

IFI CAN_XL IP

User Guide and Register Map

Core Version:	0x04165B10
Document Version:	1.0
Document Date:	04. 2022

Important Notes XL

- *In April 2012 BOSCH released the revision 1.0 of the CAN FD standard
 - for flexible Data Rate todo*
- *This standard will be converted to ISO 11898-1 (hopefully)*
-
- *The IFI CAN-XL IP core is based on the IFI CAN-FD IP-core revision 0x09115B15
passed the ISO CAN conformance tests on December 13th, 2017
According to "ISO 16845-1:2016 Road vehicles - Controller area network (CAN) - Conformance test plan" and C&S
enhancement/corrections according to "CAN CONFORMANCE TESTING Test Specification C&S Version r3d08"*
- *XL added*
- *XL differences to FD marked in **RED***
-

Important Notes FD

- *In April 2012 BOSCH released the revision 1.0 of the CAN FD standard
 - for flexible Data Rate*
- *In 2014 this standard was converted to ISO 11898-1*
- *For improving the residual error-rate there are a few changes included into the ISO standard, which make both versions incompatible to one another
The older standard is called non-ISO, the newer ISO*
- *To give our customers the possibility to switch between both standards we organized the registers in a way that
 - customers can use the old register map for the (old) BOSCH standard called „non-ISO“ and
 - the new register map for the new „ISO“ standard (signed with § in this document)*
- *To switch between both standards customers have to set or clear the ena_ISO bit in the CMD register
 - ! before ! writing all the control words*
- *In February 2015 we added a new Configuration Bit Bit 26, so customers can switch the Bittiming independent from the ena_ISO bit*
- *ALTERA® is now part of Intel®, so references to Altera in this document should be read as references to Intel*
- *The IFI CAN-FD IP-core revision 0x09115B15
passed the ISO CAN conformance tests on December 13th, 2017
According to "ISO 16845-1:2016 Road vehicles - Controller area network (CAN) - Conformance test plan" and C&S enhancement/corrections according to "CAN CONFORMANCE TESTING Test Specification C&S Version r3d08"*
- *In February 2018 we added additional IRQs (signed with # in this document)*

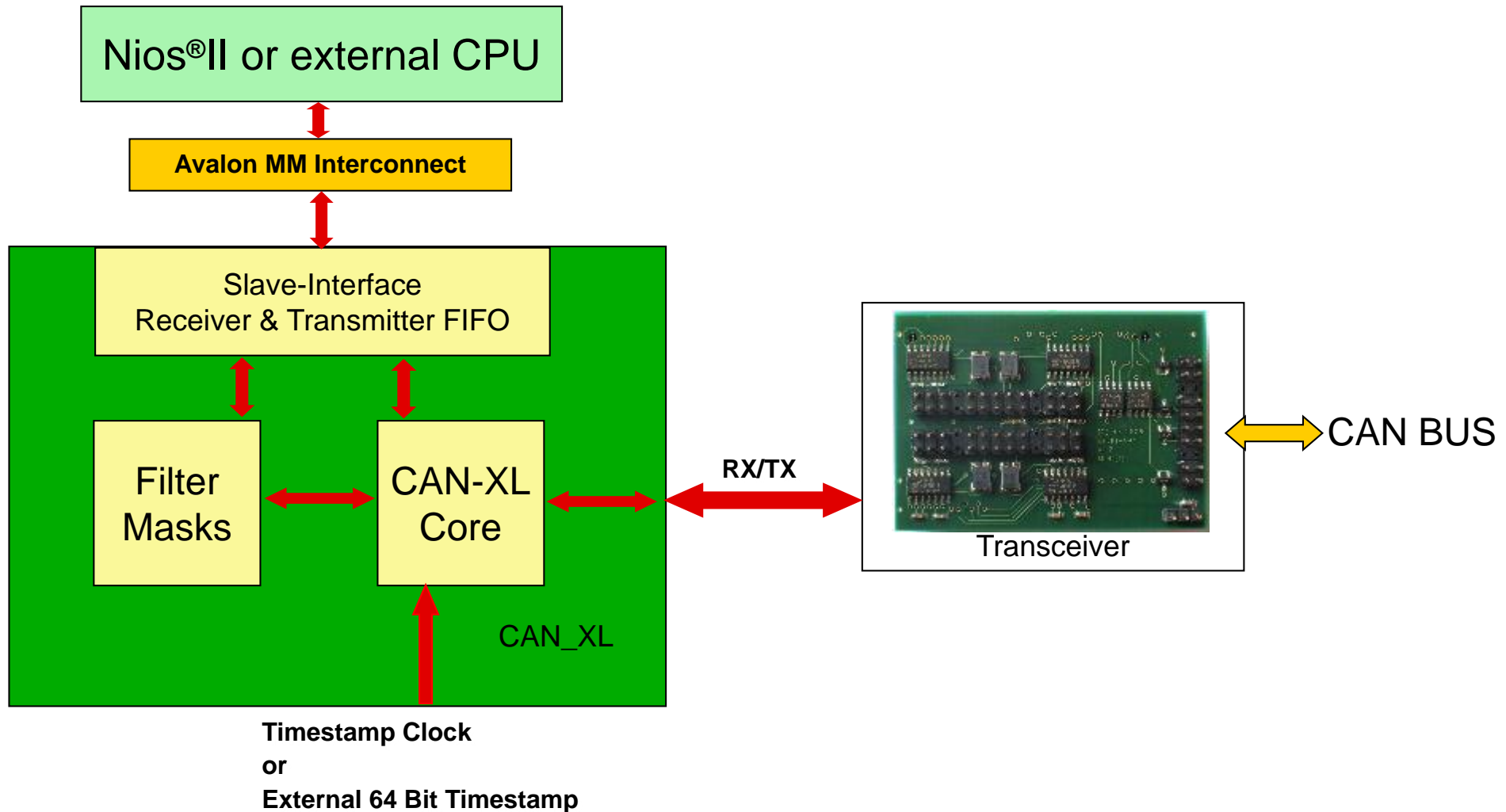
Contents

- *Overview*
- *Installation*
- *Integrating the core using QSYS or Platform Designer*
- *Reference Designs*
- *Using the Core without Nios/QSYS*
- *License Agreement*
- *Register Map*

Overview

- *Block Diagram*
- *Feature List*
- *Intel Implementation*
- *OpenCore Plus Feature*
- *Reference Design*
- *Using SignalTap II*
- *Pricing*
- *References*
- *CAN Background*
- *Contacting Technical Support*

