CoDeSys

FBE - Library

Reference Guide
for
use with
EASY242 & EASY2606

frenzel + berg
electronic

1. Content

2. Introduction

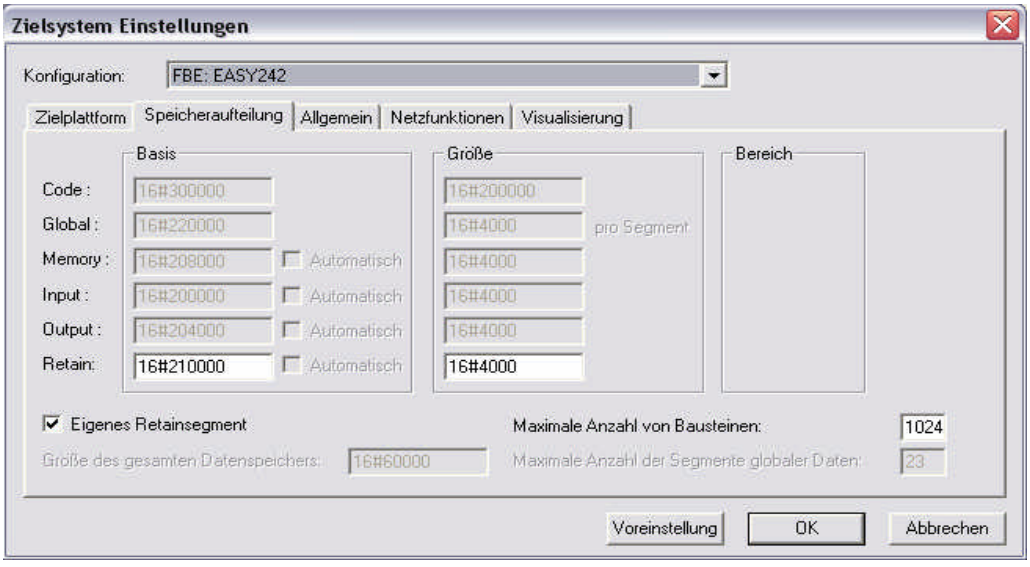
In order to support the powerful EASY PLC core modules there is a library extension for the CoDeSys development environment. Any libraries are internal, that means they are implemented in the PLC runtime system. Others are external IEC-Code libraries.
There are the following libraries:

Library name	available	Description	E242	E2606	Type
FBESysTask.lib	now	Implementation of Interrupt service routines	yes	yes	RT
FBESysEncoder.lib	now	Support of incremental encoders	yes	no	RT
FBESysUtil.lib	now	Several Utilities	yes	yes	RT
FBESysSerial.lib	now	Support of serial interfaces	yes	yes	RT
FBESysKeyboard.lib	now	Support of Simulation a key klick	yes	yes	RT
FBESysTerminal1.lib	now	Support of alpha numeric LCD module	yes	no	RT
FBESysVisuTools.lib	now	Support of Touchposition on a touch display	yes	yes	RT
FBESysInterrupt.lib	now	Support of Interrupt service routines	yes	yes	RT
FBESysCAN.lib	now	Support of two CAN-interfaces	yes	one interface	RT
FBESysIo.lib	now	Support of directly set / reset of Digital Outputs	yes	yes	RT
FBESysPulse.lib	now	Support of PWM and frequency oscillator	yes	no	IEC

RT: included in PLC Runtime system
IEC: external IEC-Code library

3. Non Volatile Memory

The EASY242 offers the possibility to connect a battery extern to the Board in order to support non volatile Retain segments on NVRAM. For the EASY2606, there is an additional Timekeeper option available. Check Tutorial: “How to install timekeeper” on www.frenzel-berg.com
The configuration dialog must be set as below (**Easy242** and **EASY2606**):



4. Utilities

The Library FBESysUtil.lib is a Library extension for the CoDeSys PLC runtime system and supports several utilities for IEC61131 applications running on systems from frenzel + berg elektronik. It is an internal library; all functions are included in the runtime system.

The following functions are implemented:

Function name	Description
SYSUTIL_GETSYSTIME	Returns the cycle time for the PLC application
SYSUTIL_GETFV	Reads the software version of the run time system
SYSUTIL_GETTARGETID	Read the PLC type product code or target ID
SYSUTIL_LEDSET	Sets the user LEDs

4.1. SYSUTIL_GETSYSTIME

Description:

Read actual cycle time for one PLC cycle. The actual cycle time is added to an offset given as parameter. This enables long time measurement.

Declaration:

```
FUNCTION SYSUTIL_GETSYSTIME : UDINT
VAR_INPUT
        SCALE : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
SCALE	UINT	Offset to add to the actual cycle time 0: Systemtime in milli seconds x: reserved

Return Value (Data type UDINT)

The function returns the sum: “actual cycle time” + OffsetVal.

4.2. SYSUTIL_GETFV

Description:

Read the software version of the firmware.

Declaration:

```
FUNCTION SYSUTIL_GETFV : UINT
VAR_INPUT
        DUMMY : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
DUMMY	UINT	Must be held at zero

Return Value (Data type UINT)

The function returns the software version

4.3. SYSUTIL_GETTARGETID

Description:

Read the hardware identification out of the PLC system. The function returns the CodeSys specific target identification.

Declaration:

```
FUNCTION SYSUTIL_GETTARGETID: UINT
VAR_INPUT
        DUMMY : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
DUMMY	UINT	Must be held at zero

Return Value (Data type UDINT)

The function returns the PLC Id

Hardware	Codesys Target (dez)	FBE-Code (hex)
EASY242	14912	
EASY2606	14918	

4.4. SYSUTIL_LEDSET

Description:

Set the User-LEDs.

Declaration:

```
FUNCTION FBEUTIL_LEDSET : BOOL
VAR_INPUT
        LED_NR : UINT;
        LIGHT    : INT;
END_VAR
```

Parameters:

Name	Data-Type	Description
LED_NR	UINT	Number of the LED to program (0 for the first LED)
LIGHT	INT	LED Behavior. 0 LED is off 1..100 Duty cycle of LED-blinking (100% means on) 10% means short on time and long off time 90% means long on time and short off time -1 .. -9 LED flicker times. The LED will be switched on for –LIGHT times and then will be switched off for approx 1 second.

Return Value (Data type BOOL)

The function returns true, if the LED was programmed successfully.

5. Task / CycleTime

The Library FBESysTask.lib is a Library extension for the CoDeSys PLC runtime system and returns the processing / interval time of the last cycle for the chosen task.

The following functions are implemented:

Function name	Description
SYSTASK_GETPROCESSTIME	Returns the processing time of the last cycle for chosen task
SYSTASK_GETTIMECYCLE	Returns the interval time of the last cycle for chosen task

5.1. SYSTASK_GETPROCESSTIME

Description:

Returns the processing time of the last cycle for the chosen task in milli seconds.

Declaration:

```
FUNCTION SYSTASK_GETPROCESSTIME : UINT
VAR_INPUT
        TASKID          : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
TASKID	UINT	Number of the Task, which will be monitored 0: PLC_PRG Task 1: VISU_TASK … all other tasks

Return Value (Data type UINT)

The function returns the processing time for the chosen task in milliseconds.

5.2. SYSTASK_GETTIMECYCLE

Description:

Returns the interval time of the last cycle for the chosen task in milli seconds.

