	

VMS Status Request
order message

where:

FIFO-reset	= 0 for no FIFO reset
	= 1 for FIFO reset first

The reply to a VMS Status Request is either an Asynchronous Status Reply or a Dummy Status Reply.

Asynchronous Status Reply

In an Asynchronous Status Reply the control task reports a status change in one of the VMS units or in the VMS controller. All Asynchronous status response messages are reported via a FIFO buffer so that they are not lost. If in a VMS controller a status change occurs and no VMS Status Request has been received, then the VMS Status Response will be stored in the FIFO until the moment that it can be reported.

3		task = 1
4	Command status	
5	Type = 2	Command = 4
6	message-type = 1 (Async.s)	
7	VMS-number (0..255)	
8	old Sign status	
9	new Sign status	

Asynchronous Status
reply message

where:

Command status	= S$OK
old_sign_status	= S$OK
	= S$RESET
	= S$PICT$WARNING
	= E$FIFO$OVERFLOW
	= E$PICT$ERROR
new_sign_status	= same as old sign status

Status reports for the controller contain a zero in the VMS number field.

Dummy Status Reply

If within a minute after receive of a VMS Status request no status change occurs in one of the VMS units, then the control task issues a Dummy Status Reply to the master task.

3		task = 1
4	Command status	
5	Type = 2	Command = 4
6	message-type = 2 (Dummy)	

Dummy Status reply message

where:

Command status	= S$OK

Request Configuration

In the Request Configuration command a master task replies to return
the number of connected VMS units and the supported pictograms on a specific unit.

3		task = 1
4	Type = 2	Command = 5
5	VMS-number (0, or 1..255)	

Request Configuration order message

If the VMS-unit number equals zero then a Configuration Information reply is returned in which the number of connected VMS units is indicated. If a specific unit 1..255 is specified, then a message is returned with the supported pictograms of that unit.

Configuration Information

In a configuration buffer the VMS service task supplies information on the configuration of the VMS controller, with connected VMS-units. If information from a specific VMS is requested then information on the number and type of the supported pictograms is returned. The collection of pictograms is returned in a bitmap.

3		task = 1	
4	Command status		
5	Type = 2	Command = 5	
6	VMS-unit = n		VMS-units = 0
7	VMS-type		number VMS-units (1..16)
8	supported pictograms		
16	64 bits		

Where:

Command status	= S$OK
	= E$ERROR (one of the pictograms has an error status)
	= E$WARNING (one of the pictograms has a warning status)
VMS-type	= 1 (fibre optic VMS)
	= 2 (LED VMS)

VMS Maintenance Commands

Maintenance commands and replies are specific for the VMS-controller technology applied and can also be specific for a supplier.

The maintenance commands for a specific implementation, namely for fibre optic display units, is:

id	command	reply
1	get operation information	operation information reply
2	get VMS maintenance status	VMS maintenance status reply
3	reset VMS	reset VMS reply

The operation information reply provides extensive information on a VMS unit, including supplier and available functionality. A VMS maintenance status reply provides extensive information on the status of the lamps in a VMS unit. With the reset VMS command it is possible to reset the micro processor of the VMS controller.

Get operation information

In a Get Operation Information maintenance command the master task requests a master task for specific maintenance information to a VMS controller.

	task = 2
Type = 2	Command = 0
maintenance command = 1	
VMS number (0..255)	

where:

- VMS number = 1..255 (get operation info of group and specified VMS)
  = 0 (if only group operation info is required)

Operation Information reply

In an Operation Information reply the VMS controller returns the answer to the Get Operation Information Command. The message contains manufacturer information about the controller and a specified connected VMS :

3		task = 2
4	command status	
5	Type = 2	0
6	command =1	
7	group vendor	
8	group type	
9	light indicator sensor	
10	number of VMS's	

11	VMS number
12	VMS vendor
13	VMS type
14	number of available signs
15	programmable VMS

where:

- Command Status	= S$OK
- Group vendor	= 0(unknown)
	= 1-255 (list of 255 vendors)
- Group type	= 0 (unknown)
	= 1 - 255 (list with 255 types)
- light indication sensor	
	= 0 not available
	= 1 available, automatic brightness adjustment available
- number of VMS's	= 1..255
- VMS number	= 1..255 (the following bytes contain info about specified VMS number)
	= 0 (the following bytes do not contain valid information)
- VMS vendor	= 0 (unknown)
	= 1 - 255 (list with 255 vendors)
- VMS type	= 0 (unknown)
	= 1 - 255 (list with 255 types)
- number of available signs	
	= 1 - 255
- VMS programmable	= 0 not programmable
	= 1 programmable

