

**VDE** SPEC

SWARCO TRAFFIC SYSTEMS GmbH

# Interface for digital Signals and Sensors for Traffic Light Control Devices

VDE SPEC 90013 V1.0 (en)

**VDE**

## Preface

Publication date of this VDE SPEC: 10 March 2022.

No draft has been published for the present VDE SPEC.

This VDE SPEC was developed according to the VDE SPEC procedure. VDE SPEC 90013 is developed in project groups and not necessarily with the involvement of all interested parties.

This VDE SPEC is **not** part of the VDE set of regulations or the German set of standards. In particular, this VDE SPEC is **not** a technical rule within the meaning of Section 49 EnWG.

The authors of this VDE SPEC are:

- Jürgen Weingart, SWARCO TRAFFIC SYSTEMS GmbH
- Gerhard Kral, SWARCO Futurit Verkehrssignalsysteme GmbH
- Matthias Nolle, SWARCO TRAFFIC SYSTEMS GmbH
- Klaus Löwenhagen, RTB GmbH & Co. KG
- Jan Göbel, Langmatz GmbH

The present VDE SPEC is on the subject:

“Interface for digital signals and sensors for Traffic Light Control Devices”.

Please send your feedback to the email address: [info@swarco.de](mailto:info@swarco.de)

If you are interested in actively participating in the project group, please contact [info@swarco.de](mailto:info@swarco.de).

At this point, many thanks to the experts for their valuable input:

*Artur Schmidt, DKE*

*Kai Siemer, Langmatz GmbH*

*Christian Ehring, RTB GmbH & Co. KG*

*Andreas Seiler, SWARCO Futurit Verkehrssignalsysteme GmbH*

*Kim Dalgaard, SWARCO TECHNOLOGY APS*

*Hans-Jürgen Mauser, SWARCO TRAFFIC SYSTEMS GmbH*

*Dr. Thomas Novak, SWARCO TRAFFIC SYSTEMS GmbH*

*Andreas Wortmann, TÜV Rheinland Rail Certification B.V.*

Despite great efforts to ensure the correctness, reliability and precision of technical and non-technical descriptions, the VDE SPEC project group can neither explicitly nor implicitly guarantee the correctness of the document. This document is used in the knowledge that the VDE SPEC project group cannot be made liable for damage or loss of any kind. The application of the present VDE SPEC does not release the user from responsibility for their own actions and is therefore at their own risk.

In the course of the manufacture and / or introduction of products into the European internal market, the manufacturer shall carry out a risk analysis in order to first determine which risks the product may entail. After performing the risk analysis, he evaluates these risks and, if necessary, takes suitable measures to effectively eliminate or minimize the risks (risk assessment). The present VDE SPEC does not release the user from this responsibility.

In this document the standard conformant use of must, shall, should and may is used as follows [1]:

- **Mandatory** requirements are expressed with **shall** or **have to**
- **Recommended** requirements are expressed with **should**
- Requirements that are **optional** are expressed with **may** or **could**
- **Must** is used only for **external constraints**

Attention is drawn to the possibility that some elements of this document may affect patent rights. VDE is not responsible for identifying any or all of the related patent rights.

## Executive Summary

This VDE SPEC deals with the introduction of an open, intelligent, digital interface for units that are typically mounted on the traffic light pole. These can be of a safety-relevant nature such as signal heads, but also of a non-safety-relevant nature like push buttons.

This VDE SPEC specifies the parameters and the physical characteristics of the interface for bidirectional communication of traffic light controllers with distributed signalling and detection devices in an architecture according to EN 50556:2018, 3.2.8 "Centralized power supply, distributed intelligent system".

# Inhalt

<b>Introduction</b>	<b>1</b>
<b>1 Scope</b>	<b>1</b>
<b>2 Normative References</b>	<b>2</b>
<b>3 Terms and definitions</b>	<b>2</b>
<b>4 Symbols and abbreviations</b>	<b>3</b>
<b>5 All ILT devices</b>	<b>4</b>
5.1 General	4
5.1.1 General requirements	4
5.1.2 Communication Interface	4
5.1.3 Power supply	4
5.1.4 Limited functionality	7
5.1.5 Document & Protocol Version	7
5.2 Overall Description	8
5.2.1 Network topologies	8
5.2.2 Network addressing	8
5.2.3 CAN packet	12
5.2.4 Safety integrity	17
5.2.5 Error handling	26
5.2.6 Error reporting	29
5.2.7 Firmware update	33
5.2.8 Parameter update & read out	36
5.2.9 Time sync concept	43
5.3 CAN bus commands	44
5.3.1 General functionality	44
5.3.2 System commands and parameters which apply to all ILT components.	49
5.3.3 Combination of command sets	54
5.4 General technical annotations on and boundary conditions to ILT devices	55
5.4.1 Technical annotation A, Telegram command reference	55
5.4.2 Boundary condition B, Dimming	58
5.4.3 Boundary condition C, Service functions	59
5.4.4 Boundary condition D, Signal state telegram sequencing requirements / example	59
5.5 Proposals	61
<b>6 Aspects</b>	<b>61</b>
6.1 Command set Aspect and parameters	61
6.2 Command SetOperationParameter	64
6.3 Command GetOperationData	68
6.4 Command AliveAck	70
6.5 Technical annotation E, Multipoint Signals	70
<b>7 Pedestrian Push Button</b>	<b>73</b>
7.1 Command set Pedestrian push buttons	73
7.2 Command SetOperationParameter	77

7.3	Command GetOperationData	81
7.4	Command AliveAck	83
7.5	Technical annotations on Pedestrian push Buttons	83
7.5.1	Technical annotation F, manually operated demands and logic inputs	83
<b>8</b>	<b>Acoustic</b>	<b>84</b>
8.1	Command set Acoustic	84
8.2	Command SetOperationParameter	89
8.3	Command GetOperationData	93
8.4	Command AliveAck	95
8.5	Boundary condition regarding to Acoustic	96
8.5.1	Boundary condition G, volume limits	96
<b>9</b>	<b>Traffic Sensor</b>	<b>96</b>
9.1	Command set Traffic sensors	96
9.2	Command SetOperationParameter	99
9.3	Command GetOperationData	100
9.4	Command AliveAck	101
9.5	Technical annotations on Traffic Sensors	101
9.5.1	Technical annotation H, Vehicle classes TLS / BAST, ASTRA	101
	<b>Bibliography</b>	<b>103</b>

## List of figures

Figure 1 – ILT concept	1
Figure 2 – AliveAck during voltage dip	6
Figure 3 – Network topologies	8
Figure 4 – 29-bit CAN ID	11
Figure 5 – 29-bit CAN ID in bit and byte scale	11
Figure 6 – CAN Packet	12
Figure 7 – Basic start-up sequence	13
Figure 8 – Start-up sequence including identification	14
Figure 9 – CAN-bus power-up state-machine	15
Figure 10 – Start-up sequence including identification and command SetNetworkIDMask	16
Figure 11 – Sending telegrams redundantly	21
Figure 12 – State-machine for safety-related telegrams	22
Figure 13 – Alive sequence with one Ack missed	25
Figure 14 – Alive sequence with two missing Ack's from the same ILT component	26
Figure 16 – Error handling	28
Figure 16 – Firmwarebu-update	34
Figure 17 – Parameter-update	37
Figure 18 – Time synchronisation	44
Figure 19 – Dim-levels	59
Figure 20 – Multi Point Signals (1)	71
Figure 21 – Multi Point Signals (2)	71
Figure 22 – Valid and invalid light source masks	72
Figure 23 – Synchronous blinking	73
Figure 24 – Manually operated demands	83
Figure 25 – Logical inputs	84
Figure 26 – Volume limits	96

## List of tables

Table 1 – Abbreviations and Conventions	3
Table 2 – ILT component power consumption	5
Table 3 – Voltage dip	5
Table 4 – Signals at M8-connector	7
Table 5 – Component designator	9
Table 6 – Manufacturer ID specification	9
Table 7 – Device-type specification	9
Table 8 – Sub-type specification	10
Table 9 – Example using Network IDs	16
Table 10 – PFH values necessary as input to FTA of TLC	17
Table 11 – Safety measures	18
Table 12 – Common Status in safety-relevant telegrams	21
Table 13 – Alive telegrams	24
Table 14 – Status mask	25
Table 15 – KnownState	28
Table 16 – FailureState	29
Table 17 – Warnings	29
Table 18 – Commands for error reporting	30
Table 19 – Telegrams concerning firmware update	34
Table 20 – Telegrams concerning parameter update	38
Table 21 – Telegrams concerning parameter read out	41
Table 22 – PowerUp-ID addressed system commands	45
Table 23 – Direct-addressed system commands	47
Table 24 – System commands	50
Table 25 – Command GetWarningAck	54
Table 26 – Cross reference of system commands	55
Table 27 – Aspect standard commands	61
Table 28 – Command SetOperationParameter	65
Table 29 – Aspect operation parameter specification	65
Table 30 – Command GetOperationData	68
Table 31 – Aspect operation data specification	68
Table 32 – Aspect command AliveAck	70
Table 33 – Light source masks	72
Table 34 – Push Button standard commands	74
Table 35 – Command SetOperationParameter	78
Table 36 – Push Button operation parameter specification	78
Table 37 – Command GetOperationData:	81
Table 38 – Push Button operation data specification	82
Table 39 – Push Button command AliveAck	83
Table 40 – Acoustic standard commands	85
Table 41 – Command SetOperationParameter	89
Table 42 – Acoustic operation parameter specification	89

Table 43 – Command GetOperationData	93
Table 44 – Acoustic operation data specification	94
Table 45 – Acoustic command AliveAck	95
Table 46 – Detector command and status telegrams	97
Table 47 – Command SetOperationParameter Traffic Sensors	99
Table 48 – Detector operation parameters	100
Table 49 – Command GetOperationData	100
Table 50 – Traffic sensor command AliveAck	101
Table 51 – Classification in 8+1 vehicle classes acc. to TLS / BAST	101
Table 52: Classification in 5+1 vehicle classes acc. to TLS / BAST	101
Table 53 – Classification in 2 vehicle classes acc. to TLS / BAST	102
Table 54 – Classification type SWISS10 acc. to ASTRA	102
Table 55 – Comparison TLS – ASTRA SWISS10	102



## Introduction

The scope of this document includes the ILT (Intelligent Line Technology) communication interface between the ILT components and the ILT interface box. The network topology, communication standard, and protocol are included, as well as the connector. The network between the ILT interface boxes and the traffic controller is excluded from this specification, as this is a concern of the system integrator. Thus, addressing, and other network issues are only considered between the ILT interface box and the ILT components.

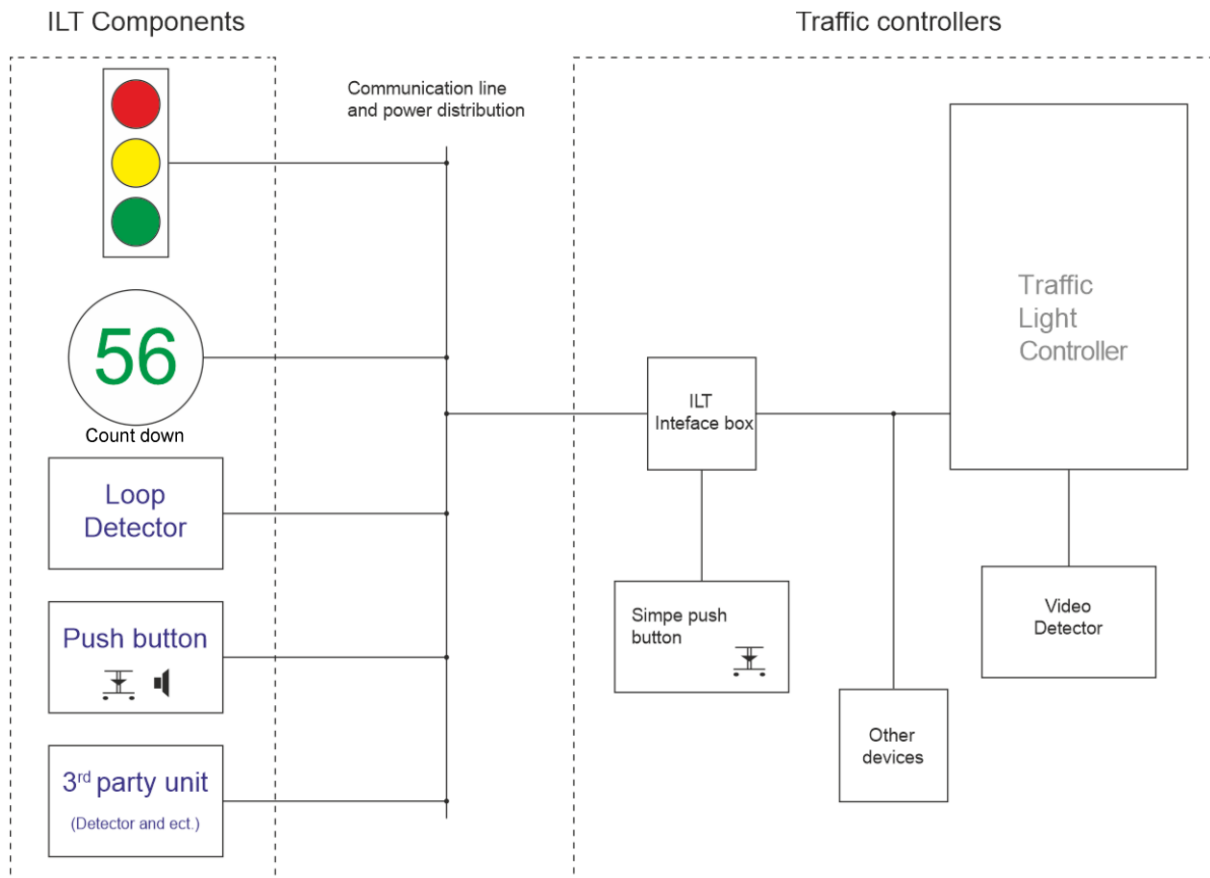


Figure 1 – ILT concept

## 1 Scope

The interface specified in this VDE SPEC is intended for use on permanently installed as well as transportable TLC. In order to be able to fully exploit the technical and economic potential of the decentralized architecture, not only safety-relevant data but also non-safety-relevant data according to the VDE 0832 series of standards are supported.

Optical signalling:

The signal heads developed along this specification provide comfortable functions like the control of luminosity dependent on operational time, to complement the physical degradation of LEDs. Thereby the lifetime can be sustainably extended. By the standardised connection, additional components can be connected without special skills.

Acoustic / tactile signalling:

To enable a consistent wire-saving technology, it is important to include the blind signalling devices, which combine not only the safety-relevant functions of the release but also non-safety-relevant functions such as the time-dependent volume adjustment, the regionally differently handled orientation tone, and the sensor system for the requests.

Sensors:

It is important to include the non-safety-relevant sensors that are independent of the blind signal generators; in the first step, this concerns components such as request buttons and traffic detectors.

The following ILT components are included in the scope of this document:

- ILT Interface box
- Aspects
- Count downs
- Traffic detectors
- Push buttons
- 3rd party units

## 2 Normative References

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

EN 12368, *Traffic control equipment - Signal heads*

EN 12675:2000, *Traffic signal controllers - Functional safety requirements*

EN 50293:2012, *Road traffic signal systems - Electromagnetic compatibility*

EN 50556, *Road traffic signal systems*

EN 61000-4-5:2014, *Electromagnetic compatibility (EMC) - Part 4-5: Testing and measurement techniques - Surge immunity test (IEC 61000-4-5:2014)*

EN 61508 (series), *Functional safety of electrical/electronic/programmable electronic safetyrelated systems*

CLC/TS 50509, *Use of LED signal heads in road traffic signal systems*

EN 61076-2-104, *Connectors for electronic equipment - Product requirements - Part 2-104: Circular connectors - Detail specification for circular connectors with M8 screw-locking or snap-locking*

This VDE SPEC is no normative document; therefor referenced documents are also of informational character but are relevant for the use.

## 3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

### 3.1

#### aspect

single light spot consisting of housing, optics, and electronics

### 3.2

#### ILT component

terminal devices as signal head, push button, etc. The ILT component is also called Slave.

### 3.3

#### ILT interface box

interface for the ILT components provided by the system integrator. Typically, the controller manufacturer. The ILT interface box is also called Master.

### 3.4

#### known state

a state for a ILT component known to be safe to the traffic. I.e. for aspects this state will be off (no light emission).

### 3.5 signal head

signalling unit on roadside, consisting of several light spots called aspects

### 3.6 Unique ID

ID given to the ILT components during manufacturing – shall be unique globally

## 4 Symbols and abbreviations

**Table 1 – Abbreviations and Conventions**

Symbol	Meaning	Unit
ASTRA	Swiss federal roads office	
BASt	German federal highway research institute	
Bit<8:11>	Syntax of bit fields. This example denotes a bit field which ranges from bit 8 up to bit 11, what implies the field consists of 4 bits.	
Byte[0:5]	Syntax of byte arrays. This example denotes a byte array which ranges from byte 0 up to byte 5, what implies the array consists of 6 bytes.	
Byte[5] <0:3>	This example denotes a bit field which is located in byte 5 and which ranges from bit 0 to bit 3.	
CAN	Controller Area Network	
CAN ID	29-bit field within CAN framework	
EMC	Electro magnetic compatibility	
ESD	Electro static discharge	
FIT	Failure in time	h <sup>-1</sup>
FTA	Fault Tree Analysis	
FW	Firmware	
ID	Identifier	
ILT	Intelligent Line Technology	
Inst	Denotations in telegram specifications for information telegrams sent instantaneously if a given event occurs (i.e. a state change in a ILT component).	
Network ID	16-bit field Network address which shall be unique within the ILT network between the ILT interface box and the ILT components. It's a part of the 29-bit CAN ID.	
Numbers / numerical values	decimal: default – all numbers consisting of only digits, without any prefix or suffix hexadecimal: prefix of "0x" or suffix of "h" / "hex" binary: suffix of "b" unless otherwise stated, signed numbers are represented in two's complement	
PFH ( $\lambda_{DU}$ )	Probability of dangerous (undetected) failure	h <sup>-1</sup>
Req	Denotations in telegram specifications for telegrams send to request an action or data.	
Resp	Denotations in telegram specifications for telegrams send in response to a request telegram.	
TLC	Traffic Light Controller	
TLS	Technical delivery terms for roadway stations	

## 5 All ILT devices

### 5.1 General

#### 5.1.1 General requirements

- It shall be easy to configure the system after installation, supporting tools should be allowed.
- A commonly known protocol standard shall be chosen.
- The protocol design shall be open for extensions with respect to future ILT components as well as extended functionality of existing ILT components.
- The design of the protocol including hardware drivers shall support a high degree of flexibility with respect to controller integration.
- During maintenance it shall be possible to replace one ILT component at a time, without having to reconfigure the system. Are more than one ILT component replaced at a time, reconfiguration of the replaced parts may be needed.
- All ILT components shall have one well defined error state, defined as “known state” which has to be entered in case of any error that cannot be reported reliably anymore (which explicitly and especially includes lost communication).
- All failures that can be reliably reported shall be reported, communication shall not be stopped unnecessarily. Defined states of reduced functionality can be entered according to failure severity, i.e. “warning state” or “failure state”.
- It shall be presumed that the ILT interface boxes are placed within the poles carrying the signal heads, pushbuttons, and etc. This implicitly implies cable lengths from the ILT interface boxes to the ILT components less than 15 m. To achieve long allowed cable lengths, the input capacitance of each component (which is like load to the bus) shall be limited.  
The ISO 11898-2 standard specifies a limit of 20 pF per CAN transceiver, but this does not include any EMC / ESD protection, which always adds some capacitance.  
To allow for a reasonable protection of ILT-components, a  
**input capacitance limit of 100 pF per component (differential, CAN-H to CAN-L)**  
is specified
- The system shall be open to development of 3<sup>rd</sup> party ILT components
- The system shall comply with EN 50556 and EN 12675. For Dutch compliance this means that green/green conflicts shall be present for max 100 ms and missing reds shall be reacted to within 200 ms. Signal heads shall comply with CLC/TS 50509 and EN 12368. The max turn-off time of a LED signal head is 50 ms according to CLC/TS 50509. An EN 61508 SIL 3 certification shall be achievable for both the hardware and the software implementation of the chosen protocol.
- Max. number of ILT Components: 32

#### 5.1.2 Communication Interface

CAN BUS: ISO 11898-2; high speed CAN, with extended ID will be used. Data alignment: little endian (unless otherwise specified).

Baud rate: 500 kBd fixed (no automatic baud rate setting)

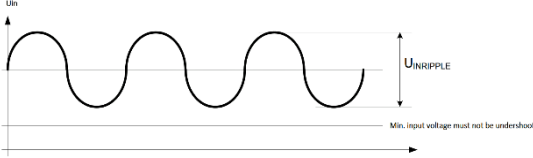
Location of sample point: 75 % – 90 %

Unpowered ILT components shall behave passive/recessive towards the bus. This also applies power failures or component breakdowns leading to a power failure.

#### 5.1.3 Power supply

The power supply for the ILT components shall provide a DC supply voltage, galvanic separated from the mains supply. Table 2 shows the estimated power consumption for various ILT components. Double, or reinforced insulation, and earthen shall be used for the power supply and the ILT-Interface box.

**Table 2 – ILT component power consumption**

		Values			Remark
		Min.	Typ.	Max.	
Supply voltage of ILT-Component	$U_{IN}$	15 V	24 V	29 V	
Average input power of ILT-Component (nominal / average continuous power consumption) <sup>a</sup>	$P_{IN}^a$			2.0 W	Aspect, 200 mm aspect
				4.0 W	Aspect, 300 mm aspect
				12 W	Countdown
				8 W	Push button with acoustics and tactile, or with acoustics only
Maximum / short-time Input current of ILT-Component <sup>a</sup>	$I_{IN}$			500 mA 800 mA	<p>@ <math>U_{IN} \geq 24</math> V @ <math>U_{IN} &lt; 24</math> V</p> <p>Value shall not be exceeded at any time (regardless of PowerUp or later). There is no restriction to any current slope. Resistive loads shall be considered on the IF-Box, also when connecting an ILT-Component due to plug&amp;play.</p> <p><i>NOTE Short spikes impossible to avoid, i.e. from parasitic capacitances caused by overvoltage protection at the supply input, may be slightly higher for a few milliseconds. The range of this parasitic capacitance is expected to be below 10 nF per ILT component.</i></p> <p><i>All other loads, even input filters / capacitors, shall be supplied by any kind of (inrush) current control / delay circuit, ensuring the allowed maximum input current at any time.</i></p>
Ripple on Supply voltage	$U_{INRIPPLE}$			1.0 Vpp 300 mVpp	<p>@ <math>F_{IN} \leq 300</math> Hz @ <math>F_{IN} &gt; 300</math> Hz</p> <p>The specified minimum supply voltage min <math>U_{IN}</math> will not be undershoot!</p> 

<sup>a</sup> While  $P_{IN}$  describes the average input power of an ILT-Component, peak currents (at PowerUp or due to i.e. turning ON optical or acoustic outputs) are defined via  $I_{IN}$ .

### Voltage dips

The target is that during a voltage dip there is no impact to the overall system behaviour.

#### definition of voltage dip

following Table 3 shows the maximum pulse width  $T_{DIP}$  of a voltage dip. voltage values are based on the IEC61000-6-7

**Table 3 – Voltage dip**

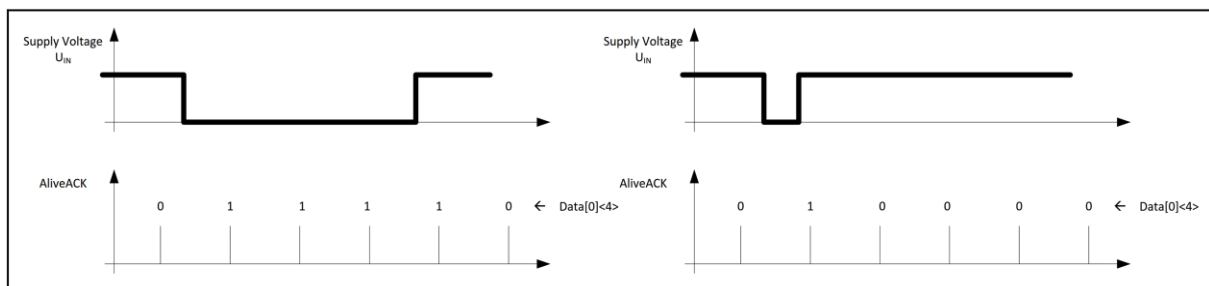
Supply voltage of ILT-Component $U_{IN}$	$T_{DIP}$
$15 \text{ V} > U_{IN} \geq 9,6 \text{ V}$	$\leq 50 \text{ ms}$
$9,6 \text{ V} > U_{IN} \geq 0 \text{ V}$	$\leq 1 \text{ ms}$

For voltage dips with a pulse with  $> T_{DIP}$ : a Powerup Notification will follow

- after a Delay of 20 ms (to prevent the transmission of multiple Powerup Notifications in case of contact bouncing)
- when the Supply voltage rises above min  $U_{IN}$

during a voltage dip

- A momentarily restricted functionality may occur for the duration of the voltage dip (i.e. light OFF at an aspect or no sound at an acoustic) which is covered by the standards (i.e. EN 50556, up to 100 ms) and will not cause any unsafe situation
- The CAN communication shall continue without any restriction
- The occurrence of a voltage dip is communicated to the ILT-Interface box by the AliveAck. During a voltage dip, but at least one time the bit in the AliveAck, Data[0]<4>, is set



**Figure 2 – AliveAck during voltage dip**

### Output Current of ILT Interface Box

The ILT interface box (resp. its power supply system) shall be able to deliver not only the required output current to supply all connected ILT components for normal operation (see definition  $I_{IN}$  in Table 2), but it also shall be able to handle overcurrent in any failure mode of an ILT component in a fail-safe way, so that overall system / traffic controller safety is still ensured, and no additional risks are added to the system.

Realisation is up to the manufacturer of the controller / ILT interface box. For example, this could be done within the ILT-Interface-Box by turning OFF its output voltage before the maximum achievable output current is reached, or by providing a reasonable amount of short-term peak load power while still preventing a continuous overload / short-circuit condition.

Background: ILT-Components with input currents up to 800 mA (according definition  $I_{IN}$  in Table 2) need a fuse in the range of 1,6 A (according IEC). Those fuses need  $>6$  A to blow within 100 ms.

### Cables

The cables connecting the ILT components shall include two power wires and one twisted pair for communication. The twisted pair shall be shielded. The shield shall only be terminated at the interface box. The square of the power wires depends on the actual load of the ILT component. The voltage drops over cables and connectors shall be considered.

Suggested cable type: LIYCY TP 2 x 2 x (0.35 mm, 0.5 mm)

Alternative cable type: Sensor-Actor-Cable SAC-4P

### Connection

The cables are connected to the ILT-Interface Box by a standardised M8 Connector as specified in EN 61076-2-104. The pinning is given in Table 4, with pairing for suggested cable.

**Table 4 – Signals at M8-connector**

Pin number	Signal name	LIYCY TP	Sensor-Actor-Cable SAC-4P	Wires
1	CAN H	brown	brown	Pair 1
2	CAN L	white	white	
3	GND	green	blue	Pair 2
4	+24V	yellow	black	

**EMC – test of transient over voltages (surge) according EN 61000-4-5:2014**

- there is no requirement to the ILT-Components to withstand a transient overvoltage according EN 61000-4-5:2014 on the CAN interface in Normal-Mode (line to line, here CAN H to CAN L). It is the responsibility of the system integrator to set up an environment (cabling and overvoltage protection) to avoid such a test or to keep the residual clamping voltage between CAN H and CAN L  $U_c < 35\text{ V}$   
for example, following solutions are sufficient (only one of the listed methods is required):
- use a shielded cable, connected to earth on both sides (EN 61000-4-5:2014, figure 12)
- use a symmetrical cable (EN 61000-4-5:2014, figure 10) – this requirement should be fulfilled in (almost) all practical cases, as any reasonable CAN / fieldbus cable is symmetrical by design
- reduce the cable length out of a metallic pole to  $<10\text{ m}$  (EN 50293:2012, table 5) – this requirement is fulfilled whenever a regular metallic pole is used and the ILT component is installed in or at the pole

Reason: Performing a test in Normal Mode with  $U_{pk} = 0,5\text{ kV}$  and  $R_i = 42\text{ Ohm}$  makes it necessary to use overvoltage protection devices with high pulse rating (12 Apk,  $>450\text{ W}$ ). Such devices are available but only with high capacitance, which makes the use with 500 kBd and 32 ILT-Components over 15 m cable impossible. On the other hand, when using an overvoltage protection suitable for high-speed CAN, the pulse rating of such a device is much too small ( $<8\text{ Apk}$ ,  $<350\text{ W}$ ) and the ILT-Component is not protected.

**5.1.4 Limited functionality**

It shall be possible to differentiate the functionality of the ILT components developed for different customers. The limited edition of a ILT component may be realized by programming different versions of the FW into the component or by configuring the component. This shall be done during the production of the ILT component. It is important that the full function set of a limited edition ILT component, cannot be accessed in any way, even by hacking.

The function set of the ILT component is not defined at protocol level. The functionality/features of ILT components are defined during manufacturing and are specified by the product designator. It shall be possible to read out the product designator via the protocol.

**5.1.5 Document & Protocol Version**

This chapter describes how versioning of the ILT communication specification is handled, as there are multiple items which have to be versioned:

- the ILT specification document as VDE SPEC 90013
- the document containing the list of registered ILT manufacturers with the manufacturer ID.
- the ILT communication protocol itself, as it is implemented technically

The following rules apply to these versions and their handling:

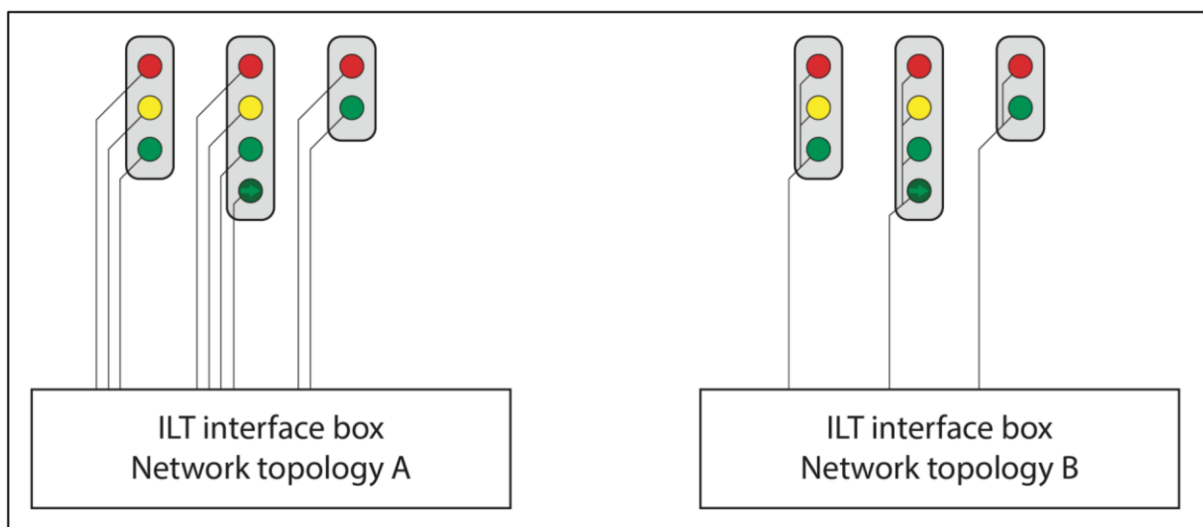
- 1) Version numbers of **documents** and the **protocol specification** are **independent** from each other. Document and Protocol versions are incremented individually.
- 2) Document versions:
  - a) If a document is **edited in any way** – no matter if the change has a functional effect on the protocol or is just a change in explanations or corrections – its version number is increased

- b) The document containing the list of registered ILT manufacturers is updated individually to this VDE SPEC 90013 version and vice versa.
- 3) Protocol versions:
- a) The protocol version is increased within the documents whenever any functional change is specified and is used by the ILT components / interface boxes as soon as features according to this specification version are implemented.
  - b) The protocol is kept completely backward-compatible, changes have to be only extensions (i.e. additional data fields, additional commands, ...) but no changes to or redefinitions of existing commands and/or data.

## 5.2 Overall Description

### 5.2.1 Network topologies

The chosen network topology has a great impact on installation, configuration, and maintenance. The two alternatives shown in Figure 3.



**Figure 3 – Network topologies**

- For network topology A, all the ILT components are connected directly to the ILT interface box. This simplifies the installation and configuration but increases the cost and size of the ILT interface box and the number of cables.
- For network topology B, all ILT components within an aspect are connected in parallel. This gives an advantage of having fewer connectors at the ILT interface box but complicates the configuration and service. The same considerations are valid for other types of ILT components such as push buttons and detectors.

Due to these two topologies and the required flexibility of the system, the termination resistor shall not be placed inside the ILT components. The termination shall be a part of the harness.

### 5.2.2 Network addressing

In order to avoid addressing conflicts and mistakes during configuration, installation, and service, all ILT components shall be given a component designator, separated into a standardized and a unique ID (Table 5). The component designator is assigned during production. It shall not be possible to modify the component designator during the lifetime of the ILT component.



**Table 5 – Component designator**

Standardized ID:	Data	Number of bits
Device-type	Aspect, Push button, etc.	8 [256 device types]
Sub-type	red, yellow, green, etc.	8 [256 subtypes]
Manufacturer ID	SWT, STS, SWF, etc.	8 [256 ID's]
Unique ID:	Data	Number of bits
Manufacturer specific serial number	Serial number, freely definable by the manufacturer.	39 bit for unique serial number

Manufacturer ID (Table 6), Device-type (Table 7), and Sub-type (Table 8) are specified as follows.