Block Diagram



IFI CAN_FD Feature List

- CAN 2.0B
 - Standard or Extended Identifier
 - Remote Frames
 - Error-Handling
- CAN FD ISO and non-ISO
 - up to 64 Byte Data
 - Flexible Data rate
- Separate Message FIFO for Receive / Transmit
 - each can be configured 4, 8, 16, 32 or 64 kBytes
 - dynamic size for each message to save memory resources
- 256 Message Filters
 - Every Message Filter contains one MASK- and one Identifier Register
- NIOS Interface
 - Example software included
 - HAL drivers for NIOS II included
- Separate Clock for CAN and Avalon-IF possible
- One High Priority Message (**Classic or FD Message only**)
- Bus-Statistic possible
- 32 Bit Timestamp (64 bit when Timestamp from external):
 - for received messages
 - possible for transmitted messages with frame number other than 0
- For external CPU support :
 - 8,16, 32 or 64 Bit Interface
- Reading of the compile time parameters possible

IFI CAN_XL Feature List (in addition to FD)

- CAN XL
 - todo

IFI Feature Details

- **Timestamp**
 - “OFF” disabled Timestamp, saves about 100 LE
 - “ON_internal” Timestamp with 1 us resolution, TriggerPosition selectable SOF or ACK
 - “ON_External” External 64 bit Timestamp used
 - “ON_with_ExternalControl” for Clock and Reset for TimestampCounter, additional outputs for custom Timestamp

- **Dual_Clock**
 - “OFF” system-clock is used for CAN-timing, saves about 350 LE, CAN-timing may be limited because of the usually slower system-clock
 - “ON” separate Clock for system and CAN-timing, means with a faster CAN-Clock a higher Fast Baudrate can be achieved, optimal Clockrate for CAN-clock can be selected depending on Baudrate in external PLL

- **Bus_statistic**
 - “OFF” disabled Bus_statistic, saves about 300 LE
 - “ON” selectable Bus_statistic value in Total Frames or Bus_load in %

- **Filter with 256 Mask+Identifier entries**
 - “OFF” disabled filter, saves about 200 LE and 2 M9k
 - “ON” filter for ID, 11bit only ID, 29bit only ID, CAN-FD only ...

IFI Feature Details

■ ID Format

- “IFI_legacy”, identical to IFI CAN IP and IFI Advanced CAN IP
- “CANalyzer”, identical to Vector Informatik CANalyzer
- “ID_other”, 3rd ID format used in the CAN market
-

■ Additional Errorcounters for CAN_FD debug

- count errors during arbitration phase if transmitter (slow Baudrate)
- count errors during data phase if transmitter (fast Baudrate)
- count errors during arbitration phase if receiver (slow Baudrate)
- count errors during data phase if receiver (fast Baudrate)

■ PWM

- support for SIC Transceivers in FAST Mode
- when using FAST Mode these Transceivers need to be driven by a programmable PWM
- for switching between SIC Mode (dominant/recessive) (like CAN transceivers)
- and TX Node push/pull, RX node adjust threshold
- for more documentation
 - search the web for “CAN XL fast sic transceivers”

IFI Additional Feature Details

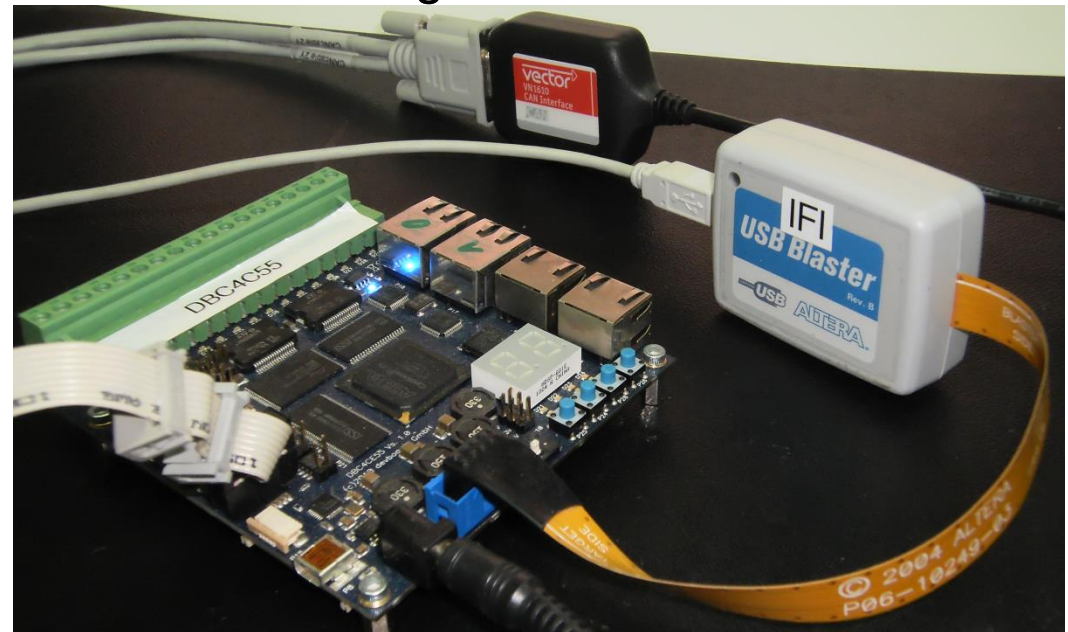
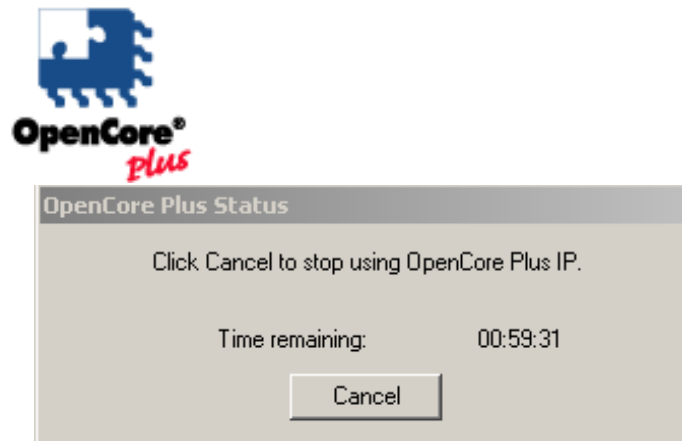
- Programmable Transmission Suspend
 - 0 => no suspend, start next transmission as soon as valid (default)
 - <value> => suspend next transmission after a successful transmission or after abort when maximum repeat count is reached in <value> microseconds
 - global value or per individual message
- Programmable Transmission RepeatCount
 - 0 => endless repeat of transmitted message until successful transmission (default)
 - 1 => just one try => single shot mode, no repeat after error
 - 2 => repeat ones, in case first try was not successful
 - <value> => make maximum <value> attempts to transmit message
 - global value or by individual message
- Bus_monitoring Mode
 - CAN_FD spec 3.3.1
- Restricted Operation Mode
 - CAN_FD spec 3.3.2
- Hard-Reset
 - CAN-core can be reseted hard