Get VMS maintenance Status

In a Get VMS Maintenance Status command the master task requests a VMS controller for status information about the lamps:

3		task = 2
4	type =2	command = 0
5	maintenance command = 2	
6	VMS-number	

where:

- VMS number	=	1..255 (status request for specified VMS and complete group)
	=	0 (status request for the entire group)

VMS Maintenance Status Reply

In a VMS maintenance Status reply, the VMS returns the status of the lamps

3		task = 2
4	command status	
5	type = 2	0
6	command = 2	
7	VMS number	
8	current light indication	
9	current sign number	
10	status 1	
11	status 2 (reserved)	
12	failure primary lamp 1. byte	
13	failure primary lamp 2. byte	
14	failure primary lamp 3. byte	
15	failure secondary lamp 1. byte	
16	failure secondary lamp 2. byte	
	failure secondary lamp 3. byte	

where:

- current light indication
  = 1 = Day
  2 = dimmed stage 1
  3 = dimmed stage 2
  4 = night
- status 1 =
  bit 0: failure primary lamp
  bit 1: failure secondary lamp
  bit 2: reserved
  bit 3: door contact
  bit 4: sign partial failure
  bit 5: reserved
  bit 6: control active
  bit 7: manual operation

- status 2 =
  reserved
- failure primary lamp 1. byte
  bit 0 set: primary lamp 1

bit 1 set: primary lamp 2
bit 2 set: primary lamp 3
bit 3 set: primary lamp 4
bit 4 set: primary lamp 5
bit 5 set: primary lamp 6
bit 6 set: primary lamp 7
bit 7 set: primary lamp 8

- failure primary lamp 2. byte
  bit 0 set: primary lamp 9
  bit 1 set: primary lamp 10
  bit 2 set: primary lamp 11
  bit 3 set: primary lamp 12
  bit 4 set: primary lamp 13
  bit 5 set: primary lamp 14
  bit 6 set: primary lamp 15
  bit 7 set: primary lamp 16
- failure primary lamp 3. byte
  bit 0 set: primary lamp 17
  bit 1 set: primary lamp 18
  bit 2 set: primary lamp 19
  bit 3 set: primary lamp 20
  bit 4 set: primary lamp 21
  bit 5 set: primary lamp 22
  bit 6 set: primary lamp 23
  bit 7 set: primary lamp 24
- failure secondary lamp 1. byte
  bit 0 set: secondary lamp 1
  bit 1 set: secondary lamp 2
  bit 2 set: secondary lamp 3
  bit 3 set: secondary lamp 4
  bit 4 set: secondary lamp 5
  bit 5 set: secondary lamp 6
  bit 6 set: secondary lamp 7
  bit 7 set: secondary lamp 8
- failure secondary lamp 2. byte
  bit 0 set: secondary lamp 9
  bit 1 set: secondary lamp 10
  bit 2 set: secondary lamp 11
  bit 3 set: secondary lamp 12
  bit 4 set: secondary lamp 13
  bit 5 set: secondary lamp 14
  bit 6 set: secondary lamp 15
  bit 7 set: secondary lamp 16
- failure secondary lamp 3. byte

bit 0 set: secondary lamp 17
bit 1 set: secondary lamp 18
bit 2 set: secondary lamp 19
bit 3 set: secondary lamp 20
bit 4 set: secondary lamp 21
bit 5 set: secondary lamp 22
bit 6 set: secondary lamp 23
bit 7 set: secondary lamp 24

- VMS number
  = 1..255 (the following bytes contain information about specified VMS number)
  = 0 (the following bytes do not contain valid information)

Reset VMS

3		task = 2
4	type = 2	command = 0
5	maintenance command = 3	

where:
- VMS number = 1..255 (reset specified VMS)
  = 0 (reset all VMS')

Reset VMS reply

3		task = 2
4	command status	
5	type = 2	0
6	maintenance command = 3	

where:
- Command Status = S$OK

Status Return Codes

The following status codes can be replied by the VMS controller through field 4 in a reply message:

Condition	Value	Meaning
S$OK	00	Command executed ok
E$UNKNOWN$CMD	01	Non existing command
E$UNKNOWN$TYPE	02	Type conflict
E$UNKNOWN$VMS$NR	05	Non existing VMS unit
E$UNKNOWN$PICT	06	Non supported pictogram
E$UNSUPP$LIGHTN	07	Non supported light setting
E$PICT$ERROR	08	Fatal error in pictogram
E$FIFO$OVERFLOW	09	Fifo overflow in controller messages have gone lost
W$WARNING	64H	Command executed with warning status
W$PICT$WARNING	65H	Pictogram using backup facilities
S$RESET		Controller reset