**Table 6 – Manufacturer ID specification**

Manufacturer ID	
0	Don't care
1	Manuf. #1 (reserved)
2	Manuf. #2 (reserved)
3	Manuf. #3 (reserved)
4-255	Free

NOTE Manufacturers will be added with a manufacturer identification and the ID to a registration list as soon as they start developing / offering ILT components.

The registration list and the description of the registration procedure are hosted as “Register of Manufacturer Identifier for VDE SPEC 90013” [2] at the VDE platform:

[www.dke.de/manufacturer-identifier](http://www.dke.de/manufacturer-identifier)

**Table 7 – Device-type specification**

Device-type (according to its main appearance / primary feature)	
0	<i>reserved / start of class “Aspects”</i>
<b>1</b>	<b>ASPECT</b>
2..9	Aspect combinations and variants (i.e. tramway, Bern type, car2x, ...)
10..19	<i>reserved</i>
20	<i>reserved / start of class “Pedestrian push buttons”</i>
<b>21</b>	<b>Pedestrian push button (without acoustic/tactile signalling)</b>
22	Pedestrian push button with included acoustic <sup>a</sup> and tactile signalling
23	Pedestrian push button with tactile signalling (without acoustic)
24	Pedestrian push button with acoustic <sup>a</sup> signalling (without tactile)
25..29	Pedestrian push button combinations and variants
30..39	<i>reserved</i>
40	<i>reserved / start of class “Acoustic signals”</i>
<b>41</b>	<b>Acoustic signal</b>
42	Acoustic signal with included tactile signalling
43..49	Acoustic signal combinations and variants
50..59	<i>reserved</i>

Device-type (according to its main appearance / primary feature)	
60	<i>reserved / start of class "Traffic sensors / detectors"</i>
<b>61</b>	<b>Traffic sensor</b> / loop detector with single loop
62	Traffic sensor / loop detector with double loop and simple classification, speed, length, direction
63	Traffic sensor / loop detector with double loop and accurate classification, speed, length, direction
64	loop detector with double loop for bicycle detection
65	Traffic sensor / video detector
66	Traffic sensor / thermal detector
67	Traffic sensor / radar detector
68-79	Traffic sensor / combinations and variants
80	<i>reserved / start of class "Environment sensors"</i>
<b>81</b>	<b>Environment sensor</b>
82..89	Environment sensor combinations / variants
90..99	<i>reserved</i>
100	<i>reserved / start of class "Communication devices"</i>
<b>101</b>	<b>Communication device</b>
102..109	Communication device combinations / variants (i.e. public transport, ...)
110..119	<i>reserved</i>
120..244	reserved for future device types
245..249	reserved for manufacturer specific test devices
250..254	reserved for future command set extensions
<sup>a</sup> A push button with acoustic signalling shall always support at least orientation AND walk sound.	

NOTE The supported command sets have to be requested by the *GetProfile* command, see chapter 5.3.1. The functionality provided and supported by any device is defined by the combination of the properties "Device Type", "Sub type", the supported command sets and the **generic functionality explicitly defined** within the ILT specification (this document and all parts). All devices with extended functionality (datasheet-defined) can/shall be able to be operated identical to "generic" devices using the generic command definitions, so full compatibility is provided to an interface box supporting the generic feature sets. In case of unknown "extended" device types, compatibility is provided by resorting to the "basic" device type (i.e. a push button identifying as device type 25 can be operated by assuming a device type of 20).

**Table 8 – Sub-type specification**

Sub-type		
Aspect	0	Not used
	1	White
	2	Red
	3	Yellow
	4	Green
	5-255	Not used

Sub-type		
Push button	0	Not used
	1	generic Push-button functions (without optical information)
	2	Push-button with optical information
	3	Push-button with optical information and additional demand classes
	4	Push-button with optical information and additional demand classes, Acoustic can include additional Speech output
	5-255	Not used
Acoustic signal	0	Not used
	1	Orientation Sound only (logical "red")
	2	Walk Sound only (logical "green")
	3	Orientation and Walk Sound
	4	Orientation and Walk Sound and additional acoustic / speech output
	5-255	Not used
Traffic Sensor	0	Not used
	1	<b>Traffic sensor</b> / loop detector with single loop
	2	Traffic sensor / loop detector with double loop and simple classification, speed, length, direction
	3	Traffic sensor / loop detector with double loop and accurate classification, speed, length, direction
	4	loop detector with double loop for bicycle detection
	5	Traffic sensor / video detector
	6	Traffic sensor / thermal detector
	7	Traffic sensor / radar detector
	8 - 255	Not used

NOTE "additional demand classes" are any demand / detection features that go beyond the basic demand features for pedestrians and visually impaired as described in chapter 7.1 for command "PushbuttonStatusAck".

In addition to this it shall be possible to give the ILT components a network address, also called "network ID". This network ID is defined to be a part of the CAN ID and shall be unique within the local CAN network. The network ID shall fit into the 29-bit standard CAN ID field together with the priority, command, and direction fields. The usage of the CAN ID for the ILT protocol is specified below:

<b>Priority</b> 3 bit <28:26>	<b>Network ID</b> 16 bit <25:10>	<b>Type</b> 1 bit <9>	<b>Command</b> 8 bit <8:1>	<b>Direction</b> 1 bit <0>
----------------------------------	-------------------------------------	--------------------------	-------------------------------	-------------------------------

Figure 4 – 29-bit CAN ID

	29 bit ILT CAN identifier																												
Bit number	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Meaning	Priority			Network ID																Type	Command								Dir.
Byte grouping	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Figure 5 – 29-bit CAN ID in bit and byte scale

Priority: 0-7

0 = Error priority (PrioError) for severe error message

1 = High priority (PrioHigh) for safety-critical commands and safety-critical error messages.

- 2 = reserved
- 3 = Normal priority (PrioNorm) for non-safety-critical commands and warning messages
- 4 = reserved
- 5 = Low priority (PrioLow) for retrieving component information, warnings, and firmware download.
- 6 = reserved
- 7 = PowerUp priority (PrioPowerUp)

NOTE The behaviour of the ILT-Component when a command is received with a priority other than specified shall be to accept and execute the command even with the wrong priority and send the ACK with the specified priority.

*Network-ID: 0 – 0xFFFF*

- 0x0000 = Broadcast-identifier (Broadcast-ID).
  - Only a limited number of commands can be sent as broadcast.
  - Only the traffic controller, by means of the ILT interface box, which is the master, can send commands as broadcast.
- 0x0001 to 0xFFFFE = Direct-identifier (Direct-ID).
  - - These Network-ID's can be assigned to ILT-components.
  - - These Network-IDs are indicated by priority 1, 3 or 5 (PrioHigh, PrioNorm, PrioLow).
- 0x0001 to 0xFFFFE = PowerUp-identifier (PowerUp-ID).
  - This PowerUp-identifier is generated by the ILT-component as a random number
  - These Network-IDs are generated by the ILT-component during the PowerUp sequence or for severe error messages.
  - These Network-IDs are indicated by priority 0 or 7 (PrioError, PrioPowerUp).
- 0xFFFF = ID-less-identifier (ID-less).
  - This identifier is used by the IF-Box for addressing before a network-ID is assigned (including the ACK from the ILT-Component)
  - These Network-IDs are indicated by priority 7 (PrioPowerUp).

*Type of telegram 0-1*

- 0 = Regular telegram
- 1 = Redundant telegram

*Command: 0-FFh*

- 0x00 to 0x55 = Reserved for system commands, shall be implemented for all ILT components.
- 0x56 to 0xA9 = Standardized commands for individual device types.
- 0xAA to 0xFF = Manufacturer specific commands for individual device types.

*Direction: 0-1*

- 0 = Master (ILT-Interface-box) to Slave (ILT-component)
- 1 = Slave to Master

Programming the Network ID is the responsibility of the system integrator. The system integrator programs all 16 bits of the network ID field freely. It may contain information such as type of ILT component, colour of aspect, Group number of aspect, and shall form a unique network address, but this is up to the system integrator. The ILT interface box hands over the Network ID to the ILT component at each power up sequence (details see chapter 5.2.3.1).

### 5.2.3 CAN packet

CAN ID 29 bit	DLC 4 bit	Data[0] 8 bit	Data[1] 8 bit	Data[2] 8 bit	Data[3] 8 bit	Data[4] 8 bit	Data[5] 8 bit	Data[6] 8 bit	Data[7] 8 bit
------------------	--------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------

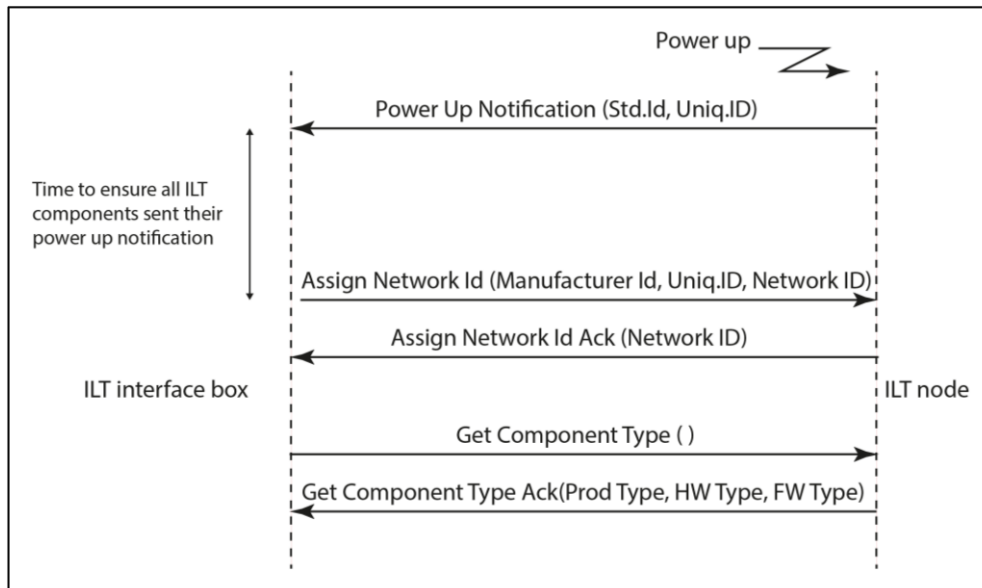
Figure 6 – CAN Packet

- For Acknowledges without data the 8 bit command shall be repeated in *Data[0]* for packet verification purposes (this allows CRC checking and sending inversed bits in redundant packet).
- The DLC shall be checked by ILT-Component against telegram length. In case of inconsistency (refer to backward compatibility described in chapter “Document & Protocol Version”) between DLC and CMD, the command shall be ignored unless otherwise specified in this document(s).

### 5.2.3.1 Power up addressing, identification, Hot Plug

→ For a description of the commands, see section 5.3.1 .

The *GetComponentType* command provides extra information of the ILT component. The Command *Powerup Notification* is repeated by the ILT-component every 1 second in case of missing answer from the traffic-controller (*AssignNetworkID* or *Identify*).



(all notes count for both, topology A and topology B)

NOTE 1 From the instant of time the command *AssignNetworkIDAck* has been sent, no more commands using priority 7 (PrioPowerUp) are accepted anymore by the ILT-component.

NOTE 2 At power-up, the ILT-component:

- initializes the masks and filters in a way that they are disabled.
- stops the timer for the Alive-timeout.

NOTE 3 With receiving of command *AssignNetworkID*, the ILT-component starts the timer for the Alive-timeout. Thus, it starts responding to Alive broadcasts received from the ILT interface box, so the interface box and / or the traffic controller can monitor any present ILT component.

**Figure 7 – Basic start-up sequence**

The 16-bit PowerUp-ID is a hash value based on the Manufacturer-ID and the Unique-ID. Only in case of transmission error (resulting in a telegram collision and detected by an error frame) the ILT-Component will create a new hash value.

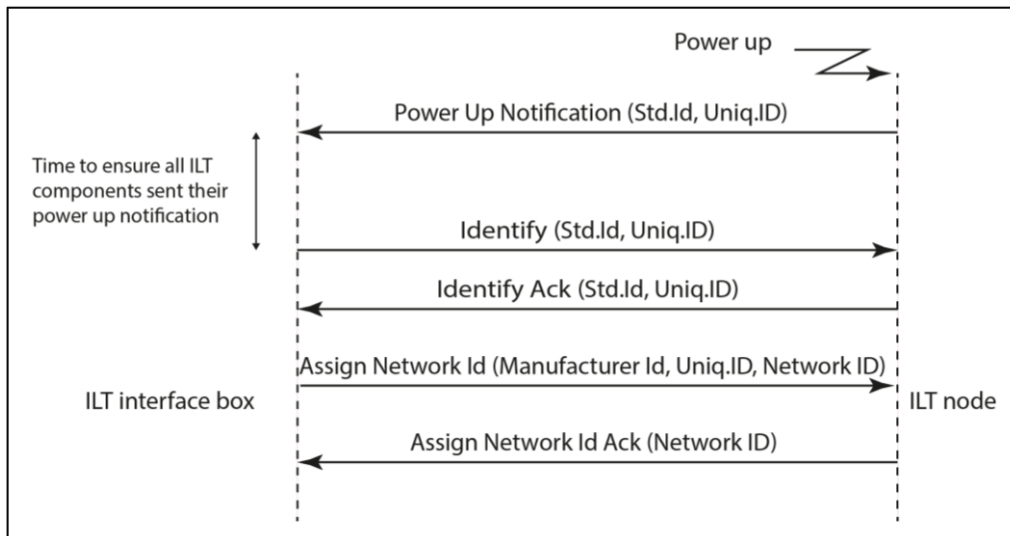
$T_{PUN} < 200$  ms Max. time between power-up (input voltage greater than or equal min.  $U_{IN}$ ) and sending out command *Powerup Notification* or the command *StuckOnError* (without considering delay due to collisions on CAN bus). It's recommended to use a random time delay for sending out command *Powerup Notification* in order to reduce the probability of CAN bus collisions. However, this random time delay shall not violate  $T_{PUN}$ .

Identification procedure for topology B:

For topology Type B an additional identification process is needed. The following diagram shows the start-up sequence extended with an identification procedure.

By the reception of the identification command, the SLT component must perform an action that makes it possible to identify its physical position.

This could be done by flashing a lantern shortly or make a push button beep, in order to be able to identify the physical position of the SLT component before assigning a network address. This feature would be mandatory for the network topology B defined in section 5.2.1.



NOTE 1 For specification of the action performed by the ILT-component, see the related chapter of the ILT-component.

NOTE 2 Independent of the device type, the resulting action shall be activated only once within a two second time period and shall be kept as short as possible to ensure it cannot be interpreted as any GO, FREE or WALK signalisation on the traffic junction (i.e. green aspect or walk-sound on an acoustic signal).

For components with safety requirements, the activation period is strongly recommended to last less than 100 ms, the defined process safety time.

The exact *Identify* behaviour has to be specified within the ILT component product datasheet.

When the reception of an *Identify* command falls into that period of 2 seconds after having received a previous *Identify* command, then the related action is not started and no answer-telegram is sent by the ILT-component.

In other words: a subsequent *identify* telegram is only processed if a time period of two seconds has passed after the previous *identify* has been received.

**Figure 8 –Start-up sequence including identification**

The following state machine summarizes the power-up-sequence of the CAN communication from the ILT component's point of view.

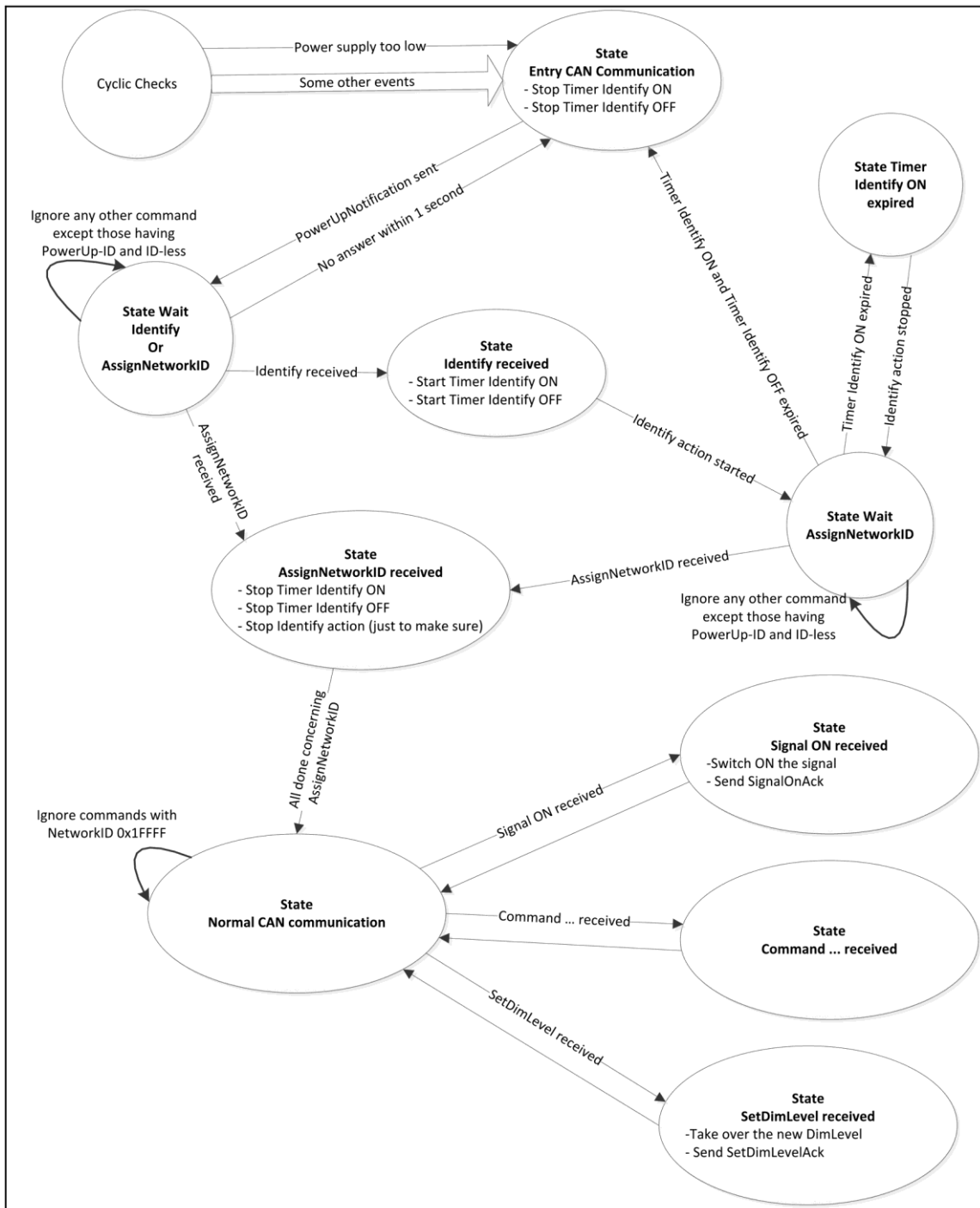
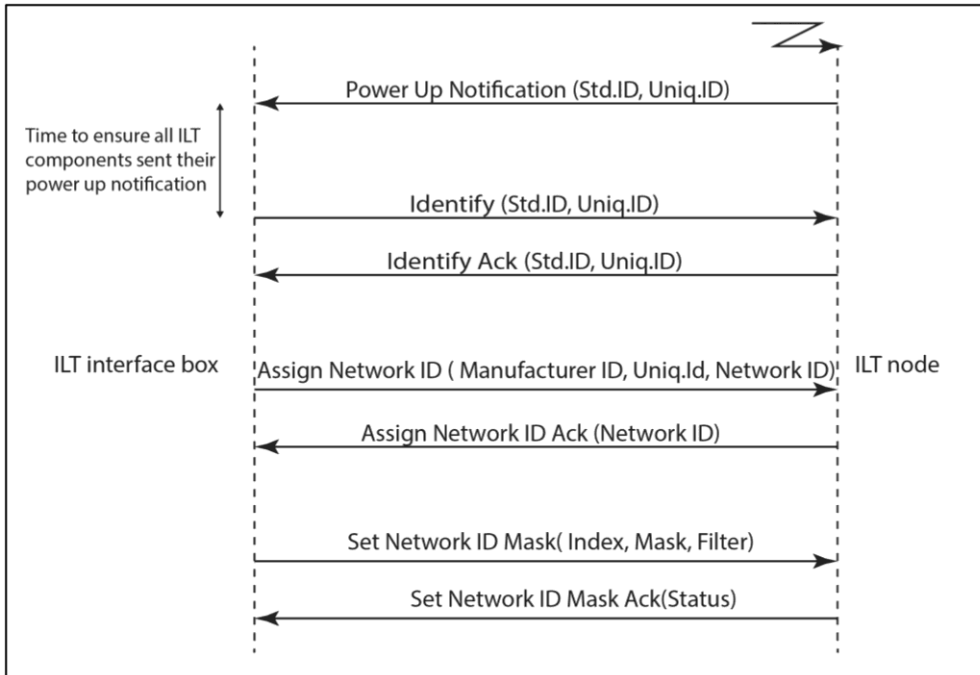


Figure 9 – CAN-bus power-up state-machine

### 5.2.3.2 Addressing and grouping features

➔ For a description of the commands, see section 5.3.1.

In order to utilize the CAN grouping facilities, a *SetNetworkIDMask* command for programming the CAN filter within the CAN controller of the ILT components shall be available via the protocol.



**Figure 10 – Start-up sequence including identification and command SetNetworkIDMask**

The Network-ID is defined by the system integrator and shall be programmed into the ILT-components at each power up. If the addressing is done in a structured way, it will be possible to utilize the group addressing features of the CAN controller. An example is given in Table 9.

**EXAMPLE** Consider the sample Network-ID structure in Table 9, its definition is completely free to the system integrator (controller manufacturer):

**Table 9 – Example using Network IDs**

Type 4 bit	Subtype 4 bit	Group 4 bit	ID 4 bit
Type = 0..15 0 = don't care 1 = Aspect 2 = Pedestrian push button 3 = Acoustic signal 4 = Traffic sensor	Aspect: 0=don't care 1=red 2=yellow 3=green  Pedestrian push button: No subtype defined  Acoustic signal 0=don't care 1=red 2=not defined 3=green  Traffic sensor: 0 = don't care 1 = induction loop 2 = video camera 3 = thermal camera 4 = radar detector	Group number = 1..15 0=don't care (= any)	Individual ID = 1..15 0=don't care (= any)



This layout enables broadcasts for groups of components, such as:

SignalOn (ID=0x1100) => turn on all red aspects

SignalOff (ID=0x1000) => turn off all aspects (regardless of colour)

SignalOn (ID=0x2320) => turn on the green sound in all touch sounds belonging to group 2

SignalOn (ID=0x2125) => turn on the red sound touch in group 2 sound with ID=5

The CAN filters should be set up as follows for the aspects and touch sounds:

Example for aspects: A yellow aspect belonging to group 4 with an ID = 7 would use a Network-ID =0x1247 and interpret it as follows:

Filter[1] = 0x01247 - Individual address of aspect 7 in group 4

Filter[2] = 0x01240 – Any yellow aspect in Group 4 is addressed

Filter[3] = 0x01200 – Type and sub-type (yellow aspect) is addressed in any group, with any ID

### 5.2.4 Safety integrity

The safety integrity of the total system shall be evaluated according to EN 50556, EN 12675, and EN 61508 SIL-3. The latter is optional. EN 50556 together with EN 12675 sets some timing restraints for the signal supervision. In general, a green/green conflict shall be prevented within 100 ms (Dutch requirement), and missing signals shall be handled within 200 ms allowing 50 ms for an LED signal to turn off.

⇒ Thus, process safety time for aspects is defined to be 100 ms

EN 50556 refers to EN 50159, where section 5 identifies threats to be considered and the table *Safety measures* lists measures against the various threats. This shall be considered for the safety aspects of the protocol implementation.

Safety and safety certification for the components is in the responsibility of the manufacturer of each component. To enable the system integrator to design a safe system it is necessary to specify all application relevant safety-related requirements.

⇒ Therefore, the description/specification of the component needs to include the safety related application conditions.

In the traffic light control system in total, the rate of a hazardous signalling (i.e. probability of dangerous failures per hour = PFH) needs to be below  $10^{-7}$  per hour for a SIL 3 system according EN 61508 and referenced by DIN VDE 0832 standards. Depending on the number of components at an intersection and on its complexity, this PFH is valid for all system components together and each of them will contribute to this PFH.

⇒ The requirement for a Probability of dangerous undetected errors (PFH) per hour threshold is dependent on the degree of the expansion of an intersection, a single threshold for a maximum PFH value can therefore hardly be specified.

Nevertheless, to dimension the entire system, the result of the fault tree analysis (FTA) of each component, i.e. the PFH of each component, is necessary as input to the FTA of the TLC. These PFH values shall be included in the description of the component. It shall be structured according to the hazardous failures

⇒ given in Table 10 and accompanied with the related proof test interval (PTI).

**Table 10 – PFH values necessary as input to FTA of TLC**

State of Signal	State reported	Description	Justification/Remark
Signal unintended off	reports on	Signal is NOT ON (as intended) but reports on CAN that it is ON	Red hazardous
			Yellow not hazardous
			Green not hazardous
			Acoustic not hazardous
			Tactile not hazardous

State of Signal	State reported	Description	Justification/Remark	
Signal unintended on	reports off	Signal is NOT OFF (as intended) but reports on CAN that it is OFF	Red Yellow Green Acoustic Tactile	hazardous, if combined with Yellow hazardous, if combined with Red hazardous hazardous hazardous
Signal stuck at ON	not reported	Signal state is recognised as ON but cannot be reported	Red Yellow Green Acoustic Tactile	not hazardous not hazardous hazardous hazardous hazardous
Signal stuck at ON	reports on	Signal state is recognised as ON (stuck on) and is reported within the Alive ACK	Red Yellow Green Acoustic Tactile	not hazardous not hazardous hazardous, if no reaction of TLC is possible hazardous, if no reaction of TLC is possible hazardous, if no reaction of TLC is possible

REMARK For combined acoustic and tactile signals the values are regarded as independent, separate values are expected. As no safety related stop-signal exists, the state "Signal unintended off" is not requested.

### 5.2.4.1 Safety features

For the data transmission a white channel approach is taken, and additional measures against message corruption are implemented. The Table 11 defines various transmission failures and the measures that shall be taken against those.

Table 11 – Safety measures

No	Failure	Failure Type	Effect	Safety Requirement or measure	Comment
1	Corruption (single bit destroyed)	A	A new message can be produced which can cause malfunctions.	Check with a CRC	CAN bus provides a Hamming distance (by means of CRC) of 6, all single-bit errors are detected and corrected
2	Corruption (Lots of bits destroyed)	A	A new message can be produced which can cause malfunctions.	Duplication of message and comparison of both messages (redundant transmission), in addition to integrity measures provided by CAN bus	
3	Malfunction of the network (i.e. broken wire), entity	A	A safety related message cannot reach the intended ILT component	Send messages periodically (alive telegram) and use a watchdog	Entity is responsible for going into safe / known state
4	Malfunction of entity (of a receiver)	B	A safety related message cannot reach the intended ILT component	Send messages periodically (alive telegram) and use a watchdog	Entity is responsible for going into safe / known state
5	Malfunction of the network access unit (CAN transceiver)	B	A message is received, but the content of the message does not correspond to the expected values.	Use of a watchdog, Duplication of message and comparison of both messages (redundant transmission), life telegrams will also be affected and lead to reaction	

No	Failure	Failure Type	Effect	Safety Requirement or measure	Comment
6	Loss of a message	A	A safety related message cannot reach the ILT component	Use of acknowledge telegram, Duplication of message and comparison of both messages (redundant transmission), sequence counter for cyclic <i>Alive</i> telegrams	2 telegrams AND ACK telegram are missing, which is discovered, for <i>Alive</i> telegrams a missing sequence counter value is detected
7	Insertion of a message	A	An old message, which has been stored for a certain time reappears, or the message is doubled, or another entity adds a new message.	Use of acknowledge telegram, Duplication of message and comparison of both messages (redundant transmission), sequence counter for cyclic <i>Alive</i> telegrams	Detection at ILT interface box: ACK telegram received without a telegram transmitted, for <i>Alive</i> telegrams an old / added / repeated sequence counter value is detected
8	Repetition of a message	A	An old message, which has been stored for a certain time reappears, or a message is doubled by the ILT component.	Message ID in the payload, duplication of message and comparison of both messages (redundant transmission), acknowledge of repeated message to alleged sender makes repetition obvious for sender, sequence counter for cyclic <i>Alive</i> telegrams	For <i>Alive</i> telegrams an old / repeated sequence counter value is detected by the ILT interface box
9	Repetition of a message	A	An old message, which has been stored for a certain time reappears, or a message is doubled by the ILT interface box.	Message ID in the payload, duplication of message and comparison of both messages (redundant transmission), acknowledge of repeated message to alleged sender makes repetition obvious for sender, sequence counter for cyclic <i>Alive</i> telegrams	For <i>Alive</i> telegrams a repeated sequence counter value is detected by every ILT component
10	Wrong sequence of messages	C	The sequence of messages differs from the actual one.	---	Design decision: All safety-relevant telegrams are self-contained; thus, sequence is not relevant.
11	Delay of a message	B	The message may be delayed due to a network entity. The value is not valid anymore.	Use of a time-out.	If no valid <i>Alive</i> telegrams are received within the process safety time of 100 ms, the receiver (ILT component) enters the know state.
12	Masquerade	A	An entity claims to be a different entity.	Use of acknowledge telegram, payload comparison against identifier, Duplication of message and comparison of both messages (redundant transmission). In case of <i>Alive</i> telegrams, duplicates will be detected by the ILT interface box.	If no valid <i>Alive</i> telegrams are received within the process safety time of 100 ms, the receiver (ILT component) enters the know state.
13	Non-safety related message	A	A non-safety related message is considered as a safety-related message and results in a non-intended action at a ILT component.	Message ID in the payload, Duplication of message and comparison of both messages (redundant transmission)	

NOTE Sequence numbers as safety measures are only needed for the cyclic *Alive* telegrams due to other implemented measures mentioned above.

- Sequence numbering prevents the following incidents:
- Bad software repeats the same command.
- Protocol stack switches the order of commands.
- A ILT component pushes a wrong ID to the bus.
- A command is missing.

*The consequence of these misbehaviours – **for all telegrams except the Alive mechanism** – is that the resulting signal “picture” may be dangerous to the traffic. The main traffic controller shall collect the actual signal states within the process safety time of 100 ms and check for dangerous situations. This checking shall be redundant for SIL 3. Thus, errors caused by repetition, deletion, insertion, and re-sequencing are disclosed by the main controller safety checking within the process safety time. While each ILT component is responsible for the correct processing of commands, and the correct setting of the state field within the AliveAck, the responsibility of the ILT interface box and or the controller is to monitor the combination of the actual state of the signals.*

#### **5.2.4.1.1 Sending telegrams redundantly**

For EN 61508 SIL 3 compliance, all safety relevant telegrams shall be sent redundantly.

- The first telegram sent is the "regular telegram", and the second telegram is the "redundant telegram".
- The transmission shall always be started with the regular telegram followed by the redundant Telegram.
- The entire payload of the redundant telegram shall be sent inverted<sup>1</sup>. This applies also to the Ack telegrams.
- The distinction between regular and redundant telegram is made by the *Type* bit of the CAN ID.
- To be successfully accepted and processed, the regular- and redundant-telegrams shall be sent within a time window of 10 ms and in immediate succession (a command, consisting of regular and redundant telegram, shall be transmitted completely before sending another safety relevant command to the same component(s)).
- The Ack for both telegrams shall be sent after the ILT component has processed the information. A Nack shall be send immediately, if the redundant telegram is not received within the time frame, or the sequence is wrong (the redundant telegram is received prior to the regular telegram, or a different safety-related telegram is received before the redundant telegram of the previous command – in this case the Nack telegram can be delayed by about 10 ms which is equivalent to the timeout period (reason: avoiding two Nack telegrams for the same root cause)).
- The processing of the received command shall only be started if both the regular- and the redundant-telegrams are consistent with respect to each other.
- For an example of sequencing of multiple commands, please see 5.4.4.

---

<sup>1</sup> The payload shall be inverted as follows: The order of data bytes and data bits shall be reversed, and the data bits shall be inverted.

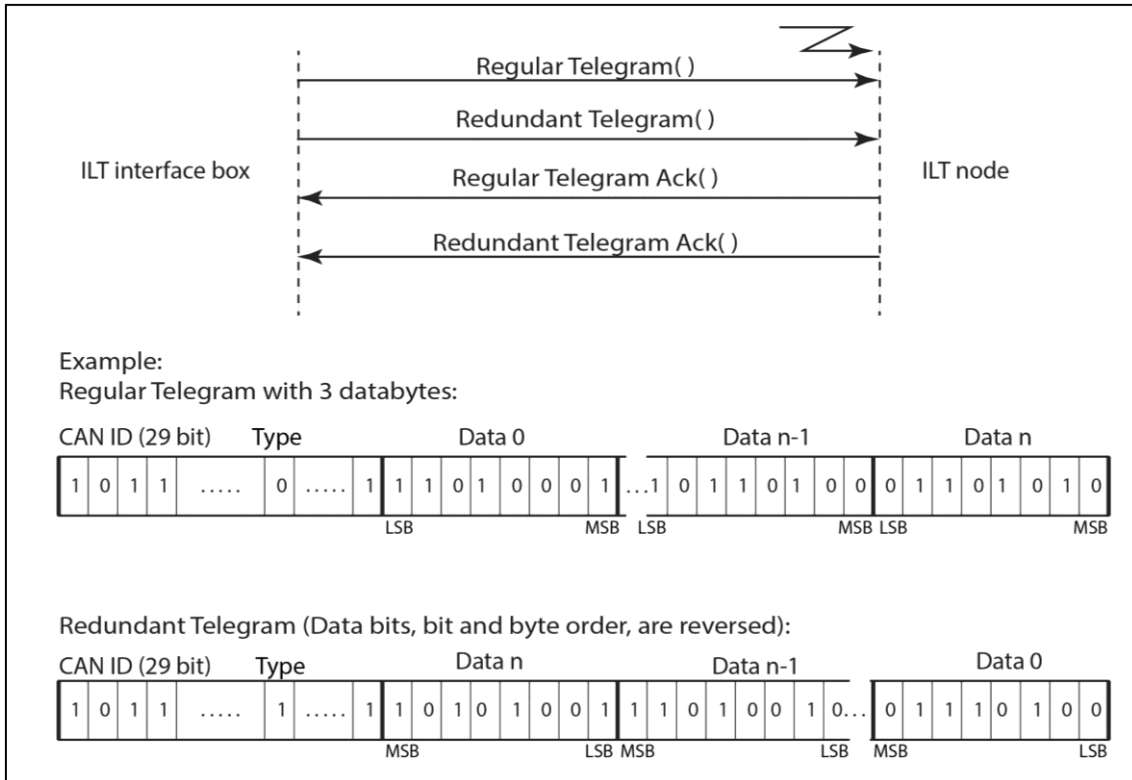


Figure 11 – Sending telegrams redundantly

#### 5.2.4.1.2 Answer (Ack / Nack) for safety-relevant-telegrams

The answer-telegram for safety-relevant-telegrams consists of at least a Status-byte in the payload which is split into a common part and an individual part. The bit-mapping of the common part is shown in Table 12 below.