Declaration:

```
FUNCTION SYSTASK_GETTIMECYCLE : UINT
VAR_INPUT
        TASKID          : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
TASKID	UINT	Number of the Task, which will be monitored 0: PLC_PRG Task 1: VISU_TASK … all other tasks

Return Value (Data type UINT)

The function returns the interval time of the chosen task in milliseconds.

6. Interrupt

The Library FBESysInterrupt.lib is a Library extension for the CoDeSys PLC runtime system and enables implementation of Interrupt services for IEC61131 applications. It is an internal library; all functions are included in the runtime system.

Each Interrupt channel is assigned to a dedicated interrupt input pin. The interrupts are edge sensitive and may be configured to positive, negative or both transitions at the corresponding interrupt input pin. The interrupt priority may be selected from thirtytwo levels.

An interrupt may not only be activated from hardware signal transitions, but also by IEC61131 application software. This feature enables implementation of program units at different CPU priorities.

The following functions are implemented:

Function name	Description
SYSINTERRUPT_REGSERVICE	Registers a specific program module for the use with interrupt control. Sets the requested interrupt priority, and interrupt edge
SYSINTERRUPT_ENABLE	Enables a dedicated interrupt channel
SYSINTERRUPT_DISABLE	Disables a dedicated interrupt channel
SYSINTERRUPT_SETREQUEST	Set Interrupt request from IEC61131 application
SYSINTERRUPT_CLRREQUEST	Clear Interrupt request from IEC61131 application
SYSINTERRUPT_DELETESERVICE	Delete a dedicated interrupt channel for use as normal input

The program module to call from interrupt should be implemented as "Program" in the CoDeSys Development environment. There must be no parameters into this module.

Calling of the library functions is according to the IEC61131 standard. See Library Manager of the CoDeSys programming tool for detailed parameter information of the library functions.

6.1. Hardware Cross Reference

This reference shows the connections of the EASY242 and EASY2606.

Type / IRQ Number	EASY2606	EASY242
0..3	In Byte 1.0 ..1.3 (X8)	IRQ 0..3
4..7	In Byte 1.4 ..1.7 (X8)	reserved
8..15	not available	IRQ 8..15 IN2.0..2.7

6.2. SYSINTERRUPT_REGSERVICE

Description:

Registers a function for the use with the interrupt control system. Registering of program modules as an interrupt task is done with the individual Id of this module. The Id can be checked with the function "INDEXOF" of the runtime system.
With registration of the interrupt function, the interrupt keeps still disabled. In order to use this interrupt channel, it must be enabled with function "SYSINTERRUPT_ENABLE".
See example for more information.

Declaration:

```
FUNCTION SYSINTERRUPT_REGSERVICE : BOOL
VAR_INPUT
        IRQNR          : UINT;
        NPOU_ID        : INT;
        IRQPRIORITY    : UINT;
        EDGE           : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
IRQNR	UINT	Number of the Interrupt Channel, which must be used for this Interrupt function.
NPOU_ID	INT	Number of the program module that must be registered for this interrupt. (INDEXOF())
IRQPRIORITY	UINT	Priority level of the Interrupt channel 0 : Lowest Priority 32 : Highest Priority Use with care !!! Higher than all other interrupt sources. Only for very short interrupt program Note! Only one irq at the same priority
EDGE	UINT	Enables the active edge for interrupt activation (Both edges may be enabled at the same time) Bit0 Enables/Disables interrupt enable on rising edge of input signal at dedicated interrupt pin Bit1 Enables/Disables interrupt enable on falling edge of input signal at dedicated interrupt pin Setting of the bits is interpreted as follows Bitx = 0 Edge disabled, Interrupt is not activated at this transition Bitx = 1 Edge enabled, Interrupt is activated at this transition

Return Value (Data type BOOL)

The function returns TRUE, if the interrupt function was successfully registered to the PLC runtime system. Otherwise FALSE is returned.

Example

A program module named "CallMeFromIrq" is to call with rising edge at interrupt input pin IRQ0. The requested Priority for this interrupt is 2.
The registration of the interrupt is done with:

```
…
VAR
        …
        Success: BOOL;          (* Use this var for Success of Boolean functions *)
        …
END_VAR
…
…
Success:= SYSINTERRRUPT_REGSERVICE(0, INDEXOF(CallMeFromIrq), 2, 1);
…
```

6.3. SYSINTERRUPT_ENABLE

Description:

Enables an interrupt channel for reception of interrupt requests. Previously set request bits will be cleared.
With registration of the interrupt function, the interrupt keeps still disabled. In order to use this interrupt channel, it must be enabled with this function.
The user must take care of the correct handling of registering and enabling of interrupts. This function does not check, whether there is a interrupt task registered to the channel, that should be enabled.

Declaration:

```
FUNCTION SYSINTERRUPT_ENABLE : BOOL
VAR_INPUT
        IRQNR: UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
IRQNR	UINT	Number of the Interrupt Channel, which must be enabled.

Return Value (Data type BOOL)

The function returns TRUE, if the interrupt function was successfully enabled. Otherwise FALSE is returned.

Example

A program module named "CallMeFromIrq" is to call with both edges at interrupt input pin IRQ0. The requested Priority for this interrupt is 1. Afterwards the interrupt must be enabled.

```
…
VAR
        …
        Success: BOOL;              (* Use this var for Success of Boolean functions *)
        …
END_VAR
…
Success:= SYSINTERRRUPT_REGSERVICE(0, INDEXOF(CallMeFromIrq), 1, 3);
Success:= SYSINTERRUPT_ENABLE(0);
```

6.4. SYSINTERRUPT_DISABLE

Description:

Disables an interrupt channel for reception of interrupt requests.

Declaration:

```
FUNCTION SYSINTERRUPT_DISABLE : BOOL
VAR_INPUT
        IRQNR : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
IRQNR	UINT	Number of the Interrupt Channel, which must be disabled.

Return Value (Data type BOOL)

The function returns TRUE, if the interrupt function was successfully disabled. Otherwise FALSE is returned.

6.5. SYSINTERRUPT_SETREQUEST

Description:

Sets the interrupt request flag of a dedicated interrupt channel. If this channel was previously enabled, the interrupt will be called.

Declaration:

```
FUNCTION SYSINTERRUPT_SETREQUEST : BOOL
VAR_INPUT
        IRQNR : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
IRQNR	UINT	Number of the Interrupt Request Channel, which must be set.

Return Value (Data type BOOL)

The function returns TRUE, if the interrupt request was successfully set. Otherwise FALSE is returned.

Example

The interrupt request for Interrupt channel 3 must be set.

```
VAR
        Success: BOOL;                  (* Use this var for Success of Boolean functions *)
END_VAR

Success:= SYSINTERRUPT_SETREQUEST(3);
```

6.6. SYSINTERRUPT_CLRREQUEST

Description:

Clears the interrupt request flag of a dedicated interrupt channel.

Declaration:

```
FUNCTION SYSINTERRUPT_CLRREQUEST : BOOL
VAR_INPUT
        IRQNR : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
IRQNR	UINT	Number of the Interrupt Request Channel, which must be cleared.

Return Value (Data type BOOL)

The function returns TRUE, if the interrupt request was successfully cleared. Otherwise FALSE is returned.

