SIGMATEK

3.5“ Terminal

Project Implementation

Date of creation: 10.02.2014 | Version date: 28.04.2016 | Article number: XX-XXX-XXXX-E

Publisher: SIGMATEK GmbH & Co KG
A-5112 Lamprechtshausen
Tel.: 06274/4321
Fax: 06274/4321-18
Email: office@sigmatek.at
WWW.SIGMATEK-AUTOMATION.COM

Translation from German

3.5” Terminal Project Implementation

This document serves as a guide for implementing projects on a 3.5" terminal.

Contents

1 Basic Function.. 4

2 Settings in the CPU 5

3 Settings in the 3.5" Display 5

4 Creating a Class Project to Control a 3.5" Display............... 7

5 Creating a Screen Project to Show on the Display 9

6 Guidelines for using LSE Easy 12

7 Using Multiple Displays 14

7.1 Multiple Displays with the Same Visualization 14

7.2 Multiple Displays with Different Visualizations 15

7.3 Combined Operation 16

8 Loading Data for Multiple Displays with One Visualization 17

9 CAN Bus Protocol 20

9.1 SW Layer.. 20

9.2 CMD Layer 24

9.2.1 ComCMD_ALIVE PLC <=> HMI 24

9.2.2 ComCMD_UPDATE PLC <=> HMI 25

9.2.3 ComCMD_UPDATESTRING PLC => HMI...................................... 25

9.2.4 ComCMD_RESET PLC => HMI... 26

9.2.5 ComCMD_RUN PLC => HMI 27
9.2.6 ComCMD_SCREEN PLC => RUN 27
9.2.7 ComCMD_BACKLIGHT PLC => HMI 28
9.2.8 ComCMD_BACKLIGHTDIM PLC => HMI 28
9.2.9 ComCMD_ASK_TEMP PLC => HMI 29
9.2.10 ComCMD_TEMP HMI => PLC 29
9.2.11 ComCMD_ASK_ALIVE PLC<=>HMI 30
9.2.12 ComCMD_ASK_ACTSCREEN PLC => HMI 30
9.2.13 ComCMD_ACTSCREEN HMI => PLC 31
9.2.14 ComCMD_ASK_FILE_CRC PLC => HMI 31
9.2.15 ComCMD_FILE_CRC HMI => PLC 32

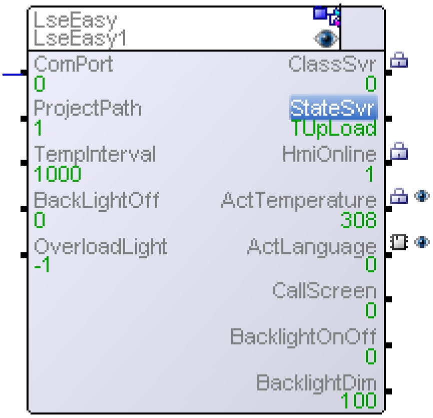
10 Command Examples 33

11 EasyMap.txt 35

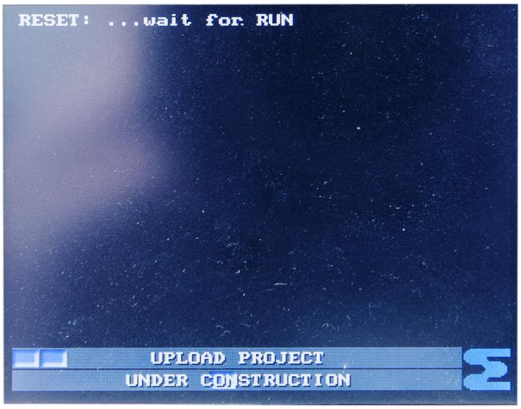
1 Basic Function

The display communicates with the CPU via the CAN bus. A visualization will be created using the LASAL Screen Editor, which will then be loaded to the CPU (files to the CPU).

The LSEEasy class establishes communication with the display. The selected project is then loaded. The class checks whether the project in the display matches the project in the control. If they differ, the class automatically sends the project to the display in which it is then saved. This process can be observed in the StateSvr of the LSEEasy class.



```
LseEasy
LseEasy1
ComPort              ClassSvr
0                           0
ProjectPath          StateSvr
1                     TUpLoad
TempInterval        HmiOnline
1000                        1
BackLightOff   ActTemperature
0                         308
OverloadLight     ActLanguage
-1                          0
                   CallScreen
                            0
               BacklightOnOff
                            0
                 BacklightDim
                          100
```



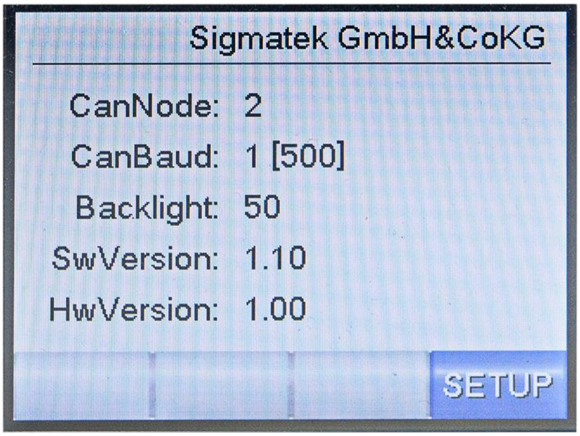
While downloading, the display shows the following: The UPLOAD PROJECT bar shows the progression of the download. The UNDER CONSTRUCTION bar runs continuously. This can however, sometimes remain at one location since the program is then written in memory of the display.

Next, the visualization is started. If the project is the same, the visualization is started immediately.

2 Settings in the CPU

```
autoexec.lsl :SET CAN 1 BAUD 1 STATION 0
```

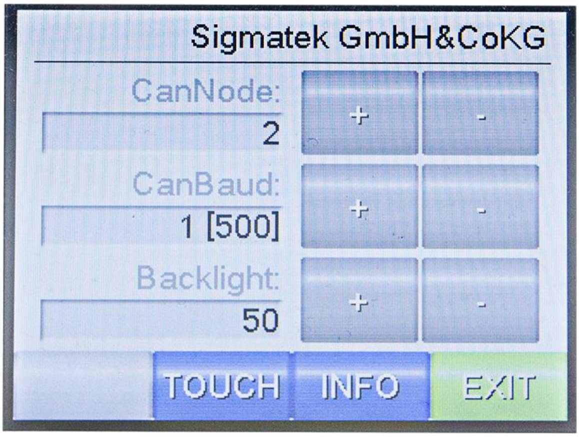
3 Settings in the 3.5" Display



Sigmatek GmbH&CoKG

CanNode: 2
CanBaud: 1 [500]
Backlight: 50
SwVersion: 1.10
HwVersion: 1.00

Press SETUP button!

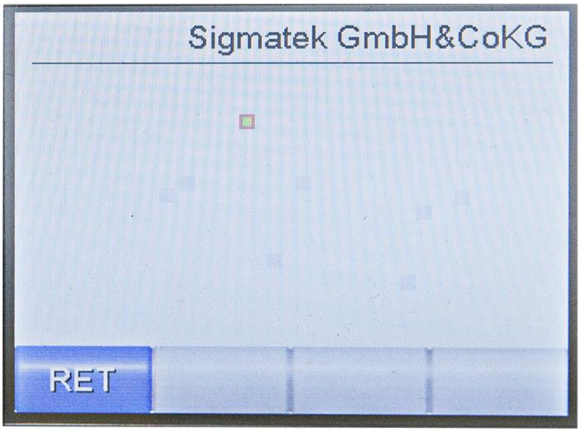


Sigmatek GmbH&CoKG

CanNode: 2
CanBaud: 1 [500]
Backlight: 50

CanNode : setting.