8.2 Vehicle Detector

Schematically the ES15 Vehicle Detector can be represented as follows:

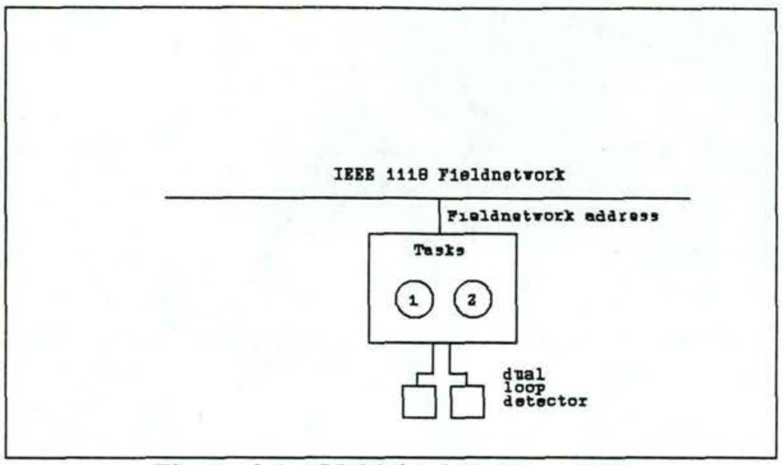


Figure 2.1 : Vehicle detector set-up

A vehicle detector has two separate tasks that can each handle one or more commands via the Bitbus network:

1. A measurement tasks that continuously monitors the output from the road sensors. If a change occurs a message is queued to the network
2. A service task which performs maintenance commands and also replies to requests for actual status information.

The communication over the fieldnetwork contains the following messages for vehicle detectors:

id	command	task	reply
0	maintenance command	2	maintenance reply
1	detector data request	1	vehicle data reply asynchronous status change status reply
2	detector status request	2	synchronous status reply

Both tasks can simultaneously handle one command each. During normal operation a vehicle detector is handling a Detector Data Request. The reply to this request can be one out of three different messages that after some time are issued via a FIFO buffer:
- On detection of a vehicle, a Vehicle Data Reply is prepared and added to an output FIFO.

When on that moment the vehicle detector has a Detector Data Request outstanding, then the message is sent directly from the FIFO to the fieldnetwork.
- At the occurrence of a status change, an Asynchronous Status Change message is constructed and added to the output FIFO.
- Every 30 seconds and after a FOFO reset a Dummy Reply message is added to the output FIFO.

During handling of a Detector Data Request a detector can simultaneously handle a Maintenance Command or a Detector Status Request via the service task. The reply to this request is issued immediately.

Detector Data Request

A master task can request a vehicle detector for detector data in a Detector Data Request command:

3		task = 1
4	type = 1	command = 1
5	Fifo clear flag (0 or 1)	

where:

fifo clear flag	= 1 reset FIFO before
	0 after fifo reset

The reply to a Detector Data Request command can be a Vehicle Data reply, Asynchronous Status Reply or a Dummy reply. Each of these replies are issued via a Fifo outputbuffer. When at reception of the command the fifo buffer is empty, then the reply is postponed until either a vehicle passes, or a status change in the detector occurs, or a timeout expires. It is possible to denote in the command if the fifo has to be reset first. If within 30 seconds after reception of the Detector Data request no status change in the detector occurs a dummy reply is issued.

Vehicle Data Reply

Each time a vehicle passes a Vehicle Data reply message is filled in and issued to the output fifo:

quantity	abbreviation	unity	range
travel direction:	td		positive/negative/unknown
speed:	s	0.1 m/s	0 - 4095
normalized traveltime	nt	ms/1 m	0 - 1023
reliability speed/traveltime:	rs		0 - 3
length:	l	cm	0 - 4095
normalized occupancy time	ot	10 ms	0 - 4095
reliability length/occupancytime:	rl		0 - 3

time	t	10 ms	0 - 4294967305
vehicle count	co		0 - 15
category	ca		0 - 15

3			task = 1	
4	Command Status = S$VEHICLE$DATA$REPLY			
5	type = 1		command = 1	
6	b1 td b0	b1 rs b0	b1 rl b0	
7	b7 speed b0			
8	b3 travel time b0		b11 speed b8	
9		b9 travel time b4		
10	b7 length b0			
11	b11 occ time b8		b11 length b8	
12	b7 occupancy time b0			
13	b7 time b0			
14	b15 time b8			
13	b23 time b16			
14	b32 time b24			
15	b3 vehicle co. b0		b3 category b0	