Example

The interrupt request for Interrupt channel 0 must be cleared.

```
…
VAR     …
        Success: BOOL;              (* Use this var for Success of Boolean functions *)
        …
END_VAR
…
Success:= SYSINTERRUPT_CLRREQUEST (0);
…
```

6.7. SYSINTERRUPT_DELETESERVICE

Description:

This function deletes the Interrupt service of a requested interrupt channel. To enable the Interrupt again it must be used the function “SYSINTERRUPT_REGSERVICE” and then “SYSINTERRUPT_ENABLE”.

Declaration:

```
FUNCTION SYSINTERRUPT_DELETESERVICE : BOOL
VAR_INPUT
        IRQNR : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
IRQNR	UINT	Number of the Interrupt Request Channel, which service must be deleted.

Return Value (Data type BOOL)

The function returns TRUE, if the interrupt request was successfully cleared. Otherwise FALSE is returned.

7. Encoder / Event Counter

The Library FBESysEncoder.lib is a Library extension for the CoDeSys PLC runtime system and enables access to incremental encoders with 2 tracks for IEC61131 applications. The encoder counters are 32 bit counters.
The number of supported encoder channels depends on the target system.
It is an internal library; all functions are included in the runtime system.

Attention: This Library is **NOT** available for **EASY2606**!

The following functions are implemented:

Function name	Description
SYSENCODER_CONTROL	Encoder Controller function This function is used to: Start/stop/Clear/Preset the encoder channel and read the count value.
SYSENCODER_INIT	Setup or reconfiguration of an encoder channel The setup of the encoder channels is done with application startup. So the use of this function is optional and only required, if a channel must be reconfigured during run time.
SYSENCODER_READ32BIT	Read the value from the encoder as long value (32 bit position)
SYSENCODER_READ16BIT	Read the value from the encoder as integer value (16 bit position)
SYSENCODER_START	Starts the encoder
SYSENCODER_STOP	Stops the encoder

7.1. Data Types

In order to implement the Encoder library functions there is a new data type declared in the library FBESysEncoder.lib. It is strongly recommended to use this data types with the library functions, even if the replaced constant would be also processed by the CoDeSys compiler without any problems.

Type_ENCODER

Data type to select the encoder channel.

```
TYPE type_ENCODER : (
        ENCODER0:= 0,
        ENCODER1:= 1,
        ENCODER2:= 2,
        ENCODER3:= 3
);
END_TYPE
```

7.2. SYSENCODER_INIT

Description:

Performs a setup or reconfiguration for the selected encoder channel. The setup of the encoder channels is done with application startup. So the use of this function is optional and only required, if a channel must be reconfigured during run time.

Declaration:

```
FUNCTION SYSENCODER_INIT : BOOL
VAR_INPUT
        Encoder     : type_ENCODER;
        Enable      : BOOL;
        HighRes     : BOOL;
        InvertDir   : BOOL;
        Reserved    : BOOL;
END_VAR
```

Parameters:

Name	Data-Type	Description
ENCODER	type_ENCODER	Number of Encoder Channel for this function call
ENABLE	BOOL	If set to TRUE, the setup for the selected encoder channel is performed
HIGHRES	BOOL	If set to TRUE, the input frequency of the encoder channel is doubled.
INVERTDIR	BOOL	If set to TRUE, the encoder channel decrements else if set to FALSE, the encoder channel increments
RESERVED	BOOL	reserved

Return Value (Data type BOOL)

The function returns TRUE if the setup for the selected encoder channel was performed, otherwise false.

7.3. SYSENCODER_CONTROL

Description:

Controller cycle for complete access to the encoder counter and control flags.

Declaration:

```
FUNCTION SYSENCODER_CONTROL : DINT
VAR_INPUT
        Encoder     : type_ENCODER;
        Enable      : BOOL;
        Clear       : BOOL;
        Preset      : BOOL;
        PresetValue : DINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
ENCODER	type_ENCODER	Number of Encoder Channel for this function call
ENABLE	BOOL	Enable of Encoder Channel If Channel is enabled counting of encoder pulses is enabled. Otherwise counting is stopped.
CLEAR	BOOL	If set to TRUE, the count value of the selected encoder channel is set to zero
PRESET	BOOL	If set to TRUE, the count value of the selected encoder channel is set to the value given with parameter PresetValue. Note: If parameters PRESET and CLEAR are set to TRUE at the same time CLEAR has priority and will be performed.
PRESETVALUE	DINT	If parameter PRESET is set to TRUE, the count value of the selected encoder channel is set to the value given with parameter PRESETVALUE.

Return Value (Data type DINT)

The function returns the count value of the selected encoder channel.

7.4. SYSENCODER_READ32BIT

Description:

Reads the complete count value from the encoder channel. The position is returned as DINT (32 bit value)

Declaration:

```
FUNCTION SYSENCODER_READ32BIT : DINT
VAR_INPUT
        ENCODER      : type_ENCODER;
END_VAR
```

Parameters:

Name	Data-Type	Description
ENCODER	type_ENCODER	Number of Encoder Channel for this function call

Return Value (Data type DINT)

Position as DINT (32 bit value)

7.5. SYSENCODER_READ16BIT

Description:

Reads only the lower word of count value from the encoder channel. The position is returned as INT (16 bit value) This function is implemented to reduce the time for encoder access for applications that request only 16 bit position range.

Declaration:

```
FUNCTION SYSENCODER_READ16BIT : INT
VAR_INPUT
        ENCODER      : type_ENCODER;
END_VAR
```

Parameters:

Name	Data-Type	Description
ENCODER	TYPE_ENCODER	Number of Encoder Channel for this function call

Return Value (Data type INT)

Position as INT (16 bit value)

7.6. SYSENCODER_START

Description:

Starts the selected encoder channel.

Declaration:

```
FUNCTION SYSENCODER_START : BOOL
VAR_INPUT
        ENCODER      : type_ENCODER;
END_VAR
```

Parameters:

Name	Data-Type	Description
ENCODER	type_ENCODER	Number of Encoder Channel for this function call

Return Value (Data type BOOL)

The function returns TRUE if the start for the selected encoder channel was performed, otherwise false.

7.7. SYSENCODER_STOP

Description:

Stops the selected encoder channel.

Declaration:

```
FUNCTION SYSENCODER_STOP : BOOL
VAR_INPUT
        ENCODER      : type_ENCODER;
END_VAR
```

Parameters:

Name	Data-Type	Description
ENCODER	type_ENCODER	Number of Encoder Channel for this function call

Return Value (Data type BOOL)

The function returns TRUE if the stop for the selected encoder channel was performed, otherwise false.

8. PULSE

The Library FBESysPulse.lib is an IEC-Code external library and may be used for generating an Oscillator with constant frequency or a Pulse Width Modulation (PWM).

Attention: This library is **NOT** available for **EASY2606**!

8.1. Hardware Cross Reference

This reference shows the connections of the EASY242.

Typ / Pulse Channel	EASY 242
0	OUT1.0 / PWM0
1	OUT1.1 / PWM1
2	OUT1.2 / PWM2
3	OUT1.3 / PWM3
Range	Max. CPU-Clock / 8

Note:

If one of the four possible channels (depending on used hardware) is initialized as an Oscillator or a PWM-Channel, this channel can't be used as digital output.