CanBaud : setting

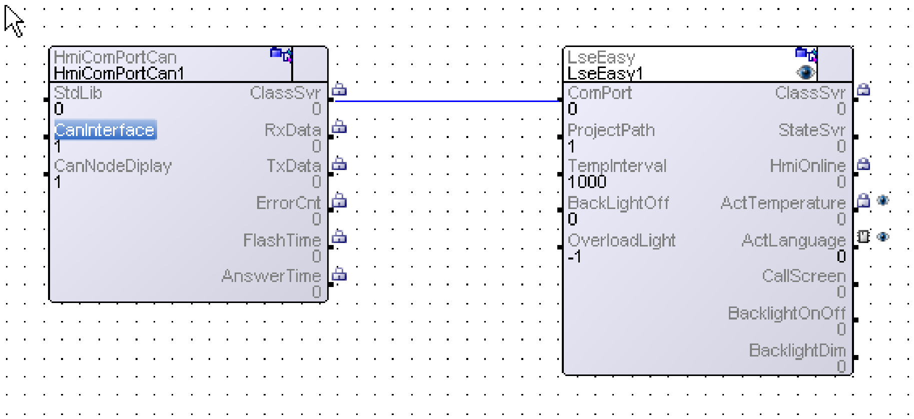


Touch: serves as touch test

RET: Return to main menu

4 Creating a Class Project to Control a 3.5" Display

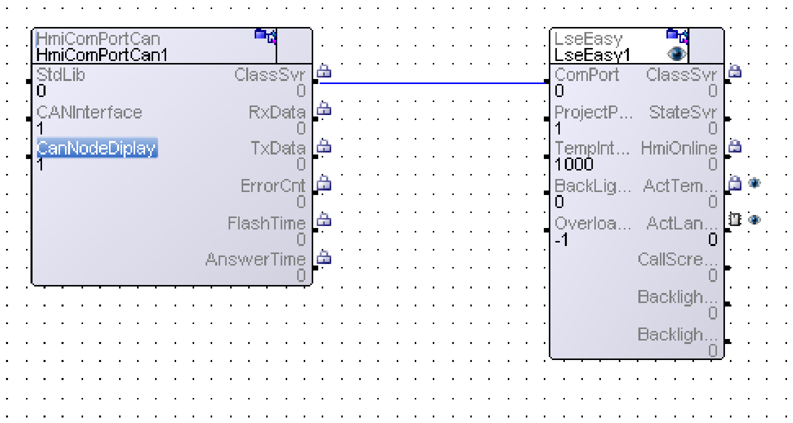
- Create a network and place the following objects. Set the CanInterface (1 .. Can1, 2 .. Can2).



HmiComPortCan
HmiComPortCan1
StdLib 0
CanInterface 1
CanNodeDiplay 1
ClassSvr 0
RxData 0
TxData 0
ErrorCnt 0
FlashTime 0
AnswerTime 0

LseEasy
LseEasy1
ComPort 0
ProjectPath 1
TempInterval 1000
BackLightOff 0
OverloadLight -1
ClassSvr 0
StateSvr 0
HmiOnline 0
ActTemperature 0
ActLanguage 0
CallScreen 0
BacklightOnOff 0
BacklightDim 0

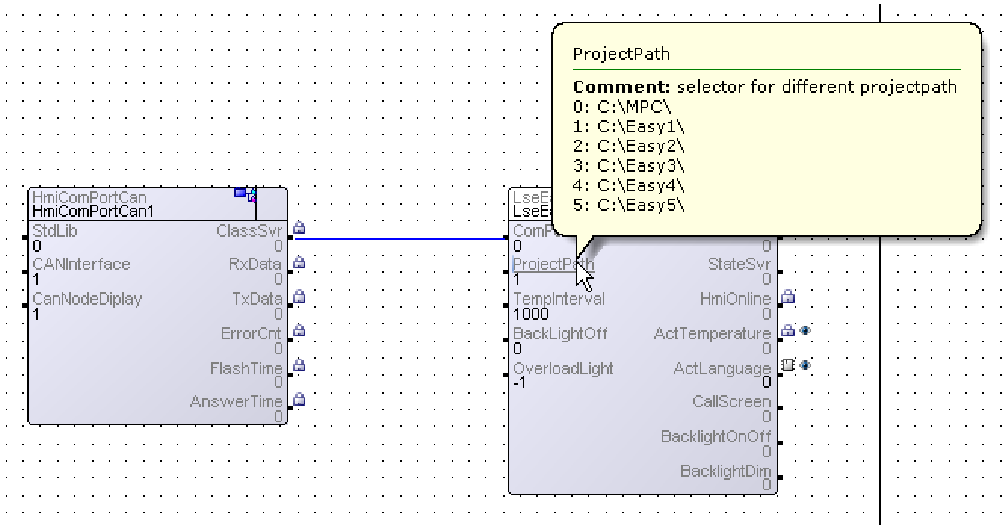
- Set the CanNodeDisplay (CanNode, which was defined in the display setup).



HmiComPortCan
HmiComPortCan1
StdLib 0
CANInterface 1
CanNodeDiplay 1
ClassSvr 0
RxData 0
TxData 0
ErrorCnt 0
FlashTime 0
AnswerTime 0

LseEasy
LseEasy1
ComPort 0
ProjectP... 1
TempInt... 1000
BackLig... 0
Overloa... -1
ClassSvr 0
StateSvr 0
HmiOnline 0
ActTem... 0
ActLan... 0
CallScre... 0
Backligh... 0
Backligh... 0

- In the ProjectPath client, an initial value must also be entered. This value depends on the path over which the visualization should be loaded into the display. This setting will be required again later in the Screen project.



ProjectPath

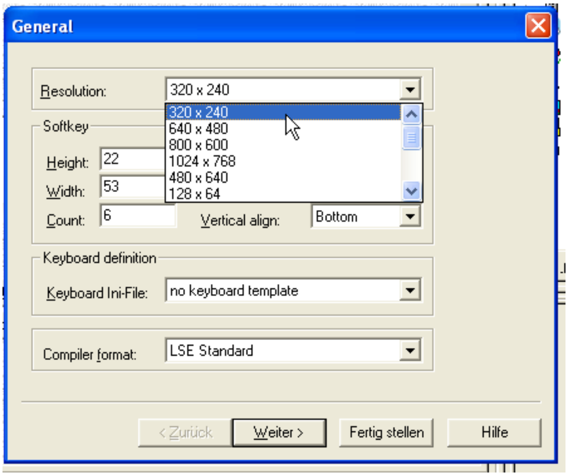
Comment: selector for different projectpath
0: C:\MPC\
1: C:\Easy1\
2: C:\Easy2\
3: C:\Easy3\
4: C:\Easy4\
5: C:\Easy5\

- The remaining clients and servers are not needed at the moment. For a description of the client functions, call the help file of the class.

5 Creating a Screen Project to Show on the Display

- Create a new project.

- Select **320x240 f**or the resolution.