Table 12 – Common Status in safety-relevant telegrams

Status <7>	Status <6>	Status <5>	Note
0	don't care	don't care	Ok
1	0	0	Error: Redundant telegram received first
1	0	1	Error: Regular telegram received doubled
1	1	0	Error: Data-mismatch between regular and redundant telegram or wrong data in both telegrams.
1	1	1	Error: Timeout of redundant telegram or other safety-related command received before redundant telegram

The bit-mapping in Status<0:4> is individual to the transmitted command. As an example, see command *SetOperationParameterAck*.

The following state-machine show the program-flow of safety-relevant-telegrams.

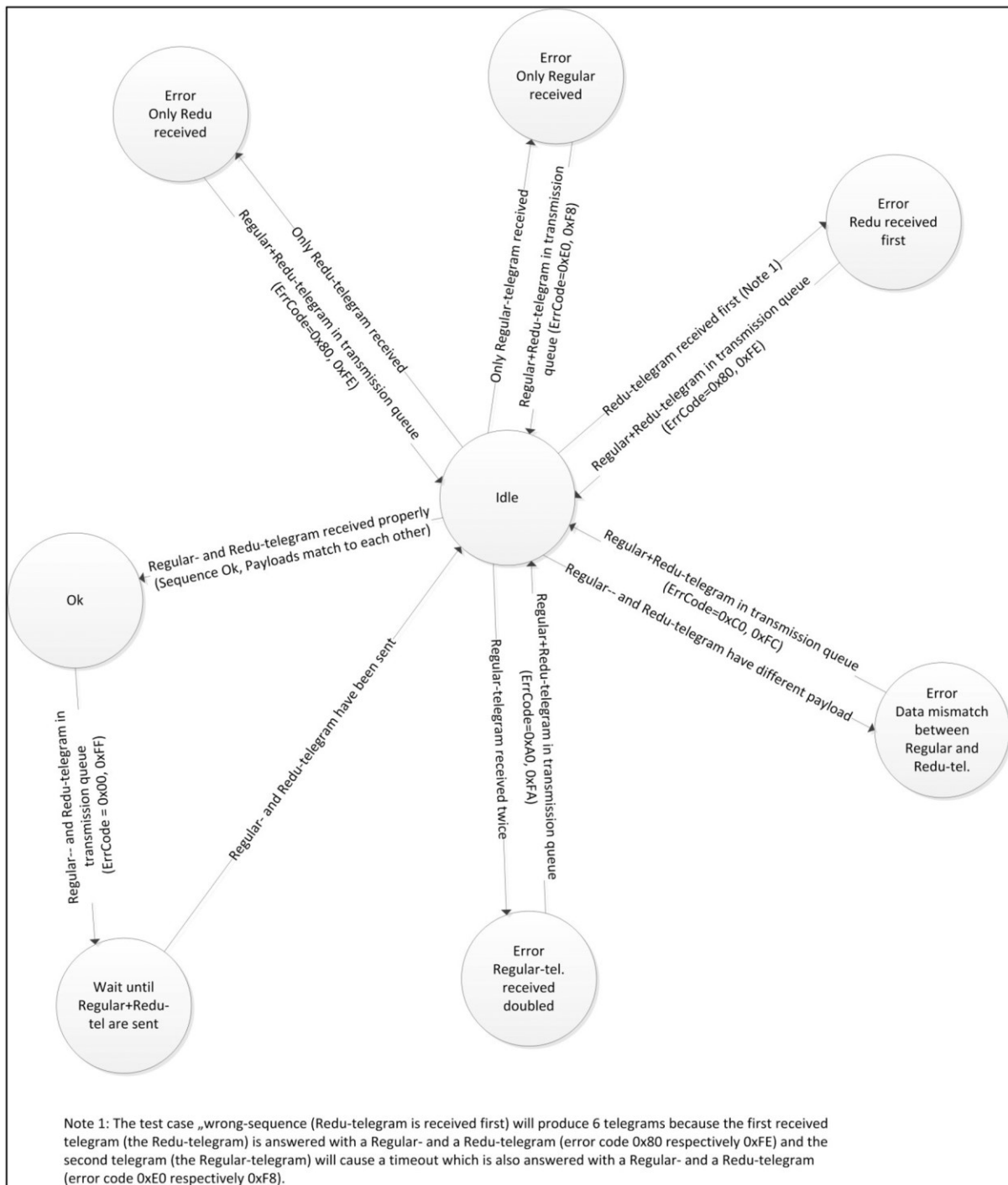


Figure 12 – State-machine for safety-related telegrams

### 5.2.4.1.3 Alive telegram for ILT components

The ILT interface box needs to know whether or not the ILT components are alive. If not, the ILT interface box (system master) shall take proper action(s) in order to prevent a dangerous situation. Therefore, the master shall send the command *Alive* as a cyclic broadcast. The Alive telegrams Table 13 are important for critical components to allow for functional safety, but to keep an overall system state of present components, any ILT component shall take part in the Alive mechanism.

- If an ILT component for any reason does not receive the *Alive* broadcast for more than 100 ms, it shall enter its known state.
- If the telegram *AliveAck* is not received by the ILT interface box, it shall register a fault for the non-responsive ILT-component(s).

- If the master does not receive **at least one** *AliveAck* from a safety critical ILT-component within the process safety time of **100 ms**, the master shall register a fatal fault at that component and act accordingly.
- The observation of the communication timeout will start by the ILT-component immediately with sending *AssignNetworkIDAck*.
- Within a time slot of 100 ms, the master shall be able to decide whether an ILT-component is alive or not (for EN 50556 and EN 12675 compliance).
- The ILT interface box shall vary the sequence counter in any way, using the whole value range (i.e. incrementing) in every Alive broadcast cycle, and check that all ILT components reply with the most recent value within their *AliveAck* responses.
- The ILT component shall check the sequence counter against immediate repetitions. If a repetition is detected, the 100 ms safety timer is NOT retriggered, so that after 100 ms of identical Alive sequence counter values the ILT component enters the known state. However, the ILT component does not make any other assumption on the behaviour of the sequence counter.
- This mechanism covers the cases: “loss”, “insertion”, “repetition”, and “masquerade” of 5.2.4.1. The previously mentioned consequences and responsibilities of both ILT interface box and ILT components ensure the compliance to these requirements.

**Table 13 – Alive telegrams**

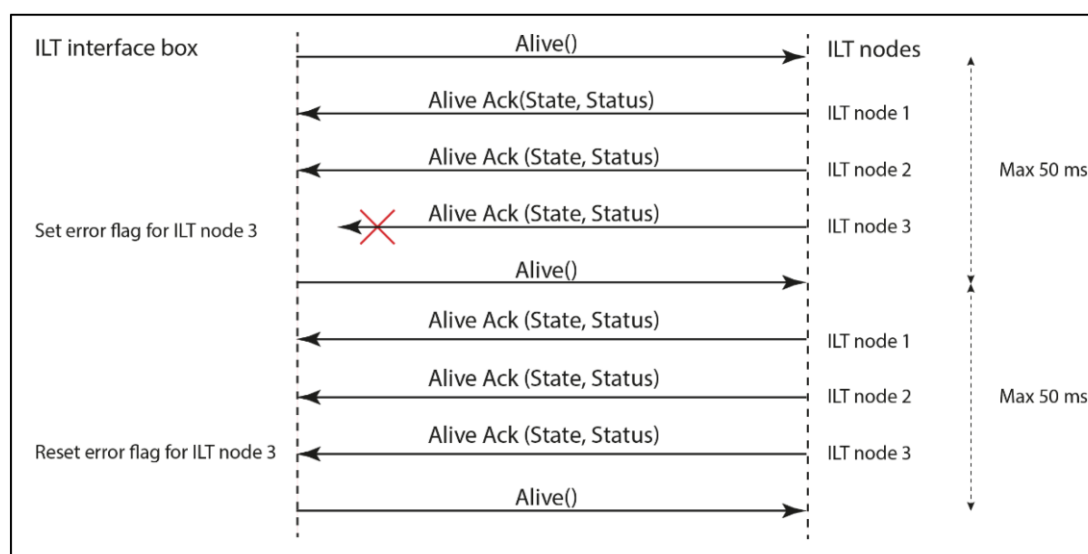
Telegram	Safety relevant	Data	Remark
Alive	Yes <sup>a</sup>	4 bit sequence counter	<p><u>Sequence counter</u> Data[0]&lt;0:3&gt; ... Sequence counter</p> <p>NOTE 1 The time <math>T_{ALIVEPER}</math> between the sending of two consecutive Alive commands shall be greater than or equal to 20 ms</p>
AliveAck	Yes <sup>b</sup>	<p>6 bit common status</p> <p>1 bit sum failure pending</p> <p>1 bit sum warning pending</p> <p>CONDITIONAL: 16 bit Status <sup>b</sup></p>	<p><u>6 bit common status</u> Data[0]&lt;0:3&gt; ... Sequence counter (see note 1) Data[0]&lt;4&gt; ... 1 = DIP occurred since last AliveAck Data[0]&lt;5&gt; ... not used (all bits 0)</p> <p>NOTE 1 The sequence counter shall be the bit-inverted sequence counter value received from master in the last Alive telegram.</p> <p><u>1 bit sum failure</u> Data[0]&lt;6&gt; ... 0 ... no failure 1 ... failure pending see note 2</p> <p><u>1 bit sum warning</u> Data[0]&lt;7&gt; ... 0 ... no warning 1 ... warning pending see note 3</p> <p><u>16 bit Status</u> For content of the status see definition of assignment below <sup>b</sup></p> <p>NOTE 2 More detailed information about the failure shall be obtained by the command GetFailure.</p> <p>NOTE 3 More detailed information about the warning shall be obtained by the command GetWarning, OpCode = 0x00</p> <p>NOTE 4 The time <math>T_{ALIVEACK}</math> between reception of command Alive and sending the acknowledge AliveAck shall not be greater than 5 ms (conditions: no commands in FIFO when receiving the Alive and no collisions on CAN bus when sending the AliveAck)</p>
<p><sup>a</sup> Not transmitted redundantly</p> <p><sup>b</sup> The content (and presence / length) of the status depends on the supported command sets (features / functionalities) of each ILT component. The assignment is (and shall be) done according to the following logic (Table 14 and Figure 13), focusing on output functionalities.</p>			



**Table 14 – Status mask**

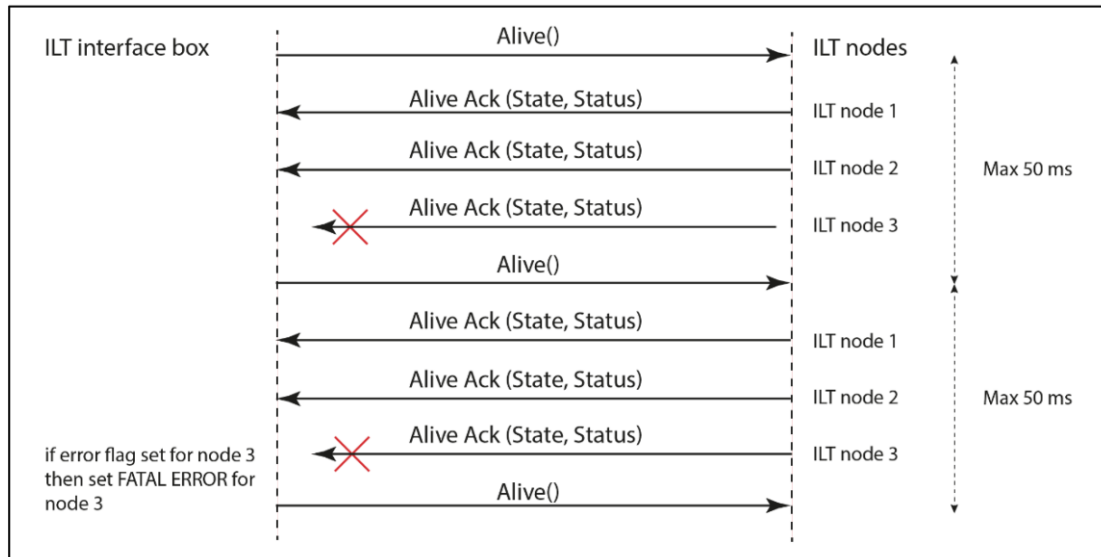
Type of ILT component	Content of status mask
single functionality <b>safety-related</b>	single safety functionality / command set
single functionality <b>non-safety-related</b>	<b>no status mask</b> , telegram shortened!
multiple functionalities all <b>non-safety-related</b>	<b>no status mask</b> , telegram shortened!
multiple functionalities one safety-related	safety-related feature / command set
multiple functionalities more than one safety-related	specified in a future specification / protocol version, providing a way to report all relevant information in as little data as possible

**EXAMPLE** Each combination of a push button with an acoustic signal will report the acoustic signal state within the AliveAck telegram, as the push button has no safety-related functionality.



**Figure 13 – Alive sequence with one Ack missed**

Note: the reactions “Set/Reset error flag for ILT node 3” in this example indicate that the state is known to and stored within the ILT interface box. The system reaction on this is defined by the traffic controller, as it may be reasonable to tolerate a lost AliveAck as long as signalling is still safe and/or there is enough time left to initiate a fail-safe reaction.



**Figure 14 – Alive sequence with two missing Ack's from the same ILT component**

If an ILT-component does not receive a command *Alive* or a command *AliveAck* is not received by the ILT Interface-box within 100 ms, it shall enter its Known-/Safe-state. For the ILT aspect, this means that the light shall be turned off. If a fatal error is diagnosed within an ILT-component, it shall immediately enter the known state defined for the ILT component and transmit an error message to the system master if possible.

## 5.2.5 Error handling

### 5.2.5.1 General concept

Relying on the retransmission feature and the CAN data link layer it may be assumed that all telegrams transmitted are received and not corrupted. This is in a reliability perspective only valid with a given probability. Assuming that missing or corrupted telegrams occur, measures against this shall be taken. Both ILT components and the master shall validate and check the incoming redundant telegrams. If these are not consistent, they shall be discarded. Both the master and the ILT components shall expect to receive at least one valid telegram during process safety time as a sign that the master or ILT component is alive. If this does not happen a fatal error shall be assumed, and the known state shall be entered.

The responsibility of the ILT components is to inform the master of their status correctly. The responsibility of the master is to ensure that the entire “signal picture” displayed is not endangering the traffic. Therefore, a fatal error in one signal may, but not necessarily has to, introduce a dangerous “signal picture” to the traffic.

The aim is not to specify a generic safe data bus, but to define the safety for our specific application where not all telegrams are safety relevant.

The ILT Safety Concept combines a high safety level with a deep diagnostic coverage. Therefore, it is based on a 3-level organization.

- Warnings (Table 17) The ILT-Component detects a minor failure outside the control system ( $\mu\text{C}$ ). In face of this failure the ILT-Components runs within the specified limits.  
i.e. failure of one LED is compensated by switching to redundant LED
- FailureState (Table 16) The ILT-Component detects a failure outside the control system ( $\mu\text{C}$ ) but the status of the STL-Component itself is reliable.  
i.e. failure in the LED-control circuitry
- KnownState (Table 15) The status of the ILT-Component control system is impaired. This may be a result due to a severe failure of a part of the control system ( $\mu\text{C}$ ) itself and/or a loss of communication between ILT-Component and ILT-IF-Box.  
i.e. memory failure of the  $\mu\text{C}$ .  
➔ The overall safety of the ILT component is not compromised in this state, this is ensured by safety / design requirements.

→ In case of lost communication, the known state is entered deliberately, thus with complete and full safety control of the ILT component.

Detailed specification and requirements of the KnownState (Table 15):

- In general, this state is defined for any type of component to be turned off completely, all outputs as well as CAN bus communication.
  - I.e. for signals, light output is off or at least below the threshold given in EN 12368; for Blind People Signals acoustic the KnownState is the off or silent state, for tactile signals it is the off or non-vibrating state.
  - For non-safety components like sensors there might be a predefined state that is assumed in the case of non-functionality or broken communication, but then it is not named "KnownState" as it does not necessarily fulfil the requirements of this definition.
- The design of the safety relevant ILT component shall be done in such a way, that the probability of a second or further independent failure that might lead to a dangerous undetected failure (which means conflicting signalling situations according to EN 12675) shall be less than  $10^{-5}$ /year (Chapter 5.2.4.2.3, EN 50556) while assuming a regular maintenance interval of 6 months (Table 1, EN 50556).
- The KnownState shall be registered as a warning in the traffic controller's log, recorded KnownStates shall be repaired in the next maintenance activity.
- The traffic controller assumes the defined state for the specific type of component for any overall signalling state evaluation, which is done in the (SIL 3 conformant) supervision system of the traffic controller.
  - If there is no conflict (i.e. mandatory red light is off) the TLC keeps its usual operation.

The picture below describes the overall interaction:

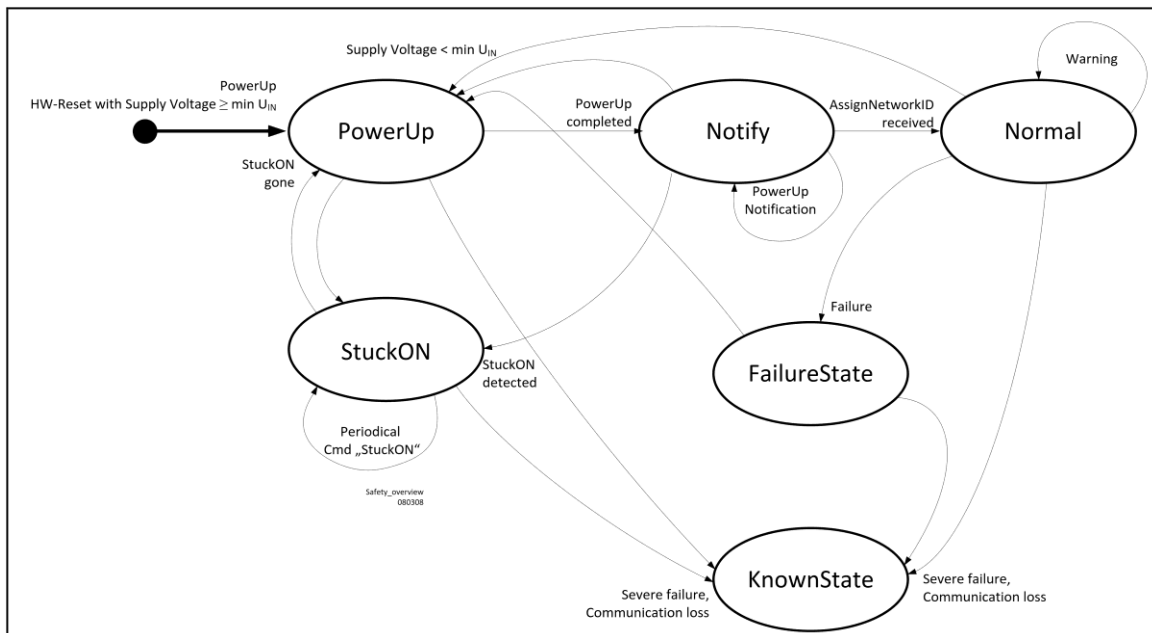


Figure 15 – Error handling

### 5.2.5.1.1 KnownState

Table 15 – KnownState

KnownState	
enter	<p>when the control system of the ILT-Component is impaired in any way, but still safe, or when CAN communication is lost.</p> <ul style="list-style-type: none"> <li>failure or bad self-diagnostic result on one or more ICs (i.e. RAM or Flash) or</li> <li>no communication to ILT-Interface Box (“Alive Timeout”), caused i.e. by broken wire or pulled plug</li> </ul>
exit	<p>with PowerUp.</p> <p>→ KnownState is a volatile status that is not stored persistent, neither in the signal, not the ILT IF box or other part of the controller.</p> <p>That means that after a voltage or power drop the standard initialisation phase occurs if there is a permanent failure in the signal and/or communication the KnownState is entered again.</p> <p>Together with the kind of addressing during initialisation, that enables the exchange of single defect signals while normal operation (on the fly).</p>
notification to ILT-Interface Box	<ul style="list-style-type: none"> <li>Command “EnterKnownState”, singular transmitted, not redundant, including diagnostic information</li> <li>no answer to Command “Alive”</li> </ul> <p>The known state is basically detected by the ILT-Interface box via the missing AliveAcks. The command “EnterKnownState” is just for detailed diagnostic information.</p> <p>→ CAN bus provides reasonable mechanisms which ensure that communication works either bidirectional or not at all, it can be assumed that unidirectional communication never happens.</p> <p>Communication losses to individual ILT components shall be detected from the supervision system of the traffic controller and handled as KnownState of these ILT components.</p>
behaviour of ILT-Component	<ul style="list-style-type: none"> <li>CAN transceiver is disabled</li> <li>ILT-Component enters OFF-state (i.e. light is turned OFF for aspect)</li> </ul>
restriction to CAN communication	ILT-component does not react and answer to any command

### 5.2.5.1.2 FailureState

Table 16 – FailureState

FailureState	
enter	when the ILT-Component detects an internal failure while the control system ( $\mu$ C) is still reliable and communication to ILT-Interface Box is still available i.e. failure detected in monitoring circuitry, turn ON/OFF-switches....
exit	with PowerUp
notification to ILT-Interface Box	<ul style="list-style-type: none"> <li>■ Command "EnterFailureState", singular, redundant transmitted including diagnostic information. This command is sent only once even in the case that successive failures occur.</li> <li>■ bit "sumfailure" within the AliveAck is set, showing that detailed diagnostic information can be obtained via the command "GetFailure"</li> </ul> <p>The failure state is basically detected by the ILT-Interface box via the command EnterFailureState. The information that the ILT-Component has entered the failure state is in fact available by the AliveAck but not safe since the AliveAck is not transmitted redundant.</p>
behaviour of ILT-Component	ILT-Component forces the OFF-state (i.e. light is turned OFF for aspect). However, it is important to notice that in failure state the ILT-Component still shows his actual state (i.e. ON or OFF for aspect) by the light source mask of the AliveAck. The ILT-Interface Box has to monitor the actual state and check if it is compliant to the overall junction state.
restriction to CAN communication	all commands that can lead to exit the OFF-state are rejected and not executed. This is also shown via a negative acknowledge (i.e. for aspects, SignalOnAck with status<0>=1)

### 5.2.5.1.3 Warnings

Table 17 – Warnings

Warnings	
enter	when the ILT-Component detects a minor failure while the behaviour and the specification of the ILT-Component is still met. i.e.: one defect LED is compensated by a redundant one, temperature rises out of an expected range (but stays below a critical limit)
exit	when the root cause of the failure is gone
notification to ILT-Interface Box	bit "sumwarning" within the AliveAck is set detailed diagnostic information is obtained via the command "GetWarning"
behaviour of ILT-Component	internal action to compensate or remove the root cause of the problem
restriction to CAN communication	none

### 5.2.6 Error reporting

The commands in Table 18 below are related to the warning and failure handling of the ILT system.

**Table 18 – Commands for error reporting**

Commands for error and warning reporting			
Telegram	Safety relevant	Data	Remark
EnterKnownState	No <sup>a</sup>	<p>8 bit standard Error code</p> <p>8 bit manufacturer and device type specific Error code</p> <p>16 bit manufacturer and device type specific advanced information</p>	<p>ILT component detected an error and has entered the KnownState.</p> <p><u>8 bit standard Error code</u></p> <p>Data[0] = 0 = Not allowed</p> <p>1 = HW error (unspecified HW error)</p> <p>2 = SW error (unspecified SW error)</p> <p>3 = Temperature (too high)</p> <p>4 = Internal CPU error (self-test failed)</p> <p>5 = Output error (Light, vibration, sound, etc.)</p> <p>6 = Safety checking failed (safety integrity violated)</p> <p>7 = Communication error (CAN communication failed caused by errors which are specified by command <i>ComWarning</i>)</p> <p>8 = Alive timeout</p> <p>9 = external Power wiring</p> <p>10 = Traffic sensor fault generic</p> <p>11 - 255 = reserved</p> <p><u>8 bit manufacturer and device type specific Error code</u></p> <p>Data[1] = Information about supported manufacturer and device type specific error codes see appropriate product datasheet, chapter “ILT communication Interface (CAN-Bus) / Manufacturer Specific error codes for KnownState”</p> <p><u>16 bit manufacturer and device type specific advanced information</u></p> <p>Data[2:3] = advanced information</p>

Commands for error and warning reporting

Telegram	Safety relevant	Data	Remark
EnterFailureState	Yes	<p>8 bit standard Error code</p> <p>8 bit manufacturer and device type specific Error code</p> <p>16 bit manufacturer and device type specific advanced information</p>	<p>ILT component detected an error and has entered the FailureState.</p> <p><u>8 bit standard Error code</u></p> <p>Data[0] = 0 = Not allowed</p> <p>1 = HW error (unspecified HW error)</p> <p>2 = SW error (unspecified SW error)</p> <p>3 = Temperature (too high)</p> <p>4 = Internal CPU error (self-test failed)</p> <p>5 = Output error (Light, vibration, sound, etc.)</p> <p>6 = Safety checking failed (safety integrity violated)</p> <p>7 = Communication error (CAN communication failed caused by errors which are specified by command <i>ComWarning</i>)</p> <p>8 = Alive timeout</p> <p>9 = external Power wiring</p> <p>10 = Traffic sensor fault generic</p> <p>11 - 255 = reserved</p> <p><u>8 bit manufacturer and device type specific Error code</u></p> <p>Data[1] = For information about supported manufacturer and device type specific error codes, see appropriate product datasheet, chapter "ILT communication Interface (CAN-Bus) / Manufacturer Specific error codes for FailureState".</p> <p><u>16 bit manufacturer and device type specific advanced information</u></p> <p>Data[2:3] = advanced information</p> <p>NOTE The command EnterFailureState shall be transmitted BEFORE the AliveAck is sent with sumfailure=1.</p>
StuckOnError	Yes <sup>b</sup>	<p>Standardized-ID, Unique-ID</p>	<p><u>Standardized-ID</u></p> <p>Data[0]&lt;0:7&gt; = Device-type &lt;0:7&gt;</p> <p>Data[1]&lt;0:7&gt; = Sub-type &lt;0:7&gt;</p> <p>Data[2]&lt;0:7&gt; = Manufacturer-ID &lt;0:7&gt;</p> <p><u>Unique-ID</u></p> <p>Data[3]&lt;0:7&gt; = Serial number &lt;0:7&gt;</p> <p>Data[4]&lt;0:7&gt; = Serial number &lt;8:15&gt;</p> <p>Data[5]&lt;0:7&gt; = Serial number &lt;16:23&gt;</p> <p>Data[6]&lt;0:7&gt; = Serial number &lt;24:31&gt;</p> <p>Data[7]&lt;0:6&gt; = Serial number &lt;32:38&gt;</p> <p>Data[7]&lt;7&gt; = 0 reserved</p>
ComWarning	No	void	Sent by ILT interface box:

Commands for error and warning reporting

Telegram	Safety relevant	Data	Remark
ComWarningAck	No	8 bit Error code 8 bit Receive error counter 8 bit Transmit error counter 8 bit Com status register of CAN controller 8 bit FIFO overflow	Sent by ILT Component (either as response to command <i>ComWarning</i> or autonomously) <u>8 bit Error code</u> Data[0]<0> = Receive error (see Note 1) Data[0]<1> = Transmit error (see Note 2) Data[0]<2> = Receive data overflow Data[0]<3> = Transmit buffer overflow Data[0]<4> = Reserved Data[0]<5:7> = Detailed error code of the last detected error according to the CAN standard: <ul style="list-style-type: none"> <li>■ 0 = No Error</li> <li>■ 1 = Stuffing error</li> <li>■ 2 = Form error</li> <li>■ 3 = Acknowledge error</li> <li>■ 4 = Bit one error</li> <li>■ 5 = Bit zero error</li> <li>■ 6 = CRC error</li> <li>■ 7 = For future use</li> </ul> NOTE 1 Data[0]<0> is set if the Receiver error counter is $\geq 96$ and reset when that counter is $< 86$ . NOTE 2 Data[0]<1> is set, if the Transmit error counter is $\geq 96$ and reset when that counter is $< 86$ . NOTE 3 The following conditions will cause an autonomous trigger of sending command <i>ComWarning</i> : <ol style="list-style-type: none"> <li>a) Receive error counter or Transmit error counter enters the Warning-level (<math>\geq 96</math>)</li> <li>b) Receive error counter or Transmit error counter leaves the Warning-level (the according counter is <math>&lt; 86</math>).</li> <li>c) Receive error counter or Transmit error counter enters the Error-passive-level (<math>\geq 128</math>).</li> <li>d) Receive error counter or Transmit error counter leaves the Error-passive-level (the according counter is <math>&lt; 118</math>).</li> <li>e) Any change in Data[0]&lt;0:4&gt;</li> </ol> <u>8 bit Receive error counter</u> Data[1]<0:7> ... Error counter in CAN controller <u>8 bit Transmit error counter</u> Data[2]<0:7> ... Error counter in CAN controller <u>8 bit Com status register of CAN controller</u> Data[3]<0:7> ... <u>8 bit FIFO overflow</u> Data[4]<0> = RXHIGH overflow Data[4]<1> = RXMISC overflow Data[4]<2> = TXFIFO overflow Data[4]<3:7> = Reserved NOTE 4 Overflow bits are reset immediately after having sent out the warning telegram.



## Commands for error and warning reporting

Telegram	Safety relevant	Data	Remark
<sup>a</sup>			Since the "KnownState" of an ILT-Component is defined by "no communication" on CAN and in case of failure of an ILT-Component the sending of command "EnterKnownState" cannot be guaranteed, this telegram can only be seen as an additional information
<sup>b</sup>			<p>This command is sent spontaneously by the ILT-component before a Network Address has been assigned.</p> <ul style="list-style-type: none"> <li>– there is no redundancy needed although this command is safety-relevant. Since this command just leads to PowerOff the complete system it cannot generate an unsafe state (or more unsafe that already appear)</li> <li>– the command is sent periodically (100 ms cycle time) with PowerUp-ID and Priority 0 (PrioError) is used</li> <li>– within <math>T_{PUN}</math> (see chapter "Power up addressing, identification, Hot Plug")</li> <li>– Remark: after network ID has been assigned, a StuckOnError is shown via EnterFailureState command and the LightSourceMask set within the AliveAck</li> </ul>

### 5.2.7 Firmware update

All ILT components should support online firmware updating for easier maintenance in the end.

To simplify development, it is acceptable that ILT components are introduced without the firmware update feature at the beginning, having the feature added later – however the hardware design shall be chosen in a way allowing to add the firmware update functionality in general.

During the upload of the firmware to the ILT component, the ILT components shall be able to operate normally. A reboot is required in order to flash the uploaded firmware into those flash-memory pages, where normal program execution takes place. This implies, that a bootloader shall be implemented which decides on power-up, if there is a new firmware available.

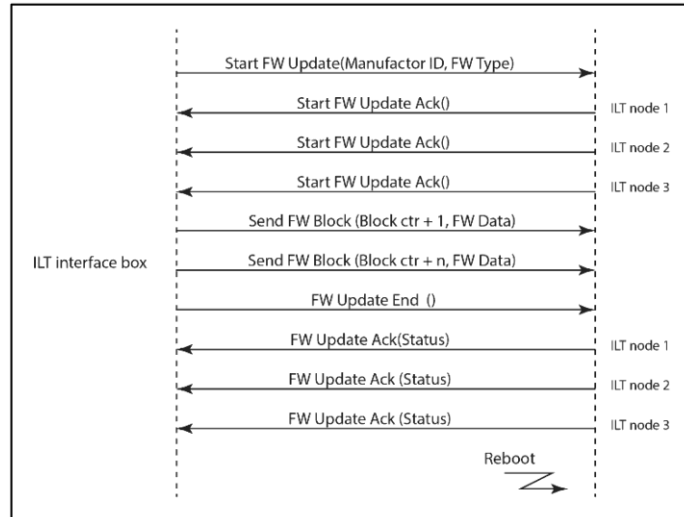
- If no new firmware is available, then the bootloader branches to normal execution.
- If a new firmware is available, then the bootloader does the flashing of the new firmware and when that has finished, the bootloader triggers a software reset (which leads to a branch for normal execution) <sup>2</sup>.

Pre-requisites: The traffic controller has to maintain an internal table where all ILT components are registered. The table consists of the Standardized ID (Device Type, Sub Type, Manufacturer ID) and the Unique ID (passed by Power Up Notification) as well as the 16-bit Network ID (generated by traffic controller). The 16-bit Network ID identifies an ILT-component uniquely.

Technician on site has to decide after receiving the Acknowledge event to power down the system (to activate the new Updated FW). After FW upload the "new" FW will be flashed after sending the command FWUpdateFlash or with the next power up. However, after the flashing of the new FW is completed, the ILT-Component generates a SW-reset which leads to a Powerup Notification.