The following functions are implemented:

Function name	Description
SYSPULSE_INITOSC	Inizialize the request Oscillator channel
SYSPULSE_INITPWM	Inizialize the request Pulse Width Modulation channel
SYSPULSE_SETFHZOSC	Set the frequency of the Oscillator channel
SYSPULSE_SETPWMSTEPS	Set the Steps of the Pulse Width Modulation channel
SYSPULSE_STOP	Stops the request channel
SYSPULSE_UNINSTALLOSC	Uninstall the request Oscillator channel
SYSPULSE_UNINSTALLPWM	Uninstall the request Pulse Width Modulation channel

8.2. Data Types

In order to implement the Pulse library functions there are several new data types declared in the library FBESysPulse.lib. It is strongly recommended to use this data types with the library functions, even if the replaced constant would be also processed by the CoDeSys compiler without any problems.

Type_PLS_DIV

Data type to select the prescaler of an pulse channel.

```
TYPE type_PLS_DIV : (
        NONE,
        DIV2,
        DIV4,
        DIV8,
        DIV16,
        DIV32,
        DIV64,
        DIV128
);
END_TYPE
```

Type_PLS_MODE

Data type to prevent the use of an output as Oscillator and a PWM at the same time. This data type is only for internal use.

```
TYPE type_PLS_MODE : (
        PLS_NO,
        PLS_PWM,
        PLS_OSC
);
END_TYPE
```

8.3. SYSPULSE_INITOSC

Description:

Inizialize the request Oscillator channel.

Declaration:

```
FUNCTION SYSPULSE_INITOSC : BOOL
VAR_INPUT
        PLSCPUCLOCKMHZ          : UINT;
        PLSCHANNEL              : UINT;
        PLSPARA                 : UINT;
        PLSPRESCALER            : type_PLS_DIV;
END_VAR
```

Parameters:

Name	Data-Type	Description
PLSCPUCLOCKMHZ	UINT	CPU Clock in MHz of the System (default 64 MHz)
PLSCHANNEL	UINT	Number of channel for Oscillator-Signal-Output (Range 0..3)
PLSPARA	UINT	Pulse length (= PLSPara * 8/CPUClock)
PLSPRESCALER	type_PLS_DIV	to select prescale of None, DIV2, DIV4, DIV8, DIV32, DIV64, DIV128 of (CPUClock/8)

Return Value (Data type BOOL)

TRUE If initialization of the oscillator channel was successful
FALSE If initialization of the oscillator channel failed

8.4. SYSPULSE_INITPWM

Description:

Inizialize the request Pulse Width Modulation channel.

Declaration:

```
FUNCTION SYSPULSE_INITPWM : BOOL
VAR_INPUT
        PLSCPUCLOCKMHZ          : UINT;
        PLSCHANNEL              : UINT;
        PLSSTEPS                : UINT;
        PLSPRESCALER            : type_PLS_DIV;
END_VAR
```

Parameters:

Name	Data-Type	Description
PLSCPUCLOCKMHZ	UINT	CPU Clock in MHz of the System (default 64 MHz)
PLSCHANNEL	UINT	Number of channel for PWM-Signal-Output (Range 0..7)
PLSSTEPS	UINT	Maximum of steps for resolution
PLSDiv2	BOOL	set FALSE for Edge Alligned PWM ; set TRUE for Center Alligned PWM
PLSViv64	BOOL	set FALSE for PWMClock resulution = CPUClock; set TRUE for PWMClock resulution = CPUClock/64

Return Value (Data type BOOL)

TRUE If initialization of the PWM channel was successful
FALSE If initialization of the PWM channel failed

8.5. SYSPULSE_SETFHZOSC

Description:

Set the frequency of the Oscillator channel. (value → Hz)

Declaration:

```
FUNCTION SYSPULSE_SETFHZOSC : BOOL
VAR_INPUT
        PLSCHANNEL          : UINT;
        PLSFRQ              : DWORD;
END_VAR
```

Parameters:

Name	Data-Type	Description
PLSCHANNEL	UINT	Number of channel for Oscillator-Signal-Output (Range 0..3)
PLSFRQ	DWORD	Frequency of the Oscillator-Signal at Hz

Return Value (Data type BOOL)

TRUE if frequency is set, else FALSE.

8.6. SYSPULSE_SETPWMSTEPS

Description:

Set the Steps of the Pulse Width Modulation channel.

Declaration:

```
FUNCTION SYSPULSE_SETPWMSTEPS : BOOL
VAR_INPUT
        PLSCHANNEL          : UINT;
        PLSVALUE            : UINT ;
END_VAR
```

Parameters:

Name	Data-Type	Description
PLSChannel	UINT	Number of channel for PWM-Signal-Output (Range 0..7)
PLSValue	UINT	Number of steps to go (PLSValue <PLSSteps)

Return Value (Data type BOOL)

TRUE if steps are set, else FALSE.

8.7. SYSPULSE_STOP

Description:

Stops the signal of the request channel.

Declaration:

```
FUNCTION SYSPULSE_STOP : BOOL
VAR_INPUT
        PLSCHANNEL          : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
PLSCHANNEL	UINT	Number of channel which has to stop Signal-Output (Range 0..7)

Return Value (Data type BOOL)

TRUE if stop is set, else FALSE

8.8. SYSPULSE_UNINSTALLOSC

Description:

Uninstalls the request Oscillator-Channel to use it as a digital Port.

Declaration:

```
FUNCTION SYSPULSE_UNINSTALLOSC : BOOL
VAR_INPUT
        PLSCHANNEL          : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
PLSCHANNEL	UINT	Number of Oscillator-Channel which shoud be uninstalled (Range 0..3)

Return Value (Data type BOOL)

TRUE if channel is successfully uninstalled, else FALSE.

8.9. SYSPULSE_UNINSTALLPWM

Description:

Uninstalls the request PWM-Channel to use it as a digital Port.

Declaration:

```
FUNCTION SYSPULSE_UNINSTALLPWM : BOOL
VAR_INPUT
        PLSCHANNEL          : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
PLSCHANNEL	UINT	Number of PWM-Channel which shoud be uninstalled (Range 0..7)

Return Value (Data type BOOL)

TRUE if channel is successfully uninstalled, else FALSE.

9. Digital Output

The Library FBESysIo.lib is a Library extension for the CoDeSys PLC runtime system and supporting several utilities for IEC61131 applications running on systems from frenzel + berg electronic. It is an internal library; all functions are included in the runtime system.

9.1. SYSIO_RESETDIGITALOUT

Description:

This function resets a single digital output channel. The output is reset directly on the hardware without waiting for the IO update cycle at the end of the PLC loop. The corresponding memory location for the output data is also cleared

Declaration:

```
FUNCTION SYSIO_RESETDIGITALOUT : BOOL
VAR_INPUT
        OUTPUTBYTENR        : UINT;
        OUTPUTBITNR         : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
OUTPUTBYTENR	UINT	Number of the Digital Output Byte which shoud be reseted
OUTPUTBITNR	UINT	Number of the Bit from the requested Output Byte which shoud be reset

Return Value (Data type BOOL)

TRUE if Output Byte was successfully reset, else FALSE.

9.2. SYSIO_SETDIGITALOUT

Description:

This function sets a single digital output channel. The output is set directly on the hardware without waiting for the IO update cycle at the end of the PLC loop. The corresponding memory location for the output data is also set.