General

Resolution: 320 x 240

320 x 240
640 x 480
800 x 600
1024 x 768
480 x 640
128 x 64

Softkey

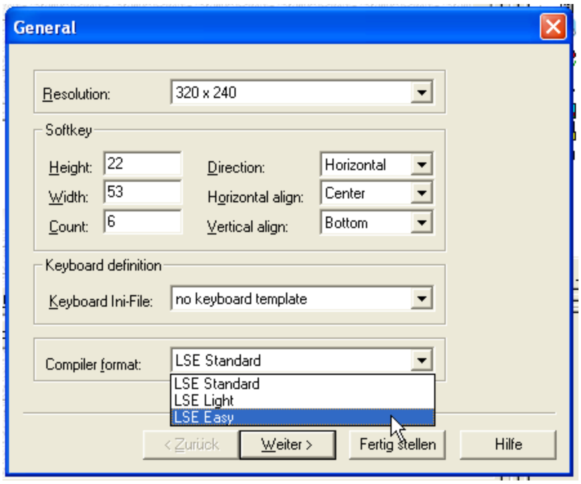
Height: 22
Width: 53
Count: 6
Vertical align: Bottom

Keyboard definition

Keyboard Ini-File: no keyboard template

Compiler format: LSE Standard

- For the compiler format, set: **LSE Easy**.


General

Resolution: 320 x 240

Softkey

Height: 22
Width: 53
Count: 6
Direction: Horizontal
Horizontal align: Center
Vertical align: Bottom

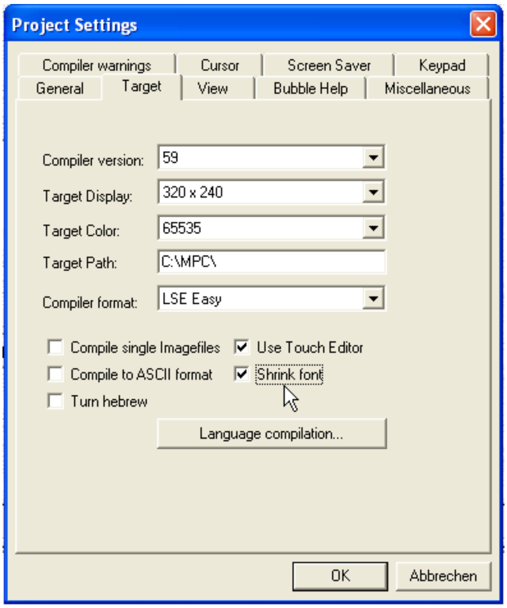
Keyboard definition

Keyboard Ini-File: no keyboard template

Compiler format: LSE Standard

LSE Standard
LSE Light
LSE Easy

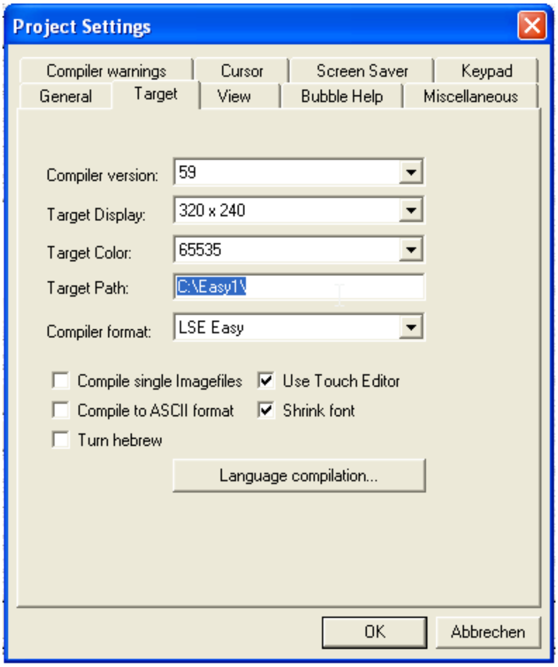
- Press the **Finish** button
- Open the project settings.
- Change to the **Target** tab.
- Select the check box **Shrink font**.



Compiler version: 59
Target Display: 320 x 240
Target Color: 65535
Target Path: C:\MPC\
Compiler format: LSE Easy

☐ Compile single Imagefiles ☑ Use Touch Editor
☐ Compile to ASCII format ☑ Shrink font
☐ Turn hebrew

- Depending on how the ProjectPath client is initialized in the Class project, the **Target Path** should now be changed in the **Settings**.


Project Settings

Compiler warnings | Cursor | Screen Saver | Keypad
General | Target | View | Bubble Help | Miscellaneous

Compiler version: 59
Target Display: 320 x 240
Target Color: 65535
Target Path: C:\Easy1\
Compiler format: LSE Easy

- [ ] Compile single Imagefiles
- [x] Use Touch Editor
- [ ] Compile to ASCII format
- [x] Shrink font
- [ ] Turn hebrew

- Close the settings with the **OK** button.

- Finished!

- The visualization can now be created.

6 Guidelines for using LSE Easy

- Only screens and global screens are available. No windows or objects can be used.

- Up to Firmware version 1.29, the user has 20 screens available; starting with FW version 1.30, 40 screens are available.

- Data servers can only be displayed by their own CPU (Reference To Variables\Connection Settings => internal),

- Units can be used, conversion possible starting with LSE Easy 1.19. Digits, Position of Decimal Point and Unit Text can be used.

- Use bitmaps that are as small as possible. Implementing an image with 200x100 is impractical when it will be used with 100x50. The terminal has little memory space. Maximal 256 Images/Files.

- Only specific button frames are available.

- Z-order on the screen is considered.

- Language conversion is possible. If the language is converted, the visualization is automatically resent from the PLC to the display; since not all text can be displayed in all languages due to memory space. The display then restarts automatically with the visualization output.

- Multiple displays are possible. The same visualization can be shown on several displays. An individual visualization can also be created and shown on each display. Combined operation is also possible (see point 7).

- When turned on, the SIGMATEK setup screen can be replaced by a bitmap. For this purpose, simply create an image with the name bootloader or in the derivation of LseEasy, used the name defined in the GetBootimage() function. To then access the setup, press the lower right corner while the boot image is running.

- An overload function is provided to display the same visualization for different Class objects (see point 8).

- String Editor is available starting with display FW 1.30.

- Real number values CANNOT be correctly displayed.

- No color schemes (only Unit, Font, Text and Image schemes)

- Image alignment for buttons hard-coded to CENTER/CENTER

7 Using Multiple Displays

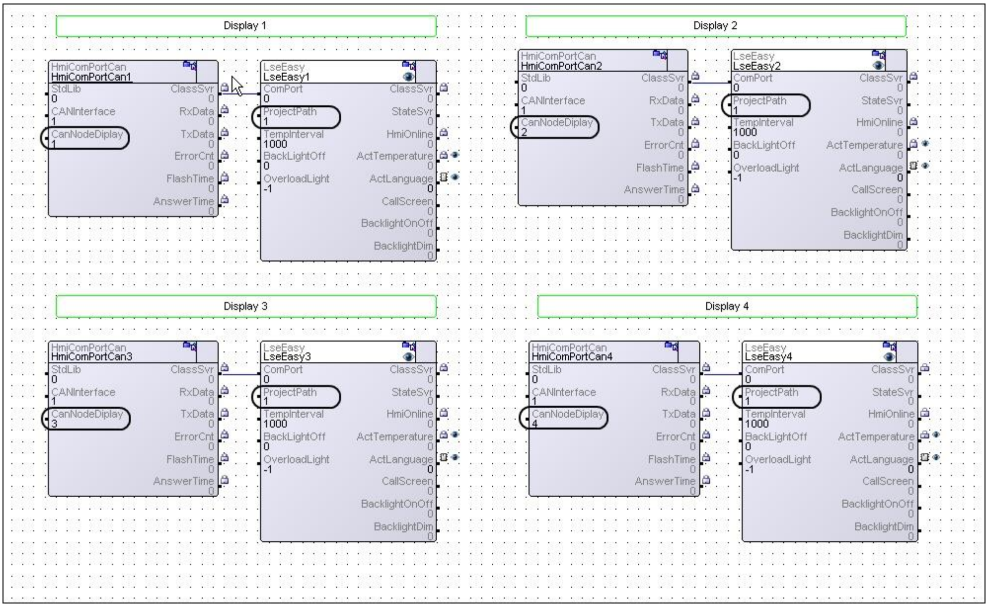
Each display must be configured separately (CanNode, CanBaud). For each display, an individual object of the HmiComPortCan and LseEasy classes is required. The CanNode setting of the display must be set to the client of the appropriate object (CanNodeDisplay).