The following Figure 16 shows the sequence of a Firmware-update, Table 19 presents the regarding Telegrams.

<sup>2</sup> Since the command *Powerup Notification* is sent by the ILT component only at normal execution, the specified time  $T_{PUN}$  will be violated (will take several seconds).



**Figure 16 – Firmwarebu-update**

**Table 19 – Telegrams concerning firmware update**

Telegram	Safety relevant	Data	Remark
FWUpdateStart	No	8 bit Device-type 8 bit Manufacturer-ID 8 bit FW-type 24 bit FW description	<u>8 bit Device-type</u> Data[0] = See Powerup Notification <u>8 bit Manufacturer-ID</u> Data[1] = See Powerup Notification <u>8 bit FW-type</u> Data[2] = See GetComponentTypeAck <u>24 bit FW-description</u> Data[3:5] = See GetComponentTypeAck NOTE 1 Header of FW-Package-File: – Byte[0] = Device-type – Byte[1] = Manufacturer-ID – Byte[2] = FW-Type – Byte[3:5] = FW-description NOTE 2 For accepting a firmware update, the device type, manufacturer ID and firmware type shall be identical.
FWUpdateStartAck	No	8 bit Status	<u>8 bit Status</u> Data[0] = Status NOTE 1 See common note concerning 8 bit Status NOTE 2 Firmware update is denied if <i>Device-type</i> <b>OR</b> <i>Manufacturer-ID</i> <b>OR</b> <i>FW-type</i> does not match with actual firmware. NOTE 3 The firmware update is accepted even if the actual firmware is the same version. NOTE 4 Every ILT-Component will send this Ack no matter if it is addressed or not. NOTE 5 Command FWUpdateStartAck is sent immediately as soon as all internal measures to prepare the FW upload (i.e. erase flash) are completed. The time $T_{FWSTART}$ between reception of command <i>FWUpdateStart</i> and sending the acknowledge <i>FWUpdateStartACK</i> shall not be greater than 5 Sec.

Telegram	Safety relevant	Data	Remark
FWUpdateSendBlock	No	16 bit BlockCtr (Block counter) 48 bit FWData (Firmware data)	<p><u>16 bit BlockCtr (Block counter)</u> Data[0] = BlockCtr&lt;0:7&gt; Data[1] = BlockCtr&lt;8:15&gt;</p> <p>NOTE 1 Up to 65536 blocks each have 6 Bytes =&gt; 393216 Bytes before overflow occurs. NOTE 2 The checksum is part of the FWData. NOTE 3</p> <ul style="list-style-type: none"> <li>– BlockCtr is 0x0000 in first FWUpdateSendBlock</li> <li>– BlockCtr + 1 for each subsequent FWUpdateSendBlock</li> </ul> <p>NOTE 4 In order to write received data into a non-volatile memory a minimum time <math>T_{FWSEND} = 30</math> ms between two subsequent commands <i>FWUpdateSendBlock</i> has to be met.</p> <p><u>48 bit FWData</u> Data[2] = FWData &lt;0:7&gt; Data[3] = FWData &lt;8:15&gt; Data[4] = FWData &lt;16:23&gt; Data[5] = FWData &lt;24:31&gt; Data[6] = FWData &lt;32:39&gt; Data[7] = FWData &lt;40:47&gt;</p>
FWUpdateEnd	No	void	
FWUpdateEndAck	No	8 bit Status	<p><u>8 bit Status</u> Data[0] = Status</p> <p>NOTE 1 See common note concerning 8 bit Status NOTE 2 Only those ILT-Component will send this Ack which have received FwUpdateStart <b>AND</b> which have answered with “firmware update accepted” in FwUpdateStartAck. NOTE 3 Command <i>FWUpdateEndAck</i> is sent immediately as soon as all internal measures on the ILT-Component are completed. The time <math>T_{FWEND}</math> between reception of command <i>FWUpdateEnd</i> and sending the acknowledge <i>FWUpdateEndACK</i> shall not be greater than 10 ms.</p>
FWUpdateFlash	No	void	
FWUpdateFlashAck	No	8 bit Status	<p>Sent immediately after receipt of the command and just before starting to program the flash</p> <p><u>8 bit Status</u> Data[0] = Status Status&lt;7&gt; ...</p> <ul style="list-style-type: none"> <li>– 0 = Ok, programming will be started</li> <li>– 1 = Error, no programming will be started because any bit in Status&lt;0:6&gt; is set.</li> </ul> <p>NOTE 1 See common note concerning 8 bit Status. NOTE 2 Only those ILT-Component will send this Ack which have received FwUpdateStart <b>AND</b> which have answered with “firmware update accepted” in FwUpdateStartAck. NOTE 3 Command <i>FWUpdateFlashAck</i> is sent immediately as soon as all internal measures on the ILT-Component are completed. The time <math>T_{FWFLASH}</math> between reception of command <i>FWUpdateFlash</i> and sending the acknowledge <i>FWUpdateFlashACK</i> shall not be greater than 10 ms. It is important to notice, that the time <math>T_{FWFLASH}</math> is NOT the time for flashing itself but the time just to prepare the flashing.</p>

Telegram	Safety relevant	Data	Remark
Common note concerning 8 bit Status:			
– Status<0> ... 0 / 1			= Firmware update accepted / denied (this is an information and not an error)
– Status<1> ... 1			= FwUpdateEnd received before receiving any command FwUpdateSendBlock
– Status<2> ... 1			= FwUpdateFlash received before receiving command FwUpdateEnd
– Status<3> ... 1			= Wrong Block-counter in any command FwUpdateSendBlock (at least one block is missing or incrementation of BlockCtr has been violated)
– Status<4> ... 1			= Wrong checksum
– Status<5> ... 0			= Timeout
– Status<6> ... 0			= Internal error
– Status<7> ... 1			= Any bit in Status<1:6> is set (common error indication)

## 5.2.8 Parameter update & read out

In general, a distinction is made between DeviceParameter and OperationParameter

- DeviceParameter  
organized in device and manufacturer specific parameter block (set of general device settings, permanently stored in the device, binary data block not interpreted by the IF-Box)
- OperationParameter  
The command SetOperationParameter is used to change operation parameters temporary stored and changed per runtime, interpreted by the IF-Box

While this section handles the DeviceParameter only, the OperationParameters are described in the relevant chapters of the product types (SetOperationParameter, see chapter 6.2, 7.2, 8.2, 9.2).

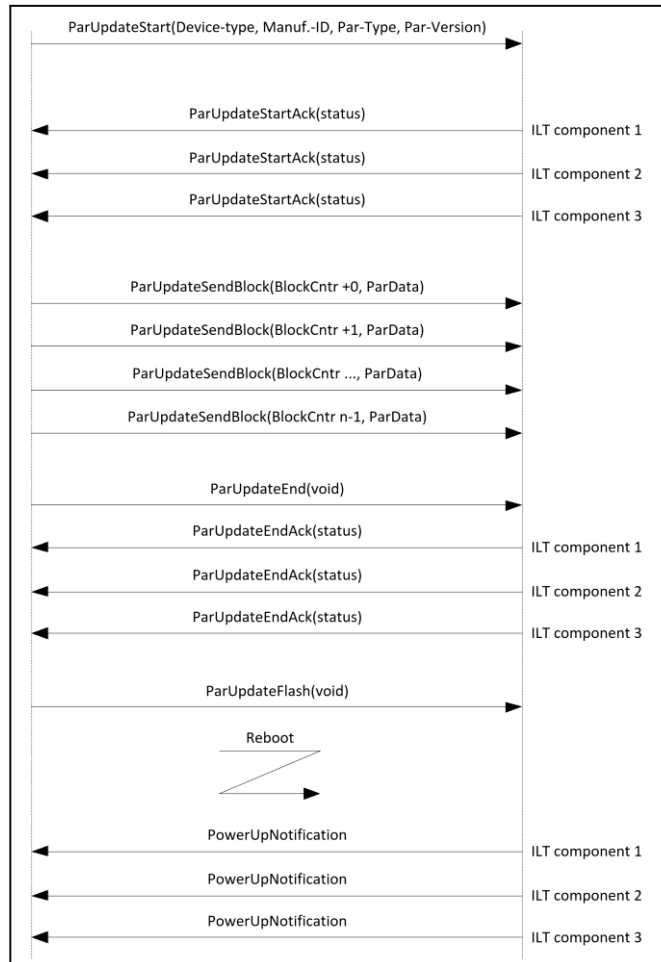
All ILT components may support online parameter updating for non-safety related, manufacturer- and/or device-specific parameters. During the update of the parameters to the ILT component, the ILT components shall be able to operate normally, the new / changed parameters are not used until the component is rebooted.

The command set in Table 20 is intended to update the DeviceParameter. For a safe and consistent activation of the new parameters by the ILT-component, a reboot is required.

In addition, it shall be possible that the parameters can be read out by the IF-Box on demand using the commands in Table 21. They are kept within the traffic controller and are updated after an ILT-component has been exchanged due to service.

Pre-requisites: The traffic controller has to maintain an internal table where all types of parameter sets are registered. The table consists of the Standardized ID (Device Type, Manufacturer ID), the ParameterType (description of the type of the parameter block) and the ParameterVersion (version of the parameter block).

The following figure shows the sequence of a parameter update.



**Figure 17 – Parameter-update**

**Table 20 – Telegrams concerning parameter update**

Telegram	Safety relevant	Data	Remark
ParUpdateStart	No	8 bit Device-type 8 bit Manufacturer-ID 8 bit Par-Type 40 bit Par Version	<p><u>8 bit Device-type</u> Data[0] = See Powerup Notification</p> <p><u>8 bit Manufacturer-ID</u> Data[1] = See Powerup Notification</p> <p><u>8 bit Par-Type</u> Data[2] &lt;0:3&gt; = See GetParameterTypeAck &lt;4:7&gt; = 0 (reserved)</p> <p><u>40 bit Par-Version</u> Data[3:7] = See GetParameterTypeAck</p> <p>NOTE 1 Header of Par-Package-File:  <ul style="list-style-type: none"> <li>– Byte[0] = Device-type</li> <li>– Byte[1] = Manufacturer-ID</li> <li>– Byte[2] = Par-Type</li> <li>– Byte[3:7] = Par-Version</li> </ul> </p> <p>NOTE 2 For accepting a parameter update, the device type, manufacturer ID and par type shall be identical.</p> <p>NOTE 3 based on the Par-Version the ILT-component is responsible to handle the upward and downward compatibility</p> <p>NOTE 4 based on the addressing mechanism, broadcasts, group-addressing (multicasts) or individual addressing can be used.</p>
ParUpdateStartAck	No	8 bit Status	<p><u>8 bit Status</u> Data[0] = Status</p> <p>NOTE 1 See common note concerning 8 bit Status</p> <p>NOTE 2 Parameter update is denied if <i>Device-type</i> <b>OR</b> <i>Manufacturer-ID</i> <b>OR</b> <i>Par-type</i> does not match with actual firmware.</p> <p>NOTE 3 Parameter update is rejected if the write is not permitted (as defined in the ParameterFunctionMask in command <i>GetParameterTypeAck</i>)</p> <p>NOTE 4 The parameter update is accepted even if the actual parameter is the same version.</p> <p>NOTE 5 Every ILT-Component will send this Ack no matter if it is addressed or not.</p> <p>NOTE 6 Command ParUpdateStartAck is sent immediately as soon as all internal measures to prepare the Par upload (i.e. erase flash) are completed. The time <math>T_{PARSTART}</math> between reception of command <i>ParUpdateStart</i> and sending the acknowledge <i>ParUpdateStartACK</i> shall not be greater than 5 Sec.</p>

Telegram	Safety relevant	Data	Remark
ParUpdateSendBlock	No	16 bit BlockCtr (Block counter) 48 bit ParData (Parameter data)	<p><u>16 bit BlockCtr (Block counter)</u> Data[0] = BlockCtr&lt;0:7&gt; Data[1] = BlockCtr&lt;8:15&gt;</p> <p>NOTE 1 Up to 65536 blocks each have 6 Bytes =&gt; 393216 Bytes before overflow occurs. NOTE 2 The checksum is part of the ParData. NOTE 3</p> <ul style="list-style-type: none"> <li>- BlockCtr is 0x0000 in first ParUpdateSendBlock</li> <li>- BlockCtr + 1 for each subsequent ParUpdateSendBlock</li> </ul> <p>NOTE 4 In order to write received data into a non-volatile memory a minimum time <math>T_{PARSEND} = 30</math> ms between two subsequent commands <i>ParUpdateSendBlock</i> has to be met.</p> <p><u>48 bit ParData</u> Data[2] = ParData &lt;0:7&gt; Data[3] = ParData &lt;8:15&gt; Data[4] = ParData &lt;16:23&gt; Data[5] = ParData &lt;24:31&gt; Data[6] = ParData &lt;32:39&gt; Data[7] = ParData &lt;40:47&gt;</p>
ParUpdateEnd	No	void	
ParUpdateEndAck	No	8 bit Status	<p><u>8 bit Status</u> Data[0] = Status</p> <p>NOTE 1 See common note concerning 8 bit Status NOTE 2 Only those ILT-Component will send this Ack which have received ParUpdateStart <b>AND</b> which have answered with "parameter update accepted" in ParUpdateStartAck. NOTE 3 Command <i>ParUpdateEndAck</i> is sent immediately as soon as all internal measures to validate the received parameters on the ILT-Component are completed. The time <math>T_{PAREND}</math> between reception of command <i>ParUpdateEnd</i> and sending the acknowledge <i>ParUpdateEndACK</i> shall not be greater than 10 ms.</p>
ParUpdateFlash	No	void	

Telegram	Safety relevant	Data	Remark
ParUpdateFlashAck	No	8 bit Status	<p>Sent immediately after receipt of the command and just before starting to program the parameter into the flash.</p> <p><u>8 bit Status</u>  Data[0] = Status  Status&lt;7&gt; ... 0 = Ok programming will be started  1 = Error no programming will be started because any bit in Status&lt;0:6&gt; is set.</p> <p>NOTE 1 See common note concerning 8 bit Status  NOTE 2 Only those ILT-Component will send this Ack which have received ParUpdateStart <b>AND</b> which have answered with "parameter update accepted" in ParUpdateStartAck .  NOTE 3 Command <i>ParUpdateFlashAck</i> is sent immediately as soon as all internal measures to prepare the storage of the parameter into the flash on the ILT-Component are completed. The time <math>T_{PARFLASH}</math> between reception of command <i>ParUpdateFlash</i> and sending the acknowledge <i>ParUpdateFlashACK</i> shall not be greater than 10 ms. It is important to notice, that the time <math>T_{PARFLASH}</math> is NOT the time for flashing itself but the time just to prepare the flashing.  NOTE 4 immediately after sending the ParUpdateFlashACK the ILT-Component starts performing the flash process. Therefore the ILT-component enters a state, identical to the KnownState:  <ul style="list-style-type: none"> <li>- CAN communication is stopped</li> <li>- all Outputs are set to "OFF"</li> </ul> The time <math>T_{PARWRITE}</math> to write the parameter to into the flash shall not be greater than 100 ms followed by the PowerUpNotification  NOTE 5 When a PowerDown occurs during the flash process (so during <math>T_{PARWRITE}</math>), the parameter can get lost, and the parameter update shall be repeated.</p>
<p>Common note concerning 8 bit Status:</p> <ul style="list-style-type: none"> <li>- Status&lt;0&gt; ... 0 / 1 = Parameter update accepted / denied (this is an information and not an error)</li> <li>- Status&lt;1&gt; ... 1 = ParUpdateEnd received before receiving any command ParUpdateSendBlock</li> <li>- Status&lt;2&gt; ... 1 = ParUpdateFlash received before receiving command ParUpdateEnd</li> <li>- Status&lt;3&gt; ... 1 = Wrong Block-counter in any command ParUpdateSendBlock (at least one block is missing or incrementation of BlockCtr has been violated)</li> <li>- Status&lt;4&gt; ... 1 = Wrong checksum</li> <li>- Status&lt;5&gt; ... 0 = Timeout</li> <li>- Status&lt;6&gt; ... 0 = Internal error</li> <li>- Status&lt;7&gt; ... 1 = Any bit in Status&lt;1:6&gt; is set (common error indication)</li> </ul>			



**Table 21 – Telegrams concerning parameter read out**

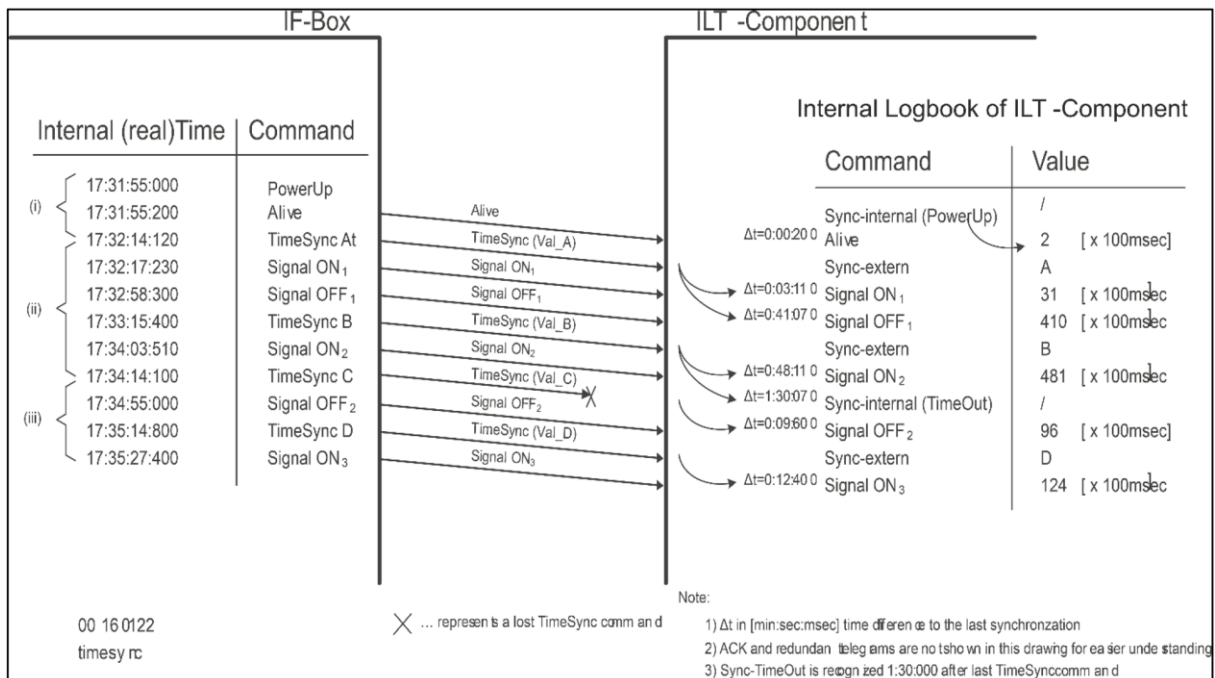
Telegram	Safety relevant	Data	Remark
GetParameterType	No	8 bit ParCode	<p>Get detailed information about the parameter.</p> <p><u>ParCode</u>            Data[0]&lt;7&gt; ... 0 read out Par-Type                              1 read out Parameter Function mask            &lt;4:6&gt; ... 0 (reserved)            &lt;0:3&gt; ... for Data[0]&lt;7&gt;=0 =&gt; Par-Type                              for Data[0]&lt;7&gt;=1 =&gt; 0</p> <p>NOTE 1 Par-Type is a unique identifier within Device-type and Manufacturer-ID. With this information, the ILT-IF-Box is able to detect all the ILT-Components that are affected by / allow a parameter update.</p>
GetParameterTypeAck	No	8 bit ParCode 40 bit Par-Version (for reading out Par-Type) 32 bit ParameterFunctionMask 8 bit Error-information	<p><u>ParCode</u>            Data[0] = see definition in GetParameterType</p> <ul style="list-style-type: none"> <li>■ for read out Par-Type (ParCode[0]&lt;7&gt;=0:             <p><u>Par-Version</u>                Data[1]&lt;0:7&gt; = Par-Version &lt;0:7&gt;                Data[2]&lt;0:7&gt; = Par-Version &lt;8:15&gt;                Data[3]&lt;0:7&gt; = Par-Version &lt;16:23&gt;                Data[4]&lt;0:7&gt; = Par-Version &lt;24:31&gt;                Data[5]&lt;0:7&gt; = Par-Version &lt;32:39&gt;</p> <p><u>Error-information</u>                Data[6]&lt;0:6&gt; = not used                Data[6]&lt;7&gt; = 0 ... Ok,                                  1 ... Error (requested information is not available).</p> </li> <li>■ for read out ParameterFunctionMask bit field for each Par-Type:             <p><u>ParameterFunctionMask</u>                Data[1]&lt;0:7&gt; ReadWrite permission for Par-Type 0-7                Data[2]&lt;0:7&gt; ReadWrite permission for Par-Type 8-15                                  0 ... ReadWrite disabled                                  1 ... ReadWrite enabled                Data[3]&lt;0:7&gt; AutoBackup permission Par-Type&lt;0:7&gt;                Data[4]&lt;0:7&gt; AutoBackup permission Par-Type&lt;8:15&gt;                                  0 ... AutoBackup disabled                                  1 ... AutoBackup enabled</p> <p><u>Error-information</u>                Data[5]&lt;0:6&gt; = not used                Data[5]&lt;7&gt; = 0 ... Ok,                                  1 ... Error (requested information is not available).</p> </li> </ul>

Telegram	Safety relevant	Data	Remark
ParRead	No	4 bit Par-Type specification 2 bit Sequence number 2 bit reserved	Read Parameter from ILT-Component. <u>4 bit Par-Type specification</u> Data[0]<0:3> specifies the type of parameter block to be read out  <u>2 bit Sequence number</u> Data[0]<4:5> = <ul style="list-style-type: none"> <li>■ 0x00 = Start or restart the <i>ParRead</i> operation (see Notes)</li> <li>■ 0x01 – 0x03 = see Note</li> </ul> NOTE 1 The <i>ParRead</i> operation is started with the sequence-number 0x00, on each subsequent command <i>ParRead</i> , the sequence-number is incremented. However, the overflow from 0x03 is 0x01 because 0x00 is reserved to indicate the start or restart of the <i>ParRead</i> operation. NOTE 2 The sender shall wait for the answer before sending a subsequent command <i>ParRead</i> . Data[0]<6:7> = 0 (reserved)

Telegram	Safety relevant	Data	Remark
ParReadAck	No	4 bit Par-Type specification 2 bit Sequence number 2 bit Status Up to 56 bits of parameter data	<p><u>4 bit Par-Type specification</u> Data[0]&lt;0:3&gt; = see command <i>ParRead</i></p> <p><u>2 bit Sequence number</u> Data[0]&lt;4:5&gt; = Copied value of the received Sequence-number.</p> <p><u>2 bit Status</u> Data[0]&lt;6:7&gt;...</p> <ul style="list-style-type: none"> <li>■ 0x00 = Continue because additional data is possibly available (Note 1)</li> <li>■ 0x01 = End of record (Note 2)</li> <li>■ 0x02 = Any error occurred (Note 3)</li> <li>■ 0x03 = Reserved (command <i>ParRead</i> is ignored)</li> </ul> <p><u>Up to 56 bits of parameter data</u>            Data[1] = Requested parameter data &lt;0:7&gt;            Data[2] = Requested parameter data &lt;8:15&gt;            Data[3] = Requested parameter data &lt;15:23&gt;            Data[4] = Requested parameter data &lt;24:31&gt;            Data[5] = Requested parameter data &lt;32:39&gt;            Data[6] = Requested parameter data &lt;40:47&gt;            Data[7] = Requested parameter data &lt;48:55&gt;</p> <p>NOTE 1 The data in the payload consists of the parameter data. The traffic-controller shall send a new command <i>ParRead</i> to see if there are additional data available.</p> <p>NOTE 2 The complete (requested) parameter data has been sent. The payload consists of only Data[0]. The traffic-controller shall start a new <i>ParRead</i> operation (with sequence-number 0x00) to bring the ILT-Component into another state.</p> <p>NOTE 3 The payload consists of only Data[0]. The traffic-controller can either continue (with the same sequence-number) or it can start a new <i>ParRead</i> operation (with sequence-number 0x00).</p> <p>NOTE 4 the first bytes of transferred parameter are used as Header information:</p> <ul style="list-style-type: none"> <li>- Byte[0] = Device-type</li> <li>- Byte[1] = Manufacturer-ID</li> <li>- Byte[2] = Par-Type</li> <li>- Byte[3:5] = Par-Version</li> </ul> <p>Possible errors:</p> <ul style="list-style-type: none"> <li>■ Wrong ParBlock-specification</li> <li>■ Wrong sequence-number</li> <li>■ for this Par-Type the read out is not permitted (as defined in the ParameterFunctionMask in command <i>GetParameterTypeAck</i>)</li> </ul>

### 5.2.9 Time sync concept

The next figure shows the general concept for synchronization of the logbooks within the IF-Box and the ILT components.



### Key

- (i) shows the phase until the first synchronization
- (ii) shows the standard behaviour
- (iii) shows what happens when a synchronization command has gone lost
  - Logbooks within IF-Box and ILT-Components will be synchronized via the time stamps (SyncValues A, B, D, in example above represents a time stamp). There is no RTC necessary (neither in the IF-Box nor in the ILT-component)
  - if no TimeSync command is received within a TimeOut (1:30:000 in example above), an internal timestamp will be set by the ILT component (but from now on, there is no more synchronization to the IF-Box anymore)
  - an RTC / UTC time can be used in the IF-Box (with 2 Byte SyncValue the “minute” value of UTC can be used for that)
  - SetTimeSync shall be sent periodically. The specification of the periodic interval is important in order to know when the internal delta-time-span-counter overflows (which has a resolution of 100 ms).

Figure 18 – Time synchronisation

## 5.3 CAN bus commands

### 5.3.1 General functionality

The functions described in this section shall be implemented in all ILT components, regardless of the manufacturer and device-type.

#### PowerUp-ID and ID-less addressed commands

The Table 22 shows the system commands which are sent with the *PowerUp-ID* as Network-ID .

Table 22 – PowerUp-ID addressed system commands

System commands	Safety relevant	Data	Remark
<p>Powerup Notification  <i>Network-ID = PowerUp-ID</i>                      Any value from                      0x0001 to 0xFFFFE</p>	<p>No                      (see Note 1)</p>	<p>Standardized-ID,                      Unique-ID</p>	<p><u>Standardized-ID</u>                      Data[0]&lt;0:7&gt; = Device-type &lt;0:7&gt;                      Data[1]&lt;0:7&gt; = Sub-type &lt;0:7&gt;                      Data[2]&lt;0:7&gt; = Manufacturer-ID &lt;0:7&gt;</p> <p><u>Unique-ID</u>                      Data[3]&lt;0:7&gt; = Serial number &lt;0:7&gt;                      Data[4]&lt;0:7&gt; = Serial number &lt;8:15&gt;                      Data[5]&lt;0:7&gt; = Serial number &lt;16:23&gt;                      Data[6]&lt;0:7&gt; = Serial number &lt;24:31&gt;                      Data[7]&lt;0:6&gt; = Serial number &lt;32:38&gt;                      Data[7]&lt;7&gt; = 0/1 ... CustomerData cleared/set                      (see Note 3)</p> <p>Range for Serial number:                      0x0000000001 to                      0x7FFFFFFF                      Serial number 0x0000000000 is                      prohibited and reserved for ILT-                      Components during production.</p> <p>This command shall be sent within <math>T_{PUN}</math> (see chapter                      "Power up addressing, identification, Hot Plug")</p> <p>In case of a collision during the transmission of the                      Powerup Notification (several ILT-Component use                      same random number for Network-ID), the                      subsequent Powerup Notification shall use a new                      random NetworkID</p>
<p>AssignNetworkID  <i>Network-ID = 0xFFFF</i></p>	<p>No                      (see Note 1)</p>	<p>Manufacturer-ID,                      Unique-ID,                      Network-ID</p>	<p><u>Manufacturer-ID</u>                      Data[0]&lt;0:7&gt; = Manufacturer-ID &lt;0:7&gt;</p> <p><u>Unique-ID</u>                      Data[1]&lt;0:7&gt; = Serial number &lt;0:7&gt;                      Data[2]&lt; 0:7&gt; = Serial number &lt;8:15&gt;                      Data[3]&lt; 0:7&gt; = Serial number &lt;16:23&gt;                      Data[4]&lt; 0:7&gt; = Serial number &lt;24:31&gt;                      Data[5]&lt; 0:6&gt; = Serial number &lt;32:38&gt;                      Data[5]&lt; 7&gt; = not used</p> <p><u>Network-ID</u>                      Data[6]&lt;0:7&gt; = Network-ID &lt;0:7&gt;                      Data[7]&lt;0:7&gt; = Network-ID &lt;8:15&gt;</p> <p>Value 0x0000 (Broadcast-ID) or 0xFFFF as Network-                      ID is not allowed. If this value is handed over, the                      ILT-component sends no answer.</p> <p>The received Manufacturer-ID and Unique-ID shall                      match to accept the telegram, otherwise the ILT                      component sends no answer.</p> <p>The assign network ID – assign network                      acknowledge should be done in sequence to avoid                      package collisions due to the network ID 0xFFFF.</p>

System commands	Safety relevant	Data	Remark
AssignNetworkIDAck	No (see Note 1)	Network-ID	<p><u>Network-ID</u></p> <p>Data[0]&lt;0:7&gt; = Network-ID &lt;0:7&gt; Data[1]&lt;0:7&gt; = Network-ID &lt;8:15&gt;</p> <p>After sending AssignNetworkIDAck, the ILT-component no longer accepts telegrams which are received with priority 3 (PrioPowerUp).</p> <p>NOTE The ILT-Component uses the just assigned Network-ID within the CAN-ID and additional within the payload. This will prevent for collisions when all the addressed ILT-Components sends their ACK also at the "same time".</p>
Identify (see Note 2)  <i>Network-ID = 0xFFFF</i>	No	Standardized-ID, Unique-ID	<p><u>Standardized-ID</u></p> <p>Data[0]&lt;0:7&gt; = Device-type &lt;0:7&gt; Data[1]&lt;0:7&gt; = Sub-type &lt;0:7&gt; Data[2]&lt;0:7&gt; = Manufacturer-ID &lt;0:7&gt;</p> <p><u>Unique-ID</u></p> <p>Data[3]&lt;0:7&gt; = Serial number &lt;0:7&gt; Data[4]&lt;0:7&gt; = Serial number &lt;8:15&gt; Data[5]&lt;0:7&gt; = Serial number &lt;16:23&gt; Data[6]&lt;0:7&gt; = Serial number &lt;24:31&gt; Data[7]&lt;0:6&gt; = Serial number &lt;32:38&gt; Data[7]&lt;7&gt; = not used</p> <p>The Device-type, the Sub-type, the Manufacturer-ID, and the Unique-ID shall match to accept the telegram, else the ILT-component sends no answer.</p>
IdentifyAck <i>Network-ID = 0xFFFF</i>	No	Standardized-ID, Unique-ID	<p><u>Standardized-ID</u></p> <p>Data[0]&lt;0:7&gt; = Device-type &lt;0:7&gt; Data[1]&lt;0:7&gt; = Sub-type &lt;0:7&gt; Data[2]&lt;0:7&gt; = Manufacturer-ID &lt;0:7&gt;</p> <p><u>Unique-ID</u></p> <p>Data[3]&lt;0:7&gt; = Serial number &lt;0:7&gt; Data[4]&lt;0:7&gt; = Serial number &lt;8:15&gt; Data[5]&lt;0:7&gt; = Serial number &lt;16:23&gt; Data[6]&lt;0:7&gt; = Serial number &lt;24:31&gt; Data[7]&lt;0:6&gt; = Serial number &lt;32:38&gt; Data[7]&lt;7&gt; = 0/1 ... CustomerData cleared/set (see Note 3)</p> <p>The Device-type, the Sub-type, the Manufacturer-ID, and the Unique-ID shall match to accept the telegram, else the ILT-component sends no answer.</p>
GetCustomerData <i>Network-ID</i> is either the PowerUp-ID or the Direct-ID  (a) Command AssignNetworkID has not been received so far:  The Network-ID is the PowerUp-ID which is the same as used for command Powerup Notification  (b) Command AssignNetworkID has already been received:  The Network-ID is the Direct-ID which is the same as assigned by command AssignNetworkID	No	void	---

System commands	Safety relevant	Data	Remark
GetCustomerDataAck <i>Network-ID</i> = same Network-ID as command GetCustomerData was received	No	8 byte Customer-data	<u>8 byte Customer-data</u> Data[0:7] = Customer-data. See Note 3.
<p>NOTE 1 There is no need to define those commands as safety critical, since the Network-ID is checked by the ILT-InterfaceBox by receiving command AssignNetworkIDAck.</p> <p>NOTE 2 The flashing of an aspect shall be so short (max. 50 ms) and combined with a long dark phase (at least 950 ms) so that this is not seen as safety critical even if the telegrams are sent out or received uncontrolled. See also chapter Identification procedure for topology B.</p> <p>NOTE 3 The 8 byte Customer-data:</p> <ul style="list-style-type: none"> <li>■ is stored in a non-volatile memory as it is.</li> <li>■ is normally used to detail the Sub-type.</li> <li>■ is seen as cleared when all bytes of it are set to 0x00.</li> </ul>			

### Direct addressed commands

The Table 23 shows the system commands which are sent with a Direct-identifier as Network-ID. The commands are ignored by the ILT-component when they are sent with the Broadcast-ID.