Notice that the contents of the message is independent of specific vehicle detector technology. The various fields in the message are explained below:

travel direction: Direction of the vehicle that passes the vehicle detector. A vehicle detector that can determine travel direction has a pre-configured (fixed) method of detecting what is 'positive' or 'negative' travel direction.

speed: Speed of the vehicle in 0.1 m/s.

normalized travel time: Time the vehicle takes to travel a distance of 1 m in ms.

reliability speed/tr.tm.: Reliability indication of speed and travel time data:
0 = not reliable (invalid data)
1 = error > 25 % possible
2 = 2 % < error <= 25 %
3 = error < 2 %

length: Length of vehicle in cm

normalised occ. time: The time in 10 ms the vehicle occupied a point of the road.

reliability length/occ.tm.: Reliability indication of length and occupancy time data:
0 = not reliable (invalid data)
1 = error > 25 % possible
2 = 2 % < error <= 25 %
3 = error < 2 %

vehicle identification: sequence count of the vehicle

time The moment the vehicle passed the detector. This value can be used to measure the time elapsed between the passage of this vehicle (vehicle count) and the one before (vehicle count - 1) in units of 10 ms. This time is initialized to 0 after every powerup or reset of the detector and can be set by a set time command.

category Category identification of the vehicle:
0 = no category identification
1 = motor cycle

```
2 = passenger car
3 = delivery van
4 = passanger car with trailer
5 = delivery van with trailer
6 = truck
7 = truck with trailer
8 = truck with semi-trailer
9 = bus
10 - 15 reserved
```

Asynchronous Status Change reply

At the occurrence of a status change in the detector, the following reply buffer is constructed and added to the output FIFO:

3		task = 1
4	Command Status = S$STATUS$REPLY	
5	type = 1	command = 1
6	detector technology code	
7	Detector technology dependent status	
n		

Status messages provide information on the status of the detector hardware. Consequently they are specific for the detector technology, identified in the sixth byte of the message.

In the following the status messages for the ES15 dual loop detector are explained.
For a dual loop detector code L_DUAL_LOOP the following status data can be generated:

1. Undefined detection/detection sequence error

status code: L_SEQ_STATUS

The cause of this status message is further described by a sub-code:

L_SQ_SEQ	- detection sequence error (only detection on primary or secondary loop while both loops were tuned)
L_SQ_SPEED	- vehicle detected with a speed > 200 km/h
L_SQ_DIFF	- difference between occupancy time of primary loop and secondary loop to large.

3		task = 1
4	Command Status = S$STATUS$REPLY	
5	type = 1	command = 1
6	L_DUAL_LOOP	
7	L_SEQ_STATUS	
8	sequence subcode	

2. Loop tune status change

status code: L_TUNE_STATUS

This status message is output when the tune status of one of the detectors changes. The cause of this status message is further described by a two sub-codes indicating the status of the primary and secondary loop:

L_OUT_TUNE_POWER	- loop not tuned after power up/reset
L_OUT_TUNE_TIME_OUT	- loop not tuned after occupancy time out
L_OUT_TUNE_FREQ	- loop not tuned after frequency change
L_TUNED	- loop tuned
L_TUNE_NO_CHANGE	- no status change of this loop

For information the resulting tune status of both loops is also send with this message in two loop status fields:

L_IN_TUNE	- loop tuned
L_OUT_TUNE	- loop out of tune

3		task = 1
4	Command Status = S$STATUS$REPLY	
5	type = 1	command = 1
6	L_DUAL_LOOP	
7	L_TUNE_STATUS	
8	primary loop subcode	
9	secondary loop subcode	
10	primary loop tune status	
11	secondary loop tune status	

3. Detector system error

```
        status code: L_SYS_ERROR
```

The cause of this status message is further described by a sub-code:

L_SY_EPROM	- EPROM checksum error
L_SY_RAM	- RAM read/write error
L_SY_FIFO_OVERFLOW	- Output message fifo overflow
L_SY_POWER	- Detector power up / reset

3		task = 1
4	Command Status = S$STATUS$REPLY	
5	type = 1	command = 1
10	L_DUAL_LOOP	
11	L_SYS_ERROR	
12	system error subcode	

Status Reply

A dummy Status Reply message is added to the output buffer every 30 seconds and each time after a FIFO restet. This prevents the higher level application to timeout and verify that the vehicle detector is still alive. It also allows the higher level application to maintain counters for reliable detector status information during system startup and after communication failures.