7.1 Multiple Displays with the Same Visualization

A visualization is created for the display. In the project settings, a path is defined under the menu item Target Path.

For example: C:\Easy1\ => ProjectPath =1   for all four displays

This is then configured in the objects of the LseEasy class. The setting in the ProjectPath client is identical for all objects.



Display 1

HmiComPortCan
HmiComPortCan1
StdLib 0
CANInterface 1
CanNodeDiplay 1
ClassSvr 0
RxData 0
TxData 0
ErrorCnt 0
FlashTime 0
AnswerTime 0

LseEasy
LseEasy1
ComPort 0
ProjectPath 1
TempInterval 1000
BackLightOff 0
OverloadLight -1
ClassSvr 0
StateSvr 0
HmiOnline 0
ActTemperature 0
ActLanguage 0
CallScreen 0
BacklightOnOff 0
BacklightDim 0

Display 2

HmiComPortCan
HmiComPortCan2
StdLib 0
CANInterface 1
CanNodeDiplay 2
ClassSvr 0
RxData 0
TxData 0
ErrorCnt 0
FlashTime 0
AnswerTime 0

LseEasy
LseEasy2
ComPort 0
ProjectPath 1
TempInterval 1000
BackLightOff 0
OverloadLight -1
ClassSvr 0
StateSvr 0
HmiOnline 0
ActTemperature 0
ActLanguage 0
CallScreen 0
BacklightOnOff 0
BacklightDim 0

Display 3

HmiComPortCan
HmiComPortCan3
StdLib 0
CANInterface 1
CanNodeDiplay 3
ClassSvr 0
RxData 0
TxData 0
ErrorCnt 0
FlashTime 0
AnswerTime 0

LseEasy
LseEasy3
ComPort 0
ProjectPath 1
TempInterval 1000
BackLightOff 0
OverloadLight -1
ClassSvr 0
StateSvr 0
HmiOnline 0
ActTemperature 0
ActLanguage 0
CallScreen 0
BacklightOnOff 0
BacklightDim 0

Display 4

HmiComPortCan
HmiComPortCan4
StdLib 0
CANInterface 1
CanNodeDiplay 4
ClassSvr 0
RxData 0
TxData 0
ErrorCnt 0
FlashTime 0
AnswerTime 0

LseEasy
LseEasy4
ComPort 0
ProjectPath 1
TempInterval 1000
BackLightOff 0
OverloadLight -1
ClassSvr 0
StateSvr 0
HmiOnline 0
ActTemperature 0
ActLanguage 0
CallScreen 0
BacklightOnOff 0
BacklightDim 0

7.2 Multiple Displays with Different Visualizations

Four different visualizations are created. Each display contains a different visualization. In the project settings of each visualization, a different path is set under the item Target Path.

For example:

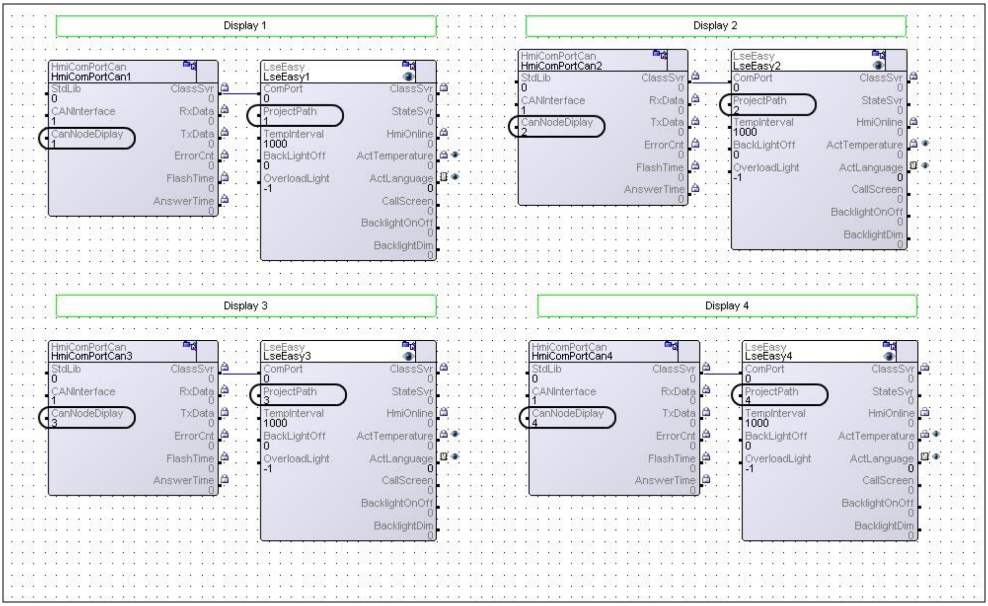
C:\Easy1\ => ProjectPath =1 for visualization 1 on display 1

C:\Easy2\ => ProjectPath =2 for visualization 2 on display 2

C:\Easy3\ => ProjectPath =3 for visualization 3 on display 3

C:\Easy4\ => ProjectPath =4 for visualization 4 on display 4

These are now set in the objects of the LseEasy class. The ProjectPath client is now configured for to the corresponding visualization.



7.3 Combined Operation

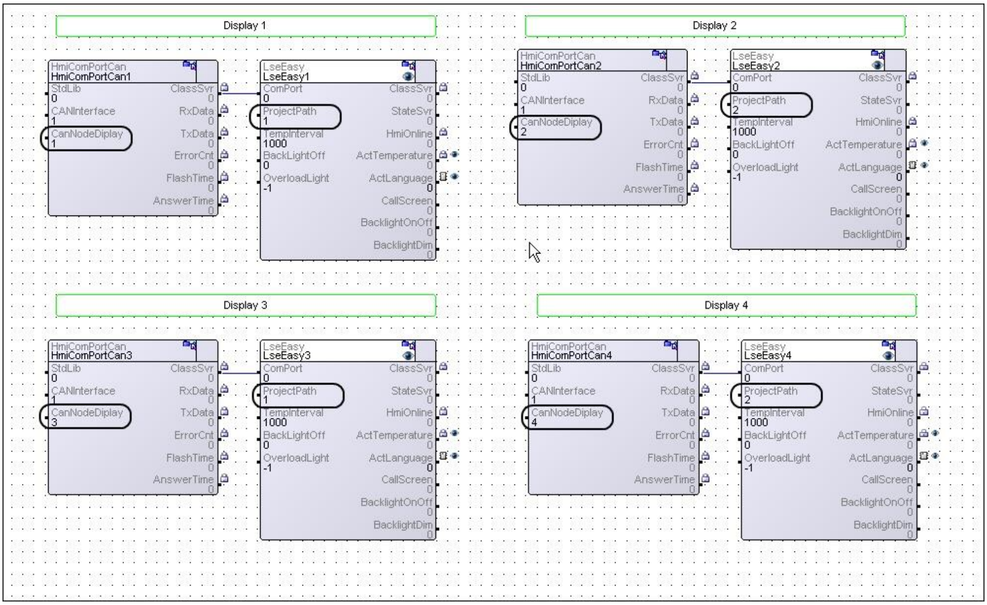
The above display options can be easily combined. Two different visualizations are created and displayed on two screens each. In the project settings, the two visualizations are assigned different paths under the item Target Path.