**Table 23 – Direct-addressed system commands**

System commands	Safety relevant	Data	Remark
GetComponentType	No	Void	
GetComponentTypeAck	No	FW-type 8 bit HW-description 24 bit FW-description 24 bit Error-information 8 bit	<p><u>FW-Type</u> Data[0]&lt;0:7&gt; = FW-type &lt;0:7&gt;</p> <ul style="list-style-type: none"> <li>■ Unique identifier within Device-type and Manufacturer-ID.</li> <li>■ With this information, the STL-IF-Box is able to detect all the ILT-Components that are affected by a FW-update.</li> </ul> <p><u>HW-description</u> Data[1]&lt;0:7&gt; = HW-description &lt;0:7&gt; Data[2]&lt;0:7&gt; = HW-description &lt;8:15&gt; Data[3]&lt;0:7&gt; = HW-description &lt;16:23&gt;</p> <p><u>FW-description</u> Data[4]&lt;0:7&gt; = FW-description &lt;0:7&gt; Data[5]&lt;0:7&gt; = FW-description &lt;8:15&gt; Data[6]&lt;0:7&gt; = FW-description &lt;16:23&gt;</p> <p><u>Error-information</u> Data[7]&lt;0:6&gt; = not used Data[7]&lt;7&gt; = 0 ... Ok, 1 ... Error (requested information not available).</p>

System commands	Safety relevant	Data	Remark
SetNetworkIDMask (see Note 1)	No	Index, Mask, Filter	<u>Index</u> Data[0]<0:3> = Index<0:3>  <u>Mask</u> Data[0]<4:7> = Mask<0:3> Data[1]<0:7> = Mask<4:11> Data[2]<0:7> = Mask<12:19> Data[3]<0:7> = Mask<20:27> Data[4]<0> = Mask<28> Data[4]<1> = not used  <u>Filter</u> Data[4]<2:7> = Filter<0:5> Data[5]<0:7> = Filter<6:13> Data[6]<0:7> = Filter<14:21> Data[7]<0:6> = Filter<22:28> Data[7]<7> = not used  Value 0x00000 (Broadcast-ID) for Filter is not allowed.
SetNetworkIDMaskAck	No	Reply Index, Mask and Filter or Error-byte	<u>Index</u> Data[0]<0:3> = Index<0:3>  <u>Mask</u> Data[0]<4:7> = Mask <0:3> Data[1]<0:7> = Mask<4:11> Data[2]<0:7> = Mask<12:19> Data[3]<0:7> = Mask<20:27> Data[4]<0> = Mask<28> Data[4]<1> = not used  <u>Filter</u> Data[4]<2:7> = Filter<0:5> Data[5]<0:7> = Filter<6:13> Data[6]<0:7> = Filter<14:21> Data[7]<0:6> = Filter<22:28> Data[7]<7> = not used  In case of error, only Data[0] is returned as <u>Error-byte</u> in the following way: Data[0]<0> = 1 = invalid index Data[0]<1> = 1 = invalid mask Data[0]<2> = 1 = invalid filter Data[0]<3:6> = 1 = not used Data[0]<7> = 1 = Common error indication (if any bit of Data[0]<0:6> is set or if another error has occurred).
SetCustomerData	No	8 byte Customer-data	<u>8 byte Customer-data</u> Data[0:7] = Customer-data Details to Customer-data see command GetCustomerDataAck.
SetCustomerDataAck	No	8 bit Status	<u>8 bit Status</u> Data[0] ... <ul style="list-style-type: none"> <li>■ 00h = Ok</li> <li>■ 80h = Customer data is not stored</li> </ul>
GetProfile	No	void	--  NOTE Request is not defined as safety-relevant as it cannot cause any negative impact, it only makes the component send data.



System commands	Safety relevant	Data	Remark
GetProfileAck	No	8 bit Safety / protocol information 16 bit Command set information 16 bit reserved for future command sets 16 bit Communication protocol specification version	<u>8 bit Safety / protocol Information</u> Data[0]<0> = 1 = approval according EN 50556 Data[0]<1> = 1 = approval according EN 61508 Data[0]<2:6> = 0 = reserved Data[0]<7> = 1 = compliant to ILT specification <u>16 bit Command set information</u> Data[1]<0> = 1 Command set Aspect Data[1]<1> = 1 Command set Pedestrian push button Data[1]<2> = 1 Command set Acoustic Data[1]<3> = 1 Command set Traffic sensor Data[1]<4..7> = 0 (reserved) Data[2]<0..7> = 0 (reserved) <u>16 bit reserved for future command sets</u> Data[3]<0..7> = 0 (reserved) Data[4]<0..7> = 0 (reserved) <u>16 bit Communication protocol specification version</u> Data[5]<0..7> = protocol main (major) version Data[6]<0..7> = protocol sub (minor) version (see Note 2) NOTE The command set bits correspond to the specified type-specific command sets described within the type-specific communication protocol document parts 2 to 5 (and probably following).
GetDeviceID	No	void	--
GetDeviceIDAck	No	Standardized-ID, Unique-ID	<u>Standardized-ID</u> Data[0]<0:7> = Device-type <0:7> Data[1]<0:7> = Sub-type <0:7> Data[2]<0:7> = Manufacturer-ID <0:7> <u>Unique-ID</u> Data[3]<0:7> = Serial number <0:7> Data[4]<0:7> = Serial number <8:15> Data[5]<0:7> = Serial number <16:23> Data[6]<0:7> = Serial number <24:31> Data[7]<0:6> = Serial number <32:38> Data[7]<7> = 0
<p>NOTE 1 Concerning to command <i>SetNetworkIDMask</i></p> <ul style="list-style-type: none"> <li>Since the parameters Mask and Filter follows the rules of how a 29 bit ILT-CAN identifier is built, only the 16-bit-field for the Network-ID is used, all other bits are ignored.</li> <li>ILT-Components have to support at least 4 filters. Internally, there are 2 additional filters which are not configurable by the traffic-controller. One of these 2 filters is configured to receive broadcasts from the traffic-controller and the second one is configured to receive direct addressed commands (using Network-ID) from the traffic-controller.</li> </ul> <p>NOTE 2 Concerning command <i>GetProfileAck</i></p> <ul style="list-style-type: none"> <li>The “communication protocol specification version” names the protocol-version of the implemented communication specification on which the implementation of the ILT component software is based, and which is supported COMPLETELY.</li> <li>For the first usable versions of ILT components designed for field testing, which do NOT completely support the features and requirements of a specification version <math>\geq 4.0</math>, the following <b>number range is reserved: main version == 3, sub version <math>\geq 5</math></b>. A table has to be supplied with each of these software releases clearly explaining the differences.</li> </ul>			

### 5.3.2 System commands and parameters which apply to all ILT components.

The Table 24 shows the system commands which apply to all ILT components.

Table 24 – System commands

System commands					
Telegram	Type	Safety relevant	Data	Remark	
GetInternalData	Req	No	4 bit InternalData specification 2 bit Sequence number	<p>Get internal data defined for individual ILT components.</p> <p><u>4 bit InternalData specification</u> Data[0]&lt;0:3&gt; = See section of the individual ILT components.</p> <p><u>2 bit Sequence number</u> Data[0]&lt;4:5&gt; =</p> <ul style="list-style-type: none"> <li>■ 0x00 = Start or restart the <i>GetInternalData</i> operation (see Notes)</li> <li>■ 0x01 – 0x03 = see Note</li> </ul> <p>NOTE 1 The <i>GetInternalData</i> operation is started with the sequence-number 0x00, on each subsequent command <i>GetInternalData</i>, the sequence-number is incremented. However, the overflow from 0x03 is 0x01 because 0x00 is reserved to indicate the start or restart of the <i>GetInternalData</i> operation.</p> <p>NOTE 2 The sender shall wait for the answer before sending a subsequent command <i>GetInternalData</i>.</p>	

System commands				
Telegram	Type	Safety relevant	Data	Remark
GetInternalDataAck	Resp	No	4 bit InternalData specification 2 bit Sequence number 2 bit Status Up to 56 bits of internal data	<p><u>4 bit InternalData specification</u></p> <p>Data[0]&lt;0:3&gt; = see command <i>GetInternalData</i></p> <p><u>2 bit Sequence number</u></p> <p>Data[0]&lt;4:5&gt; = Copied value of the received Sequence-number.</p> <p><u>2 bit Status</u></p> <p>Data[0]&lt;6:7&gt;...</p> <ul style="list-style-type: none"> <li>■ 0x00 = Continue because additional data is possibly available (Note 1)</li> <li>■ 0x01 = End of record (Note 2)</li> <li>■ 0x02 = Any error occurred (Note 3)</li> <li>■ 0x03 = Reserved (command <i>GetInternalData</i> is ignored)</li> </ul> <p><u>Up to 56 bits of internal data</u></p> <p>Data[1] = Requested internal data &lt;0:7&gt;            Data[2] = Requested internal data &lt;8:15&gt;            Data[3] = Requested internal data &lt;15:23&gt;            Data[4] = Requested internal data &lt;24:31&gt;            Data[5] = Requested internal data &lt;32:39&gt;            Data[6] = Requested internal data &lt;40:47&gt;            Data[7] = Requested internal data &lt;48:55&gt;</p> <p>NOTE 1 The data in the payload consists of the internal data. The traffic-controller shall send a new command <i>GetInternalData</i> to see if there are additional data available.</p> <p>NOTE 2 The complete (requested) internal data has been sent. The payload consists of only Data[0]. The traffic-controller shall start a new <i>GetInternalData</i> operation (with sequence-number 0x00) to bring the ILT-Component into another state.</p> <p>NOTE 3 The payload consists of only Data[0]. The traffic-controller can either continue (with the same sequence-number) or it can start a new <i>GetInternalData</i> operation (with sequence-number 0x00).</p> <p>Possible errors:</p> <ul style="list-style-type: none"> <li>■ Wrong block-specification</li> <li>■ Wrong sequence-number</li> </ul>
SetTimeSync	Req	No	32 bit timestamp	<p>Periodically for synchronisation of Logbook</p> <p><u>32 bit timestamp</u></p> <p>Data[0] = timestamp &lt;0:7&gt;            Data[1] = timestamp &lt;8:15&gt;            Data[2] = timestamp &lt;16:23&gt;            Data[3] = timestamp &lt;24:31&gt;</p> <p>NOTE It makes no sense what the payload means, the ILT-Component is just requested to store the received payload.</p> <p>It is recommended to optimize the memory management within the ILT-components, that the time <math>T_{STS}</math> between two subsequent SetTimeSync commands stays in the range of <math>9 \text{ Min} \leq T_{STS} \leq 11 \text{ Min}</math>.</p>
SetTimeSyncAck	Resp	No	Void	

System commands				
Telegram	Type	Safety relevant	Data	Remark
GetFailure	Req	No	-	Get diagnostic information about the FailureState
GetFailureAck	Resp	No	8 bit standard Error code 8 bit manufacturer and device type specific Error code 16 bit manufacturer and device type specific advanced information	The same information as transmitted in the command EnterFailureState <u>8 bit standard Error code</u> Data[0] = 0 = Not allowed 1 = HW error (unspecified HW error) 2 = SW error (unspecified SW error) 3 = Temperature (too high) 4 = Internal CPU error (self-test failed) 5 = Output error (Light, vibration, sound, etc.) 6 = Safety checking failed (safety integrity violated) 7 = Communication error (CAN communication failed caused by errors which are specified by command ComWarning) 8 = Alive timeout 9 = external power wiring 10 = Traffic sensor fault generic 11 - 255= reserved <u>8 bit manufacturer and device type specific Error code</u> Data[1] = For aspect, see Table 18 "aspect command EnterFailureState". NOTE In case of the occurrence of successive failures, the LATEST standard and manufacturer specific error code is sent (so with periodically sending out the command GetFailure, the ILT-Interface Box is also able to detect the appearance of successive failures of the ILT Component. <u>16 bit manufacturer and device type specific advanced information</u> Data[2:3] = advanced information
GetWarning	Req	No	8 bit OpCode	Get detailed warning information (which is defined for individual ILT components) associated to the status field in command Alive. <u>8 bit OpCode</u> Data[0] ... Warning specification, see Table 25 below "command GetWarningAck" OpCode = 0 generic Warning class 1 - 16 manufacturer specific additional information to each warning class

System commands				
Telegram	Type	Safety relevant	Data	Remark
GetWarningAck	Resp	No	8 bit OpCode specification 8 bit Status 16 bit Info	<u>8 bit OpCode specification</u> Data[0] = Copy of received OpCode  <u>8 bit Status</u> Data[1] ... <ul style="list-style-type: none"> <li>■ 00h = Ok</li> <li>■ 80h = Invalid Warning specification</li> </ul> <u>16 bit Info</u> Data[2:3] = see Table 25 – Command GetWarningAck  NOTE In case an unknown OpCode was received by command GetWarning, then the payload consists of only Data[0:1].

Table 25 – Command GetWarningAck

Command <i>GetWarningAck</i>			
Detailed information concerning to bit-field Status in command <i>AliveAck</i>			
Warning specification	Title	Data	Remark
00	Warning class	8 bit OpCode (Warning spec.) 8 bit Status 8 bit standard WarningInfo 8 bit manufacturer and device type specific WarningInfo	<u>8 bit OpCode</u> Data[0] = 0x00  <u>8 bit Status</u> Data[1] = see Table 24 GetWarningAck  <u>8 bit standard WarningInfo</u> Data[2]<0>: Warning specification 01 Actuator (i.e. LED for aspect) Data[2]<1>: Warning specification 02 Temperature Data[2]<2>: Warning specification 03 Sensors (i.e. Touch-Sensor) Data[2]<3>: Warning specification 04 Reserved Data[2]<4>: Warning specification 05 Reserved Data[2]<5>: Warning specification 06 Reserved Data[2]<6>: Warning specification 07 Reserved Data[2]<7>: Warning specification 08 Reserved  <u>8 bit manufacturer and device type specific WarningInfo</u> Data[3]<0>: Warning specification 09 Data[3]<1>: Warning specification 10 Data[3]<2>: Warning specification 11 Data[3]<3>: Warning specification 12 Data[3]<4>: Warning specification 13 Data[3]<5>: Warning specification 14 Data[3]<6>: Warning specification 15 Data[3]<7>: Warning specification 16  0 ... no warning 1 ... Warning pending  NOTE For details and specification of used Warning info refer to the appropriate product datasheet in chapter "ILT communication Interface (CAN-Bus) / Warning classes"
01 - 16	Warning specification → Detailed information for each warning class For detailed description of warning specifications see appropriate chapter "ILT communication Interface (CAN-Bus) / Warning classes" Warning specification    01    corresponds with WarningInfo    Data[2]<0> (= Actuator) 02    corresponds with WarningInfo    Data[2]<1> (= Temperature) : 16    corresponds with WarningInfo    Data[3]<7>		

### 5.3.3 Combination of command sets

Often, several functions are combined in one unit, for example Pedestrian push buttons and acoustic signals (including vibrators). Since every single physical device can only have one address or NetworkID, the features provided are indicated by the device-type and the supported CAN commands are defined via the *GetProfile* (in the example above, for a combined unit of Pedestrian push button and acoustic signals the bit <1>"Command set Pedestrian push button" and bit<2>"Command set Acoustic" are set)

## 5.4 General technical annotations on and boundary conditions to ILT devices

### 5.4.1 Technical annotation A, Telegram command reference

For all telegrams, the direction bit within the ID field of the command shall be set to '0' when the Master sends telegrams to Slaves. The direction bit shall be set to '1' when Slaves transmit to the Master. For a Slave to send an Ack to a command the Slave shall send a telegram with the actual command and the direction bit set to '1' back to the master. All responses have the same priority as the commands, unless specified differently. Only the Master may send a telegram as broadcast.

In Table 26 an overview about all telegrams and their use is given.

**Table 26 – Cross reference of system commands**

Commands									
Telegram	Cmd (hex) (Range = 0x00...0xA9)	Priority	Network-ID (16 bit)	Safety relevant	System	Aspect	Pedestrian PushButton	Acoustic	Traffic Sensor
Powerup Notification	00	PowerUp	Random number from PowerUp-ID	No	x				
AssignNetworkID	01	PowerUp	ID-less	No	x				
AssignNetworkIDAck	01	PowerUp	assigned Network-ID	No	x				
Identify	02	PowerUp	ID-less	No	x				
IdentifyAck	02	PowerUp	ID-less	No	x				
GetComponentType	03	Low	Direct	No	x				
GetComponentTypeAck	03	Low	Direct	No	x				
SetNetworkIDMask	04	Normal	Direct	No	x				
SetNetworkIDMaskAck	04	Normal	Direct	No	x				
Alive	05	Normal	Broadcast	Yes <sup>a</sup>	x				
AliveAck	05	Normal <sup>b</sup>	Direct	Yes <sup>a</sup>	x				
EnterKnownState	06	High	Direct	No <sup>c</sup>	x				
ComWarning	07	Normal	Direct	No	x				
ComWarningAck	07	Normal	Direct	No	x				
FWUpdateStart	08	Low	Broadcast or Direct	No	x				
FWUpdateStartAck	08	Low	Direct	No	x				
FWUpdateSendBlock	09	Low	Broadcast or Direct	No	x				
FWUpdateEnd	0A	Low	Broadcast or Direct	No	x				
FWUpdateEndAck	0A	Low	Direct	No	x				
FWUpdataFlash	0B	Low	Broadcast or Direct	No	x				
FWUpdataFlashAck	0B	Low	Direct	No	x				
SetOperationParameter	0C	Normal	Direct	Yes		x			
SetOperationParameterAck	0C	Normal	Direct	Yes		x			

Commands									
Telegram	Cmd (hex) (Range = 0x00...0xA9)	Priority	Network-ID (16 bit)	Safety relevant	System	Aspect	Pedestrian PushButton	Acoustic	Traffic Sensor
GetWarning	0D	Normal	Direct	No	x				
GetWarningAck	0D	Normal	Direct	No	x				
GetOperationData	0E	Normal	Direct	No		x			
GetOperationDataAck	0E	Normal	Direct	No		x			
GetInternalData	0F	Low	Direct	No	x				
GetInternalDataAck	0F	Low	Direct	No	x				
SetTimeSync	10	Low	Direct	No	x				
SetTimeSyncAck	10	Low	Direct	No	x				
SetCustomerData	11	Low	Direct	No	x				
SetCustomerDataAck	11	Low	Direct	No	x				
GetCustomerData	12	Low	Direct or PowerUp-ID	No	x				
GetCustomerDataAck	12	Low	Direct or PowerUp-ID	No	x				
GetProfile	13	Low	Direct	No	x				
GetProfileAck	13	Low	Direct	No	x				
StuckOnError	14	Error	PowerUp-ID	Yes <sup>a</sup>	x				
EnterFailureState	15	High	Direct	Yes	x				
GetFailure	16	Normal	Direct	No	x				
GetFailureAck	16	Normal	Direct	No	x				
GetDeviceID	17	Low	Direct	No	x				
GetDeviceIDAck	17	Low	Direct	No	x				
ParUpdateStart	18	Low	Broadcast or Direct	No	x				
ParUpdateStartAck	18	Low	Direct	No	x				
ParUpdateSendBlock	19	Low	Broadcast or Direct	No	x				
ParUpdateEnd	1A	Low	Broadcast or Direct	No	x				
ParUpdateEndAck	1A	Low	Direct	No	x				
ParUpdateFlash	1B	Low	Broadcast or Direct	No	x				
ParUpdateFlashAck	1B	Low	Direct	No	x				
GetParameterType	1C	Low	Direct	No	x				
GetParameterTypeAck	1C	Low	Direct	No	x				
ParRead	1D	Low	Direct	No	x				
ParReadAck	1D	Low	Direct	No	x				



Commands									
Telegram	Cmd (hex) (Range = 0x00...0xA9)	Priority	Network-ID (16 bit)	Safety relevant	System	Aspect	Pedestrian PushButton	Acoustic	Traffic Sensor
SignalOn	56	High	Direct	Yes		x			
SignalOnAck	56	High	Direct	Yes		x			
SignalOff	57	High	Direct	Yes		x			
SignalOffAck	57	High	Direct	Yes		x			
SetDimLevel	58	Normal	Direct	Yes		x			
SetDimLevelAck	58	Normal	Direct	Yes		x			
ForcedOn	59	Normal	Direct	Yes		x			
ForcedOnAck	59	Normal	Direct	Yes		x			
ForcedOff	5A	Normal	Direct	Yes		x			
ForcedOffAck	5A	Normal	Direct	Yes		x			
SetOutputStatePushButton	60	High	Direct	Yes			x		
SetOutputStatePushButtonAck	60	High	Direct	Yes			x		
InformationSound	61	Normal	Direct	No			x		
InformationSoundAck	61	Normal	Direct	No			x		
PushbuttonStatus	62	Normal	Direct	No			x		
PushbuttonStatusAck	62	Normal	Direct	No			x		
SetOperationParameterPushButton	63	Normal	Direct	Yes			x		
SetOperationParameterPushButtonAck	63	Normal	Direct	Yes			x		
GetOperationDataPushButton	64	Normal	Direct	No			x		
GetOperationDataPushButtonAck	64	Normal	Direct	No			x		
SetDimLevelPushButton	65	Normal	Direct	Yes			x		
SetDimLevelPushButtonAck	65	Normal	Direct	Yes			x		
SetOutputStateAcoustic	70	High	Direct	Yes				x	
SetOutputStateAcousticAck	70	High	Direct	Yes				x	
ForcedStateAcoustic	71	Normal	Direct	Yes				x	
ForcedStateAcousticAck	71	Normal	Direct	Yes				x	
ForcedSelftest	72	Normal	Direct	Yes				x	
ForcedSelftestAck	72	Normal	Direct	Yes				x	
SetOperationParameterAcoustic	73	Normal	Direct	Yes				x	
SetOperationParameterAcousticAck	73	Normal	Direct	Yes				x	
GetOperationDataAcoustic	74	Normal	Direct	No				x	
GetOperationDataAcousticAck	74	Normal	Direct	No				x	
SetAcousticLevel	75	Normal	Direct	Yes				x	
SetAcousticLevelAck	75	Normal	Direct	Yes				x	
DetectionStatus	80	Normal	Direct	No					x

Commands									
Telegram	Cmd (hex) (Range = 0x00...0xA9)	Priority	Network-ID (16 bit)	Safety relevant	System	Aspect	Pedestrian PushButton	Acoustic	Traffic Sensor
DetectionStatusAck	80	Normal	Direct	No					x
GetDetectionStatus	81	Normal	Direct	No					x
GetDetectionStatusAck	81	Normal	Direct	No					x
RisingEdgeStatus	82	Normal	Direct	No					x
FallingEdgeStatusStandard	83	Normal	Direct	No					x
SetOperationParameterTrafficSensor	84	Normal	Direct	No					x
SetOperationParameterTrafficSensorAck	84	Normal	Direct	No					x
GetOperationDataTrafficSensor	85	Normal	Direct	No					x
GetOperationDataTrafficSensorAck	85	Normal	Direct	No					x
FallingEdgeStatusClassification	86	Normal	Direct	No					x
OccupancyTimeInterval	87	Normal	Direct	No					x
OccupancyTimeIntervalAck	87	Normal	Direct	No					x
ResetDetectorAlignmentZone	88	Normal	Direct	No					x
ResetDetectorAlignmentZoneAck	88	Normal	Direct	No					x
<p><sup>a</sup> → logically safety-relevant, but not sent redundantly.</p> <p><sup>b</sup> → advantage of “normal”: initial occurrences of urgent errors, which are more important, are received faster</p> <p><sup>c</sup> → safety-related error message</p>									

#### 5.4.2 Boundary condition B, Dimming

It shall not be possible to set the dim level below or above the light intensity defined in CLC/TS50509 and EN 12368. This varies depending on country specific regulations and defined classes for the signal head.

In the *Signal-head*'s product datasheet, there is a min and a max brightness specified for day and night operation (according the standards mentioned above or customer requirements).

Unless the *ILT interface box* does specify which Range has to be used (using command *SetDimLevel*), the default value for *Range 0* is used.

Note, that any dim-level can be freely assigned to any Range and there are no restrictions which brightness a dim-level is assigned to. For instance, a lower dim-level can even consist of a higher brightness.

Figure 19 assumes, that:

- Dim-levels 0 to 7 are assigned to Range 1 (night-time).
- Dim-levels 8 to 15 are assigned to Range 0 (daytime).
- Assigned brightness to Dim-levels increases with the Dim-level

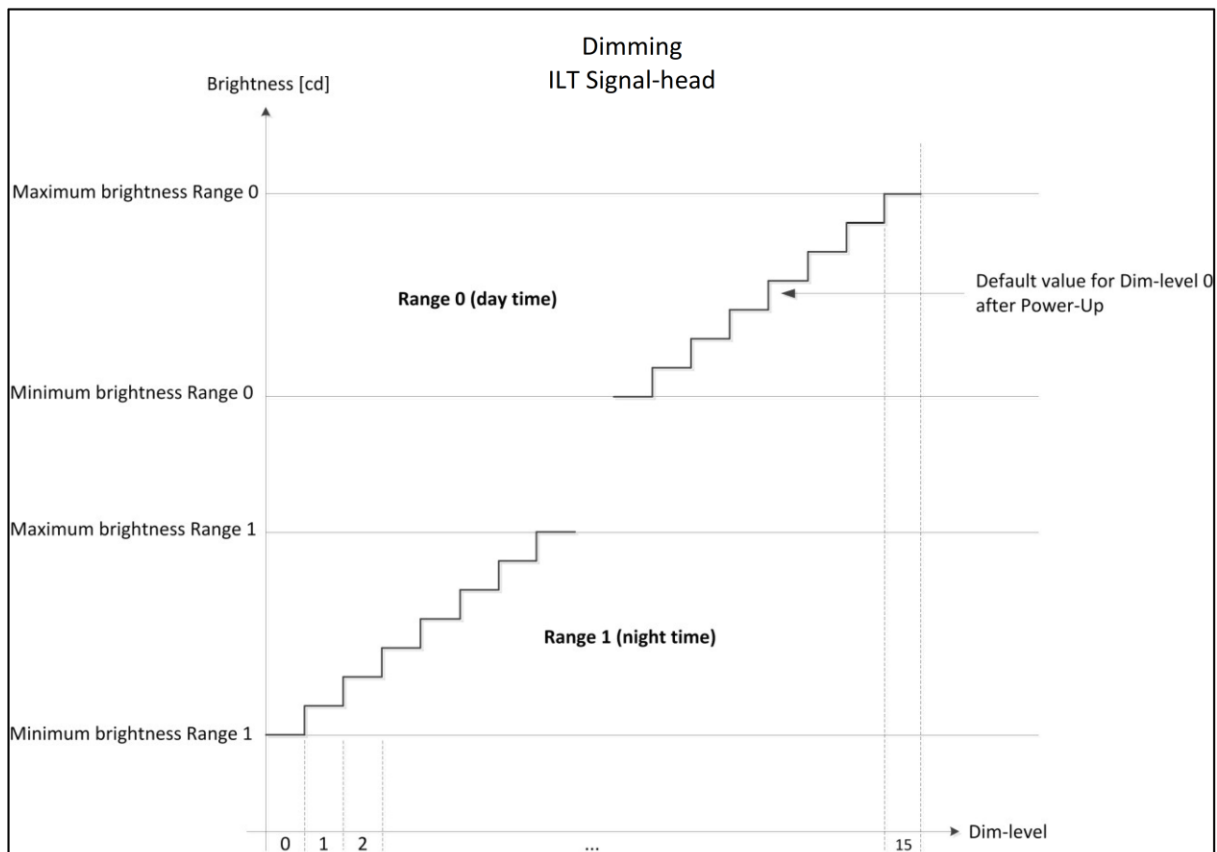


Figure 19 – Dim-levels

### 5.4.3 Boundary condition C, Service functions

Due to periodically maintenance & test on the traffic junction (according EN 50556) at a distance of 6 months we need a mechanism to simulation the complete failure chain of the complete system. This means the signal shall be forced ON (unintentionally ON) and forced OFF (unintentionally OFF).

#### Concept for implementation

- Creation of a “forced ON” and “forced OFF” command for safety reason a sequence is necessary, but redundant commands seem to be sufficient
- On ILT-Component (signal head) following action will be performed when receiving an “forced ON” or “forced OFF” command ...
  - The light source mask within the command AliveAck will be set (@forced ON) or reset (@forced OFF). This is done by the Main- $\mu$ C, there is no impact to the safety- $\mu$ C
  - The state of the LEDs is NOT changed (with a “forced ON” the LED will stay dark and with a “forced OFF” the LED will stay bright)
  - After a timeout of 100 ms the light source mask within the command AliveAck will be set correct again
- On ILT-IF-Box (traffic controller) following action shall be performed ...  
The traffic controller shall detect this “unintentionally ON” or “unintentionally OFF” and shall perform all necessary actions.

### 5.4.4 Boundary condition D, Signal state telegram sequencing requirements / example

#### 5.4.4.1 Signal command ACK and AliveAck

The change of the signal state is reported within the SignalOnAck / SignalOffAck telegrams. Since the AliveAck telegrams shows the actual state of the signal, the information within these telegrams shall

be consistent at any time, the latest telegram always shows the most recent state of the signal. That means that a telegram sent out prior to the SignalOnAck/SignalOffAck shows the previous state.

Example, sending a command sequence of: SignalOn(regular) => SignalOn(redundant) => Alive

- ⇒ possible response variant A: SignalOnAck (regular) => SignalOnAck(redundant) => AliveAck[state: on]  
(reason: delay between SignalOn(redundant) and Alive was long enough for the signal to complete the SignalOn command first)
- ⇒ possible response variant B: AliveAck[state: off] => SignalOnAck(regular) => SignalOnAck(redundant)  
(reason: delay between SignalOn(redundant) and Alive was shorter than the necessary time to execute the SignalOn command)
- ⇒ forbidden response (will be interpreted as failure by the ILT -Interface Box):  
SignalOnAck(regular) => SignalOnAck(redundant) => AliveAck[state: off]

#### 5.4.4.2 Signal commands in fast sequences

While in normal case a handshake (between the interface-box and the ILT-component) shall be met, meaning a new command can only sent out if the ACK of the previous command is received, for safety reasons it may be necessary for the ILT interface box to send out contrary signal commands in direct succession, possibly even without waiting for a previous SignalOnAck / SignalOffAck, for example if one signal fails and a second one has to change its state to retain a consistent and safe signalisation.

➔ **ignoring acknowledge responses is only allowed in case of exceptional situations caused by safety reasons! Regular communication (namely the interface box) is required to wait for acknowledges to each command.**

As the signal processes each command immediately after it has been received completely (which means both regular and redundant telegrams), there will be a correct response to each command without any added error or sequence reorder.

In any case of error, the most recent command is executed (as long as it is received successfully) and the cancelled command is negative acknowledged:

- ⇒ regular command / acknowledge sequence:  
(Interface box): SignalOn(regular+redundant) => SignalOff(regular+redundant)  
➔ (ILT component): SignalOnAck=> SignalOffAck
- ⇒ error caused by sequencing, still executing most recent command:  
(Interface box): SignalOn(regular) => SignalOff(regular+redundant) => SignalOn(redundant)  
➔ (ILT component): SignalOnNACK[Error: timeout/sequence] => SignalOffAck  
=> SignalOnNACK[Error: redu first]
- ⇒ sequencing error in both commands, none of them is executed:  
(Interface box): SignalOn(regular) => SignalOff(regular) => SignalOn(redundant) => SignalOff(redundant)  
➔ (ILT component): SignalOnNACK[Error: timeout/sequence] => SignalOffNACK[Error: timeout/sequence] => SignalOnNACK[Error: redu first] => SignalOffNACK[Error: redu first]

#### 5.4.4.3 Measures for ILT components addressing handling of fast command sequences

To be able to handle fast command sequences under all circumstances, the following concept has to be implemented within all ILT components:

- telegrams shall be separated according the telegram priority (see 5.2.2):
  - for all telegrams with either priority values  $\geq$  HIGH or commands seen as safety:  
at least 16 elementary telegrams shall be handled (which equals to 8 safety commands due to redundancy), suggested as “RXHIGH”.
  - for all telegrams with **priority values < HIGH** and **not seen as safety**:  
at least 8 elementary telegrams shall be handled, suggested as “RXMISC”

- the processing of the telegrams shall be weighted in such a way that
  - telegrams of RXHIGH have to be preferred
  - after processing 4 consecutive telegrams from RXHIGH, exactly one telegram from RXMISC has to be processed
  - → in normal operation this logic has no influence on sequential processing and only starts prioritizing as soon as multiple commands are waiting

## 5.5 Proposals

Following issues have been noted but not yet handled in detail:

- For the security of an access to ILT-components it seems interesting to provide an authorisation service via the ILT-interface. This could be relevant for the on-site configuration of (safety relevant) data of signals provision of new acoustic tone on blind peoples signal or other issues via a manufacturer individual Bluetooth or InfraRed interface.

## 6 Aspects

### 6.1 Command set Aspect and parameters

The *ILT interface box* shall be able to control the *aspect's* On and Off state of as well as its dimming level.

In case of a fatal error within the *aspect* it shall turn Off all LED's. This is named as *Known-State* of the *aspect*, not to be confused with the safe state of the system.

If a component supports the command set given in Table 27, this is reported by *GetProfile*.

**Table 27 – Aspect standard commands**

Standard commands for controlling aspects				
Telegram	Type	Safety relevant	Data	Remark
SignalOn	Req	Yes	16 bit Light Source Mask	Turns aspect on using the defined dim level. <u>16 bit Light Source Mask</u> Data[0]<0:7> Data[1]<0:7> 0 ... no change 1 ... turn light source ON  NOTE 1 Details to the use of multipoint signals see technical annotation C, "Multipoint Signals" NOTE 2 Information about used bits in the Light Source Mask and correlation to the physical light source see appropriate product datasheet of the signal, chapter "ILT communication Interface (CAN-Bus) / Signal ON/OFF" NOTE 3 The time TONLIGHT between the complete reception of command <i>SignalOn</i> and reporting the state Signal-On in command <i>AliveAck</i> and <i>SignalOnAck</i> shall not be greater than 20 ms in order to fulfil the value TONLIGHT which is specified in norm TS50509.