3			task = 1
4	Command Status = S$STATUSY$REPLY		
5	type = 1		command = 1
6	Tp	Ts	

Where:

Tp	Tune status primary loop 1 = in-tune; 0 = out of tune
Ts	Tune status secundary loop 1 = in-tune; 0 = out of tune

Synchronous status reply

The master can request detailed technical information from the loopdetector with the following synchronous command:

3					task = 1
4	Command Status = S$OK				
5	type = 1				command = 2
6	Tp	Ts	Bp	Bs	0
7	loopfrequency primary loop				
8	loop frequency secundary loop				
9	occupancy state primary loop				
10	occupancy state secundary loop				

Where:

Tp	tune status primary loop: 1=tuned;0=not-tuned
Ts	tune status secundary loop
Bp	Occupancy status primary loop 1=occupied;0=not-occupied
Bs	Occupancy status secundary loop

8.3 Meteo Controller

Meteorological data collected alongside roads, comprises the following data:

1. Visibility conditions
2. Road conditions
3. Wind conditions

Each of these categories can be provided by specific fieldnetworkcontrollers with a limited set of peripherals connected to them, or they can be provided by a single common fieldnetworkcontroller with a large range of peripherals. In this chapter three logical nodes are described, leaving freedom to the manufactures on implementation choices. The general structure of the meteo fieldnetwork controler can be depicted as follows:

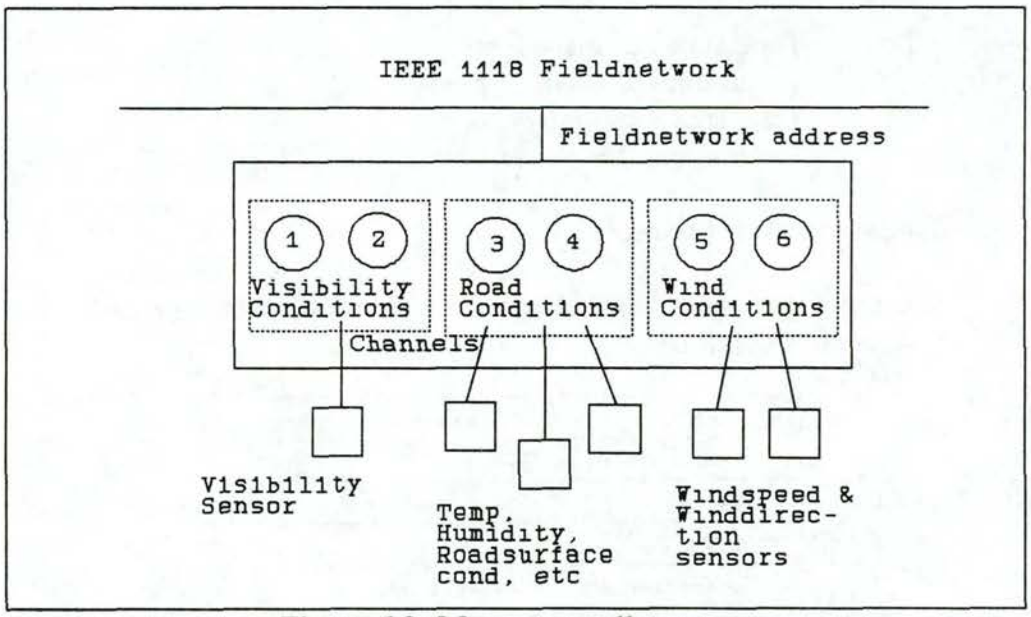


Figure 6.3: Meteo controller set-up

Maintenance Command

Maintenance commands are specific for the sensor technology used. Each of the logical controllers will have the ability to request for maintenance information. A general maintenance command that all logical controllers have is a request for the last sampled data of a specific channel and the results of its processing.

The layout of the command is as follows:

3		task = 1,3 or 5
4	type = 4,5 or 6	command = 0
5	command = C$CHANNEL$DATA	
6	channel [1..64]]	

The reply to this command is of the following format:

3		task = 2
4	command status = S$OK	
5	type = 4,5 or 6	command = 0
6	command = C$CHANNEL$DATA	
7	channel [1..64]	
8	status channel data	
9	channel data	

The status of the channel data and the channel data itself are specific for the controller and may be specific for each manufacturer.

8.3.1 Visibility Conditions

Visibility is calculated over a period of 12 seconds, within that period the visibility sensor is sampled each second.