For example:

C:\Easy1\ => ProjectPath =1 for visualization 1 on display 1 and 3

C:\Easy2\ => ProjectPath =2 for visualization 2 on display 2 and 4

These are now set in the objects of the LseEasy class. The ProjectPath client is now configured for to the corresponding visualization.



8 Loading Data for Multiple Displays with One Visualization

Each display must be configured separately (CanNode, CanBaud).

For each display, an individual object of the HmiComPortCan and LseEasy classes is required. The CanNode setting of the display must be set to the client of the appropriate object (CanNodeDisplay).

A visualization is created for all displays. In the project settings, a path is defined under the menu item Target Path.

For example: C:\Easy1\ => ProjectPath =1   for all four displays

This is then configured in the objects of the LseEasy class. The setting in the ProjectPath client is identical for all objects.

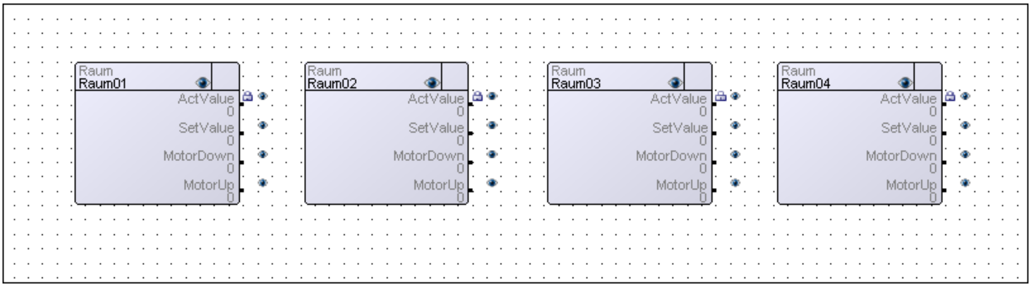
To show different data on each display, the OverloadLight client was added to the LseEasy class. The default value is -1. Value range 0 to 99.

If a number is entered, all Object.Servers in the variables list are searched while loading the project. If the character string XX is found anywhere, it is replaced with the number defined in the client; whereby the number is always formatted for 2 digits.

Example:	OverLoadLight client is initialized with 9.	
ObjectXX.SetValue	=>	is converted to Object09.SetValue.
Object.TempXXMax	=>	converted to Object.Temp09Max.

There are now four displays on which the same visualization should be run, however, with different data. This can be solved as follows:

Class in LASAL2 :	Room	.ActValue	(Server)
		.SetValue	(Server)
		.MotorDown	(Server)
		.MotorUp	(Server)

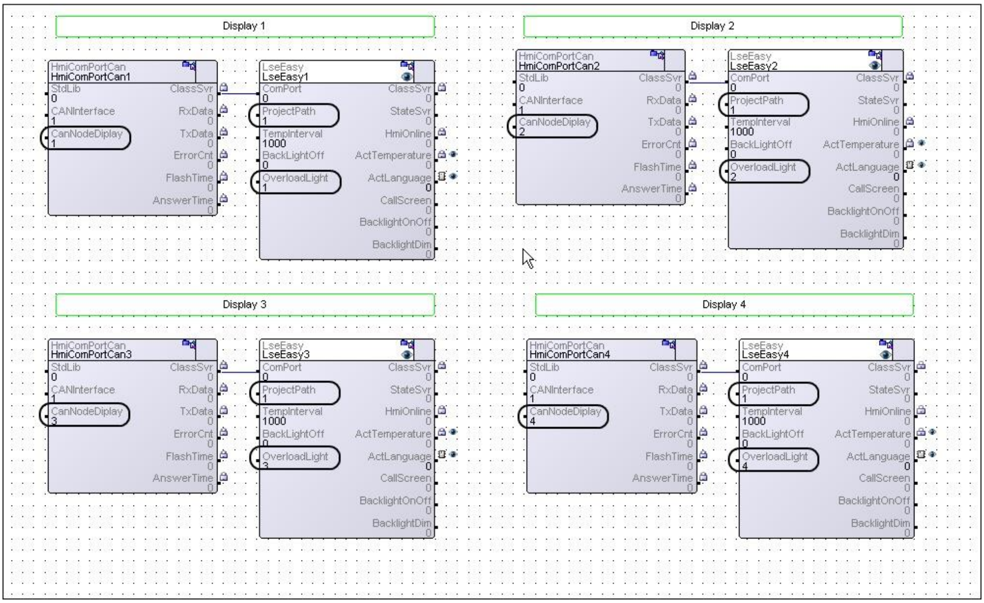


Objects in LASAL2: Room01, Room02, Room03, Room04

When the objects imported into the screen, the objects Room02, Room03 and Room04 are deleted. The object Room1 must be renamed RoomXX.

The servers of the RoomXX object can now be used.

In the LASAL2 project the OverloadLight client must be configured as follows:



Display 1

HmiComPortCan
HmiComPortCan1
StdLib 0
CANInterface 1
CanNodeDiplay 1
ClassSvr 0
RxData 0
TxData 0
ErrorCnt 0
FlashTime 0
AnswerTime 0

LseEasy
LseEasy1
ComPort 0
ProjectPath 1
TempInterval 1000
BackLightOff 0
OverloadLight 1
ClassSvr 0
StateSvr 0
HmiOnline 0
ActTemperature 0
ActLanguage 0
CallScreen 0
BacklightOnOff 0
BacklightDim 0

Display 2

HmiComPortCan
HmiComPortCan2
StdLib 0
CANInterface 1
CanNodeDiplay 2
ClassSvr 0
RxData 0
TxData 0
ErrorCnt 0
FlashTime 0
AnswerTime 0

LseEasy
LseEasy2
ComPort 0
ProjectPath 1
TempInterval 1000
BackLightOff 0
OverloadLight 2
ClassSvr 0
StateSvr 0
HmiOnline 0
ActTemperature 0
ActLanguage 0
CallScreen 0
BacklightOnOff 0
BacklightDim 0

Display 3

HmiComPortCan
HmiComPortCan3
StdLib 0
CANInterface 1
CanNodeDiplay 3
ClassSvr 0
RxData 0
TxData 0
ErrorCnt 0
FlashTime 0
AnswerTime 0

LseEasy
LseEasy3
ComPort 0
ProjectPath 1
TempInterval 1000
BackLightOff 0
OverloadLight 3
ClassSvr 0
StateSvr 0
HmiOnline 0
ActTemperature 0
ActLanguage 0
CallScreen 0
BacklightOnOff 0
BacklightDim 0

Display 4

HmiComPortCan
HmiComPortCan4
StdLib 0
CANInterface 1
CanNodeDiplay 4
ClassSvr 0
RxData 0
TxData 0
ErrorCnt 0
FlashTime 0
AnswerTime 0

LseEasy
LseEasy4
ComPort 0
ProjectPath 1
TempInterval 1000
BackLightOff 0
OverloadLight 4
ClassSvr 0
StateSvr 0
HmiOnline 0
ActTemperature 0
ActLanguage 0
CallScreen 0
BacklightOnOff 0
BacklightDim 0

9 CAN Bus Protocol

Basically, the protocol is divided into two layers.