Standard commands for controlling aspects				
Telegram	Type	Safety relevant	Data	Remark
SignalOnAck	Resp	Yes	8 bit Status + 16 bit light source mask error status ( <b>only in case of failure</b> )	<p><u>8 bit Status</u> See Table 12 – Common Status in safety-relevant telegrams</p> <p>Telegram-specific extension: Status&lt;0&gt;: 0 = OK 1 = Error setting light source, see added 16 bit light source mask error status (<b>only transmitted when this bit is set!</b>) Each bit in the returned light source mask error status set to “1” indicates that the corresponding light source has not reached the desired state.</p> <p>Status&lt;1&gt;: 0 = OK 1 = invalid light source was selected, no state change</p> <p>NOTE Command SignalOnAck is sent immediately as soon as the state of the aspect is recognized as On.</p>
SignalOff	Req	Yes	16 bit Light Source Mask	<p>Turns the signal off.</p> <p><u>16 bit Light Source Mask</u> Data[0]&lt;0:7&gt; Data[1]&lt;0:7&gt; 0 ... no <b>change</b> 1 ... turn light source OFF</p> <p>NOTE 1 Details to the use of multipoint signals see technical annotation C, “Multipoint Signals”</p> <p>NOTE 2 Information about used bits in the Light Source Mask and correlation to the physical light source see chapter “ILT communication Interface (CAN-Bus) / Signal ON/OFF”</p> <p>NOTE 3 The time <math>T_{OFFLIGHT}</math> between the complete reception of command <i>SignalOff</i> and reporting the state Signal-Off in command <i>AliveAck</i> and <i>SignalOffAck</i> shall not be greater than 15 ms in order to fulfil the value <math>T_{OFFLIGHT}</math> which is specified in norm TS50509.</p>
SignalOff Ack	Resp	Yes	8 bit Status + 16 bit light source mask error status (only in case of failure)	<p><u>8 bit Status</u> See Table 12 – Common Status in safety-relevant telegrams</p> <p>Telegram-specific extension: Status&lt;0&gt;: 0 = OK 1 = Error setting light source, see added 16 bit light source mask error status (<b>only transmitted when this bit is set!</b>) Each bit in the returned light source mask error status set to “1” indicates that the corresponding light source has not reached the desired state.</p> <p>Status&lt;1&gt;: 0 = OK 1 = invalid light source was selected, no state change</p> <p>NOTE Command SignalOffAck is sent immediately as soon as the state of the signal is recognized as Off.</p>

Standard commands for controlling aspects				
Telegram	Type	Safety relevant	Data	Remark
SetDimLevel	Req	Yes	8 bit Dim-level 8 bit Range	<p>Setting the level for all aspects, shifting to the new Dim-level over the specified time.</p> <p><u>8 bit Dim-level</u> Data[0]&lt;0:7&gt; = 0 (<i>Dim-level 0</i>) to 15 (<i>Dim-level 15</i>), any value higher than 15 is invalid.</p> <p><u>8 bit Range</u> Data[1]&lt;0:7&gt; ...</p> <ul style="list-style-type: none"> <li>■ 0 = Range 0 (Day-time)</li> <li>■ 1 = Range 1 (Night-time)</li> <li>■ 2 to 255 = not used</li> </ul> <p>NOTE 1 In case, the telegram <i>SetDimLevel</i> is never sent by the traffic-controller, the ILT-component uses <i>Dim-level 0</i> as actual Dim-level. The same applies when the telegram <i>SetDimLevel</i> is sent only with invalid or uninitialized Dim-levels.</p> <p>NOTE 2 The Dim-level shall be valid and initialized in order the Dim-level is taken over by the <i>ILT-component</i>.</p> <p>NOTE 3 The Range shall match to the Range which was set by <i>DefDimLevel</i>. The Range is explicitly used here as redundant (safety) information. Answer telegram consists of according error-message if it does not match.</p> <p>NOTE 4 Default value for <i>Dim-level 0</i> see <i>SetOperationParameter</i> for operation <i>DefDimLevel</i>.</p>
SetDimLevel Ack	Resp	Yes	8 bit Status	<p><u>8 bit Status</u> Data[0]&lt;0&gt; ...</p> <ul style="list-style-type: none"> <li>■ 0 = Dim-level valid,</li> <li>■ 1 = Dim-level invalid</li> </ul> <p>Data[0]&lt;1&gt; ...</p> <ul style="list-style-type: none"> <li>■ 0...Dim-level initialized</li> <li>■ 1...Dim-level not initialized by <i>SetOperationParameter</i></li> </ul> <p>Data[0]&lt;2&gt; ...</p> <ul style="list-style-type: none"> <li>■ 0...Range matches to the Range which was set by <i>DefDimLevel</i></li> <li>■ 1...Range does not match to the Range which was set by <i>DefDimLevel</i></li> </ul> <p>Data[0]&lt;3:4&gt; = unused bits Data[0]&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</p>

Standard commands for controlling aspects				
Telegram	Type	Safety relevant	Data	Remark
SignalForcedOn	Req	Yes	Cmd	Service function for simulation of aspect failure. With this command all bits representing a valid light source within the light source mask of the AliveAck will be set while the state of the LEDs is NOT changed After a timeout of 100 ms the light source mask within the command AliveAck will be set correct again  for details see chapter "service function" see Note 1 see Note 2
ForcedOnAck	Resp	Yes	8 bit Status	<u>8 bit Status</u> Status<0>: 0 = OK 1 = interval too short or ForcedOff is running; command ignored Status<1:4>: 0 = reserved Status<5:7>: See Table 12 – Common Status in safety-relevant telegrams
ForcedOff	Req	Yes	Cmd	Service function for simulation of aspect failure. With this command all bits within the light source mask of the AliveAck will be reset while the state of the LEDs is NOT changed After a timeout of 100 ms the light source mask within the command AliveAck will be set correct again  for details see chapter "service function" see Note 1 see Note 2
ForcedOffAck	Resp	Yes	8 bit Status	<u>8 bit Status</u> Status<0>: 0 = OK 1 = interval too short or ForcedOn is running; command ignored Status<1:4>: 0 = reserved Status<5:7>: See Table 12 – Common Status in safety-relevant telegrams

NOTE 1 Retriggerring of the command is not allowed. The smallest interval between two subsequent commands of the same type is at least 1 second. Within this interval, any command of the same type is rejected, and a negative status is reported.

NOTE 2 A running ForcedOn command cannot be interrupted or overruled by a ForcedOff and vice versa.

## 6.2 Command SetOperationParameter

The commands in Table 28 shall be supported by all ILT-components using the aspect command set. If the specified functionality is not implemented the component shall still send a suitable answer to the master (Traffic-controller).

Example, if a not implemented parameter is accessed by command *SetOperationParameter*, a NACK shall be returned.



**Table 28 – Command SetOperationParameter**

Command SetOperationParameter				
Telegram	Type	Safety relevant	Data	Remark
SetOperationParameter	Req	Yes <sup>a</sup>	8 bit OpParameter Up to 56 bit OpData	Set operation parameter defined for individual ILT components.  <u>8 bit OpParameter</u> Data[0]<0:7> ... Details see Table 31 – Aspect operation data specification.  OpParameter = 0 - 127 generic Operational Parameter 128 - 255 manufacturer specific Operational Parameter  <u>Up to 56 bit OpData</u> Data[1:7] ... How many bytes respectively bits are used depends on the specified operation parameter.
SetOperationParameterAck	Resp	Yes <sup>a</sup>	8 bit OpParameter 8 bit Status	<u>8 bit OpParameter</u> Data[0]<0:7> = Copy of received OpParameter  <u>8 bit Status</u> Data[1]<0:4> ... Details see Table 31 – Aspect operation data specification  Data[1]<5:7> = Table 12 – Common Status in safety-relevant telegrams
<sup>a</sup> Telegram has to be defined “safety critical” because the definition of brightness in Range 0 (Day-mode) and in Range 1 (Night-mode) becomes safety critical since too dark light level is safety relevant.				

**Table 29 – Aspect operation parameter specification**

Aspect operation parameter specification (command SetOperationParameter) May be set by the IF-Box using command SetOperationParameter			
OpParameter	Title	Data	Remark
00	---	---	Not used
01	DefDimLevel	8 bit operation parameter (OpParameter) 8 bit Dim level 16 bit Brightness [cd] 8 bit Range	Defining the brightness for the specified Dim-level.  <u>8 bit OpParameter</u> Data[0]<0:7> = 0x01  <u>8 bit Dim-level</u> Data[1]<0:7> = Dim-level 0 to Dim-level 15, any value higher than 15 is invalid.  <u>16 bit Brightness</u> Data[2]<0:7> = Brightness [cd] <0:7> Data[3]<0:7> = Brightness [cd] <8:15>  <u>8 bit Range</u> Data[4]<0:7> = 0 = Range 0 (day-mode) 1 = Range 1 (night-mode) Any value higher than 1 is invalid.  NOTE 1 The default values for Dim-level 0 are: <ul style="list-style-type: none"> <li>Brightness [cd] = set via parametrization of the device during manufacturing</li> <li>Range = 0</li> </ul>

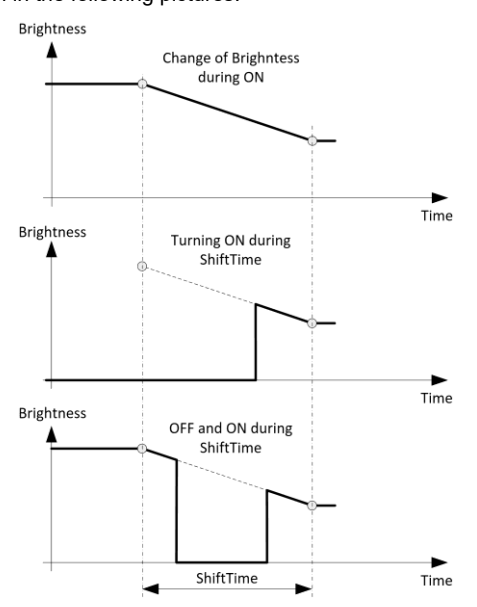
Aspect operation parameter specification (command *SetOperationParameter*)

May be set by the IF-Box using command *SetOperationParameter*

OpParameter	Title	Data	Remark
			<p>NOTE 2 The default values for Dim-level 1 to Dim-level 15 are (values means the according Dim-level is disabled):</p> <ul style="list-style-type: none"> <li>Brightness = 0 [cd]</li> <li>Range = 2 (or any invalid number for Range)</li> </ul> <p>NOTE 3 If an invalid Dim-level and/or an invalid range is received, then the values are not taken over by the ILT-component and the Status in the Ack-telegram is:</p> <ul style="list-style-type: none"> <li>Status&lt;0&gt; = 1 (in case of invalid Dim-level)</li> <li>Status&lt;1&gt; = 0</li> <li>Status&lt;2&gt; = 1 (in case of invalid Range)</li> <li>Status&lt;3&gt; = 0</li> <li>Status&lt;4&gt; = 1 (common error indication)</li> <li>Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</li> </ul> <p>NOTE 4 If a brightness of 0 [cd] is received for Dim-level 0 or for the active Dim-level, then the values are not taken over by the ILT-component and the Status in the Ack-telegram is:</p> <ul style="list-style-type: none"> <li>Status&lt;0&gt; = 1 (error dim level)</li> <li>Status&lt;1&gt; = 1 (error brightness)</li> <li>Status&lt;2&gt; = 0</li> <li>Status&lt;3&gt; = 0</li> <li>Status&lt;4&gt; = 1 (common error indication)</li> <li>Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</li> </ul> <p>NOTE 5 If a value of 0 [cd] is received for Dim-level 1 to Dim-level 15 and the according Dim-level is not the active dim-level, then the according dim-level is disabled, and the Status in the Ack-telegram is:</p> <ul style="list-style-type: none"> <li>Status&lt;0:4&gt; = all bits are 0.</li> <li>Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</li> </ul> <p>NOTE 6 In case, the brightness [cd] is lower than the minimum brightness specified by the according Range, then the brightness is changed to the minimum brightness of that Range and the Status in the Ack-telegram is:</p> <ul style="list-style-type: none"> <li>Status&lt;0&gt; = 0</li> <li>Status&lt;1&gt; = 1 (warning brightness)</li> <li>Status&lt;2&gt; = 0</li> <li>Status&lt;3&gt; = 0</li> <li>Status&lt;4&gt; = 0</li> <li>Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</li> </ul> <p>NOTE 7 In case, the brightness [cd] is higher than the maximum brightness specified by the according Range, then the brightness is changed to the maximum brightness of that Range and the Status in the Ack-telegram is:</p> <ul style="list-style-type: none"> <li>Status&lt;0&gt; = 0</li> <li>Status&lt;1&gt; = 1 (warning brightness)</li> <li>Status&lt;2&gt; = 0</li> <li>Status&lt;3&gt; = 0</li> <li>Status&lt;4&gt; = 0</li> <li>Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</li> </ul>

Aspect operation parameter specification (command *SetOperationParameter*)

May be set by the IF-Box using command *SetOperationParameter*

OpParameter	Title	Data	Remark
			NOTE 8 Sending this telegram by the traffic-controller is optional.
02	DefDimShift Time	8 bit operation parameter (OpParameter) 8 bit DimShiftTime	<p>Defining the time span for linear changing the brightness.</p> <p><u>8 bit OpParameter</u> Data[0]&lt;0:7&gt; = 0x02</p> <p><u>8 bit DimShiftTime</u> Data[1]&lt;0:7&gt; = 0 [bit] to 255 [bit]</p> $DimShiftTime[s] = \frac{0.5 [s]}{bit} * Data[1]$ <p>NOTE 1 The default dim-shift-time is 1 second what relates to a decimal value of 2 [bit].</p> <p>NOTE 2 The maximum dim-shift-time is 127.5 [s] what relates to a decimal value of 255 [bit].</p> <p>NOTE 3 With the linear change in brightness, a uniform brightness change over the entire intersection can be achieved. Therefore, this linear change in brightness shall also be considered during the OFF-state of the signal as shown in the following pictures:</p>  <p>The figure consists of three vertically stacked graphs, each with 'Brightness' on the vertical axis and 'Time' on the horizontal axis. Vertical dashed lines indicate the start and end of the 'ShiftTime' interval.</p> <ul style="list-style-type: none"> <li><b>Top Graph:</b> Labeled 'Change of Brightness during ON'. It shows a horizontal line at a high brightness level. At the start of the ShiftTime interval, the brightness begins to decrease linearly until the end of the interval, where it levels off at a lower brightness.</li> <li><b>Middle Graph:</b> Labeled 'Turning ON during ShiftTime'. It shows a horizontal line at zero brightness. At the start of the ShiftTime interval, the brightness jumps to a high level and then decreases linearly until the end of the interval, where it levels off at a lower brightness.</li> <li><b>Bottom Graph:</b> Labeled 'OFF and ON during ShiftTime'. It shows a horizontal line at zero brightness. At the start of the ShiftTime interval, the brightness jumps to a high level, then drops to zero, and then increases linearly until the end of the interval, where it levels off at a lower brightness.</li> </ul>
3 - 127	Reserved for future use of generic Aspect operation parameter		
128 - 255	Reserved for manufacturer specific operational parameter Information about supported Operational Parameters see chapter "ILT communication Interface (CAN-Bus) / Operational Parameter"		

### 6.3 Command GetOperationData

Table 30 – Command GetOperationData

Command <i>GetOperationData</i>				
Telegram	Type	Safety relevant	Data	Remark
GetOperationData	Req	No	8 bit OpData specification	<p>Get operation data defined for the individual ILT components.</p> <p><u>8 bit OpData specification</u></p> <p>Data[0] = Details, see Table 31 – Aspect operation data specification.</p> <p>OpData = 0 - 127 generic Operational Data 128 - 255 manufacturer specific Operational Data</p>
GetOperationDataAck	Resp	No	8 bit OpData specification 8 bit Status Up to 48 bit Data	<p><u>8 bit OpData specification</u></p> <p>Data[0] = Copy of received OpData</p> <p><u>8 bit Status</u></p> <p>Data[1] ...</p> <ul style="list-style-type: none"> <li>■ 00h = Ok</li> <li>■ 80h = Invalid OpData specification</li> </ul> <p><u>Up to 48 bit Data</u></p> <p>Data[2:7] = Operation Data defined for individual ILT components. Details see Table 31 – Aspect operation data specification.</p> <p>NOTE In case an unknown OpData specification was received by command GetOperationData, then the payload consists of only Data[0:1].</p>

Table 31 – Aspect operation data specification

Aspect operation data specification (command GetOperationDataAck)			
May be retrieved by the system master using command GetOperationData (Table 30)			
OpData	Title	Data	Remark
00	Working Hour Meter	8 bit OpData specification 8 bit Status 24 bit Working hours	<p><u>8 bit OpData specification</u></p> <p>Data[0] = 0x01</p> <p><u>8 bit Status</u></p> <p>Data[1] = see GetOperationDataAck</p> <p><u>24 bit Working hours</u></p> <p>Data[2] = Working hours&lt;0:7&gt; Data[3] = Working hours&lt;8:15&gt; Data[4] = Working hours&lt;16:23&gt;</p>
01	Actual LED temperature	8 bit OpData specification 8 bit Status 8 bit LED temperature	<p><u>8 bit OpData specification</u></p> <p>Data[0] = 0x01</p> <p><u>8 bit Status</u></p> <p>Data[1] = see GetOperationDataAck</p> <p><u>8 bit LED junction temperature</u></p> <p>Data[2] = -128 [°C] to + 127 [°C]</p>

Aspect operation data specification (command GetOperationDataAck)

May be retrieved by the system master using command GetOperationData (Table 30)

OpData	Title	Data	Remark
02	Version of manufacturer specific information set	8 bit Version	<u>8 bit Version</u> Data[0] = 0 - 255 Version of the manufacturer specific set of information, containing Operational Parameter, Operational Data and additional information of Warning classes according appropriate product datasheet of the aspect, chapter "ILT communication Interface (CAN-Bus) / Version of information set"
03	Lifetime prediction	8 bit OpData specification 8 bit Status 24 bit remaining Lifetime	<u>8 bit OpData specification</u> Data[0] = 0x03 <u>8 bit Status</u> Data[1] = see <i>GetOperationDataAck</i> <u>24 bit remaining Lifetime</u> Data[2] = RLT <0:7> Data[3] = RLT <8:15> Data[4] = RLT <16:23> NOTE RLT ... remaining Lifetime in [hrs] (for details and definition see corresponding device specific section)
04	ItemNumber_0	8 bit OpData specification 8 bit Status 48 bit Item number 0 (Low)	<u>8 bit OpData specification</u> Data[0] = 0x04 (Low) or 0x05 or 0x06 or 0x07 or 0x08 (High) <u>8 bit Status</u>
05	ItemNumber_1	8 bit OpData specification 8 bit Status 48 bit Item number 1	Data[1] = see <i>GetOperationDataAck</i> <u>48 bit Item Number (n):</u>
06	ItemNumber_2	8 bit OpData specification 8 bit Status 48 bit Item number 2	Data[2] = Item number (n) <0:7> Data[3] = Item number (n) <8:15> Data[4] = Item number (n) <16:23> Data[5] = Item number (n) <24:31> Data[6] = Item number (n) <32:39> Data[7] = Item number (n) <40:47>
07	ItemNumber_3	8 bit OpData specification 8 bit Status 48 bit Item number 3	NOTE 1 (n) = 0 (Low), 1, 2, 3, 4 (High). NOTE 2 Up to 30 digit product item number is available by this OpData.
08	ItemNumber_4	8 bit OpData specification 8 bit Status 48 bit Item number 4 (High)	NOTE 3 ItemNumber (n) can be seen as an extension of the Unique ID. NOTE 4 ItemNumber is ASCII-String, filled with 0x00
9 - 127	Reserved for future use of generic Aspect operation data		
128 - 255	Reserved for manufacturer specific operational data Information about supported Operational Data see chapter 9.4, Table 50		

## 6.4 Command AliveAck

Table 32 – Aspect command AliveAck

Aspect command <i>AliveAck</i>		
Telegram	Data	Remark
AliveAck	6 bit common status 1 bit sum failure 1 bit sum warning 16 bit Status	<u>6 bit common status</u> Data[0]<0:5> ... see chapter "Alive telegram for critical components" <u>1 bit sum failure</u> Data[0]<6> ... see AliveAck Table 13 <u>1 bit sum warning</u> Data[0]<7> ... see AliveAck Table 13 <u>16 bit Status</u> - Light source status  Data[1]<0:7> Data[2]<0:7> 0 ... light source OFF 1 ... light source ON there is an exact 1:1 relation to the actual activated light sources, correlating with the light source mask of "Signal ON/OFF" commands  NOTE 1 The sum failure bit shall be set consistently and at the same time with a deviating light source status if the failure is the cause for the deviation.

## 6.5 Technical annotation E, Multipoint Signals

With the implementation of the Signal ON/OFF commands multipoint signals would be supported.

- Up to 16 light sources per aspect
- Synchronous blinking of different aspects
- Using the same command (ON/OFF) for 1 point signals as well as for multipoint signals

### Command ON/OFF:

- A bit set in the SignalON commands turns ON the assigned light source, a bit set in the SignalOFF command turns OFF the assigned light source  
 => 0000 0000 0100 1100 @ SignalON-command turns ON light source 2, 3 and 6  
 => 0000 0000 0000 1000 @ SignalOFF-command turns OFF light source 3
- one bit in light source mask corresponds with a light source
- The correlation between the bit position within the light source mask in the command and the position of the light source in the aspect shall be explicitly specified in the product datasheet of the signal. To avoid misunderstandings this information shall be found in a specific chapter within the product datasheet: "ILT communication interface (CAN-Bus) / SignalON".
- for ILT-components with only one light source it is mandatory to use bit <0>.
- The numbering of the light sources is not important but should follow the following rule: from Up to Down and left to right.

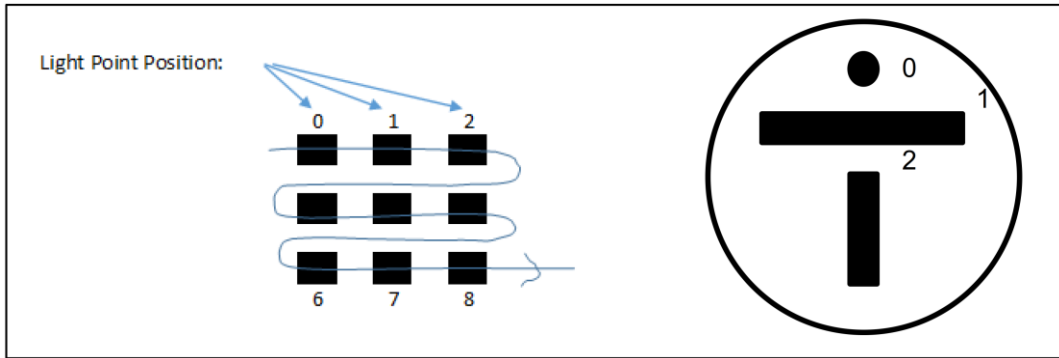


Figure 20 – Multi Point Signals (1)

- additional rule: for signals with only partially equipped light sources, the really equipped light source shall have the numbering shall be ident to the fully equipped signal.  
Example:

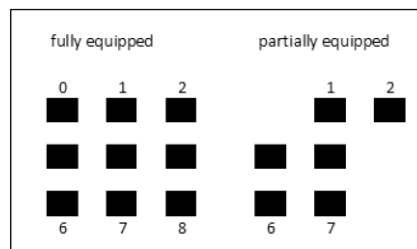


Figure 21 – Multi Point Signals (2)

- Based on the planning of the signal system, the position of a signal on the junction is known and during commissioning the relation allocation to the “unique ID” of the signal is done. Just as a “RED – left lane” is assigned to signal X, a “Negenooog 5point” is assigned to signal Y and a “Bern 5point” is assigned to signal Z (a combination of a Negenooog and Bern signal at the same junction is quite unrealistic but this example will show how to handle different multipoint signals with the same number of light sources and therefore maybe the same light source mask)

And from the data sheets of the Negenooog and Bern signal it is known which bit in the command belongs to which light source.

- The ILT-component checks if the bit set in the ON-Command and Off-Command corresponds with an existing light source. An illegal set bit in the ON command and Off-Command in the light source mask will result in an error, command is ignored. Error means, that a negative acknowledge is sent.
  - a) Exception Off-Command: With a light source mask 0xFFFF every signal turns OFF every light source.
  - b) An ON command and Off-Command with no single bit set (0x0000) will result in an error, command is ignored. Error means, that a negative acknowledge is sent.
- The ILT-component DOES NOT check if the combination of set bits in the light source mask is valid (valid combination of existing light sources) or makes sense!!
  - => The traffic controller (including the ILT interface box) is responsible to set only valid combination of light sources!
  - => This is an open solution and gives the most flexibility to the traffic controller (including the ILT-interface box).

EXAMPLE Following example in Figure 22 and Table 33 explains the valid and invalid light source masks in the ON- and OFF-Commands:

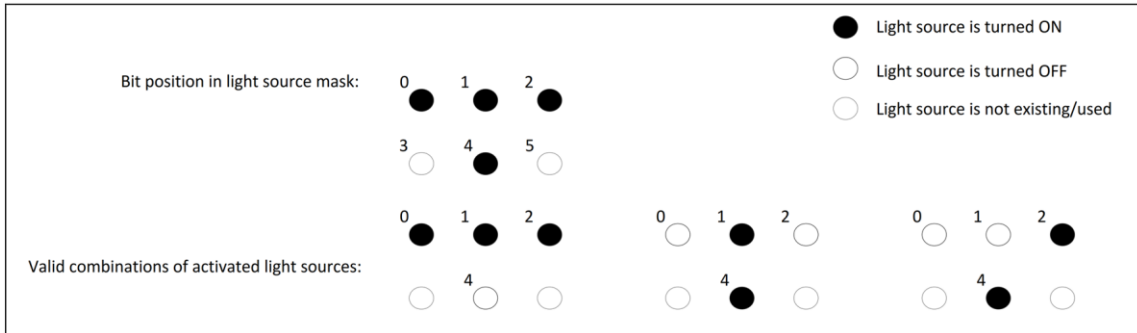


Figure 22 – Valid and invalid light source masks

Table 33 – Light source masks

Light source mask	Symbol shown at ILT-component		Remark
	Before the command	After the command	
ON-Command			
0x00	○ ● ○ ○ ● ○	Invalid, command ignored	no valid bit set in light source mask
0x01	○ ● ○ ○ ○ ○	● ● ○ ○ ○ ○	Although this is an invalid combination of activated light sources, the light sources are activated by the ILT-component. This failure is / shall be handled by the ILT-interface box.
0x18	● ● ● ○ ○ ○	Invalid, command ignored	Bit 3 is set in light source mask, but the light source is not existing
0x12	○ ○ ○ ○ ○ ○	○ ● ○ ○ ● ○	



Light source mask	Symbol shown at ILT-component		Remark
	Before the command	After the command	
OFF-Command			
0xFF			All light sources are turned OFF
0x04			Light source (bit 2 of light source mask) is already OFF
0x20		Invalid, command ignored	Bit 5 is set in light source mask, but the light source is not existing
0x01			Although this is an invalid combination of activated light sources, the light sources are deactivated by the ILT-component. This failure is / shall be handled by the ILT-interface box.

### Synchronous blinking

- The ILT-Interface box generates the blinking interval via turning ON and OFF the signal, but to reduce the bus load and to achieve a synchronous blinking, the group-addressing is used - there is only one command for all signals of the same type (same correlation of bit in light source mask and position of light source in signal is required).
- The blinking sequence is controlled by the ILT-Interface box!
- With the split ON/OFF-commands it is possible, that the same light source of several signals with a different actual appearance is turned ON & OFF.

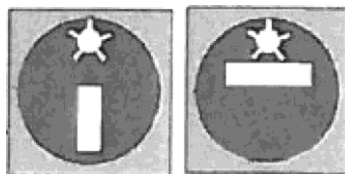


Figure 23 – Synchronous blinking

## 7 Pedestrian Push Button

### 7.1 Command set Pedestrian push buttons

This category includes: Push buttons, touch, and proximity sensors.

If a component supports the command set given in Table 34, this is reported by *GetProfile*.

Table 34 – Push Button standard commands

Standard commands for controlling Pedestrian-push-buttons				
Telegram	Type	Safety relevant	Data	Remark
SetOutputStatePushButton	Req	Yes <sup>a</sup>	8 bit state sequence counter 16 bit Optical information State Mask	<p>Turns optical information on, using the defined brightness.</p> <p><u>8 bit state Sequence counter</u> Data[0]&lt;0:3&gt; ... state Sequence counter Data[0]&lt;4:7&gt; ... 0 - reserved</p> <p><u>16 bit Optical information State Mask</u> Data[1:2]&lt;0:7&gt;: xxxx xxxx xxxx xx00 b = OpticalInformation OFF xxxx xxxx xxxx xx01 b = Demand ON xxxx xxxx xxxx xx10 b = Idle ON xxxx xxxx xxxx xx11 b = All ON (Demand &amp; Idle)</p> <p>"x": Product specific; Information about used bits in the Optical information State Mask and correlation to the performed optical information see section Push Button, chapter "ILT communication Interface (CAN-Bus) / Optical information State"</p> <p>NOTE 1 The content of the state sequence counter is not checked by the ILT-component.</p> <p>NOTE 2 The time T<sub>SIGNAL</sub> between the complete reception of command <i>SetOutputState</i> and reporting the information state in command <i>AliveAck</i> and <i>SetOutputStateAck</i> shall not be greater than 20 ms Within that time the push button shall check the correctness of the redundant commands. It is not expected that the selected signalling is already visible.</p>
SetOutputStatePushButton Ack	Resp	Yes <sup>a</sup>	8 bit Status 8 bit state sequence counter 16 bit Optical information State Mask <b>(only in case of failure)</b>	<p><u>8 bit Status</u> Data[0]&lt;5:7&gt;: See Table 12 – Common Status in safety-relevant telegrams</p> <p>Telegram-specific extension: Data[0]&lt;0&gt;:0 = OK 1 = Error setting information state, see added 16 bit information state mask error status (<b>only transmitted when this bit is set!</b>) In case of failure the actual state mask is transmitted, according to the <i>AliveAck</i>.</p> <p>Data[0]&lt;1&gt;: 0 = OK 1 = invalid information state was selected, no state change</p> <p>Data[0]&lt;2:4&gt;: 0 reserved</p> <p><u>8 bit state Sequence counter</u> Data[1]&lt;0:3&gt; ... state Sequence counter as received in corresponding <i>SetOutputState</i> telegram</p> <p>Data[1]&lt;4:7&gt; ... 0 - reserved</p> <p>NOTE Command <i>SetOutputStateAck</i> is sent immediately as soon as the state starts changing.</p>

Standard commands for controlling Pedestrian-push-buttons

Telegram	Type	Safety relevant	Data	Remark
InformationSound	Req	No	16 bit SoundMask	<p>With this command a selected sound is emitted once</p> <p><u>16 bit SoundMask</u></p> <p>Data[0]&lt;0&gt;: 0 = no sound 1 = activate acknowledge sound</p> <p>Data[0]&lt;2:3&gt;: 0 = Reserved</p> <p>Data[0]&lt;4:7&gt;: product specific sounds</p> <p>Data[1]&lt;0:7&gt;: product specific sounds</p> <p>NOTE 1 Data[0]&lt;0:3&gt; are reserved for standardized information sounds, Data[0]&lt;4:7&gt; and Data[1]&lt;0:7&gt; are reserved for manufacture/product specific information sounds.</p> <p>NOTE 2 This command is seen similar to the SignalON of an aspect, whereas the information sound is emitted only once. Therefore there is no command similar to the SignalOFF of an aspect</p> <p>NOTE 3 Sound mask "0000 0000 0000 0000" is invalid</p> <p>NOTE 4 The time TINFOSOUND between the complete reception of command <i>FlashSound</i> and <i>FlashSoundAck</i> shall not be greater than 20 ms. Within that time the push button shall check the plausibility of the command. It is not expected that the selected signalling is already audible.</p> <p>NOTE 5 The state of this sound is not shown within the AliveAck</p> <p>NOTE 6 The command is not seen as safety relevant since the ILT-Component has to check a minimum interval between two consecutive commands to prevent from a possible creation of a "walk sound" in case of a combined acoustic push button.</p>
InformationSoundAck	Resp	No	8 bit Status	<p><u>8 bit Status</u></p> <p>Data[0]&lt;0&gt;: 0 = SoundMask is valid 1 = invalid SoundMask</p> <p>Data[0]&lt;1&gt;: 0 = selected sound is available 1 = selected sound is not available - command ignored</p> <p>EXAMPLES:</p> <p>(a) sound is disabled by parametrisation</p> <p>(b) handled autonomous</p> <p>(c) walk-sound is being output</p> <p>Data[0]&lt;2&gt;: 0 = repetition out of timeout 1 = repetition within timeout - command ignored</p> <p>Data[0]&lt;3:6&gt;:0 = reserved</p> <p>Data[0]&lt;7&gt;: 0 = no error 1 = sumerror set when any bit &lt;0:6&gt; is set</p>
PushbuttonStatus	Req	No	void	Sent by IF-Box to get actual status of Push Button inputs

Standard commands for controlling Pedestrian-push-buttons

Telegram	Type	Safety relevant	Data	Remark
PushbuttonStatusAck	Resp	No	24 bit DemandClass	<p>Sent by ILT-Component (either as response to command PushbuttonStatus or spontaneously)</p> <p><u>24 bit DemandClass</u></p> <p>Data[0]&lt;0&gt;: Pedestrian 0 = not activated 1 = activated</p> <p>Data[0]&lt;1&gt;: impaired 0 = not activated 1 = activated</p> <p>Data[0]&lt;2:3&gt;: 0 = Reserved</p> <p>Data[0]&lt;4:7&gt;: product specific demands.</p> <p>Data[1]&lt;0:7&gt;: Information about used bits and</p> <p>Data[2]&lt;0:7&gt;: assignment to the physical demands see section Push Button, chapter "ILT communication (CAN-Bus) / demand classes"</p> <p>NOTE 1 Data[0]&lt;0:3&gt; are reserved for standardized inputs, Data[0]&lt;4:7&gt; are reserved for manufacture/product specific inputs. "Pedestrian" is seen as regular input for pedestrians, while "impaired" is seen as input for visually impaired persons.</p> <p>NOTE 2 For definition if command is sent spontaneously or not refer to Operational Parameter <i>DemandMode</i>.</p> <p>NOTE 3 For demand classes, not parametrized for spontaneously transmission, but activated and also de-activated again between two consecutive requests, the corresponding bit shall be set once.</p> <p>NOTE 4 The command is sent spontaneously as soon as one of the inputs as parametrized as "spontaneous" has changed, but it is transmitting the actual status of the complete demand class, including previous demands. Those previous demands are explained in technical annotation A under *) meaning that short-time demands shall not get lost.</p> <p>NOTE 5 Context to detection time see 7.5.1 technical annotation F, chapter "manually operated demands and logic inputs".</p>

Standard commands for controlling Pedestrian-push-buttons				
Telegram	Type	Safety relevant	Data	Remark
SetDimLevelPushButton	Req	Yes <sup>b</sup>	8 bit Dim-level 8 bit Range	<p>Setting the brightness of optical information, setting to the new brightness immediately.</p> <p><u>8 bit Dim-level</u> Data[0]&lt;0:7&gt; = 0 (<i>Dim-level 0</i>) to 15 (<i>Dim-level 15</i>), any value higher than 15 is invalid.</p> <p><u>8 bit Range</u> Data[1]&lt;0:7&gt; ...</p> <ul style="list-style-type: none"> <li>■ 0 = Range 0 (Day-time)</li> <li>■ 1 = Range 1 (Night-time)</li> <li>■ 2 to 255 = not used</li> </ul> <p>NOTE 1 In case, the telegram <i>SetDimLevel</i> is never sent by the traffic-controller, the ILT-component uses <i>Dim-level 0</i> as actual Dim-level. The same applies when the telegram <i>SetDimLevel</i> is sent only with invalid or uninitialized Dim-levels.</p> <p>NOTE 2 The Dim-level shall be valid and initialized in order the Dim-level is taken over by the <i>ILT-component</i>.</p> <p>NOTE 3 The Range shall match to the Range which was set by <i>DefDimLevel</i>. The Range is explicitly used here as redundant (safety) information. Answer telegram consists of according error-message if it does not match.</p> <p>NOTE 4 Default value for <i>Dim-level 0</i> see <i>SetOperationParameter</i> for operation <i>DefDimLevel</i>.</p>
SetDimLevelPushButton Ack	Resp	Yes <sup>b</sup>	8 bit Status	<p><u>8 bit Status</u> Data[0]&lt;0&gt; ...</p> <ul style="list-style-type: none"> <li>■ 0 = Dim-level valid,</li> <li>■ 1 = Dim-level invalid</li> </ul> <p>Data[0]&lt;1&gt; ...</p> <ul style="list-style-type: none"> <li>■ 0...Dim-level initialized</li> <li>■ 1...Dim-level not initialized by <i>SetOperationParameter</i></li> </ul> <p>Data[0]&lt;2&gt; ...</p> <ul style="list-style-type: none"> <li>■ 0...Range matches to the Range which was set by <i>DefDimLevel</i></li> <li>■ 1...Range does not match to the Range which was set by <i>DefDimLevel</i></li> </ul> <p>Data[0]&lt;3:4&gt; = unused bits Data[0]&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</p>
<p><sup>a</sup> no safety-relevant information, but for compatibility with other SetOutputState commands</p> <p><sup>b</sup> no safety-relevant information, but for compatibility with other SetDimLevel commands</p>				

## 7.2 Command SetOperationParameter

The commands in Table 35 shall be supported by all ILT-components using the Push-Button command set. If the specified functionality is not implemented the component shall still send a suitable answer to the master (Traffic-controller).