Declaration:

```
FUNCTION SYSIO_SETDIGITALOUT : BOOL
VAR_INPUT
        OUTPUTBYTENR        : UINT;
        OUTPUTBITNR         : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
OUTPUTBYTENR	UINT	Number of the Digital Output Byte which shoud be set
OUTPUTBITNR	UINT	Number of the Bit from the requested Output Byte which shoud be set

Return Value (Data type BOOL)

TRUE if Output Byte was successfully set, else FALSE.

9.3. SYSIO_WRALLDIGITALOUT

Description:

This functions writes the complete digital output data of the EASY242 related outputs to the hardware. This function can be used to force an additional hardware output update within the PLC task or from interrupt level

Declaration:

```
FUNCTION SYSIO_WRALLDIGITALOUT : BOOL
VAR_INPUT
        MODE  : UINT;
END_VAR
```

Parameters:

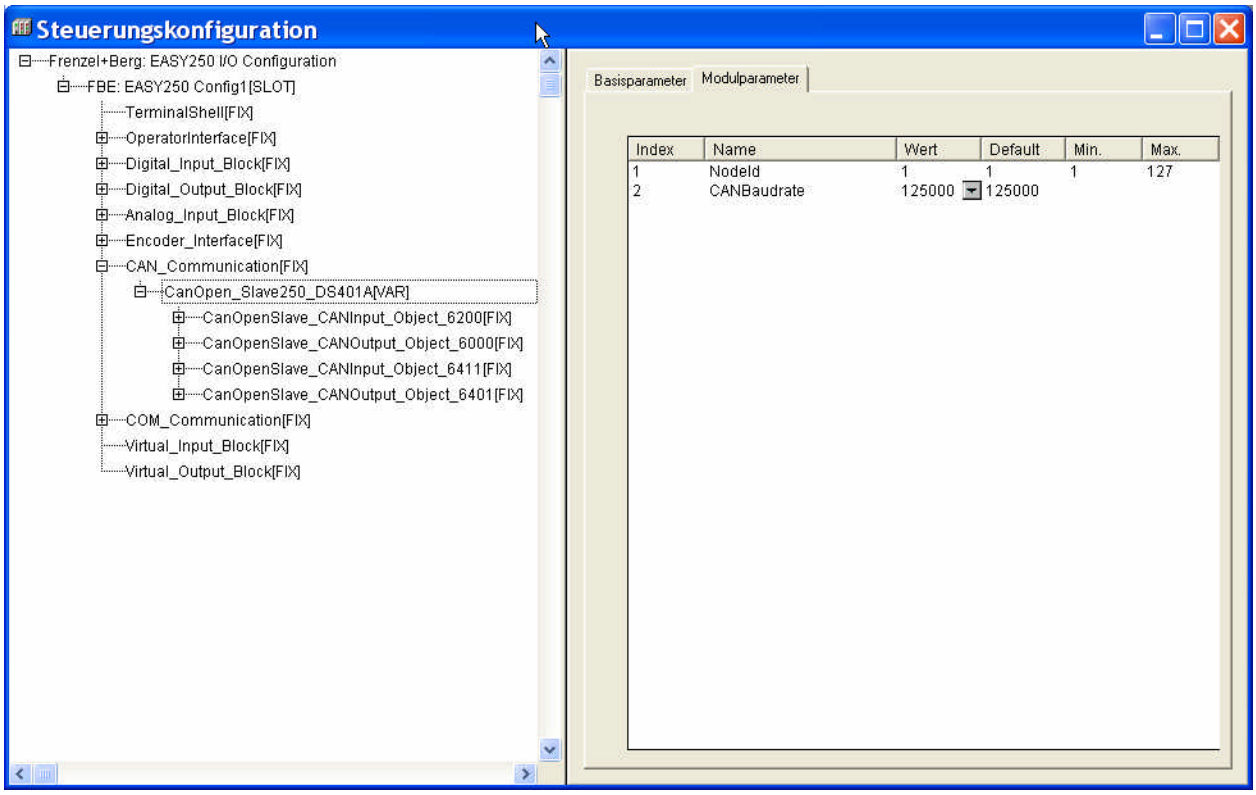
Name	Data-Type	Description
MODE	UINT	Reserved for future use, set to 0

Return Value (Data type BOOL)

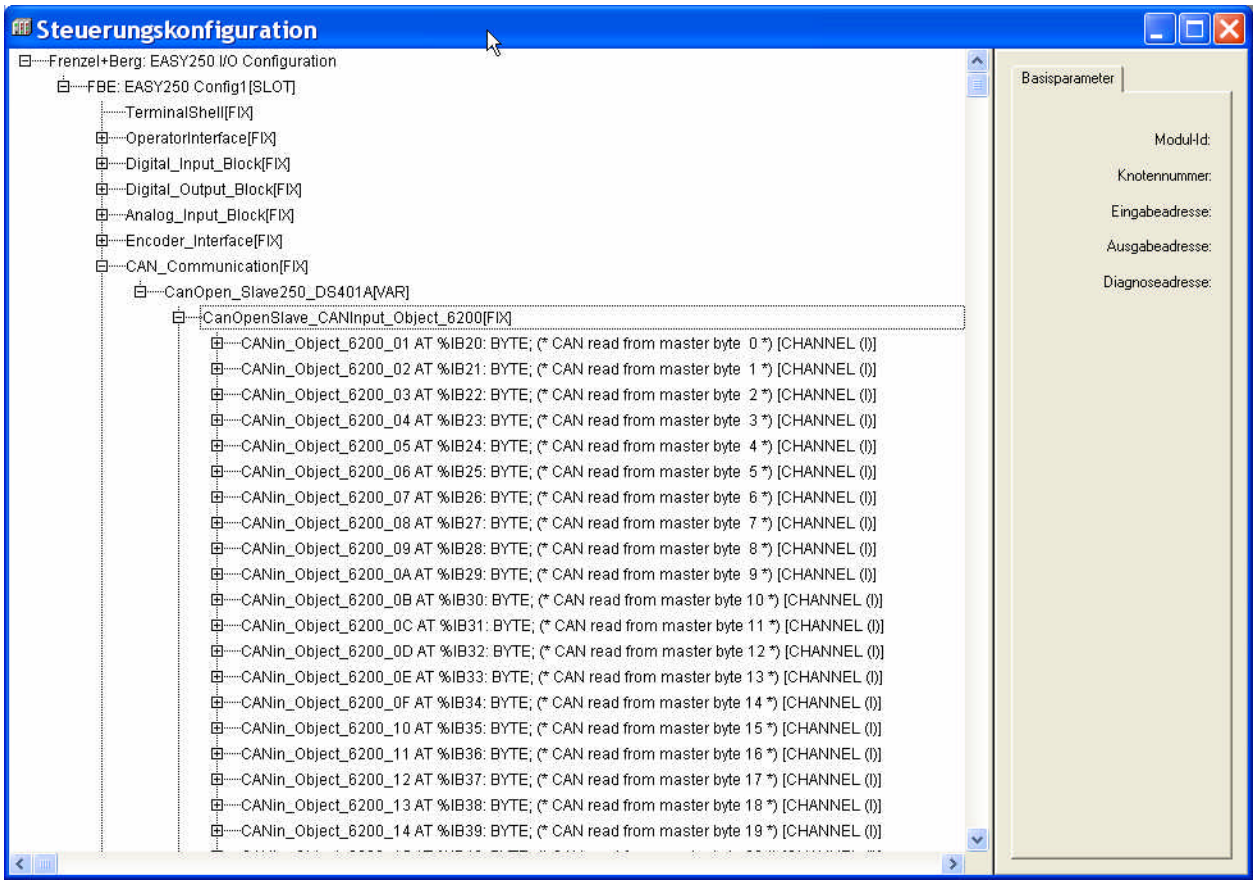
TRUE if function was successfully, else FALSE.

10. CANopen Slave

The run time system includes a CANopen slave module. If selected, the EASYxxx system acts as a CANopen slave according to DS401 (analog and digital I/O module) in a CANopen network.
The Node-ID and baud rate must be selected from the configuration dialog window.



The target system offers a number of digital I/O bytes and analog I/O integers for data transfer purposes. This data transfer memory is implemented as CANopen Object Dictionary area covering the objects 6000, 6200, 6401 and 6411. In order to transmit data from the slave to the master, it is only necessary to write to the variables named as “CANout_Object..”. It is the masters job to map the related data to a PDO and set the transfer mode needed.



11. CANopen Master

The run time system includes a very powerful CANopen master. The CANopen interface is based on a two level software structure. Adding a CANopen master to the PLC configuration activates the lower level according to DS302 level. This layer handles the complete network boot up and PDO transfer automatically.

Configuration:

The configuration of this master is done with the CoDeSys PLC configuration dialog. The master is enabled if there is CANopen master added to the system configuration.
The functionality and the maximum number of slaves depends on the target system.

The following functions are implemented:

For the DS302 implementation there are no special functions required. The complete network startup and PDO data transmission is handled automatically.

12. CAN Interface

The Library FBESysCAN.lib is a Library extension for the CoDeSys PLC runtime system and provides sending and receiving of CAN frames.

The following functions are implemented:

Function name	Description
SYSCAN_INITBASICCAN	Initializes the Basic-CAN-Interface
SYSCAN_RXMSG	Receives a CAN message
SYSCAN_ISRXMSG	Checks whether there are received CAN frames
SYSCAN_TXMSG	Transmits a CAN message

12.1. Data Types

In order to implement the CAN library functions there are several new data types declared in the library FBESysCAN.lib. It is strongly recommended to use this data types with the library functions, even if the replaced constant would be also processed by the CoDeSys compiler without any problems.

type_SYSCAN_CANNODE

Data type to select the requested CAN channel.