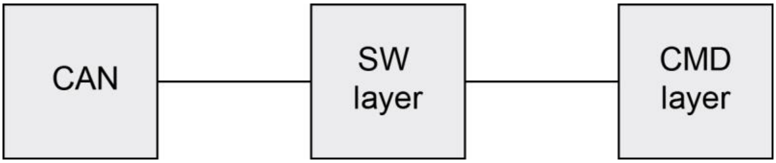
SW Layer

Prepares the data to send (to the HMI) for the CAN bus and sends it to the CAN interface.

Prepares the received data (from the HMI) from the CAN bus and sends it the CMD layer (see point 9.1).

CMD Layer

Maps the individual commands (see point 9.2).



9.1 SW Layer

In the CMD layer, the command is assembled. These data are then transferred to the SW layer using a SendData method, which sends the data via the CAN bus. The answers are again received from this layer and analyzed accordingly, assembled and resent to the CMD layer.

The following object number are used:

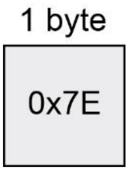
```
#define CAN_TX_OBJECT    0x020
#define CAN_RX_OBJECT    0x040
```

The CAN node of the display must be added to these ID's

Basically, there are 4 different packet types:

- Acknowledge

#define ID_ACK_MESSAGE	0x7E	1 byte

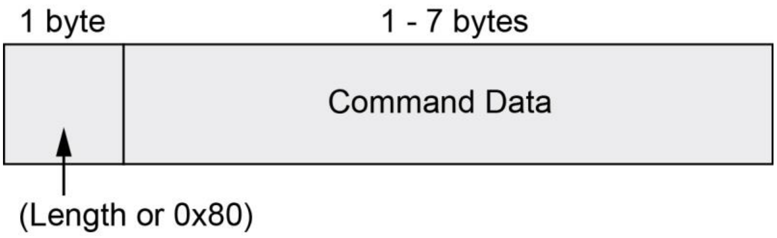


Example:
```
unsigned char tmp;
tmp = ID_ACK_MESSAGE;
CanSend(tmp, 1);
```

- Single Pack Message

This is the case if the command stream is less than or equal to 7 bytes. A remote station answers each single pack message with Acknowledge. Only after the Acknowledge is received, is the command accepted by the remote station.

The following data stream configuration is required.



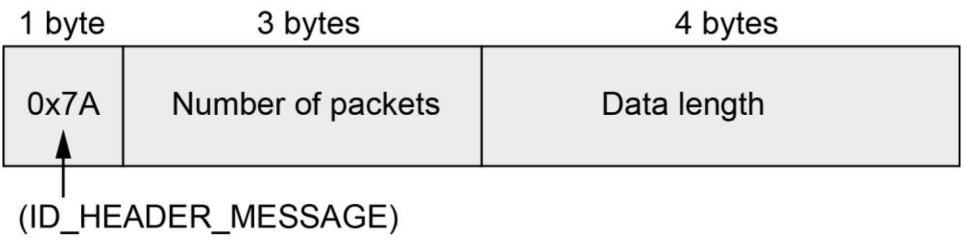
Example:

```
if(mydatalengt <= 7)
{
 unsigned char tmp[8];
tmp[0] = mydatalength | 0x80; // userdatalänge + MSB=1
memcpy(&tmp[1], mydata, mydatalength);  // userdata
CanSend(tmp, 8); // send
// WaitForAcknowledgeWithTimeout(); // wait for Acknowledge
from the remote station
}
```

- Multi Pack Message Header

If the command stream is greater than 7 bytes, the data must be divided into multiple packets. The first of these packets must then be formatted as follows: This command is then answered from a remote station with Acknowledge. Only after the Acknowledge is received, is the command accepted by the remote station.

#define ID_HEADER_MESSAGE	0x7A	8 bytes



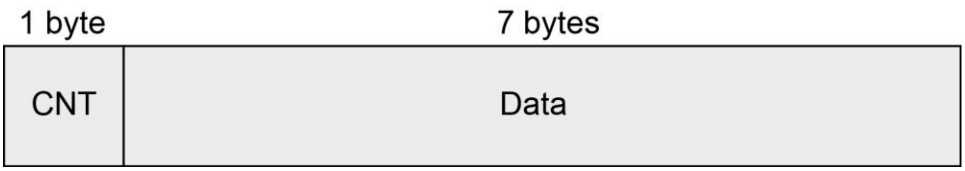
Example:

```
if(mydatalength > 7)
{
unsigned char tmp[8];
tmp[0] = ID_HEADER_MESSAGE;
*(unsigned long*)&tmp[1] = (mydatalength + 6) / 7; // number
of following packets
*(unsigned long*)&tmp[4] = mydatalength; // total length of
the user data in bytes
CanSend(tmp, 8);
// WaitForAcknowledgeWithTimeout(); // wait for Acknowledge
from the remote station
}
```

- Multi Pack Message

Packets with a length of 8 bytes (7-byte payload data) are sent until the last packet is reached, which can have a length of 1 – 7 bytes. If the number of packets is greater than 8, the remote station returns Acknowledge after the 8th packet. Only after this confirmation has been received, can the next packet then be sent (otherwise the last 8 packets must be repeated). The last packet does not have to consist of 7 bytes. This depends on how many bytes are remaining. After the last packet, the remote station also returns Acknowledge. Once the Acknowledge is received, the data is successfully transferred.

Data packet (0..n-1)

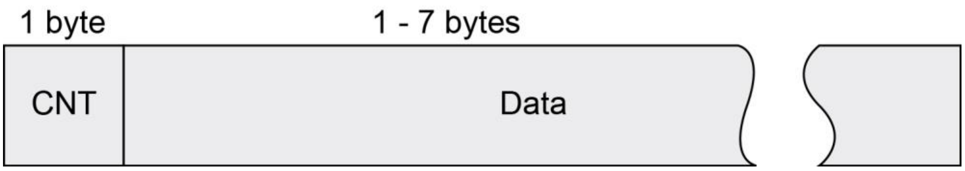


CNT => PaketIndex (0..7)

Example:

```
unsigned char tmp[8];
tmp[0] = idx; // Packet index (0-7)
memcpy(tmp[1], mydata, 7); //
CanSend(tmp, 8);
```

Data packets (n)



CNT => PaketIndex (0..7)

Example:

```
unsigned char tmp[8];
tmp[0] = idx; // Packet index (0-7)
memcpy(tmp[1], mydata, 3); //
CanSend(tmp, 4);
```

9.2 CMD Layer

All commands and data types for the HMI are defined in the MiniSrcData.h file for the program application.

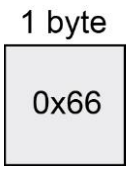
The most important commands are described below, all others require specialized knowledge and should not be used.

9.2.1 ComCMD_ALIVE PLC <=> HMI

This command is a so-called Alive signal and signals the respective receiver to the presence (functionality) of the sender. This command can be sent from the HMI to the PLC as well as from the PLC to the HMI. The PLC should send data to the HMI every 1000 ms via the CAN bus. This command should be sent if no data is currently available to send to the HMI. Otherwise, an offline message appears in the HMI, which indicates that the PLC is missing.