IFI CAN_XL Intel Implementation

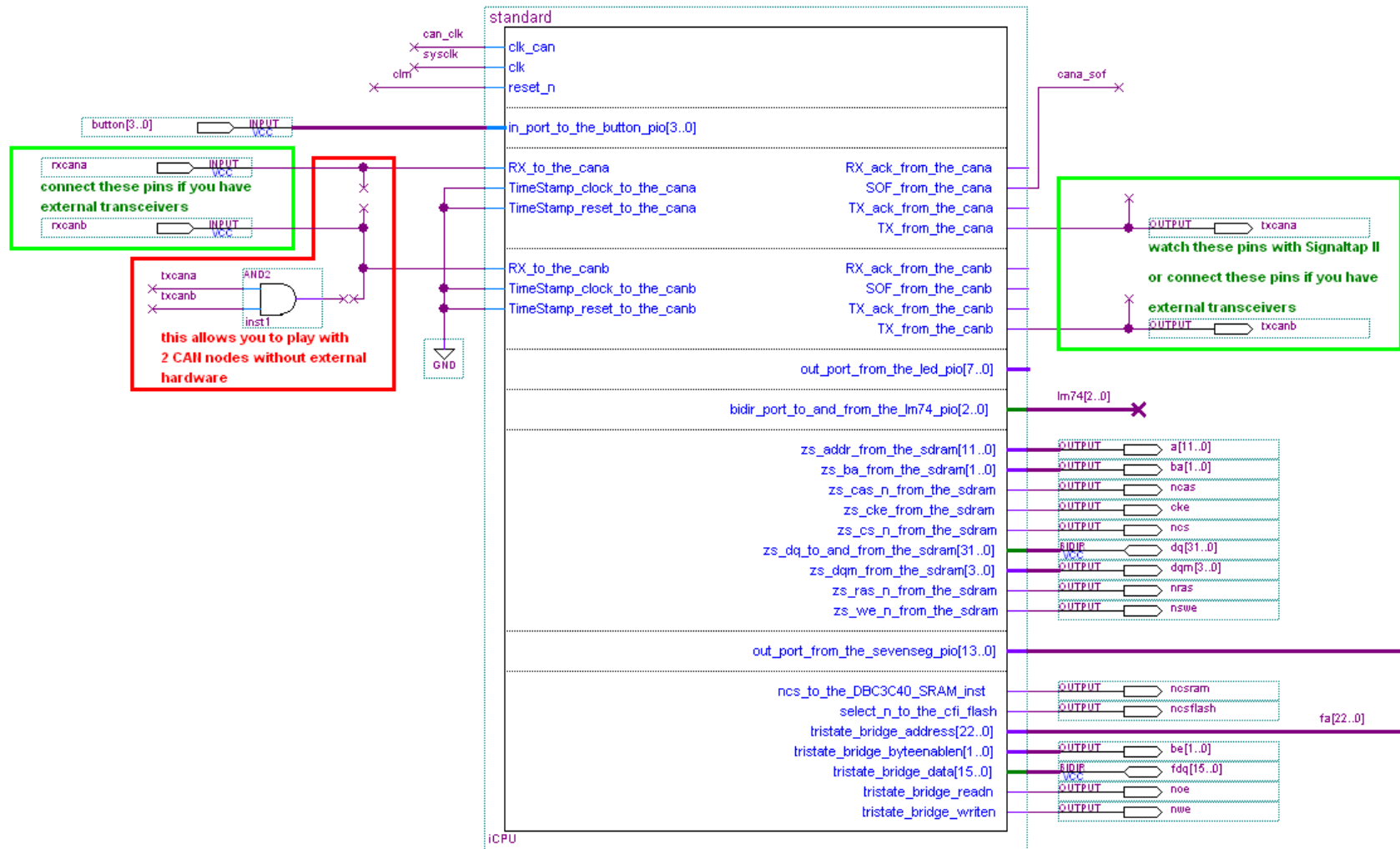
- Design flows supported by
 - QSYS Interconnect
 - Platform Designer
- Device families targeted
 - CYCLONE V
 - ARRIA 10, CYCLONE 10 GX
 - STRATIX 10
- Resources depend on the QuartusII version, compiler settings and options
- Device resource utilization CYCLONE V
 - about 2700 .. 3200 needed ALMs
 - minimum without filter 14 M10k
 - minimum with filter 16 M10k
 - maximum depend on the user setting Fifo size
- Device resource utilization ARRIA 10, CYCLONE 10 GX
 - about 2700 .. 3400 needed ALMs
 - minimum without filter 7 M20k
 - minimum with filter 9 M20k
 - maximum depend on the user setting Fifo size
- Device resource utilization STRATIX 10
 - about 2900 .. 3900 needed ALMs
 - minimum without filter 10 M20k
 - minimum with filter 12 M20k
 - maximum depend on the user setting Fifo size

OpenCore Plus Feature

- Test the CAN_XL Module on your board or on development boards from INTEL or devboards.de
 - There is no time limit with an established connection between the device and the Quartus programmer.
 - If you remove the connection the time remaining is ~ 1 hour.