```
TYPE SYSCAN_CANNODE : (
        CAN0:=          0,
        CAN1:=          1
);
END_TYPE
```

type_SYSCAN_DIR

Data type to indicate the receive or transmit message.

```
TYPE SYSCAN_DIR : (
        CAN_RXD:=    1,
        CAN_TXD:=    2
);
END_TYPE
```

12.2. SYSCAN_InitBasicCan

Description:

This function initializes the CAN-Channel.

Declaration:

```
FUNCTION SYSCAN_INITNODE : BOOL
VAR_INPUT
        NODE            : SYSCAN_CANNODE;
END_VAR
```

Parameters:

Name	Data-Type	Description
NODE	SYSCAN_CANNODE	CAN Interface Number

Return Value (Data type BOOL)

TRUE If initialization of the CAN channel was successful
FALSE If initialization of the CAN channel failed

12.3. SYSCAN_ISRXMSG

Description:

This function checks whether there is one or more messages in the receive buffer.

Declaration:

```
FUNCTION SYSCAN_ISRXMSG : BOOL
VAR_INPUT
        NODE  : SYSCAN_CANNODE;
END_VAR
```

Parameters:

Name	Data-Type	Description
NODE	SYSCAN_CANNODE	CAN Interface Number

Return Value (Data type BOOL)

TRUE If 1 or more messages available
FALSE If no message abailable

12.4. SYSCAN_RXMSG

Description:

Reads the next CAN message from receiver buffer.

Note:

Each message can only be read for one time from the receiver queue
Reading of the message deletes the message from the FIFO automatically !
This function only works with 11 Bit identifiers

Declaration:

```
FUNCTION_BLOCK SYSCAN_RXMSG
VAR_INPUT
        ENABLE      : BOOL;
        NODE        : SYSCAN_CANNODE;
END_VAR
VAR_OUTPUT
        SUCCESS     : BOOL;
        ID          : UINT;
        DATA        : ARRAY[0..7] OF BYTE;
        LEN         : UINT;
        FLAGS       : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
ENABLE	BOOL	Must be TRUE in order to enable this function block
NODE	SYSCAN_CANNODE	CAN Interface Number
SUCCESS	BOOL	TRUE if the received CAN message was detected as valid CAN frame
ID	UDINT	CAN identifier of this message
DATA	ARRAY[0..7 OF BYTE]	Data bytes of the CAN message
LEN	BYTE	Length of CAN message / received number of data bytes
FLAGS	UINT	Reserved for future use

12.5. SYSCAN_TXMSG

Description:

Stores a CAN message into the transmitter buffer.

Declaration:

```
FUNCTION_BLOCK SYSCAN_TXMSG
VAR_INPUT
        ENABLE      : BOOL;
        NODE        : SYSCAN_CANNODE;
        ID          : UINT;
        DATA        : ARRAY[0..7] OF BYTE;
        LEN         : UINT;
        FLAGS       : UINT;
END_VAR
VAR_OUTPUT
        SUCCESS     : BOOL;
END_VAR
```

Parameters:

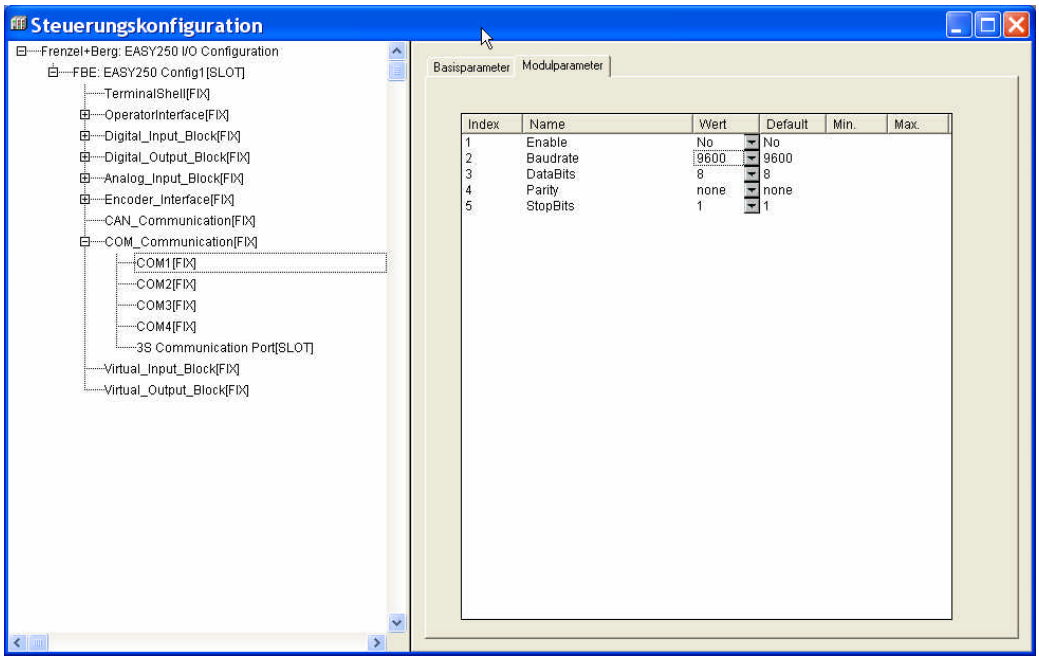
Name	Data-Type	Description
ENABLE	BOOL	Must be TRUE in order to enable this function block
NODE	SYSCAN_CANNODE	CAN Interface Number
ID	UDINT	CAN identifier of this message
DATA	ARRAY[0..7] OF BYTE	Data bytes of the CAN message to send
LEN	BYTE	Number of data bytes to transmit
FLAGS	UINT	Reserved for future use
SUCCESS	BOOL	TRUE if the message content is valid

13. Serial Interface / COM

The Library FBESysSerial.lib is a Library extension for the CoDeSys PLC runtime system to support serial COM interfaces from within IEC61131-3 applications. The COM interfaces are handled by the operation system. For each COM interface there is a transmit and a receive FIFO buffer.