The command requires no additional parameters.

#define ComCMD_ALIVE	0x66	1 byte



Example:
```
        unsigned char cmd =  ComCMD_ALIVE;
        DataSend(&cmd, 1); // Call SW layer
```

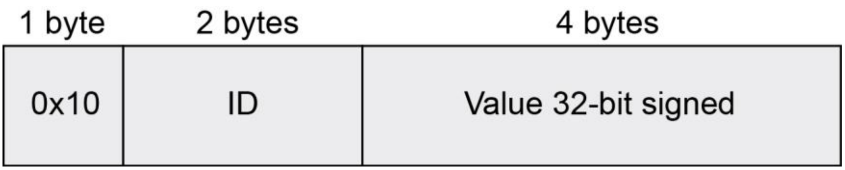
9.2.2 ComCMD_UPDATE PLC <=> HMI

The Update command exchanges values between the PLC and HMI.

Using an ID created by LSE, the value of the corresponding server on the display can be changed. This command can also be run in both directions. Changes in the actual values are sent to the HMI and thereby communicated to the PLC by confirming value changes in the HMI.

It should be noted that values sent with this command are processed as signed 32-bit values. This means the only values from -2,147,483,648 to +2,147,483,647 are allowed.

#define ComCMD_UPDATE	0x10	7 bytes

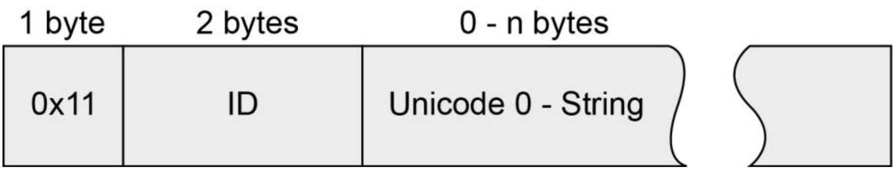


Example:
```
        unsigned char tmp[7];
        tmp[0] = ComCMD_UPDAT;
        *(unsigned short*)&tmp[1] = varid;
        *(unsigned short*)&tmp[3] = value;
        DataSend(tmp, 7); // Call SW layer
```

9.2.3 ComCMD_UPDATESTRING PLC => HMI

The UpdateString command is only available in one direction, since strings can only be changed but not entered. Using the ID, it is assigned to the appropriate server. Only Unicode strings are processed; ASCII strings must be converted accordingly.

#define ComCMD_UPDATESTRING	0x11	5 - n bytes



Example:

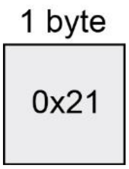
```
unsigned long textlen;
unsigned short uni_string [10];
uni_string[0] = 'H';
uni_string[1] = 'e';
uni_string[2] = 'l';
uni_string[3] = 'l';
uni_string[4] = 'o';
uni_string[5] = 0;

unsigned char tmp[255]
tmp[0] =  ComCMD_ALIVE;
*(unsigned short*)&tmp[1] = id; // id of variable
textlen  =  (StrLenUni(uni_string)  +  1)  *  sizeof(unsigned
short);
memcpy(&tmp[3], uni_string, textlen);
DataSend(&tmp, textlen + 3); // Call SW layer
```

9.2.4 ComCMD_RESET PLC => HMI

Used to put the display into RESET status.

#define ComCMD_RESET	0x21	1 byte



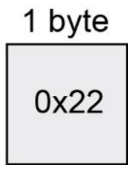
Example:

```
unsigned char cmd =  ComCMD_RESET;
DataSend(&cmd, 1); // Call SW layer
```

9.2.5 ComCMD_RUN PLC => HMI

Used to put the display into RUN status.

#define ComCMD_RUN	0x22	1 byte



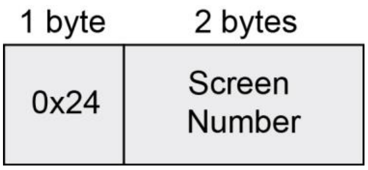
Example:

```
unsigned char cmd =  ComCMD_RUN;
DataSend(&cmd, 1); // Call SW layer
```

9.2.6 ComCMD_SCREEN PLC => RUN

The command is used to change the screen on the display from the PLC.

#define ComCMD_SCREEN	0x24	3 bytes



Example:

```
unsigned char cmd[3] ;
cmd[0] = ComCMD_SCREEN;
*(unsigned short*)&cmd[1] = screen number;
DataSend(&cmd, 3); // Call SW layer
```

9.2.7 ComCMD_BACKLIGHT PLC => HMI

Allows the display backlighting to be turned on/off when one does not want to use the integrated function (see ComCMD_BACKLIGHTTIME).

#define ComCMD_BACKLIGHT	0x26	2 bytes

1 byte	1 byte
0x26	0 ... off 1 ... on

Example:

```
unsigned char cmd[2] ;
cmd[0] = ComCMD_SCREEN;
cmd[1] = 1; // cmd[1] = 0;
DataSend(&cmd, 2); // Call SW layer
```

9.2.8 ComCMD_BACKLIGHTDIM PLC => HMI

This command can be used to adjust the brightness of the backlighting (value 0 ... 100 %, whereby 0 % does not turn it off). To turn off the display, use the ComCMD_BACKLIGHT command.

#define ComCMD_BACKLIGHTDIM	0x30	2 bytes

1 byte	1 byte
0x30	0 - 100 %

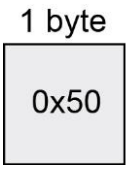
Example:

```
unsigned char cmd[2] ;
cmd[0] = ComCMD_BACKLIGHTDIM;
cmd[1] = 70; // Backlight is set to 70%.
DataSend(&cmd, 2); // Call SW layer
```

9.2.9 ComCMD_ASK_TEMP PLC => HMI

Used to query the actual temperature of the display and then answers with the ComCMD_TEMP command.

#define ComCMD_ASK_TEMP	0x50	1 byte



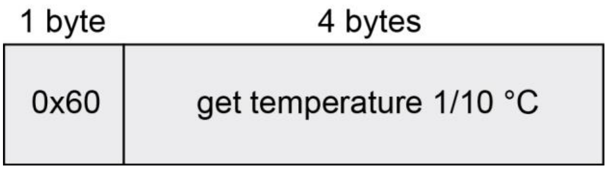
Example:

```
        unsigned char cmd ;
        cmd = ComCMD_ASK_TEMP;
        DataSend(&cmd, 1); // Call SW layer
```

9.2.10 ComCMD_TEMP HMI => PLC

This is the answer to the query from the PLC (ComCMD_ASK_TEMP command). The current temperature is returned in 1/10 °C.

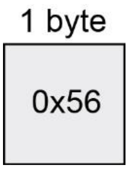
#define ComCMD_TEMP	0x60	5 bytes



9.2.11 ComCMD_ASK_ALIVE PLC<=>HMI

Can be sent from the PLC as well as the HMI. The ComCMD_ALIVE command must be sent as an answer (see 9.2.1).

#define ComCMD_ASK_ALIVE	0x56	1 byte

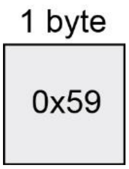


Example:
```
        unsigned char cmd ;
        cmd = ComCMD_ASK_ALIVE;
        DataSend(&cmd, 1); // Call SW layer
```

9.2.12 ComCMD_ASK_ACTSCREEN PLC => HMI

With this command, the currently displayed screen can be read. The answer is then returned with the ComCMD_ACTSCREEN command.