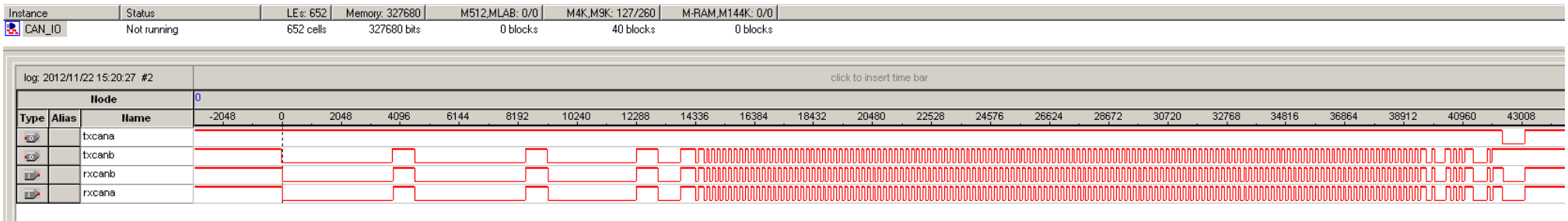


IFI CAN_XL Reference Design



■ Test the IFI CAN_XL with / without external hardware

IFI CAN_XL Reference Design



- Use Signaltap II to watch the TX + RX pins

IFI CAN-XL Pricing (encrypted netlist) ?

- Node-Locked License: please contact IFI
 - 1 year maintenance included
 - T-Guard or NIC-ID Fixed Node
 - Maintenance: please contact IFI
- Floating License: please contact IFI
 - 1 year maintenance included
 - Single or Multi Server
 - Maintenance: please contact IFI
- Licensing:
 - Unlimited (no timelimit) License, Multiproject,
 - Royalty Free with IFI
 - The CAN-NETWORK-PROTOCOL-License is not included
 - => Available by Bosch
 - search [www](#) for "Bosch_CAN_Protocol_License_Conditions"

IFI CAN-XL References

- Hardware tested on
 - **todo**

CAN-Background

■ CAN Messages:

- Every CAN message consists of a certain number of bits that are divided into fields. There are fields such as the Arbitration Field, the Data Field, CRC, the End of Frame...
- The Arbitration Field is different for CAN 2.0 A and CAN 2.0 B messages. It's a logical address with 11 bits for CAN 2.0 A and 29 bits for CAN 2.0 B. The lowest value is the highest priority = 0.
- The Data Field contains the application data of the message with 0 to 64 bits (0 to 8 bytes) (0 to 64 bytes with CAN-FD).
- With exception of the CRC delimiter, the ACK field and the EOF, the bits are stuffed. That means, 5 consecutive bits with identical value are followed by a complementary bit.
- The Error Frame and the Overload Frame are of a fixed form and not coded with bit stuffing.
- For XL additional features are implemented.

■ Error Detection:

- The error management unit is able to detect five different error types.
 - Bit Error
 - Bit Stuffing Error
 - CRC Error
 - Form Error
 - ACK Error

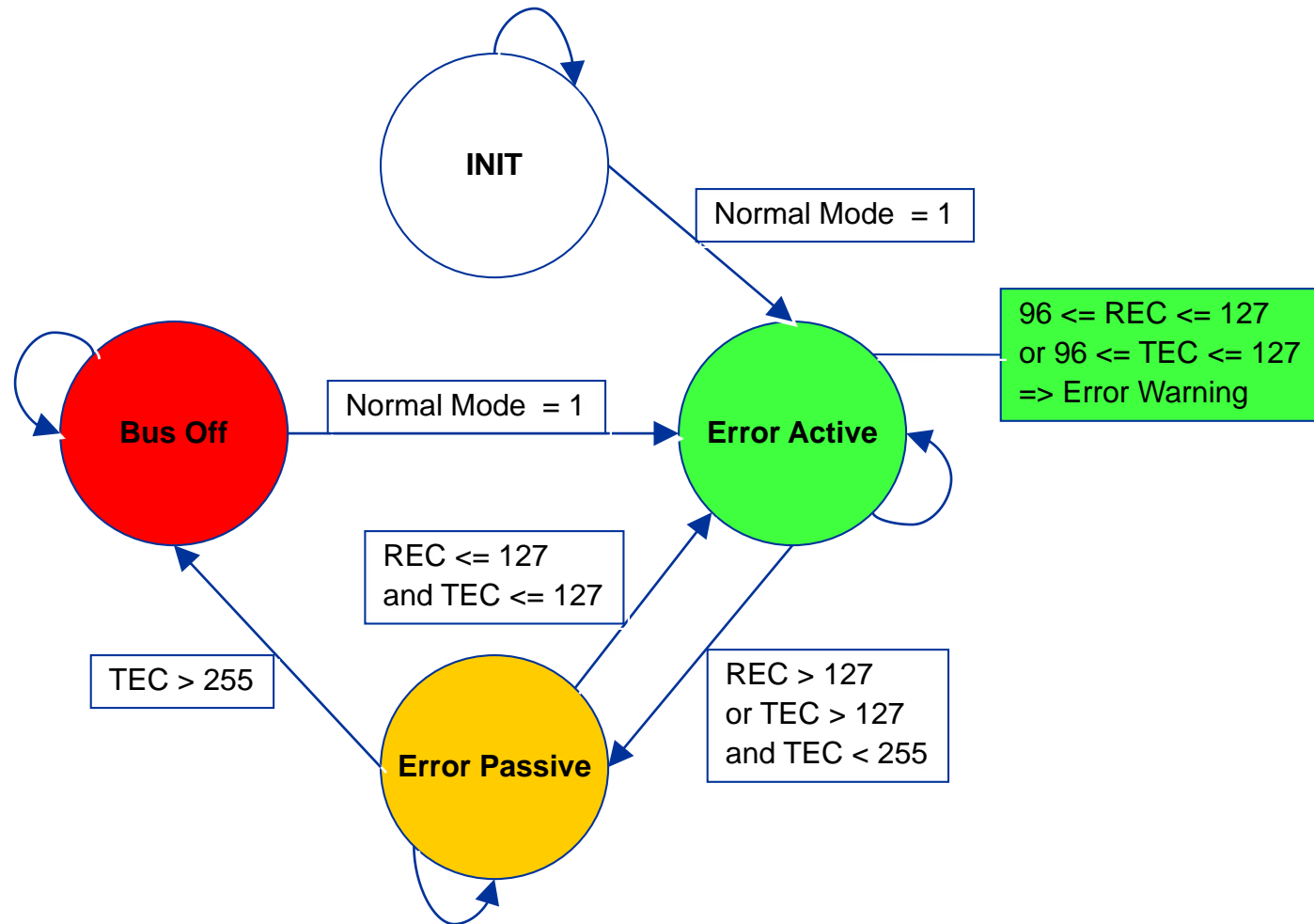
■ Error Handling:

- Error detected
- Transmitting of error frame
- Message will be discarded
- Error counters are incremented
- Transmission will be repeated

■ Error Limitation:

- To prevent a permanently disturbed bus each CAN controller has three error states.
 - Error Active
 - Error Passive
 - Bus Off

CAN-Error States



Contacting Technical Support

Although we have made every effort to ensure that this QSYS OpenCore Package works correctly, there might be problems that we have not encountered. If you have a question or problem that is not answered by the information provided in this README file, please contact the IP Vendor or Intel (Altera is now part of Intel)

For questions about the core's features, functionality and parameter settings please contact:

IFI Ingenieurbüro Für Ic-Technologie
Franz Sprenger
Am Tannenbergr 28
97877 Wertheim
Germany
Phone: (+49) 9342 91091
E-Mail: ifi@ifi-pld.de
<http://www.ifi-pld.de>



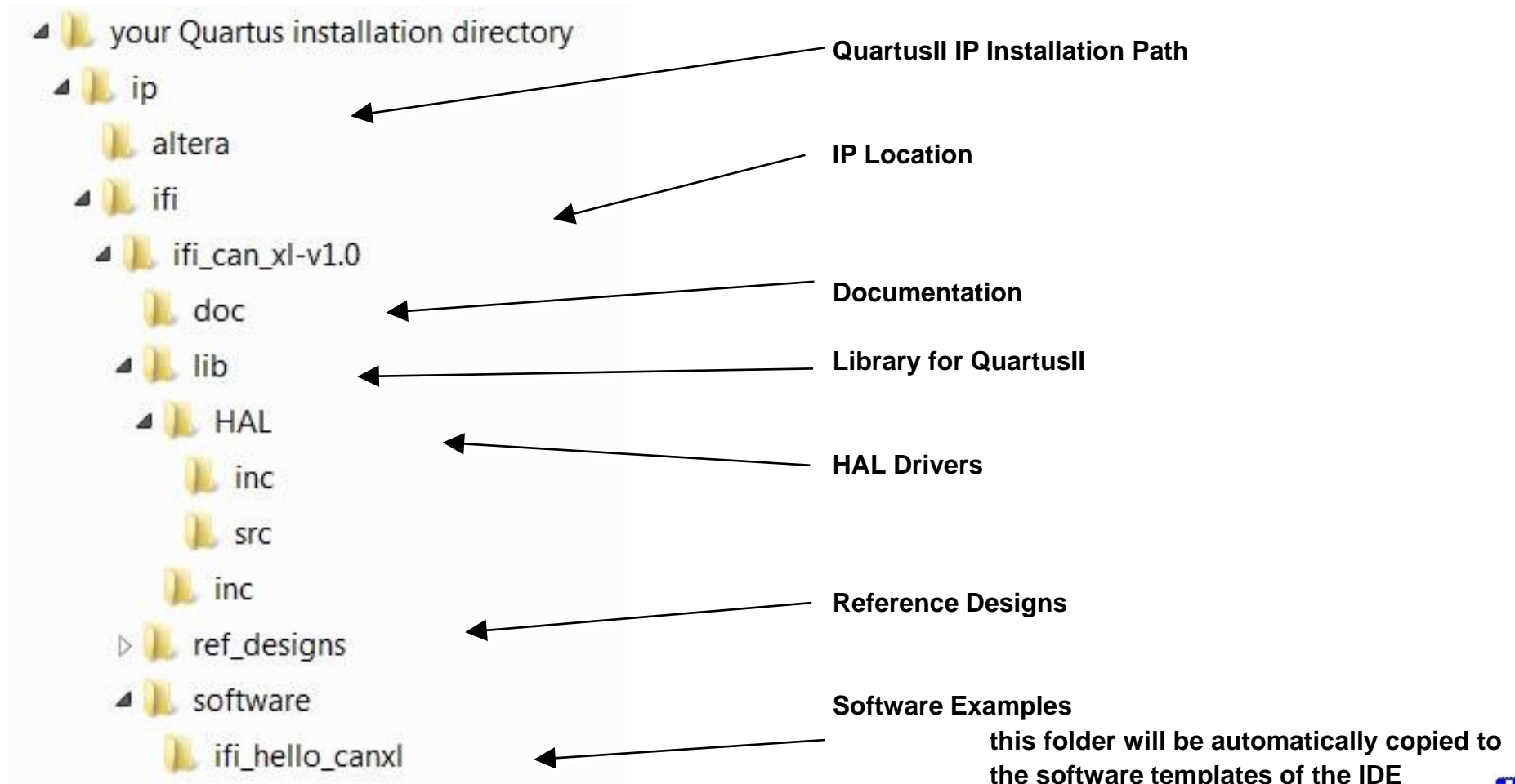
Installation

Install the IFI CAN_XL

- Before you can start using IFI CAN_XL functions, you must install the IFI CAN_XL files on your computer. The following instructions describe this process.
- **Close QuartusII and IDE / SBT**
- **The installed Quartus II version has to be 9.1 or a newer version**
- **Install the IFI CAN_XL files**
 - The following instructions describe how you install IFI CAN_XL on computers running the Windows, Linux or Solaris operating systems.
 - If you don't change the installation path, the Platform designer and QSYS will find the IP automatically
 - Windows
 - Follow these steps to install the IFI CAN_XL on a PC running a supported version of the Windows operating system:
 - Choose Run (Windows Start menu).
 - Type <path name>\<filename>.exe, where <path name> is the location of the downloaded IFI CAN_XL function and <filename> is the filename of the IFI CAN_XL function.
 - Click OK. The IFI CAN_XL Installation dialog box appears. Follow the on-screen instructions to finish installation.
 - Solaris & Linux
 - Follow these steps to install the IFI CAN_XL on a computer running supported versions of the Solaris and Linux operating systems:
 - Decompress the package by typing the following command:
 - `gzip -d<filename>.tar.gz`
 - where <filename> is the filename of the IFI CAN_XL function.
 - Extract the package by typing the following command:
 - `tar xvf <filename>.tar`

QSYS OpenCore Package

The QSYS OpenCore Package contains all files required for plug-and-play integration of this core into Altera's QSYS or Platform Designer tool, allowing the user to easily evaluate the core within his Avalon-based system.



Licensing

■ OpenCorePlus License

The download of this IP-core is completed by an individual OpenCorePlus license named `license_CAN_XL_ocp.dat` or similar

When the FEATURE line from this license is appended to the user's Quartus II license file, the encrypted files can be read into Quartus II and place and route can be performed.