At system startup and each time the controller has suffered a powerfailure a time synchronisation message must be given to the visibility controller. This allows the controller to maintain a 12 second interval in real time and to transmit measurement results at the end of each interval. Measurement results are calculated at the end of each 12 second interval starting at 0:00 hours.

All measurement results are transmitted via an output FIFO buffer. At the end of each 12 second interval a message with measurement results is prepared and added to the output FIFO. Through a fieldnetwork command these FIFO messages can be requested.

The visibility controller has two tasks that can perform the following commands:

id	command	task	reply
0	maintenance command	1	maintenance reply
1	time command	1	time reply
2	visibility data request	2	visibility data reply
			powerup message

Task 1 performs service functions. The commands to this task are performed in a synchronous manner, meaning that the reply is given immediately. The other task performs the actual measurement cycle. Its output is via the output FIFO buffer.

Time Command

In a Time Command the Master Node informs the Visibility Controller with the real time, the command allows the controller to start its measurement cycles in real time and to connect an accurate time indication to measurement results. In reaction to the command the internal clock is adjusted. To the higher level application this could mean that a data reply comes sooner or later than expected, with a different number of samples than normal.

3		task = 2
4	type = 5	command = 1
5 6	second time in tenth of seconds [0..600]	
7	minute time [1..60]	
8	hours [1..12]	

Time Reply

In a Time Reply the Visibility controller acknowledges the reception of a time command:

3		task = 2
4	command status = S$OK	
5	type = 5	command = 1

Visibility data request

The visibility request asks for the measurement results from the connected visibility sensor:

3		task = 2
4	type = 5	command = 2
5	FIFO clear flag (0 or 1)	

Where:

FIFO clear flag	= 1 for reset FIFO beforehand
	= 0 for no FIFO reset

It is possible to denote in the command if the FIFO buffer must be reset before the execution of the command. All outstanding measurements or powerup messages are wiped than.

The reply to the visibility data request can be a visibility reply or a powerup message.

Visibility Reply

The Visibility Reply contains visibility measurement information. The reply is either given directly or after a certain delay. When messages are waiting for transmission in the output FIFO the reply is transmitted directly. When the output FIFO is empty (the normal situation) the transmission of a reply is delayed until the end of the current measurement period.

3			task = 2
4	command status = S$VISIBILITY$OK		
5	type = 5		command = 2
6	time [0..300] in units of 12 seconds		
7		timesync	Smplcnt [0..32]
8	visibility [0-1000]		
9		NERR samples [0..16]	

Where:

time	= endtime of measurement period in units of 12 seconds to a total alapstime of 3 hours
timesync	= 1 for timesynchronisation lost because of powerfailure
	= 0 for timesynchronisation OK
Smplcnt	= count of visibility samples in current interval
Nerr samples	= count rejected samples from sensor
Visibility	= visibility in metres [0 - 1000]

Powerup message

After a powerfailure the controller has wiped its internal memory, including the output FIFO and internal clock. In the controller's startup sequence the internal clock is reset to zero and a powerup message is added to the output FIFO, in which the higher level application is informed about the powerloss condition.

3		task = 2
4	command status = S$POWERFAIL	
5	type = 5	command = 2

8.3.2 Roadconditions

Roadconditions comprise detailed information about the environmental conditions of the road. The information concerns: air- & roadtemperature, relative humidity, road surface conditions and precipitation. A roadcondition controller can handle a maximum of 16 temperature sensors, one relative humidity sensor, 13 roadsurface detectors, one precipitation detector and an optional radiation sensor.

The Roadsurface Condition controller has two tasks that can perform the following commands:

id	command	task	reply
0	maintenance command	3	maintenance reply
1	time command	3	time reply
2	Roadcondition request	4	Temperature reply
			Roadsurface conditions
			Radiation Reply (optional)
			Precipitation Reply
			Relative humidity reply
			powerup message

Task 3 performs service functions. The commands to this task are performed in a synchronous manner, meaning that the reply is given imediately. The other task performs the actual measurement cycle. Its output is via the output FIFO buffer.

The output FIFO buffer can contain a variable number messages depending on the timepoint and the number of connected sensors. In each interval the Relative Humidity Reply message is added last to the FIFO. To the master application this message is therefor an indicator that all data for the past period has been received.

Time Command

In a Time Command the Master Node instructs the Visibility Controller to start its measurement cycles in real time. When measurement cycles are already in progress the internal clock is adjusted. To the higher level application this could mean that a data reply comes sooner or later than expected, with a different number of samples than normal.