#define ComCMD_ASK_ACTSCREEN	0x59	1 byte



Example:
```
        unsigned char cmd ;
        cmd = ComCMD_ASK_ACTSCREEN;
        DataSend(&cmd, 1); // Call SW layer
```

9.2.13 ComCMD_ACTSCREEN HMI => PLC

This is the answer to the query from the PLC (ComCMD_ASK_ACTSCREEN command), which returns the number of the currently displayed screen.

#define ComCMD_ACTSCREEN	0x2B	3 bytes

1 byte	2 bytes
0x2B	actual screen number

9.2.14 ComCMD_ASK_FILE_CRC PLC => HMI

Required to call the current CRC of the project stored on the display. This should be checked before the variable update is started, since if the CRC is not identical to that in the EasyMap.txt file, the ID may be different. The answer is returned as the ComCMD_FILE_CRC command.

Theoretically, the server IDs can change after each LSE compiler process. For this reason, the ID should not be written into the code, but assigned as a constant. To retrieve the project CRC, 0 must be entered for the data.

#define ComCMD_ASK_FILE_CRC	0x54	3 bytes

1 byte	2 bytes
0x54	0

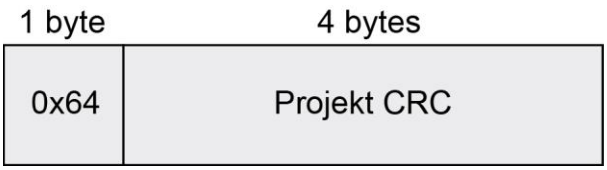
Example:

```
        unsigned char cmd[2] ;
        cmd[0] = ComCMD_ASK_FILE_CRC;
        cmd[1] = 0; // to retrieve the project CRC.
        DataSend(&cmd, 2); // Call SW layer
```

9.2.15 ComCMD_FILE_CRC HMI => PLC

This is the answer to the ComCMD_ASK_FILE_CRC, and returns the CRC of the current project in the display.

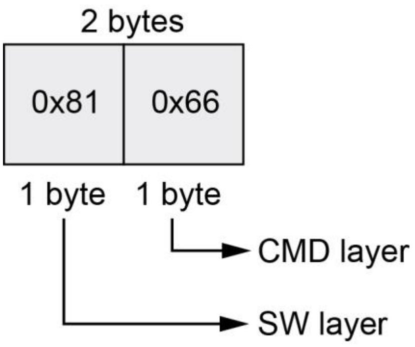
#define ComCMD_FILE_CRC	0x64	5 bytes



10 Command Examples

- The ComCMD_ALIVE command should be sent.

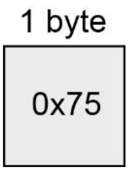
The data, which must be sent via the CAN bus, appear as follows:



```
0x66 => ComCMD_ALIVE command (1-byte length)
0x81 => Single Pack Message (0x80 OR 0x01)
       0x80 => Single Pack Message ID
       0x01 => length of the ComCMD_ALIVE)
```

This means that these 2 bytes must be sent with the object number 0x20 + CanNode display.

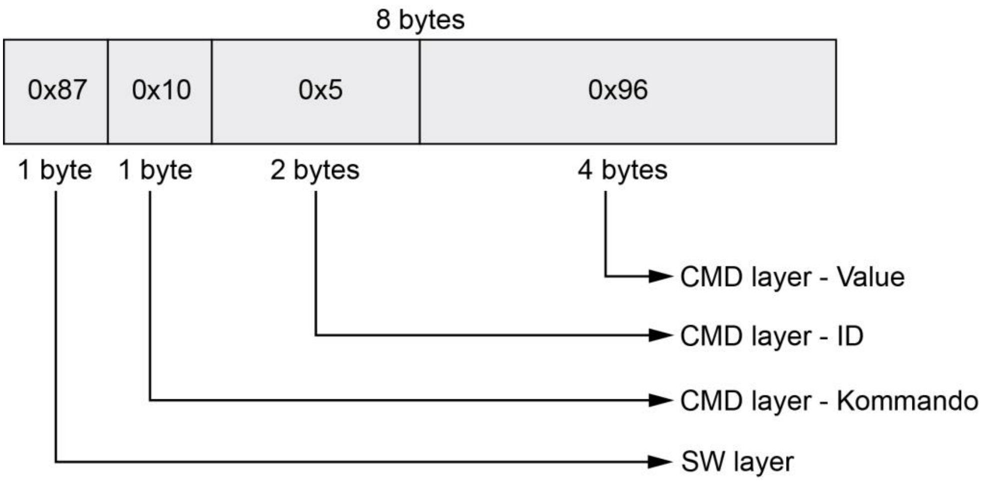
The remote station answers with an Acknowledge. This is then received at the object number 0x40 + CanNode display and appears as follows:



1 byte was received. The content of the data is 0x75. This corresponds to the packet type Acknowledge. ID_ACK_MESSAGE => 0x75

- A value in the display should be updated. The ComCMD_UPDATE command is thereby required. The variable with the ID : 5 should be changed.

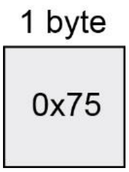
  The value of this variable should be changed to 150.



```
0x96 => value (150)
0x05 => server ID ( 5)
0x10 => ComCMD_UPDATE (7 bytes)
0x87 => Single Pack Message ( 0x80 OR 0x07 )
        0x80 => Single Pack Message ID
        0x07 => length of ComCMD_UPDATE
```

This means that these 8 bytes must be sent with the object number 0x20 + CanNode.

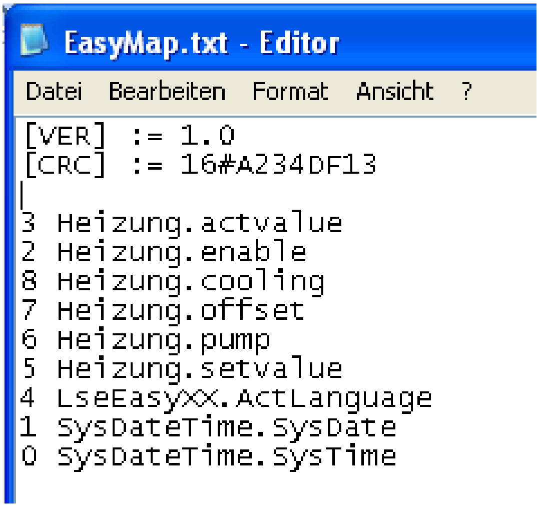
The remote station answers with an Acknowledge. These are received in the object number 0x40 + CanNode and appears as follows.



1 byte was received. The content of the data is 0x75. This corresponds to the packer type Acknowledge. ID_ACK_MESSAGE => 0x75

11 EasyMap.txt

This file creates the reference to the server used in the visualization (file format: regular expression). This file is created by the LseEasy class and is found in the path on the PLC set in the ProjectPath client.



```
[VER] := 1.0
[CRC] := 16#A234DF13

3 Heizung.actvalue
2 Heizung.enable
8 Heizung.cooling
7 Heizung.offset
6 Heizung.pump
5 Heizung.setvalue
4 LseEasyXX.ActLanguage
1 SysDateTime.SysDate
0 SysDateTime.SysTime
```

[CRC] => Project check sum

3 .. server ID : Heating.actvalue

Documentation Changes

Change date	Affected page(s)	Chapter	Note
28.04.2016		6 Guidelines for using LSE Easy	FW Updates