Target Harware	Receive FIFO size	Transmit FIFO size
EASY242	512 Bytes	512 Bytes
EASY2606	512 Bytes	512 Bytes

Configuration of the serial interfaces can be done either by using the CoDeSys System Configuration Dialog, or by using the libraries init function.



The following functions are implemented:

Function name	Description
SYSCOM_INIT	Initialize COM interface
SYSCOM_GETRXBUFNUM	Get number of received characters
SYSCOM_READ	Read single character from the serial interface
SYSCOM_READSTRING	Read string from serial channel
SYSCOM_READBLOCK	Reads len characters from the serial channel
SYSCOM_GETSTATUS	Read status of serial channel

SYSCOM_WRITE	Write single character to the serial channel
SYSCOM_WRITESTRING	Write string to the serial channel
SYSCOM_WRITEBLOCK	Write len characters to the serial channel
SYSCOM_CLEAR	Clears the receiver register and receiver buffer
SYSCOM_CLOSE	Closes a COM interface. Receiver and transmitter will be disabled.
SYSCOM_REOPEN	Opens a COM interface again with last parameters. Receiver and transmitter register and buffer will be cleared.
SYSCOM_ISRXREADY	Checks for characters in the receiver buffer. Returns TRUE if minimum one character is in the receiver buffer.

13.1. Data Types

In order to implement the serial COM library functions there are several new data types declared in the library FBESysSerial.lib. It is strongly recommended to use this data types with the library functions, even if the replaced constant would be also processed by the CoDeSys compiler without any problems.

type_COM_PORT

Data type to select the requested serial channel.

```
TYPE type_COM_PORT : (
        COM1:= 1,
        COM2:= 2,
        COM3:= 3,
        COM4:= 4,
        COM5:= 5,
        COM6:= 6
);
END_TYPE
```

type_COM_BAUD

Data type to set the requested baud rate for the serial channel

```
TYPE TYPE_COM_BAUD : UDINT;
END_TYPE
```

type_PARITY

Data type for setting parity selection for a serial channel

```
TYPE type_COM_PARITY : (
        COM_PARITY_EVEN:= 69,
        COM_PARITY_ODD:= 79,
        COM_PARITY_NONE:= 78
);
END_TYPE
```

type_COM_DATABITS, type_COM_STOPBITS

This data types are direct replacements of data type INT

```
TYPE type_COM_DATABITS : INT;
END_TYPE
TYPE type_COM_STOPBITS :  INT;
END_TYPE
```

13.2. FBE_COM_INIT

Description:

Initializes a COM interface and opens it for data transfer operations. If the user configures the serial channel within the CoDeSys system configuration dialog, there is no need to call the SYSCOM_INIT function.

Declaration:

```
FUNCTION SYSCOM_INIT : BOOL
VAR_INPUT
        COMPORT     : TYPE_COM_PORT;
        BAUD        : TYPE_COM_BAUD;
        DATABITS    : TYPE_COM_DATABITS;
        PARITY      : TYPE_COM_PARITY;
        STOPBITS    : TYPE_COM_STOPBITS;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface
BAUD	TYPE_COM_BAUD	Baudrate for serial data transmission
DATABITS	TYPE_COM_DATABITS	Number of data bits for serial data transmission
PARITY	TYPE_COM_PARITY	Parity information for serial data transmission
STOPBITS	TYPE_COM_STOPBITS	Number of Stop Bits for serial data transmission

Return Value (Data type BOOL)

TRUE If initialization of the serial channel was successful
FALSE If initialization of the serial channel failed

13.3. SYSCOM_GETSTATUS

Description:

Returns the status of a serial COM interface.

Declaration:

```
FUNCTION SYSCOM_GETSTATUS : BYTE
VAR_INPUT
        COMPORT     : TYPE_COM_PORT;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	type_COM_PORT	Number of the serial interface

Return Value (Data type BYTE)

The function SYSCOM_GETSTATUS returns the status of a serial COM interface in a Byte.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TxOVL	RxOVL	-	-	-	TxRDY	TxEM	RxRDY

RxRDY Receiver Ready
1: The COM interface has received one or more characters.
0: There are no received characters stored to the receiver FIFO buffer

TxEM Transmitter empty
1: There are no more characters in the transmission FIFO buffer.
0: There are characters in the transmitter FIFO

TxRDY Transmitter Ready
1: The transmitter FIFO buffer is ready for storing additional characters.
0: The transmitter FIFO buffer is full. Do not start any further transmissions.

RxOVL Receiver Overflow
1: There was an overflow of the receiver FIFO buffer. There are some lost characters.
0: No overflow occurred.
The Overflow flag is reset after reading the status byte using function SYSCOM_GETSTATUS. So this overflow can only be read for one time.

TxOVL Transmitter Overflow
1: There was an overflow of the transmitter FIFO buffer. There are some lost characters.
0: No overflow occurred.
The Overflow flag is reset after reading the status byte using function SYSCOM_GETSTATUS. So this overflow can only be read for one time.

13.4. SYSCOM_GETRXBUFNUM

Description:

Returns the number of characters stored in the receiver FIFO buffer of a serial COM interface.

Declaration:

```
FUNCTION SYSCOM_GETRXBUFNUM : UINT
VAR_INPUT
    COMPORT     : TYPE_COM_PORT;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface

Return Value (Data type UINT)

Number of characters stored in the receiver FIFO buffer.

13.5. SYSCOM_READ

Description:

Read one character from the receiver FIFO buffer.

Declaration:

```
FUNCTION SYSCOM_READ : BYTE
VAR_INPUT
        COMPORT      : TYPE_COM_PORT;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface

Return Value (Data type BYTE)

One character from the receiver FIFO buffer. If there is no received character in the buffer, the function returns “0”.

13.6. SYSCOM_READSTRING

Description:

Read a complete String from the receiver FIFO buffer. The string is either terminated with character ZERO or if the maximum string length is exceeded.