Example, if a not implemented parameter is accessed by command *SetOperationParameter*, a NACK shall be returned.

**Table 35 – Command SetOperationParameter**

Command <i>SetOperationParameter</i>				
Telegram	Type	Safety relevant	Data	Remark
SetOperationParameter PushButton	Req	Yes <sup>a</sup>	8 bit OpParameter Up to 56 bit OpData	Set operation parameter defined for individual ILT components. <u>8 bit OpParameter</u> Data[0]<0:7> ... Details see Table 36 – Push Button operation parameter specification. OpParameter = 0 - 127 generic Operational Parameter 128 - 255 manufacturer specific Operational Parameter <u>Up to 56 bit OpData</u> Data[1:7] ... How many bytes respectively bits are used depends on the specified operation parameter.
SetOperationParameter PushButtonAck	Resp	Yes <sup>a</sup>	8 bit OpParameter 8 bit Status	<u>8 bit OpParameter</u> Data[0]<0:7> = Copy of received OpParameter <u>8 bit Status</u> Data[1]<0:4> ... Details see Table 36 – Push Button operation parameter specification. Data[1]<5:7> = See Table 12 – Common Status in safety-relevant telegrams
<sup>a</sup> Although there is no actual need, the telegram has to be defined “safety critical” to keep it compatible to the command SetOperationParameter of aspect.				

**Table 36 – Push Button operation parameter specification**

Push-button operation parameter specification (command <i>SetOperationParameter</i> ) May be set by the IF-Box using command <i>SetOperationParameter</i>			
OpParameter	Title	Data	Remark
00	DemandMode	8 bit operation parameter (OpParameter) 24 bit DemandActiveMask 24 bit DemandInactiveMask	Defines if PushButtonStatusAck is sent spontaneously for each demand class, separately for activation and de-activation. 8 bit OpParameter Data[0]<0:7> = 0x00 <u>24 bit DemandActiveMask</u> for an activated demand (button is pressed), 0 = PushButtonStatusAck not sent spontaneously 1 = PushButtonStatusAck sent spontaneously Data[1]<0>: Pedestrian Data[1]<1>: Impaired Data[1]<2:3>: 0 = Reserved Data[1]<4:7>: product specific demands. Data[2]<0:7>: Information about used bits and Data[3]<0:7>: assignment to the physical demands see section Push Button, chapter “ILT communication (CAN-Bus) / demand classes” <u>24 bit DemandInactiveMask</u> for a de-activated demand (button is released), 0 = PushButtonStatusAck not sent spontaneously 1 = PushButtonStatusAck sent spontaneously Data[4]<0>: Pedestrian

Push-button operation parameter specification (command *SetOperationParameter*)

May be set by the IF-Box using command *SetOperationParameter*

OpParameter	Title	Data	Remark
			<p>Data[4]&lt;1&gt;: Impaired                      Data[4]&lt;2:3&gt;: 0 = Reserved                      Data[4]&lt;4:7&gt;: product specific demands.                      Data[5]&lt;0:7&gt;: Information about used bits and                      Data[6]&lt;0:7&gt;: assignment to the physical demands see section Push Button, chapter "ILT communication (CAN-Bus) / demand classes"</p> <p>NOTE 1 Default value after PowerUp is set to spontaneously transmission of PushButtonStatusAck as if command SetOperationParameter(OpParameter = 0x00) with the following data were received:                      Data[0] = 0x00 = OpParameter                      Data[1:3] = 0xFFFFFFFF = DemandActiveMask                      Data[4:6] = 0xFFFFFFFF = DemandInactiveMask</p> <p>NOTE 2 If a demand is parameterized to send PushButtonStatusAck spontaneously only for the de-activated event, then the de-activated event triggers PushButtonStatusAck for the activated event followed by PushButtonStatusAck for the de-activated event.</p> <p>Definition of Status in SetOperationParameterPushButtonAck:  <u>8 bit Status = Data[1]&lt;0:4&gt; = 0x00</u></p>
01	DefDimLevel	8 bit operation parameter (OpParameter) 8 bit Dim level 8 bit rel. Brightness [%] 8 bit Range	<p>Defining the relative brightness for the specified Dim-level.  <u>8 bit OpParameter</u>                      Data[0]&lt;0:7&gt; = 0x01</p> <p><u>8 bit Dim-level</u>                      Data[1]&lt;0:7&gt; = Dim-level 0 to Dim-level 15, any value higher than 15 is invalid.</p> <p><u>8 bit rel. Brightness</u>                      Data[2]&lt;0:7&gt; = relative Brightness 0 % ... 100 %, any value higher than 100 % is limited to 100 %</p> <p><u>8 bit Range</u>                      Data[3]&lt;0:7&gt; =                      ■ 0 = Range 0 (day-mode)                      ■ 1 = Range 1 (night-mode)                      ■ Any value higher than 1 is invalid.</p> <p>NOTE 1 The default values for Dim-level 0 are:                      • rel. Brightness = set via parametrization of the device during manufacturing                      • Range = 0</p> <p>NOTE 2 The default values for Dim-level 1 to Dim-level 15 are (values means the according Dim-level is disabled):                      • rel. Brightness = 0 [%]                      • Range = 2 (or any invalid number for Range)</p> <p>NOTE 3 If an invalid Dim-level and/or an invalid Range is received, then the values are not taken over by the <i>ILT-component</i> and the Status in the Ack-telegram is:                      • Status&lt;0&gt; = 1 (in case of invalid Dim-level)                      • Status&lt;1&gt; = 0                      • Status&lt;2&gt; = 1 (in case of invalid Range)                      • Status&lt;3&gt; = 0                      • Status&lt;4&gt; = 1 (common error indication)</p>

Push-button operation parameter specification (command *SetOperationParameter*)

May be set by the IF-Box using command *SetOperationParameter*

OpParameter	Title	Data	Remark
			<ul style="list-style-type: none"> <li>Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</li> </ul> <p>NOTE 4 If a rel. brightness of 0 [%] is received for <i>Dim-level 0</i> or for the active Dim-level, then the values are not taken over by the <i>ILT-component</i> and the Status in the Ack-telegram is:</p> <ul style="list-style-type: none"> <li>Status&lt;0&gt; = 1 (error dim level)</li> <li>Status&lt;1&gt; = 1 (error rel. brightness)</li> <li>Status&lt;2&gt; = 0</li> <li>Status&lt;3&gt; = 0</li> <li>Status&lt;4&gt; = 1 (common error indication)</li> <li>Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</li> </ul> <p>NOTE 5 If a rel. Brightness of 0 [%] is received for <i>Dim-level 1</i> to <i>Dim-level 15</i> and the according Dim-level is <u>not</u> the <u>active</u> dim-level, then the according dim-level is disabled, and the Status in the Ack-telegram is:</p> <ul style="list-style-type: none"> <li>Status&lt;0:4&gt; = all bits are 0.</li> <li>Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</li> </ul> <p>NOTE 6 If a rel. Brightness higher than 100 [%] is received, then the according rel. Brightness is limited to 100 [%] and the Status in the Ack-telegram is:</p> <ul style="list-style-type: none"> <li>Status&lt;0&gt; = 0.</li> <li>Status&lt;1&gt; = 1 (warning rel. brightness).</li> <li>Status&lt;2&gt; = 0.</li> <li>Status&lt;3&gt; = 0.</li> <li>Status&lt;4&gt; = 0.</li> <li>Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</li> </ul> <p>NOTE 7 Sending this telegram by the traffic-controller is optional.</p>
02	DetectionTime	8 bit operation parameter (OpParameter) 8 bit DetectionTime [ms]	<p>Defines the detection time (definition of and context to command <i>PushButtonStatusAck</i> see 7.5.1 technical annotation F, chapter “manually operated demands and logic inputs”)</p> <p><u>8 bit OpParameter</u> Data[0]&lt;0:7&gt; = 0x02</p> <p><u>8 bit DetectionTime</u> Data[1]&lt;0:7&gt; = 0 [ms] - 255 [ms] 0 [ms] means disabled function</p> <p>NOTE 1 The parametrized detection time can be rounded up by the <i>ILT-component</i> to a rational value without any notice to the <i>IF-Box</i> (i.e. detection time of 17 ms is rounded up to 20 ms for an <i>ILT-component</i> with 10 ms cycle time)</p> <p>NOTE 2 Default value after <i>PowerUp</i> is set according parametrization of the <i>ILT-component</i></p> <p>Definition of Status in <i>SetOperationParameterPushButtonAck</i>: <u>8 bit Status = Data[1]&lt;0:4&gt; = 0x00</u></p>
03-127	Reserved for future use of generic Push Button operation parameter		<p>Definition of Status in <i>SetOperationParameterPushButtonAck</i></p> <p><u>8 bit Status = Data[1]&lt;0:4&gt;</u> Status&lt;0&gt; = 0 Status&lt;1&gt; = 0</p>



Push-button operation parameter specification (command <i>SetOperationParameter</i> )			
May be set by the IF-Box using command <i>SetOperationParameter</i>			
OpParameter	Title	Data	Remark
			Status<2> = 0 Status<3> = 1 ... Wrong OpParameter Status<4> = 1 ... Common error indication
128 - 255	Reserved for manufacturer specific operational parameter Information about supported Operational Parameters see section Push Button, chapter "ILT communication Interface (CAN-Bus) / Operational Parameter"		Definition of Status in <i>SetOperationParameterPushButtonAck</i> <u>8 bit Status = Data[1]&lt;0:4&gt;</u> Status<0> = 0 Status<1> = 0 Status<2> = 0 Status<3> = 1 ... Wrong OpParameter Status<4> = 1 ... Common error indication <ul style="list-style-type: none"> <li>■ OpParameter, which are not listed in the appropriate product data sheet of push button, always return these status-bits.</li> <li>■ Details for OpParameter, see section push button.</li> </ul>

### 7.3 Command *GetOperationData*

Table 37 – Command *GetOperationData*:

Command <i>GetOperationData</i>				
Telegram	Type	Safety relevant	Data	Remark
<i>GetOperationDataPushButton</i>	Req	No	8 bit OpData specification	Get operation data defined for the individual ILT components. <u>8 bit OpData specification</u> Data[0] = Details, see table "Push-button operation data specification". OpData = 0 - 127 generic Operational Data 128 - 255 manufacturer specific Operational Data
<i>GetOperationDataPushButtonAck</i>	Resp	No	8 bit OpData specification 8 bit Status Up to 48 bit Data	<u>8 bit OpData specification</u> Data[0] = Copy of received OpData <u>8 bit Status</u> Data[1] ... <ul style="list-style-type: none"> <li>■ 00h = Ok</li> <li>■ 80h = Invalid OpData specification</li> </ul> <u>Up to 48 bit Data</u> Data[2:7] = Operation Data defined for individual ILT components. Details see table "Push-button operation data specification". NOTE In case an unknown OpData specification was received by command <i>GetOperationData</i> , then the payload consists of only Data[0:1].

**Table 38 – Push Button operation data specification**

Push-Button-signals operation data information (command <i>GetOperationDataAck</i> ) May be retrieved by the system master using the command <i>GetOperationData</i>			
OpData	Title	Data	Remark
00	Working Hour Meter	8 bit OpData specification 8 bit Status 24 bit Working hours	<u>8 bit OpData specification</u> Data[0] = 0x01 <u>8 bit Status</u> Data[1] = see <i>GetOperationDataAck</i> <u>24 bit Working hours</u> Data[2] = Working hours<0:7> Data[3] = Working hours<8:15> Data[4] = Working hours<16:23>
01	not used	not used	not used
02	Version of manufacturer specific information set	8 bit Version	<u>8 bit Version</u> Data[0] = 0 - 255 Version of the manufacturer specific set of information, containing Operational Parameter, Operational Data and additional information of Warning classes according appropriate product datasheet of the aspect, chapter "ILT communication Interface (CAN-Bus) / Version of information set
03	not used	not used	not used
04	ItemNumber_0	8 bit OpData specification 8 bit Status 48 bit Item number 0 (Low)	<u>8 bit OpData specification</u> Data[0] = 0x04 (Low) or 0x05 or 0x06 or 0x07 or 0x08 (High)
05	ItemNumber_1	8 bit OpData specification 8 bit Status 48 bit Item number 1	<u>8 bit Status</u> Data[1] = see <i>GetOperationDataAck</i> <u>48 bit Item Number (n):</u>
06	ItemNumber_2	8 bit OpData specification 8 bit Status 48 bit Item number 2	Data[2] = Item number (n) <0:7> Data[3] = Item number (n) <8:15> Data[4] = Item number (n) <16:23> Data[5] = Item number (n) <24:31>
07	ItemNumber_3	8 bit OpData specification 8 bit Status 48 bit Item number 3	Data[6] = Item number (n) <32:39> Data[7] = Item number (n) <40:47> NOTE 1 (n) = 0 (Low), 1, 2, 3, 4 (High).
08	ItemNumber_4	8 bit OpData specification 8 bit Status 48 bit Item number 4 (High)	NOTE 2 Up to 30 digit product item number is available by this OpData. NOTE 3 ItemNumber (n) can be seen as an extension of the Unique ID. NOTE 4 ItemNumber is ASCII-String, filled with 0x00
9-127	Reserved for future use of generic Push-Button operation data		
128 - 255	Reserved for manufacturer specific operational data For information on supported Operational Data, see the appropriate product datasheet of the Push Button Signal in chapter "ILT communication Interface (CAN-Bus) / Operational Data".		

## 7.4 Command AliveAck

Table 39 – Push Button command AliveAck

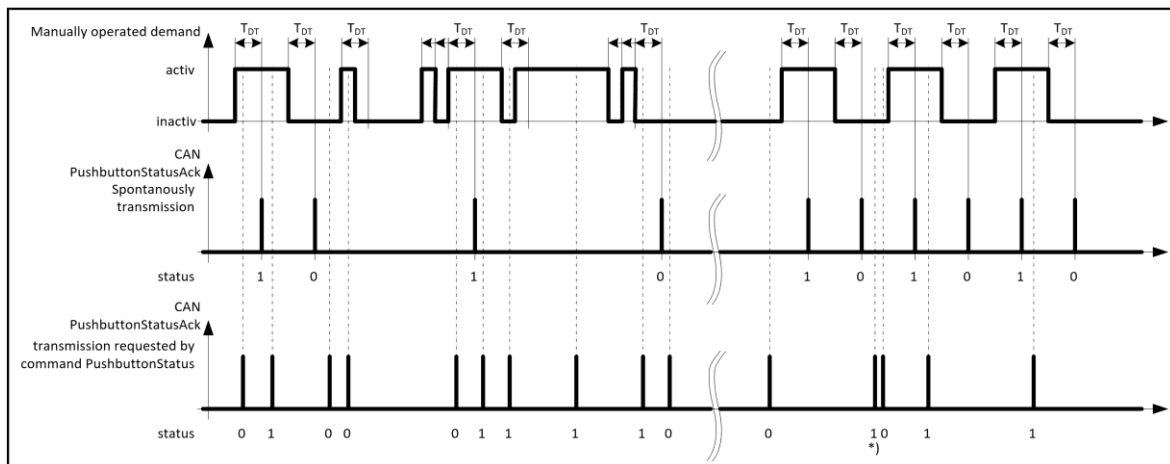
Push Button command <i>AliveAck</i>		
Telegram	Data	Remark
AliveAck	6 bit common status 1 bit sum failure 1 bit sum warning → NO Status	<u>6 bit common status</u> Data[0]<0:5> ... see chapter “Alive telegram for critical components”  <u>1 bit sum failure</u> Data[0]<6> ... see AliveAck Table 13  <u>1 bit sum warning</u> Data[0]<7> ... see AliveAck Table 13  NOTE 1 NO status bits are provided as a pure push button has no safety-related features. Telegram payload is only 1 byte!

## 7.5 Technical annotations on Pedestrian push Buttons

### 7.5.1 Technical annotation F, manually operated demands and logic inputs

While the goal of the detection time is to filter out multiple and / or short term demands to reduce the CAN-bus load, independent of manually operated demands and logic inputs, their functionality is different:

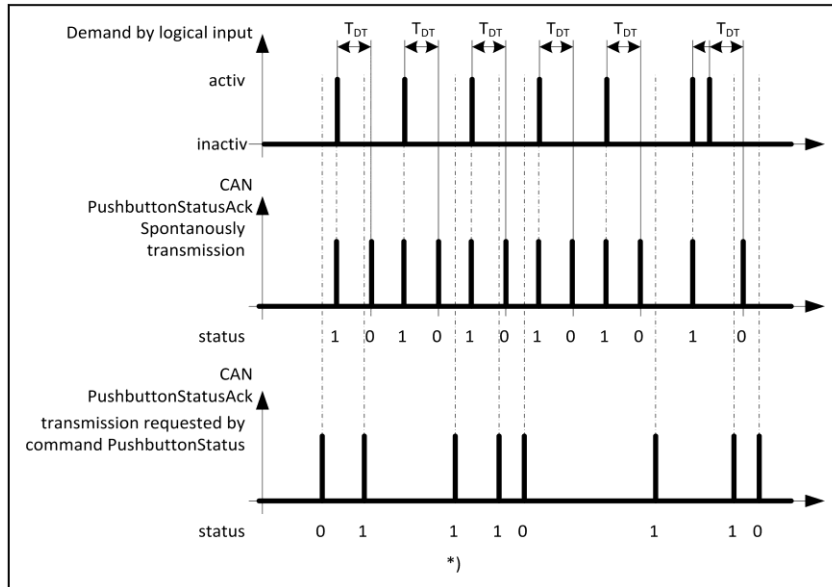
- manually operated demands (i.e. covered button or touch sensor inputs) activations (“pressing”) and de-activations (“releasing”) are notified by the demand class within the command PushbuttonStatusAck.  
 the detection time is interpreted to filter out short term demands (spikes) and leads to a delay in forwarding the demand (the delay is minimized by disabling the function with  $T_{DT} = 0$  ms)



\*) ...previous pulse is transmitted with the first following request

Figure 24 – Manually operated demands

- logical inputs (i.e. RFID)  
 For inputs like RFID that becomes active just for a very short moment of time (i.e. for the duration of any telegram transmission, when the tag comes closely to the transmitter), the information of deactivation is created by the ILT-component after  $T_{DT}$  has elapsed.



\*) previous pulse is transmitted with the first following request

**Figure 25 – Logical inputs**

## 8 Acoustic

### 8.1 Command set Acoustic

This category includes: All types of acoustic signals and vibrators.

If a component supports the command set given in Table 40, this is reported by *GetProfile*

Table 40 – Acoustic standard commands

Standard commands for controlling Acoustics				
Telegram	Type	Safety relevant	Data	Remark
SetOutputStateAcoustic	Req	Yes	8 bit state sequence counter 16 bit Sound State Mask	<p>Turns sound on using the defined volume.</p> <p><u>8 bit state Sequence counter</u> Data[0]&lt;0:3&gt; ... state Sequence counter Data[0]&lt;4:7&gt; ... 0 - reserved</p> <p><u>16 bit Sound State Mask - bitfield</u> Data[1:2]&lt;0:7&gt;: xxxx xxxx xxxx XABC b = complete bitfield</p> <p>“ABC” = basic, standardised command set  “X” = Reserved for “Walk-Sound-End” if provided.  “x” = Product / manufacturer specific bit</p> <p>“X” = 1: walk sound end signalling on  “X” = 0: walk sound end signalling off  “A” = 1: tactile signalling on  “A” = 0: tactile signalling off  “B” = 1: walk sound on  “B” = 0: walk sound off  “C” = 1: orientation sound on  “C” = 0: orientation sound off</p> <p>NOTE 1 Information about used bits and valid combinations in the Sound State Mask and correlation to the performed sound see section Acoustic signal, chapter “ILT communication Interface (CAN-Bus) / Sound State”.</p> <p>NOTE 2 The manufacturer/product-specific bits may either be interpreted as individual bits or a binary / numeric value of up to 13 bits (“state”). This shall be described within the data sheet and used accordingly.</p> <p>NOTE 3 The content of the state sequence counter is not checked by the ILT-component. This counter shall be reflected back unchanged to the controller / ILT interface box in the SetOutputStateAcousticAck, to enable the controller side ordering of telegrams.</p> <p>NOTE 4 The time TSOUND between the complete reception of command SetOutputState and reporting the sound state in command AliveAck and SetOutputStateAck shall not be greater than 20 ms  Within that time the acoustic signal shall check the correctness of the redundant commands. It is not expected that the selected sound is already audible.</p>

Standard commands for controlling Acoustics

Telegram	Type	Safety relevant	Data	Remark
SetOutputStateAcousticAck	Resp	Yes	8 bit Status + 8 bit state sequence counter 16 bit Sound State Mask <b>(only in case of failure)</b>	<p><u>8 bit Status</u> Data[0]&lt;5:7&gt;: See Table 12 – Common Status in safety-relevant telegrams</p> <p>Telegram-specific extension: Data[0]&lt;0&gt;: 0 = OK 1 = Error setting sound state, see added 16 bit sound state mask error status <b>(only transmitted when this bit is set!)</b> In case of failure the actual sound state mask is transmitted, according to the AliveAck.</p> <p>Data[0]&lt;1&gt;: 0 = OK 1 = invalid sound state was selected, no state change</p> <p>Data[0]&lt;2:4&gt;: 0 reserved</p> <p><u>8 bit state Sequence counter</u> Data[1]&lt;0:3&gt; ... state Sequence counter as received in the corresponding SetOutputState telegram Data[1]&lt;4:7&gt; ... 0 - reserved</p> <p>NOTE Command SetOutputStateAck is sent immediately as soon as the state starts changing.</p>
SetAcousticLevel	Req	Yes	8 bit Acoustic-level 8 bit Range	<p>Setting the volume level for acoustics, <u>8 bit Acoustic-level</u> Data[0]&lt;0:7&gt; = 0 (<i>Acoustic-level 0</i>) to 15 (<i>Acoustic-level 15</i>), any value higher than 15 is invalid.</p> <p><u>8 bit Range</u> Data[1]&lt;0:7&gt;: 0 = Range 0 (Day time) 1 = Range 1 (Night-time) 2 to 255 = not used</p> <p>NOTE 1 In case, the telegram <i>SetAcousticLevel</i> is never sent by the IF-Box, the ILT-component uses <i>Acoustic-level 0</i> as actual Acoustic-level. The same applies when the telegram <i>SetAcousticLevel</i> is sent only with invalid or uninitialized Acoustic-levels.</p> <p>NOTE 2 The Acoustic-level shall be valid and initialized in order the Acoustic-level is taken over by the <i>ILT-component</i>.</p> <p>NOTE 3 The Range shall match to the Range which was set by <i>DefAcousticLevel</i>. The Range is explicitly used here as redundant (safety) information. Answer telegram consists of according error-message if it does not match.</p> <p>NOTE 4 Default value for <i>Acoustic-level 0</i> see <i>SetOperationParameter</i> for operation <i>DefAcousticLevel</i>.</p>

Standard commands for controlling Acoustics				
Telegram	Type	Safety relevant	Data	Remark
SetAcousticLevelAck	Resp	Yes	8 bit Status	<p><u>8 bit Status</u></p> <p>Data[0]&lt;0&gt;: 0 = Acoustic-level valid 1 = Acoustic-level invalid</p> <p>Data[0]&lt;1&gt;: 0 = Acoustic-level initialized 1 = Acoustic-level not initialized by <i>SetOperationParameter</i></p> <p>Data[0]&lt;2&gt;: 0 = Range matches to the Range which was set by <i>DefAcousticLevel</i> 1 = Range does not match to the Range which was set by <i>DefAcousticLevel</i></p> <p>Data[0]&lt;3:4&gt;: 0 = unused bits</p> <p>Data[0]&lt;5:7&gt;: See Table 12 – Common Status in safety-relevant telegrams</p>
ForcedStateAcoustic	Req	Yes	16 bit Sound State Mask	<p>Service function for simulation of acoustic failure. With this command the Sound State Mask within the AliveAck is overwritten with Sound State Mask cited in this command while the state of the device is NOT changed. After a timeout of 100 ms the Sound State Mask within the command AliveAck will be set correct again.</p> <p><u>16 bit Sound State-Mask</u></p> <p>description of Sound State-Mask according command "SetOutputStateAcoustic"</p> <p>NOTE 1 Retriggerring of the command is not allowed. The smallest interval between two subsequent ForcedState commands at least 1 s. Within this interval, any other ForcedState command is rejected, and a negative status is reported.</p>
ForcedStateAcousticAck	Resp	Yes	8 bit Status	<p><u>8 bit Status</u></p> <p>Status&lt;0&gt;: 0 =OK 1 =interval too small; command ignored</p> <p>Status&lt;1:4&gt;: 0 = reserved</p> <p>Status&lt;5:7&gt;: See Table 12 – Common Status in safety-relevant telegrams.</p>

Standard commands for controlling Acoustics				
Telegram	Type	Safety relevant	Data	Remark
ForcedSelftest	Req	Yes	void	<p>Service function to initiate an internal self-test (exactly like it is done automatically in every power-up sequence)</p> <p>NOTE 1 the actual state of the acoustic shall be in such a way so that no acoustic output is active, otherwise the command is rejected.</p> <p>NOTE 2 The duration of the internal self-test shall not exceed TSELF = 300 ms. During this time all commands that would lead to a change in the actual output state are rejected.</p> <p>NOTE 3 A subsequent start of the self-test is only possible when a dead-time of 2 s has expired.</p> <p>NOTE 4 The self-test shall be initiated immediately with the received command.</p>
ForcedSelftestAck	Resp	Yes	8 bit Status	<p><u>8 bit Status</u></p> <p>Status&lt;0:4&gt;: 0x00 = Self-test and dead-time started; answer is sent immediately.</p> <p>Status&lt;0:4&gt;: 0x01 = Any acoustic output is active, or the command was received within the dead-time. In both cases, the command ignored. Answer is sent immediately.</p> <p>Status&lt;0:4&gt;: 0x02 = Error detected during self-test. Answer is sent when the self-test has completed but not when the detected error leads to EnterKnownState.</p> <p>Status&lt;0:4&gt;: 0x04 = No error detected during self-test. Result is sent spontaneously when the self-test has completed.</p> <p>Status&lt;0:4&gt;: 0x06 = Selftest aborted (i.e. due to voltage DIP). Result is sent spontaneously when selftest has been aborted</p> <p>Status&lt;0:4&gt;: 0x08 = Self-test in progress. Answer is sent immediately.</p> <p>Status&lt;0:4&gt;: 0x3, 0x5, 0x7, 0x09 to 0x1F = reserved</p> <p>Status&lt;5:7&gt;: common status in safety-relevant telegrams</p> <p>NOTE 1 If the self-test detects an error, the corresponding warning or failure bit within the AliveAck will be set and one of the commands, EnterKnownState or EnterFailureState, is initiated</p> <p>NOTE 2 During the self-test, no acoustic output will get active. The corresponding CAN-commands will deny the activation and autonomous functions will not trigger the activation.</p>



## 8.2 Command SetOperationParameter

The commands in Table 41 shall be supported by all ILT-components using the Acoustic command set. If the specified functionality is not implemented the component shall still send a suitable answer to the master (Traffic-controller).

Example, if a not implemented parameter is accessed by command *SetOperationParameter*, a NACK shall be returned.

**Table 41 – Command SetOperationParameter**

command <i>SetOperationParameter</i>				
Telegram	Type	Safety relevant	Data	Remark
SetOperationParameter Acoustic	Req	Yes <sup>a</sup>	8 bit OpParameter Up to 56 bit OpData	Set operation parameter defined for individual ILT components.  <u>8 bit OpParameter</u> Data[0]<0:7> ... Details see Table 42 – Acoustic operation parameter specification. OpParameter =           0 - 127 generic Operational Parameter 128 - 255 manufacturer specific Operational Parameter  <u>Up to 56 bit OpData</u> Data[1:7] ... How many bytes respectively bits are used depends on the specified operation parameter.
SetOperationParameter AcousticAck	Resp	Yes <sup>a</sup>	8 bit OpParameter 8 bit Status	8 bit OpParameter Data[0]<0:7> = Copy of received OpParameter  <u>8 bit Status</u> Data[1]<0:4> ...Details see Table 42 – Acoustic operation parameter specification.  Data[1]<5:7> = See Table 12 – Common Status in safety-relevant telegrams
<sup>a</sup> Although there is no actual need, the telegram shall be defined as “safety critical” to keep it compatible to the command SetOperationParameter of aspect.				