3		task = 3
4	type = 5	command = 1
5 6	second time in tenth of seconds [0..600]	
7	minute time [1..60]	
8	hours [1..12]	

Time Reply

In a Time Reply the Visibility controller acknowledges the reception of a time command:

3		task = 3
4	command status = S$OK	
5	type = 5	command = 1

Roadcondition request

With the following command a master application can request for outstanding roadcondition data in the output FIFO buffer:

3		task = 4
4	type = 6	command = 2
5	FIFO clear flag (0 or 1)	

Where:

FIFO clear flag	= 1 for reset FIFO beforehand
	= 0 for no FIFO reset

It is possible to denote in the command if the FIFO buffer must be reset before the execution of the command. All outstanding messages are wiped than.

The reply to this command is the next outstanding message in the output FIFO, either containg temperature data, or relative humidity data, or etc.

Temperature reply

A temperature reply message contains the measurement results of a series of connected temperature sensors. A number of these sensors can be reference resistors to allow for an accurate stability assessment. Temperature messages contain temperature information over a 12 second period. Within this interval the temperature sensors are sampled every second. At the end of each interval a series of reply messages with temperature data are prepared and added to the output FIFO.

3		task = 4
4	command status = S$TEMP$OK	
5	type = 6	command = 2
6	time [0..300] in units of 12 seconds	
7	timesync	Smplcnt [0..32]
8	sensor channel [0..32]	
9		Nerr temp [0..16]
10	12 sec temperature + 250 [0..1024]	
11	sensor channel [0..32]	
12		Nerr temp [0..16]
13	12 sec temperature + 250 [0..1024]	
14	sensor channel [0..32]	
15		Nerr temp [0..16]
16	12 sec temperature + 250 [0..1024]	

Where:

time	= endtime of measurement period in seconds
timesync	= 1 for timesynchronisation lost because of powerfailure
	= 0 for timesynchronisation OK
Smplcnt	= count of visibility samples in current interval
Nerr temp	= count rejected samples from temp sensor
12 sec temp	= 12 second avaraged temperature in 0.1 degrees

At the end of each measurement cycle the output FIFO can contain multiple of the above messages, depending on the number of temperature sensors connected.

Roadsurface conditions reply

Roadsurface conditions are calculated over a 60 second interval. At the end of each interval a series of messages is prepared and added to the output FIFO.

3		task = 4	
4	command status = S$SURF$COND$OK		
5	type = 6	command = 2	
6	time [0..300] in units of 12 seconds		
7	timesync	Smplcnt [0..32]	
8	Nerr	channel [0..32]	
9	delta freezing temp	sdft	road cond.
10	Nerr	channel [0..32]	
11	delta freezing temp	sdft	road cond.
12	Nerr	channel [0..32]	
13	delta freezing temp	sdft	road cond.
14	Nerr	channel [0..32]	
15	delta freez. temp	sdft	road cond.

Where:

Road condition = indication for road surface condition:
0 = dry
2 = wet
3 = ice
4 = salty

Delta freez.temp = change in freezing temp + 16

Sdft = Status change in freezing temp
1 = error/connot assess
0 = OK status

At the end of each measurement interval a variable number of messages, yielding to the number of connected Roadsurface condition detectors, is added to the FIFO output buffer.

Precipitation reply

This reply contains a rough indication on precipitation. Precipitation is measured every 5 seconds over an interval of 1 minute. At the end of each interval a precipitation reply is prepared and added to the output FIFO.

3		task = 4
4	command status = S$SURF$COND$OK	
5	type = 6	command = 2
6	time [0..300] in units of 12 seconds	
7	timesync	Smplcnt [0..32]
8	Precipitation ind	Nerr prec [1..16]

Where:

Precipitation ind.	= rough indication for precipitation:
	0 = dry
	1 = moist
	2 = wet
	3 = heavy rain
Smplcnt	= samplecount in current interval
Nerr prec	= count rejected samples

Radiation reply

For the improvement of short term predictions on roadsurface conditions a radiation detector can help in the assessment of a temperature gradient. An optional radiation reply contains the measurement results of a radiation detector over an interval of 12 seconds. Within this interval the detector is sampled every second. At the end of the measurement interval a radiation message is prepared and added to the output FIFO.