The license permits the generation of `<revision_name>_time_limited.sof` files.

The hardware evaluation feature will run during you have an established JTAG connection between your board and the QuartusII programmer or SignalTap. If you close the programmer it will stop working immediately. If you remove the connection it will stop working after 1 hour.

(Refer to the messages created by the programmer)

■ Full License

If you purchased a FULL LICENSE you receive an additional license file, `license_???.dat`.

Use this instead of the `license_ocp.dat`. When the FEATURE line from this license is appended to the user's Quartus II license file, the encrypted files can be read into Quartus II and place and route can be performed. The license permits the generation of `<revision_name>.pof` files and gate-level simulation netlists.

- One FEATURE line can span more than one line

Set Up Licensing

- To install your license, you can either append the license to your **license.dat** file or you can specify the IFI CAN_XL's **license_ocp.dat** file in the Quartus II software.
 - Before you set up licensing for the IFI CAN_XL, you must already have the Quartus II software installed on your computer with the licensing set-up.
- **Append the license to your license.dat file**
 - To append the license, follow these steps:
 - Open the IFI CAN_XL license file in a text editor.
 - Open your Quartus II **license.dat** file in a text editor.
 - Copy all lines from the license file and paste it into the Quartus II license file.
 - Do not delete any FEATURE lines from the Quartus II license file.
 - Save the Quartus II license file.
 - When using editors such as Microsoft Word or Notepad, ensure that the file does not have extra extensions appended to it after you save (e.g., **license.dat.txt** or **license.dat.doc**). Verify the filename in a DOS box or at a command prompt. Also, make sure that the file is saved in plain-text format without formatting characters.
- **Specify the license file in the Quartus II software**
 - To specify the IFI CAN_XL license file in Quartus II, follow these steps:
 - Altera recommends that you give the file a unique name, e.g., *<core name>*_license.dat.
 - Run the Quartus II software.
 - Choose **License Setup** (Tools menu). The **Options** dialog box opens to the **License Setup** page.
 - In the **License file** box add a semicolon to the end of the existing license path and filename.
 - Type the path and filename of the IFI CAN_XL function license file after the semicolon.
 - Do not include any spaces either around the semicolon or in the path/filename.
 - Click **OK** to save your changes.



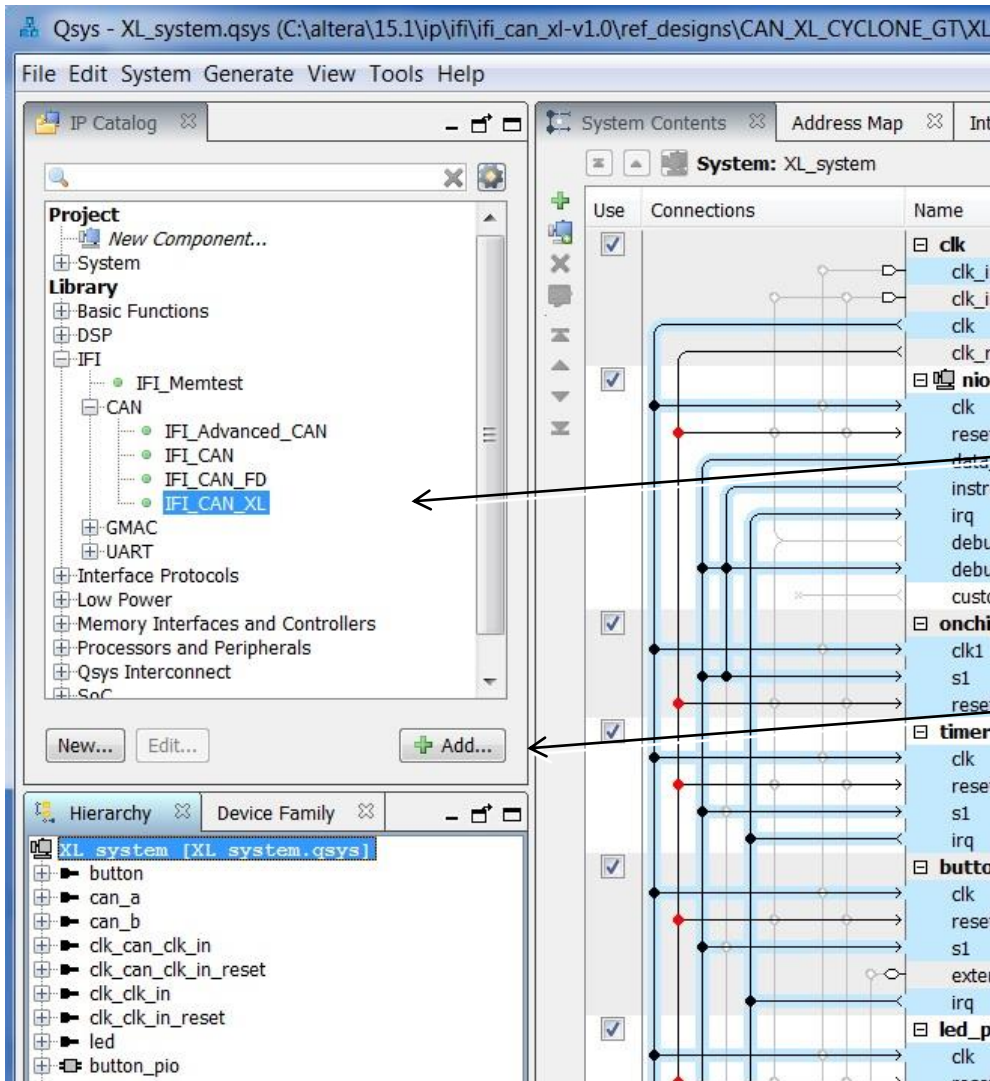
Integrating the Core using QSYS Interconnect / Platform Designer