**Table 42 – Acoustic operation parameter specification**

Acoustic operation parameter specification (command <i>SetOperationParameter</i> ) May be set by the IF-Box using command <i>SetOperationParameter</i>			
OpParameter	Title	Data	Remark
00	---	---	Not used
01	DefAcousticLevel WalkSound	8 bit operation parameter (OpParameter) 8 bit Acoustic level 8 bit lower volume limit [dB(A)] 8 bit upper volume limit [dB(A)] 8 bit raise of volume [dB] 8 bit Range 8 bit Slope rising [dB/s] 8 bit Slope falling [dB/s]	Defining the walk sound volume for the specified loudness-level.  <u>8 bit OpParameter</u> Data[0]<0:7> = 0x01  <u>8 bit Acoustic-level</u> Data[1]<0:7> = <i>Acoustic-level 0 to Acoustic-level 15</i> , any value higher than 15 is invalid.  <u>8 bit lower volume limit [dB(A)]</u> Data[2]<0:7> = lower volume limit 0 db(A) to 255 dB(A)

			<p><u>8 bit upper volume limit [dB(A)]</u> Data[3]&lt;0:7&gt; = upper volume limit 0 dB(A) to 255 dB(A)</p> <p><u>8 bit raise of volume [dB]</u> Data[4]&lt;0:7&gt; = -128 dB(A) to +127 dB(A) represented in 2's complement</p> <p><u>8 bit Range</u> Data[5]&lt;0:7&gt;: 0 = Range 0 (day-mode) 1 = Range 1 (night-mode) 2-255 = invalid.</p> <p><u>8 bit Slope rising [dB/s]</u> Data[6]&lt;0:7&gt; = 0 dB/s to 255 dB/s</p> <p><u>8 bit Slope falling [dB/s]</u> Data[7]&lt;0:7&gt; = 0 dB/s to 255 dB/s</p> <p>NOTE 1 The default values for <i>Acoustic-level 0</i> are: lower volume limit = set via parametrization of the device during manufacturing upper volume limit = set via parametrization of the device during manufacturing raise of volume = set via parametrization of the device during manufacturing Range = 0 Slope rising = set via parametrization of the device during manufacturing Slope falling = set via parametrization of the device during manufacturing</p> <p>NOTE 2 The default values for <i>Acoustic-level 1</i> to <i>Acoustic-level 15</i> are (these values mean the according <i>Acoustic-level</i> is disabled): lower volume limit = 0 dB(A) upper volume limit = 0 dB(A) raise of volume = 0 dB(A) Range = 2 (or any invalid number for Range)</p> <p>NOTE 3 If an invalid <i>Acoustic-level</i> and/or an invalid Range is received, then the values are not taken over by the <i>ILT-component</i> and the Status in the Ack-telegram is: Status&lt;0&gt; = 1 (in case of invalid <i>Acoustic-level</i>) Status&lt;1&gt; = 0 Status&lt;2&gt; = 1 (in case of invalid Range) Status&lt;3&gt; = 0 Status&lt;4&gt; = 1 (common error indication) Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</p> <p>NOTE 4 If an upper or lower volume limit of 0 [dB(A)] is received for <i>Acoustic-level 0</i> or for the active <i>Acoustic-level</i>, then the values are not taken over by the <i>ILT-component</i> and the Status in the Ack-telegram is: Status&lt;0&gt; = 1 (error acoustic level) Status&lt;1&gt; = 1 (error upper or lower volume limit) Status&lt;2&gt; = 0 Status&lt;3&gt; = 0 Status&lt;4&gt; = 1 (common error indication)</p>
--	--	--	--

Acoustic operation parameter specification (command *SetOperationParameter*)

May be set by the IF-Box using command *SetOperationParameter*

OpParameter	Title	Data	Remark
			<p>Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</p> <p>NOTE 5 If an upper or lower volume limit of 0 [dB(A)] is received for <i>Acoustic-level 1</i> to <i>Acoustic-level 15</i> and the according Acoustic-level is <u>not</u> the <u>active</u> acoustic-level, then the according acoustic-level is disabled, and the Status in the Ack-telegram is:                      Status&lt;0:4&gt; = all bits are 0                      Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</p> <p>NOTE 6 In case, the lower volume limit [dB(A)] is lower than the minimum lower volume limit specified by the according Range, then the lower volume limit is changed to the minimum lower volume limit of that Range and the Status in the Ack-telegram is:                      Status&lt;0&gt; = 0                      Status&lt;1&gt; = 1 (warning volume limit)                      Status&lt;2&gt; = 0                      Status&lt;3&gt; = 0                      Status&lt;4&gt; = 0                      Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</p> <p>NOTE 7 In case, the upper volume limit [dB(A)] is higher than the maximum upper volume limit specified by the according Range, then the upper volume limit is changed to the maximum upper volume limit of that Range and the Status in the Ack-telegram is:                      Status&lt;0&gt; = 0                      Status&lt;1&gt; = 1 (warning volume limit)                      Status&lt;2&gt; = 0                      Status&lt;3&gt; = 0                      Status&lt;4&gt; = 0                      Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</p> <p>NOTE 8 the value for raise of volume is not checked</p> <p>NOTE 9 In case, the upper volume limit [dB(A)] is lower than the lower volume limit [dB(A)], the upper volume limit is set equal with the lower volume limit. The Status in the Ack-telegram is:                      Status&lt;0&gt; = 0                      Status&lt;1&gt; = 1 (warning volume limit)                      Status&lt;2&gt; = 0                      Status&lt;3&gt; = 0                      Status&lt;4&gt; = 0                      Status&lt;5:7&gt; = See Table 12 – Common Status in safety-relevant telegrams</p> <p>NOTE 10 Sending this telegram by the traffic-controller is optional.</p>
02	DefAcousticLevel OrientationSound	8 bit operation parameter (OpParameter) 8 bit Acoustic level 8 bit lower volume limit [dB(A)] 8 bit upper volume limit [dB(A)] 8 bit raise of volume [dB] 8 bit Range 8 bit Slope rising [dB/s] 8 bit Slope falling [dB/s]	Defining the orientation sound volume for the specified loudness-level.  <u>8 bit OpParameter</u> Data[0]<0:7> = 0x01  <u>8 bit Acoustic-level</u> Data[1]<0:7> = <i>Acoustic-level 0</i> to <i>Acoustic-level 15</i> , any value higher than 15 is invalid.  <u>8 bit lower volume limit [dB(A)]</u> Data[2]<0:7> = lower volume limit 0 db(A) to 255 dB(A)

Acoustic operation parameter specification (command *SetOperationParameter*)

May be set by the IF-Box using command *SetOperationParameter*

OpParameter	Title	Data	Remark
			<p><u>8 bit upper volume limit [dB(A)]</u> Data[3]&lt;0:7&gt; = upper volume limit 0 dB(A) to 255 dB(A)</p> <p><u>8 bit raise of volume [dB]</u> Data[4]&lt;0:7&gt; = -128 dB(A) to +127 dB(A) represented in 2'scomplement</p> <p><u>8 bit Range</u> Data[5]&lt;0:7&gt;: 0 = <i>Range 0</i> (day-mode) 1 = <i>Range 1</i> (night-mode) 2-255 = invalid.</p> <p><u>8 bit Slope rising [dB/s]</u> Data[6]&lt;0:7&gt; = 0 dB/s to 255 dB/s</p> <p><u>8 bit Slope falling [dB/s]</u> Data[7]&lt;0:7&gt; = 0 dB/s to 255 dB/s</p> <p>NOTE See notes as defined for "DefAcousticLevelWalkSound" above.</p>
03	DefAcousticLevel Supplement Sounds	<p>8 bit operation parameter (OpParameter)</p> <p>8 bit Acoustic level</p> <p>8 bit lower volume limit [dB(A)]</p> <p>8 bit upper volume limit [dB(A)]</p> <p>8 bit raise of volume [dB]</p> <p>8 bit Range</p> <p>8 bit Slope rising [dB/s]</p> <p>8 bit Slope falling [dB/s]</p>	<p>Defining the sound volume for all sounds NOT being a walk or orientation sound (i.e. speech, information, ...) for the specified loudness-level.</p> <p><u>8 bit OpParameter</u> Data[0]&lt;0:7&gt; = 0x01</p> <p><u>8 bit Acoustic-level</u> Data[1]&lt;0:7&gt; = <i>Acoustic-level 0</i> to <i>Acoustic-level 15</i>, any value higher than 15 is invalid.</p> <p><u>8 bit lower volume limit [dB(A)]</u> Data[2]&lt;0:7&gt; = lower volume limit 0 dB(A) to 255 dB(A)</p> <p><u>8 bit upper volume limit [dB(A)]</u> Data[3]&lt;0:7&gt; = upper volume limit 0 dB(A) to 255 dB(A)</p> <p><u>8 bit raise of volume [dB]</u> Data[4]&lt;0:7&gt; = -128 dB(A) to +127 dB(A) represented in 2'scomplement</p> <p><u>8 bit Range</u> Data[5]&lt;0:7&gt;: 0 = <i>Range 0</i> (day-mode) 1 = <i>Range 1</i> (night-mode) 2-255 = invalid.</p> <p><u>8 bit Slope rising [dB/s]</u> Data[6]&lt;0:7&gt; = 0 dB/s to 255 dB/s</p> <p><u>8 bit Slope falling [dB/s]</u> Data[7]&lt;0:7&gt; = 0 dB/s to 255 dB/s</p> <p>NOTE See notes as defined for "DefAcousticLevelWalkSound" above.</p>
4	Enabling of the service interfaces	8 bit activation time	<p>Defining the time for service.</p> <p><u>8 bit activation time</u> Data[0]&lt;0:7&gt; = 0x00 minute to 255 minutes</p>

Acoustic operation parameter specification (command *SetOperationParameter*)

May be set by the IF-Box using command *SetOperationParameter*

OpParameter	Title	Data	Remark
			<p>NOTE 1 The telegram enables a third-party interface for a specified time, to avoid permanently open interfaces.</p> <p>NOTE 2 A repetition of the telegram shall set the remaining time to the new value. A value of 0x00 disables an active channel immediately.</p> <p>NOTE 3 Components without third-party interface ignore this telegram.</p> <p>NOTE 4 A second factor shall be activated by manual action. The maintenance user interface of the traffic controller shall give grant to the command. The maintenance time window should be after entering the 2nd factor. It shall not be possible to change the safety-relevant traffic data of the control unit except by an authorised operation. In addition, all changes should be logged.</p>
05-127	Reserved for future use of generic Acoustic operation parameter		<p>Definition of Status in <i>SetOperationParameterAcousticAck</i></p> <p><u>8 bit Status = Data[1]&lt;0:4&gt;</u></p> <p>Status&lt;0&gt; = 0                      Status&lt;1&gt; = 0                      Status&lt;2&gt; = 0                      Status&lt;3&gt; = 1 ... Wrong OpParameter                      Status&lt;4&gt; = 1 ... Common error indication</p>
128 - 255	Information about supported Operational Parameters see section Acoustic, chapter "ILT communication Interface (CAN-Bus) / Operational Parameter"		<p>Definition of Status in <i>SetOperationParameterAcousticAck</i></p> <p><u>8 bit Status = Data[1]&lt;0:4&gt;</u></p> <p>Data[1]&lt;0&gt; = 0                      Data[1]&lt;1&gt; = 0                      Data[1]&lt;2&gt; = 0                      Data[1]&lt;3&gt; = 1 ... Wrong OpParameter                      Data[1]&lt;4&gt; = 1 ... Common error indication</p> <ul style="list-style-type: none"> <li>■ OpParameter, which are not listed in the appropriate product data sheet of push button, always return these status-bits.</li> <li>■ Details for OpParameter, which are listed in the appropriate product datasheet of push button, see into this product datasheet.</li> </ul>

### 8.3 Command *GetOperationData*

Table 43 – Command *GetOperationData*

Command <i>GetOperationData</i>				
Telegram	Type	Safety relevant	Data	Remark
<i>GetOperationDataAcoustic</i>	Req	No	8 bit OpData specification	<p>Get operation data defined for the individual ILT components.</p> <p><u>8 bit OpData specification</u>                      Data[0] = Details, see Table 44 – Acoustic operation data specification.</p> <p>OpData = 0 - 127 generic Operational Data                      128 - 255 manufacturer specific Operational Data</p>

Command GetOperationData				
Telegram	Type	Safety relevant	Data	Remark
GetOperationDataAcousticAck	Resp	No	8 bit OpData specification 8 bit Status Up to 48 bit Data	<p><u>8 bit OpData specification</u> Data[0] = Copy of received OpData</p> <p><u>8 bit Status</u> Data[1] ...</p> <ul style="list-style-type: none"> <li>00h = Ok</li> <li>80h = Invalid OpData specification</li> </ul> <p><u>Up to 48 bit Data</u> Data[2:7] = Operation Data defined for individual ILT components. Details see Table 44 – Acoustic operation data specification.</p> <p>NOTE In case an unknown OpData specification was received by command GetOperationData, then the payload consists of only Data[0:1].</p>

**Table 44 – Acoustic operation data specification**

Acoustic-signals operation data information (command GetOperationDataAck)			
May be retrieved by the system master using the command GetOperationData			
OpData	Title	Data	Remark
00	Working Hour Meter	8 bit OpData specification 8 bit Status 24 bit Working hours	<p><u>8 bit OpData specification</u> Data[0] = 0x01</p> <p><u>8 bit Status</u> Data[1] = see GetOperationDataAck</p> <p><u>24 bit Working hours</u> Data[2] = Working hours&lt;0:7&gt; Data[3] = Working hours&lt;8:15&gt; Data[4] = Working hours&lt;16:23&gt;</p>
01	not used	not used	not used
02	Version of manufacturer specific information set	8 bit Version	<p><u>8 bit Version</u> Data[0] = 0 - 255</p> <p>Version of the manufacturer specific set of information, containing Operational Parameter, Operational Data and additional information of Warning classes according appropriate product datasheet of the aspect, chapter "ILT communication Interface (CAN-Bus) / Version of information set</p>
03	not used	not used	not used
04	ItemNumber_0	8 bit OpData specification 8 bit Status 48 bit Item number 0 (Low)	<p><u>8 bit OpData specification</u> Data[0] = 0x04 (Low) or 0x05 or 0x06 or 0x07 or 0x08 (High)</p>
05	ItemNumber_1	8 bit OpData specification 8 bit Status 48 bit Item number 1	<p><u>8 bit Status</u> Data[1] = see <i>GetOperationDataAck</i></p> <p><u>48 bit Item Number (n):</u> Data[2] = Item number (n) &lt;0:7&gt;</p>

### Acoustic-signals operation data information (command *GetOperationDataAck*)

May be retrieved by the system master using the command *GetOperationData*

OpData	Title	Data	Remark
06	ItemNumber_2	8 bit OpData specification 8 bit Status 48 bit Item number 2	Data[3] = Item number (n) <8:15> Data[4] = Item number (n) <16:23> Data[5] = Item number (n) <24:31> Data[6] = Item number (n) <32:39> Data[7] = Item number (n) <40:47>
07	ItemNumber_3	8 bit OpData specification 8 bit Status 48 bit Item number 3	NOTE 1 (n) = 0 (Low), 1, 2, 3, 4 (High). NOTE 2 Up to 30 digit product item number is available by this OpData.
08	ItemNumber_4	8 bit OpData specification 8 bit Status 48 bit Item number 4 (High)	NOTE 3 ItemNumber (n) can be seen as an extension of the Unique ID. NOTE 4 ItemNumber is ASCII-String, filled with 0x00
09	Ambient noise level	8 bit OpData specification 8 bit Status 8 bit Ambient noise level	<u>8 bit OpData specification</u> Data[0] = 0x09  <u>8 bit Status</u> Data[1] = see <i>GetOperationDataAck</i>  <u>8 bit Ambient noise level:</u> Data[2]<0:7> = 0 dB(A) - 255 dB(A)
10-127	Reserved for future use of generic Acoustic operation data		
128 - 255	Reserved for manufacturer specific operational data For information on supported Operational Data, see the appropriate product datasheet of the Acoustic-Signal, in chapter "ILT communication Interface (CAN-Bus) / Operational Data".		

## 8.4 Command *AliveAck*

Table 45 – Acoustic command *AliveAck*

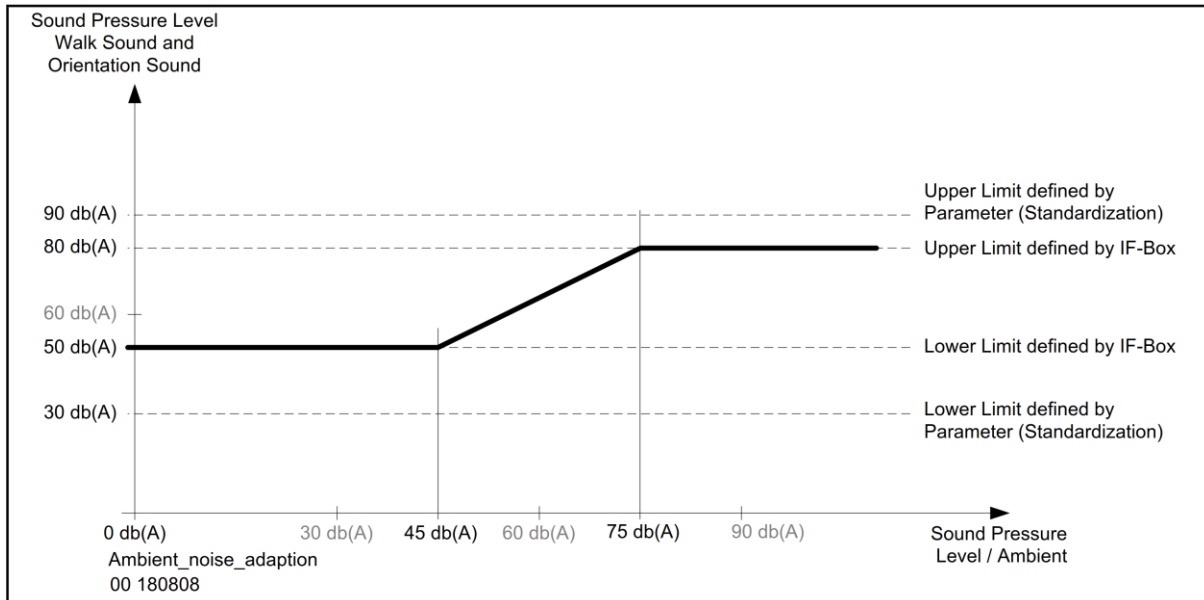
Acoustic command <i>AliveAck</i>		Data	Remark
AliveAck	6 bit common status 1 bit sum failure 1 bit sum warning 16 bit Status	6 bit common status Data[0]<0:5> ... see chapter "Alive telegram for critical components"  <u>1 bit sum failure</u> Data[0]<6> ... see AliveAck Table 13  <u>1 bit sum warning</u> Data[0]<7> ... see AliveAck Table 13  <u>16 bit Status</u> - Sound State Mask status  Data[1]<0:7> Data[2]<0:7>: 0 = sound state deactivated 1 = sound state activated  there is an exact 1:1 relation to the actual activated sound state, correlating sound state mask of "SetOutputState" commands  NOTE The sum failure bit has to be set consistently and at the same time with a deviating sound state mask status if the failure is the cause for the deviation.	

## 8.5 Boundary condition regarding to Acoustic

### 8.5.1 Boundary condition G, volume limits

It shall not be possible to set a volume below or above the volume limit defined in (country specific) regulation.

In the Acoustic's product datasheet, there is a lower and upper volume limit specified (according to the standards mentioned above or customer requirements). This volume limit can be restricted by the IF-Box (according *DefAcousticLevel*) but cannot be extended as shown in the following example:



example with Limits set via CAN: 50 dB(A) - 80dB(A)  
raise of volume ROV: 5 dB(A)

Figure 26 – Volume limits

## 9 Traffic Sensor

### Communication Interface

In order to support longer cable lengths in special applications, the detectors shall be able to communicate at lower baud rates (250 kBit/s, 125 kBit/s, 100 kBit/s, 50 kBit/s and 25 kBit/s). These shall be adjustable via the setup or a factory configuration. However, this is not part of this specification and serves as an additional note. The default setting shall be 500 kBit/s.

If components are able to operate on a different baud rate it has to be secured, that normal ILT operation is not disrupted, a change of the baud rate shall only be possible at boot time.

### 9.1 Command set Traffic sensors

This category includes: All types of "decisive" vehicle detectors (i.e. loops, cameras, thermal, and radar).

If a component supports the command set given in Table 46, this is reported by *GetProfile*



Table 46 – Detector command and status telegrams

Standard commands for controlling Detectors				
Telegram	Type	Safety relevant	Data	Remark
DetectionStatus	Inst	No	8 bit: current detector occupation / activation status for 8 zones, 1 = occupied / triggered, 0 = free 8 bit: detector state change since last DetectionStatus telegram 8 bit: direction of detection event (0 = default = intended direction, 1 = reverse direction) 8 bit: error status for detection zones (1 = incorrect operation) 4 bytes: 32-bit integer timestamp in 1 or 10 ms increments <b>(to be defined)</b>	Sends the actual state of the individual zones as soon as a single zone changes its state. Occupation or trigger has to be decided by configuration or according to type of detector. The “state change since last DetectionStatus” information is intended for the configuration options where only one type of instantaneous telegrams is requested, otherwise no change detection would be possible. Additionally, it allows unambiguous edge detection. The direction bit defaults to 0 as intended and “no” direction which allows mixing detectors with compatibility. Direction is only supported in instantaneous report mode as it makes no sense with polling.
GetDetectionStatus	Req	No	8 bytes: reserved <b>(or reduced DLC?)</b>	
GetDetectionStatusAck	Resp	No	8 bit: current detector occupation status 8 bit: detector rising edge seen since last request 8 bit: detector falling edge seen since last request 4 bytes: 32-bit integer timestamp in 1 or 10 ms increments <b>(to be defined)</b>	Detailed response since start-up or previous request. Internal edge markers have to be reset after this response.
RisingEdgeStatus	Inst	No	1 byte: 4 bit: zone number 1-8 4 bit: reserved 3 byte: reserved 4 byte: non-occupation time before this event, 1 or 10 ms increments <b>(to be defined)</b>	Detailed edge reporting for rising edge (beginning of zone occupation) – shall be enabled explicitly (see DetectorReportMode). Non-occupation time value is the relative time since last zone freeing (or start-up if zone was free then).
RisingEdgeStatus	Inst	No	1 byte: 4 bit: zone number 1-8 4 bit: reserved 3 byte: reserved 4 byte: non-occupation time before this event, 1 or 10 ms increments <b>(to be defined)</b>	Detailed edge reporting for rising edge (beginning of zone occupation) – shall be enabled explicitly (see DetectorReportMode). Non-occupation time value is the relative time since last zone freeing (or start-up if zone was free then).

Standard commands for controlling Detectors				
Telegram	Type	Safety relevant	Data	Remark
FallingEdgeStatusStandard	Inst	No	<p>1 byte:</p> <p>4 bit: zone number 1-8</p> <p>2 bit: direction: 0b00 = disabled, 0b01 = intended dir., 0b10 = reverse dir., 0b11 = undetectable</p> <p>2 bit: reserved</p> <p>1 byte: speed, 1 LSB = 1 km/h</p> <p>0x01 to 0xFE = measured speed, 0xFF = not available / disabled</p> <p>1 byte:</p> <p>4 bit: classification, i.e.: 0x0 = simple mode small / no bus / no bicycle, 0x1 = simple mode large / bus / bicycle, 0xF = disabled</p> <p>4 bit: reserved</p> <p>1 byte: vehicle length, 1 LSB = 1 dm, 0x01 to 0xFE = measured length, 0x00 = not available / disabled, 0xFF = not detected</p> <p>4 byte: occupation time before this event, 1 or 10 ms increments <b>(to be defined)</b></p>	<p>Detailed edge reporting for falling edge (end of zone occupation) – shall be enabled explicitly (see DetectorReportMode).</p> <p>At this stage, additional information is provided if the detector is installed and configured properly and provides the information: direction, speed, type classification, and vehicle length.</p> <p>If any information cannot be provided by hardware, configuration, or installation, the value "undetected / disabled" have to be sent.</p> <p>Single loop configuration: Values for direction, speed and length are generally send with value "undetected / disabled".</p> <p>Double loop configuration: The valid data for direction, speed, classification, and length are output on the second channel in the direction of travel.</p> <p>Direction of travel first channel: Direction, speed and length are sent with value "undetected / disabled". For speed and length measurement optimized values for measuring time and sensitivity are recommended.</p> <p>Occupation time value is the relative time since beginning of zone occupation (or start-up if zone was occupied then).</p>
FallingEdgeStatusClassification	Inst	No	<p>1 byte:</p> <p>4 bit: zone number 1-8</p> <p>2 bit: direction: 0b00 = disabled, 0b01 = intended dir., 0b10 = reverse dir., 0b11 = undetectable</p> <p>2 byte: speed, 1 LSB = 1 km/h, 1 to 300 = measured speed, 0xFFFF = not available / disabled</p> <p>1 byte:</p> <p>4 bit: accurate classification i.e. TLS, ASTRA, see chapter 9.5.1</p> <p>4 bit: reserved</p> <p>1 byte: vehicle length, 1 LSB = 1 dm, 0x01 to 0xFE = measured length, 0x00 = not available / disabled, 0xFF = not detected</p> <p>3 byte: occupation time before this event 1 or 10 ms increments <b>(to be defined)</b></p>	<p>Vehicle data telegram for detectors with high classification accuracy and related specification (i.e. acc. to TLS).</p> <p>Detailed edge reporting for falling edge (end of zone occupation) – shall be enabled explicitly (see DetectorReportMode).</p> <p>At this stage, additional information is provided if the detector is installed and configured properly and provides the information: direction, speed, type classification, and vehicle length.</p> <p>If any information cannot be provided by hardware, configuration, or installation, the value "undetected / disabled" have to be sent.</p> <p>Single loop configuration: Values for direction, speed and length are generally send with value "undetected / disabled".</p> <p>Double loop configuration: The valid data for direction, speed, classification, and length are output on the second channel in the direction of travel.</p> <p>Direction of travel first channel: Direction, speed and length are sent with value "undetected / disabled".</p> <p>Occupation time value is the relative time since beginning of zone occupation (or start-up if zone was occupied then).</p>

Standard commands for controlling Detectors				
Telegram	Type	Safety relevant	Data	Remark
OccupancyTimeInterval	Req	No	1 byte: 4 bit: zone number 1-8 7 bytes: reserved ( <b>or reduced DLC?</b> )	Used to determine accurate interval occupancy rates i.e. in traffic control applications when sensor is occupied by vehicle at the time of request, Time interval (i.e. 1 minute) of occupation request defined in Master.
OccupancyTimeIntervalAck	Resp	No	1 byte: 4 bit: zone number 1-8 3 bytes: Occupancy time of interval 3 bytes: Non-occupancy time of interval (time gap between vehicles) 1 or 10 ms increments ( <b>to be defined</b> ) 1 byte: reserved ( <b>or reduced DLC?</b> )	Used to determine accurate interval occupancy rates i.e. in traffic control applications when sensor is occupied by vehicle at the time of request. Internal counters for occupancy and gap time are reset at the time of response. The sum of occupancy and non-occupancy time have to be equal the time interval (+/- 3 %).
ResetDetectorAlignmentZone	Req	No	1 byte: 4 bit: 0: reset device, zone number 1-8 for zone alignment 7 bytes: reserved ( <b>or reduced DLC?</b> )	Reset device with interrupting communication. Zone alignment without reset of device and without interrupting communication.
ResetDetectorAlignmentZoneAck	Resp	No	1 byte: 4 bit: 0: device reset , zone number 1-8 alignment 7 bytes: reserved ( <b>or reduced DLC?</b> )	Ack for device reset or zone alignment finished

## 9.2 Command SetOperationParameter

Table 47 – Command SetOperationParameter Traffic Sensors

Command SetOperationParameter Traffic Sensors				
Telegram	Type	Safety relevant	Data	Remark
SetOperationParameterTrafficSensors	Req	No	8 bit OpParameter Up to 56 bit OpData	Set operation parameter defined for individual ILT components.  <u>8 bit OpParameter</u> Data[0]<0:7> ... Details see Table 48 – Detector operation parameters and “ILT communication Interface (CAN-Bus) / Operational Parameter” OpParameter = 0 - 127 generic Operational Parameter 128 - 255 manufacturer specific Operational Parameter <u>Up to 56 bit OpData</u> Data[1:7] ... How many bytes respectively bits are used depends on the specified operation parameter.
SetOperationParameterTrafficSensorAck	Resp	No	8 bit OpParameter 8 bit Status	<u>8 bit OpParameter</u> Data[0]<0:7> = Copy of received OpParameter  <u>8 bit Status</u> Data[1]<0:4> ... Details see “ILT communication Interface (CAN-Bus) / Operational Parameter” Data[1]<5:7> = don't care (not safety relevant)

**Table 48 – Detector operation parameters**

Detector's operation parameters defining the detector operation May be set by the system master using the command <i>SetOperationParameter</i> (48-63)			
OpParameter	Title	Data	Remark
48	DetectorReportMode	8 bit: global report mode setting field bit 0 – send instantaneous status when detector gets occupied bit 1 – send instantaneous status when detector is freed bit 2-7: reserved 16 bit: enable detailed edge reporting for individual zones (2 bit per zone) zone bit 0: report on rising edge zone bit 1: report on falling edge 4 bytes: reserved ( <b>or reduced DLC?</b> )	Configuration of the reporting methods on global and zone level.

### 9.3 Command GetOperationData

**Table 49 – Command GetOperationData**

Command <i>GetOperationData Traffic Sensor</i>				
Telegram	Type	Safety relevant	Data	Remark
Get Operation Data Traffic Sensor	Req	No	8 bit OpData specification	Get operation data defined for the individual ILT components.  <u>8 bit OpData specification</u> Data[0] = Details, see the appropriate product datasheet of the traffic sensor "ILT communication Interface (CAN-Bus) / Operational Data".  OpData = 0 - 127 generic Operational Data 128 - 255 manufacturer specific Operational Data
GetOperationDataTrafficSensor Ack	Resp	No	8 bit OpData specification 8 bit Status Up to 48 bit Data	<u>8 bit OpData specification</u> Data[0] = Copy of received OpData  <u>8 bit Status</u> Data[1] ... <ul style="list-style-type: none"> <li>■ 00h = Ok</li> <li>■ 80h = Invalid OpData specification</li> </ul> <u>Up to 48 bit Data</u> Data[2:7] = Operation Data defined for individual ILT components. Details see the appropriate product datasheet of the Push Button Signal in chapter "ILT communication Interface (CAN-Bus) / Operational Data"
NOTE In case an unknown OpData specification was received by command <i>GetOperationData</i> , then the payload consists of only Data[0:1].				

## 9.4 Command AliveAck

Table 50 – Traffic sensor command AliveAck

Traffic sensor command <i>AliveAck</i>		
Telegram	Data	Remark
AliveAck	6 bit common status 1 bit sum failure 1 bit sum warning → <b>NO</b> Status	<u>6 bit common status</u> Data[0]<0:5> ... see chapter "Alive telegram for critical components" <u>1 bit sum failure</u> Data[0]<6> ... see AliveAck Table 13 <u>1 bit sum warning</u> Data[0]<7> ... see AliveAck Table 13 NOTE NO status bits are provided as a pure traffic sensor has no safety-related features. Telegram payload is only 1 byte!

## 9.5 Technical annotations on Traffic Sensors

### 9.5.1 Technical annotation H, Vehicle classes TLS / BAST, ASTRA

Definition of vehicle classes used in telegram FallingEdgeStatusClassification (high classification accuracy)

Table 51 – Classification in 8+1 vehicle classes acc. to TLS / BAST

TLS class	Code	Allocated vehicles
Cars	0	Cars from compact cars up to minivans including off-road vehicles
CarT	1	Cars up to 3.5 t perm. overall weight with trailer (also utility vehicles)
HGV	2	> 3.5 t
Utveh	3	Utility vehicles ≤ 3.5 t max. permitted overall weight
HGVT	4	HGV > 3.5 t with trailer
Art. HGV.	5	All articulated HGVs
Bus	6	Vehicles with more than 9 seats for passenger transportation; also with trailer
Nc. veh.	7	All vehicles, for which the vehicle type cannot be specified, or which do not belong to any of the other classes.
Mbike	8	Motorbikes, also with sidecar; however, no bicycles, no mopeds

Table 52: Classification in 5+1 vehicle classes acc. to TLS / BAST

TLS class	Code	Allocated vehicles
Car group	0	Mbike, cars and Utveh
CarT	1	Cars up to 3.5 t perm. overall weight with trailer (also utility vehicles)
HGV	2	> 3.5 t
HGVC	4	HGVT and Art. HGVs (HGV combinations)
Bus	6	Bus
Nc. veh.	7	All vehicles, for which the vehicle type cannot be specified, or which do not belong to any of the other classes.

**Table 53 – Classification in 2 vehicle classes acc. to TLS / BAST**

TLS class	Code	Allocated vehicles
Car-similar	0	Mbike, cars, Utveh and Nc. veh.
HGV-similar	2	CarT, HGV, HGVT, Art. HGV. and Bus
NOTE When one loop of a loop system is damaged, a classification in car- and HGV-similar vehicles with less accuracy is achieved. In this case speed and length measurement are not possible.		

**Table 54 – Classification type SWISS10 acc. to ASTRA**

SWISS10 class	Code	Allocated vehicles
Cars	0	Cars from compact cars up to minivans including off-road vehicles
CarT	1	Cars with trailer
HGV	2	> 3.5 t
Utveh	3	Utility vehicles ≤ 3.5 t max. permitted overall weight
HGVT	4	HGV > 3.5 t with trailer
Art. HGV.	5	All articulated HGVs
Bus	6	Vehicles with more than 9 seats for passenger transportation; also with trailer
Mbike	8	Motorbikes, also with sidecar; however, no bicycles, no mopeds
UtvehT	9	Utility vehicles ≤ 3.5 t max. permitted overall weight with trailer
Utveh articulated	10	Articulated utility vehicles ≤ 3.5 t max. permitted overall weight
NOTE 1 For SWISS10 there is no class corresponding to the TLS class "Nc. veh. "!		
NOTE 2 When one loop of a loop system is damaged, a classification in car- and HGV-similar vehicles with less accuracy is achieved. In this case speed and length measurement are not possible		

**Table 55 – Comparison TLS – ASTRA SWISS10**

class. type	Vehicle class with code										
TLS 2	Car-similar (0)				HGV-similar (2)						
TLS (5+1)	nk Kfz (7)	Car group (0)			HGV (2)	HGV combination (4)		Bus (6)	CarT (1)		
TLS (8+1)	nk Kfz (7)	Mbike (8)	Cars (0)	Utveh (3)	HGV (2)	HGVT (4)	Art. HGV. (5)	Bus (6)	CarT (1)		
SWISS10	-	Mbike (8)	Cars (0)	Utveh (3)	HGV (2)	HGVT (4)	Art. HGV. (5)	Bus (6)	CarT (1)	UtvehT (9)	Utveh articulated (10)

## Bibliography

- [1] *ISO/IEC Directives - Part 2: Principles and rules for the structure and drafting of ISO and IEC documents*, edition 9 (2021); Clause 7, Verbal forms for expressions of provisions
- [2] “Register of Manufacturer Identifier for VDE SPEC 90013” hosted at the VDE platform:  
[www.dke.de/manufacturer-identifier](http://www.dke.de/manufacturer-identifier)
- [3] EN 50129, *Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signalling*
- [4] EN 60529, *Degrees of protection provided by enclosures (IP Code) (IEC 60529)*
- [5] EN 60950-1:2006, *Information technology equipment - Safety - Part 1: General requirements (IEC 60950-1:2005, mod.)*

VDE Verband der Elektrotechnik  
Elektronik Informationstechnik e.V.

DKE Deutsche Kommission  
Elektrotechnik Elektronik Informationstechnik  
in DIN und VDE  
Merianstraße 28  
63069 Offenbach am Main

Tel. +49 69 6308-0  
dke@vde.com  
www.dke.de

**VDE**