3		task = 4
4	command status = S$RADIATION$OK	
5	type = 6	command = 2
6	time [0..300] in units of 12 seconds	
7	timesync	Smplcnt [0..32]
8		Nerr prec [1..16]
9	Radiation [0..100]	

Where:

Radiation	= ..... to be specified ......
Smplcnt	= samplecount in current interval
Nerr prec	= count rejected samples

Relative Humidity reply

A Relative Humidity reply message contains the measurement results of a connected Relative Humidity sensor over a period of 12 seconds. Within that period the sensor is sampled every second.

3		task = 4
4	command status = *S$REL$HUM$OK*	
5	type = 6	command = 2
6	time [0..300] in units of 12 seconds	
7	timesync	Smplcnt [0..32]
8		Nerr temp [0..16]
9	12 sec rel.humidity [0..256]	

Where:

time	= endtime of measurement period in seconds
timesync	= 1 for timesynchronisation lost because of powerfailure
	= 0 for timesynchronisation OK
Smplcnt	= count of visibility samples in current interval
Nerr temp	= count rejected samples from temp sensor
12 sec rel.hum	= 12 second avaraged rel.hum in %

Powerup message

After a powerfailure the controller has wiped its internal memory, including the output FIFO and internal clock. In the controller's startup sequence the internal clock is reset to zero and a powerup message is added to the output FIFO, in which the higher level application is informed about the powerloss condition.

3		task = 2
4	command status = S$POWERFAIL	
5	type = 5	command = 2

8.3.3 Wind Conditions

Wind conditions are calculated over a period of 12 seconds. The controller allows for 1 windspeed and 1 winddirection sensor. The wind sensors are sampled every 0.2 seconds.

The structure of the measurement cycle and the buffered output of the wind controller is comparable with the visibility controller. The wind controller has two tasks that can perform the following commands:

id	command	task	reply
0	maintenance command	5	maintenance reply
1	time command	6	time reply
2	wind data request	6	wind data reply
			powerup message

Task 5 performs service functions. The commands to this task are performed in a synchronous manner. Task 6 performs the measurement cycle, its output is via a FIFO buffer.

Time Command

In a Time Command the Master Node instructs the Visibility Controller to start its measurement cycles in real time. When measurement cycles are already in progress the internal clock is adjusted. To the higher level application this could mean that a data reply comes sooner or later than expected, with a different number of samples than normal.

3		task = 2
4	type = 5	command = 1
5 6	second time in tenth of seconds [0..600]	
7	minute time [1..60]	
8	hours [1..12]	

Time Reply

In a Time Reply the Visibility controller acknowledges the reception of a time command:

3		task = 2
4	command status = S$OK	
5	type = 4	command = 1

Wind data request

In the Wind Data Request a master application can request for wind measurement results from the Wind Controller.

3		task = 6
4	type = 4	command = 2
5	FIFO clear flag (0 or 1)	

It is possible to denote in the command if the FIFO buffer must be reset before the execution of the command. All outstanding measurements or powerup messages are wiped than.

The reply to the wind request can be a visibility reply or a powerup message.

Wind reply

A Wind Reply message contains measurement results over a 12 second measurement cycle:

3			task = 6
4	command status = S$WIND$OK		
5	type = 4		command = 2
6	time [0..300] in units of 12 seconds		
7		timesync	Smplcnt [0..32]
8			Smplcnt gust
9	12 sec wind speed [0..150]		
10		Nerr wind speed	
11	12 sec wind direction [0..360]		
12		Nerr wind gust	
13	2 second maximum wind gust [0..150]		

Where:

time	= endtime of measurement period in seconds
timesync	= 1 for timesynchronisation lost because of powerfailure

	= 0 for timesynchronisation OK
wind speed	= in metres per second averaged over 12 seconds
wind direction	= in degrees averaged over 12 seconds
maximum wind gust	= maximum wind gust in a sliding window of 2 seconds
Smplcnt	= total count samples in interval
Nerr	= count error samples

Powerup message

After a powerfailure the controller has wiped its internal memory, including the output FIFO and internal clock. In the controller's startup sequence the internal clock is reset to zero and a powerup message is added to the output FIFO, in which the higher level application is informed about the powerloss condition.

3		task = 2
4	command status = S$POWERFAIL	
5	type = 5	command = 2

9 REFERENCES

[1]
"UROP Functional Architecture Specification", DRIVE-I deliverable no. 4 of the V1048 UROP project

[2]
"UROP Open Road Communication Architecture (ORCA)", DRIVE-I deliverable no 6 of the V1048 UROP project

[3]
"Description of the CROW Control Centre", DRIVE-I deliverable 33 of the CROW V1058 project.

[4]
"Functionele specificatie Gladheidsmeldsysteem", TNO-ITI document 89-ITI-B-371

###