Declaration:

```
FUNCTION SYSCOM_READSTRING : UINT
VAR_INPUT
        COMPORT      : TYPE_COM_PORT;
        STRINGDATA   : STRING;
        MAXLEN       : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface
STRINGDATA	STRING	Destination where the function has to copy the string to.
MAXLEN	UINT	Maximum valid length of this string.

Return Value (Data type UINT)

The function returns the length of the received string in bytes

13.7. SYSCOM_READBLOCK

Description:

Reads Len characters from the serial port to the buffer at Address until end of string or Len is reached.

Declaration:

```
FUNCTION SYSCOM_READBLOCK : UINT
VAR_INPUT
        COMPORT     : TYPE_COM_PORT;
        ADDRESS     : UDINT;
        LEN         : UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface
ADDRESS	UDINT	Destination where the function has to copy the len characters to.
LEN	UINT	Maximum valid length of this string.

Return Value (Data type UINT)

The function returns the number of characters of the received string

13.8. SYSCOM_WRITE

Description:

Writes a single character to the transmitter FIFO buffer.

Declaration:

```
FUNCTION SYSCOM_WRITE : BOOL
VAR_INPUT
        COMPORT     : TYPE_COM_PORT;
        DATA        : BYTE;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface
DATA	BYTE	Data Byte to transmit

Return Value (Data type BOOL)

TRUE If transmission of the data byte was successful
FALSE If transmission of the data byte failed

13.9. SYSCOM_WRITESTRING

Description:

Writes a complete string to the transmitter FIFO buffer. The string must be terminated by a character ZERO.

Declaration:

```
FUNCTION SYSCOM_WRITESTRING : UINT
VAR_INPUT
        COMPORT     : TYPE_COM_PORT;
        STRINGDATA  : STRING;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface
STRINGDATA	STRING	String Data to transmit

Return Value (Data type UINT)

The function returns the length of the transmitted string in bytes

13.10. SYSCOM_WRITEBLOCK

Description:

Writes Len characters from a Address until end of string or Len is reached to the buffer of serial port.

Declaration:

```
FUNCTION SYSCOM_WRITESTRING : UINT
VAR_INPUT
        COMPORT      : TYPE_COM_PORT;
        STRINGDATA   : STRING;
        LEN          :UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface
STRINGDATA	STRING	String Data to transmit
LEN	UINT	Maximum valid length of this string.

Return Value (Data type UINT)

The function returns the number of the transmited characters

13.11. SYSCOM_CLEAR

Description:

Clears the receiver register and receiver buffer.

Declaration:

```
FUNCTION SYSCOM_CLEAR : BOOL
VAR_INPUT
        COMPORT      : TYPE_COM_PORT;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface

Return Value (Data type BOOL)

TRUE If function was successful
FALSE If execution of this function failed

13.12. SYSCOM_CLOSE

Description:

Closes a COM interface. Receiver and transmitter will be disabled.

Declaration:

```
FUNCTION SYSCOM_CLOSE : BOOL
VAR_INPUT
        COMPORT      : TYPE_COM_PORT;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface

Return Value (Data type BOOL)

TRUE If function was successful
FALSE If execution of this function failed

13.13. SYSCOM_REOPEN

Description:

Opens a COM interface again with last parameters. Receiver and transmitter register and buffer will cleared. The functions SYSCOM_CLOSE and SYSCOM_REOPEN may be used to block serial reception for some time.

Declaration:

```
FUNCTION SYSCOM_REOPEN : BOOL
VAR_INPUT
        COMPORT      : TYPE_COM_PORT;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface

Return Value (Data type BOOL)

TRUE If function was successful
FALSE If execution of this function failed

13.14. SYSCOM_ISRXREADY

Description:

Check for characters in the receiver buffer. Returns TRUE if minimum one character is in the receiver buffer.

Declaration:

```
FUNCTION SYSCOM_ISRXREADY : BOOL
VAR_INPUT
        COMPORT      : TYPE_COM_PORT;
END_VAR
```

Parameters:

Name	Data-Type	Description
COMPORT	TYPE_COM_PORT	Number of the serial interface

Return Value (Data type BOOL)

TRUE If one or more character is/are in receiver buffer.
FALSE If receiver buffer is empty.

14. Keyboard click simulation

The Library FBESysKeyboard.lib is a Library extension for the CoDeSys PLC runtime system. It simulates a key click in the form of adding characters to the keyboard buffer.

Key-codes of PC-keyboard:

All ASCII-keys (A .. Z, a..z, 0 .. 9, %, §, RETURN, SPACE, and so on) are translated to their associated ASCII-code-value (A = 65, 1=49, RETURN=13, ESC=27 and so on).
All function-key (F1, F2, …, F12) are translated to hex values (16#F1, 16#F2, …, 16#FC).

The other key-codes are translated as shown in the table below:

Key	Code	Key	Code
Backspace	16#08	Function Key F1	16#F1
Escape	16#1B	…	
		Function Key F10	16#FA

The following functions are implemented:

Function name	Description
SYSKBD_SIMKEYCLICK	Adds characters to the keyboard buffer

14.1. SYSKBD_SIMKEYCLICK

Description:

Simulates a key click in form of a adding character to the keyboard buffer.

Declaration:

```
FUNCTION SYSKBD_SIMKEYCLICK : BOOL
VAR_INPUT
        KEYCODE: UINT;
END_VAR
```

Parameters:

Name	Data-Type	Description
KEYCODE	UINT	ASCII-Code of the character you want to add to the keyboard buffer

Return Value (Data type BOOL)

The function returns TRUE if writing the character to the keyboard buffer was successful.

15. LCD Touch Tool

The Library FBESysVisuTools.lib is a Library extension for the CoDeSys PLC runtime system.
This function reads the Position of a cursor on a touch display by repeating the coordinates an if it was clicked on this position.

The following functions are implemented:

Function name	Description
SYSVISUTOOL_TOUCHPOS	Returns the position of a cursor and if it was clicked on this position

15.1. SYSVISUTOOL_TOUCHPOS

Description:

Returns the x,y coordiantes of a cursor on a touch display and if it was clicked

Declaration:

```
FUNCTION_BLOCK SYSVISUTOOL_TOUCHPOS
VAR_OUTPUT
        XPOS        : UINT;
        YPOS        : UINT;
        CLICKED     : BOOL;
END_VAR
```

Return Value

Name	Data-Type	Description
XPOS	UINT	X coordinate of cursor on touch display
YPOS	UINT	Y coordinate of cursor on touch display
CLICKED	BOOL	Returns TRUE if it was clicked

16. Alpha numeric LCD Tool

The Library FBESysTerminal1.lib is a Library extension for the CoDeSys PLC runtime system.
This function writes characters or strings to an alpha numeric LCD module. Also it can define the mode or position of the cursor.

Attention: This library is **NOT** available for **EASY2606**!

The following functions are implemented:

Function name	Description
FBE_LCD_CURSORBLINK	Enable cursor blinking
FBE_LCD_CURSOMOVETO	Set Cursor to new position on alpha numeric LCD module
FBE_LCD_CURSOROFF	Hide cursor on alpha numeric LCD module
FBE_LCD_CURSORON	show cursor on alpha numeric LCD module
FBE_LCD_DELAY	
FBE_LCD_INIT	Initialises an alpha numeric LCD module
FBE_LCD_READY	
FBE_LCD_WAITUNTILREADY	
FBE_LCD_WHEREX	Returns the horizontal position of the cursor
FBE_LCD_WHEREY	Returns the vertical position of the cursor
FBE_LCD_WRCHAR	Writes a single character to the alpha numeric LCD module
FBE_LCD_WRSTRING	Writes a string to the alpha numeric LCD module