Adding the Core to your QSYS System
About
Documentation
Parameterize

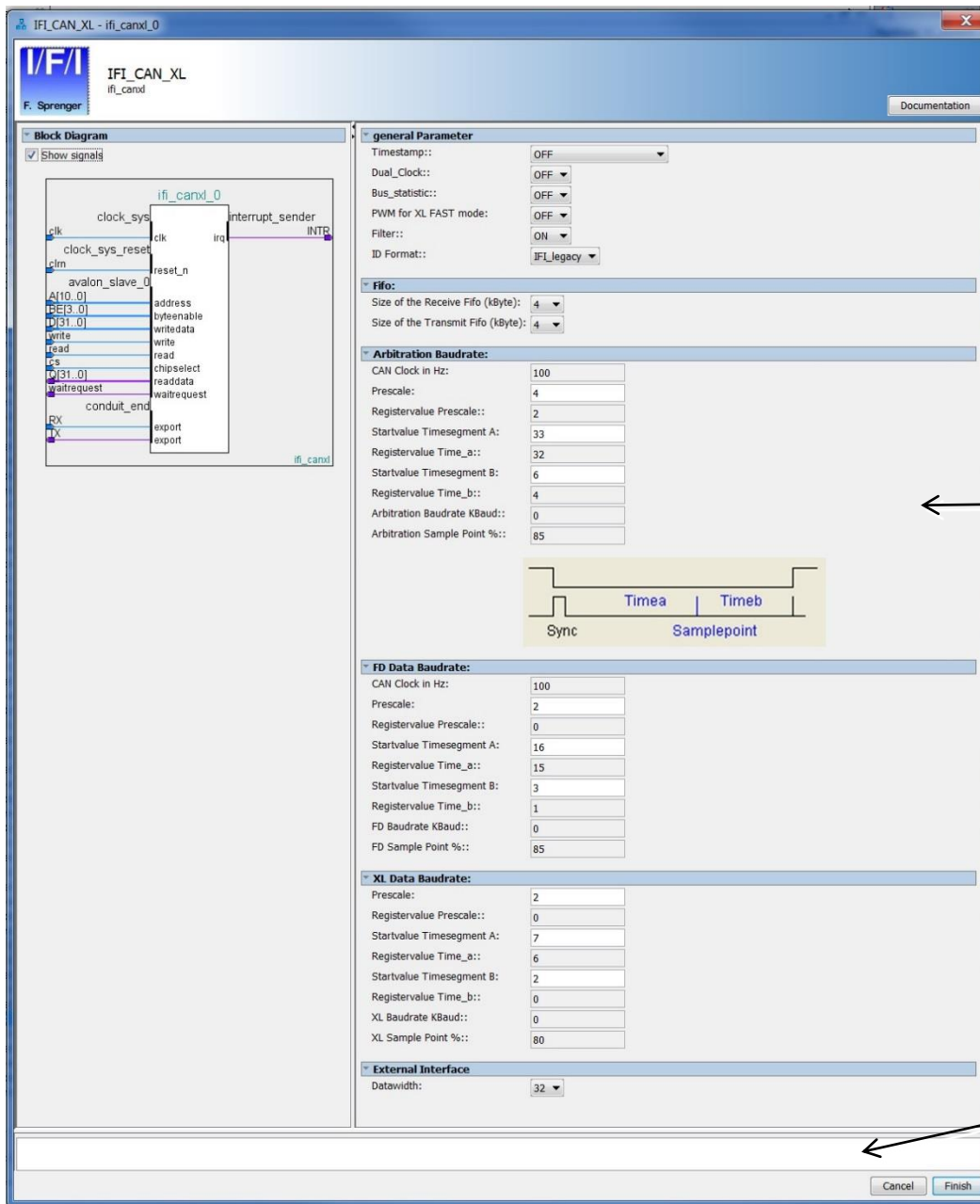
Integrating the Core with your System using QSYS

- This section contains instructions on the following:
 - Adding the core to your system
 - Running the reference design
- These instructions assume that the user is familiar with the
 - Intel OpenCore evaluation process,
 - Intel's Quartus II development software,
 - and the Altera QSYS Interconnect tool
- For more information on these prerequisites, please visit www.intel.com

Adding the Core to your QSYS System



- Launch QSYS Tool from Quartus II (Tools menu).
- Select the core by clicking on the core name
- Click "Add" to add the core to your system.



Info + Documentation

Parameters

Information

IFL CAN_XL - ifl_cand_0

IFL CAN_XL

ifl_cand

F. Springer

Documentation

Block Diagram

☒ Show signals

general Parameter

Timestamp:: OFF

Dual_Clock:: OFF

Bus_statistic:: OFF

PWM for XL FAST mode: OFF

Filter:: ON

ID Format:: IFL legacy

Fifo:

Size of the Receive Fifo (kByte): 4

Size of the Transmit Fifo (kByte): 4

Arbitration Baudrate:

CAN Clock in Hz: 100

Prescale: 4

Registervalue Prescale:: 2

Startvalue Timesegment A: 33

Registervalue Time_a:: 32

Startvalue Timesegment B: 6

Registervalue Time_b:: 4

Arbitration Baudrate KBaud:: 0

Arbitration Sample Point %:: 85

FD Data Baudrate:

CAN Clock in Hz: 100

Prescale: 2

Registervalue Prescale:: 0

Startvalue Timesegment A: 16

Registervalue Time_a:: 15

Startvalue Timesegment B: 3

Registervalue Time_b:: 1

FD Baudrate KBaud:: 0

FD Sample Point %:: 85

XL Data Baudrate:

Prescale: 2

Registervalue Prescale:: 0

Startvalue Timesegment A: 7

Registervalue Time_a:: 6

Startvalue Timesegment B: 2

Registervalue Time_b:: 0

XL Baudrate KBaud:: 0

XL Sample Point %:: 80

External Interface

Datwidth: 32

IFL CAN_XL Documentation

IFL CAN_XL

Name ifl_cand

Version 1.0

Author IFL

Description IFL CAN_XL - v1.0

Group IFL/CAN

Data Sheet [file:///C:/altera/15.1/ip/ifi/ifi_can_xl-v1.0/doc/IFL_CANXL.pdf](#)

general Parameter

Timestamp: OFF to save (about 100 LE) FPGA resources

Dual_Clock: OFF to save (about 350 LE) FPGA resources

Bus_statistic: OFF to save (about 300 LE) FPGA resources

PWM for XL FAST mode: OFF to save (about 100 LE) FPGA resources

Filter: OFF to save (about 200 LE 2 M9k) FPGA resources

ID Format: IFL legacy (like IFL CAN_IP, IFL Advanced CAN IP), CANalyzer format, ID_other: TBD

Fifo:

Size of the Receive Fifo (kByte) 4, 8, 16, 32 or 64 kByte

Size of the Transmit Fifo (kByte) 4, 8, 16, 32 or 64 kByte

Arbitration Baudrate:

CAN Clock in Hz Your CAN Clock in Hz

Prescale 2 .. 511

Registervalue Prescale: Registervalue Prescale

Startvalue Timesegment A 2 .. 511

Registervalue Time_a: Registervalue Time_a

Startvalue Timesegment B 2 .. 255

Registervalue Time_b: Registervalue Time_b

Arbitration Baudrate KBaud: Resulting Arbitration Baudrate

Arbitration Sample Point %: Resulting Sample Point in %

FD Data Baudrate:

CAN Clock in Hz Your CAN Clock in Hz

Prescale 2 .. 511

Registervalue Prescale: Registervalue Prescale

Startvalue Timesegment A 2 .. 255

Registervalue Time_a: Registervalue Time_a

Startvalue Timesegment B 2 .. 255

Registervalue Time_b: Registervalue Time_b

FD Baudrate KBaud: Resulting FD Baudrate

FD Sample Point %: Resulting Sample Point in %

XL Data Baudrate:

Prescale 2 .. 511

Registervalue Prescale: Registervalue Prescale

Startvalue Timesegment A 2 .. 255

Registervalue Time_a: Registervalue Time_a

Startvalue Timesegment B 2 .. 255

Registervalue Time_b: Registervalue Time_b

XL Baudrate KBaud: Resulting XL Baudrate

XL Sample Point %: Resulting Sample Point in %

External Interface

connected to a clock output

ust be connected to a reset source

is exported, or connected to a matching conduit.

Parameters

System: XL_system Path: cana

IFI
F. Sprenger

IFI_CAN_XL
ifi_canxl

general Parameter

Timestamp:: ON_internal

Dual_Clock:: ON

Bus_statistic:: ON

PWM for XL FAST mode: OFF

Filter:: ON

ID Format:: IFI_legacy

Fifo:

Size of the Receive Fifo (kByte): 4

Size of the Transmit Fifo (kByte): 4

Arbitration Baudrate:

CAN Clock in Hz: 160000000

Prescale: 8

Registervalue Prescale:: 6

Startvalue Timesegment A: 15

Registervalue Time_a:: 14

Startvalue Timesegment B: 4

Registervalue Time_b:: 2

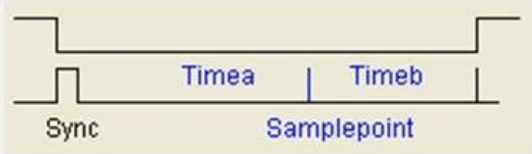
Arbitration Baudrate KBaud:: 1000

Arbitration Sample Point %:: 80

FD Data Baudrate:

CAN Clock in Hz: 160000000

Prescale: 8

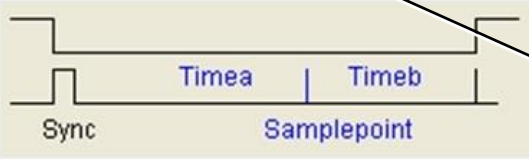


The diagram illustrates the timing of CAN bus segments. It shows a high-level signal with a 'Sync' period, followed by two segments labeled 'Timea' and 'Timeb', and finally a 'Samplepoint' period. The segments are separated by vertical lines, and the 'Samplepoint' is indicated by a horizontal line at the end.

- Enable or Disable the Timestamp
- Enable Dual Clock for System and CAN
- Enable Bus Statistic
- Disable PWM
- Enable Filters and Masks
- Select ID Format
- Receive Fifo-size in kByte
- Transmit Fifo-size in kByte

- Baudrate Calculator

Arbitration Baudrate:	
CAN Clock in Hz:	160000000
Prescale:	8
Registervalue Prescale::	6
Startvalue Timesegment A:	15
Registervalue Time_a::	14
Startvalue Timesegment B:	4
Registervalue Time_b::	2
Arbitration Baudrate KBaud::	1000
Arbitration Sample Point %::	80



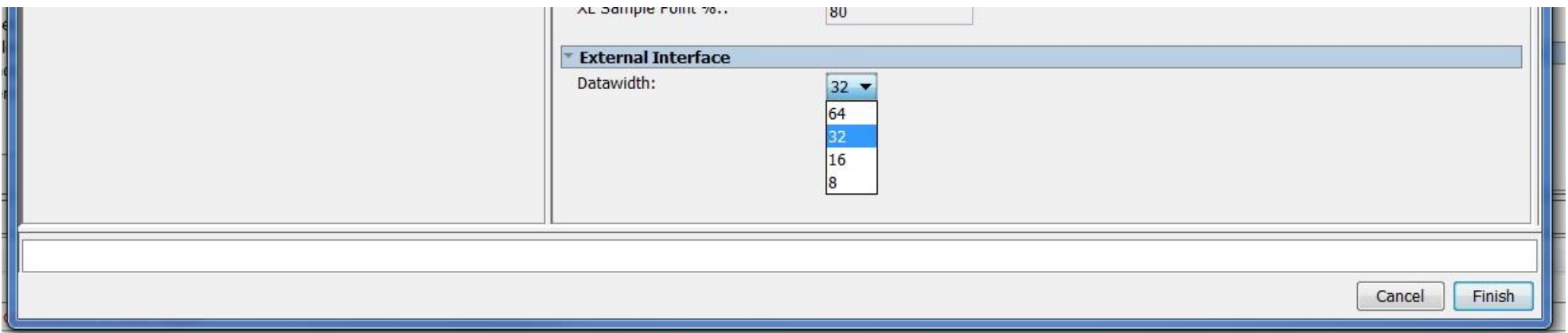
The diagram shows a CAN bus timing sequence. It starts with a 'Sync' period, followed by 'Timea' (the first part of the time segment), then 'Timeb' (the second part of the time segment), and finally the 'Samplepoint' where the signal is sampled. The segments are marked with vertical lines and labels.

FD Data Baudrate:	
CAN Clock in Hz:	160000000
Prescale:	4
Registervalue Prescale::	2
Startvalue Timesegment A:	16
Registervalue Time_a::	15
Startvalue Timesegment B:	3
Registervalue Time_b::	1
FD Baudrate KBaud::	2000
FD Sample Point %::	85

XL Data Baudrate:	
Prescale:	2
Registervalue Prescale::	0
Startvalue Timesegment A:	16
Registervalue Time_a::	15
Startvalue Timesegment B:	3
Registervalue Time_b::	1
XL Baudrate KBaud::	4000
XL Sample Point %::	85

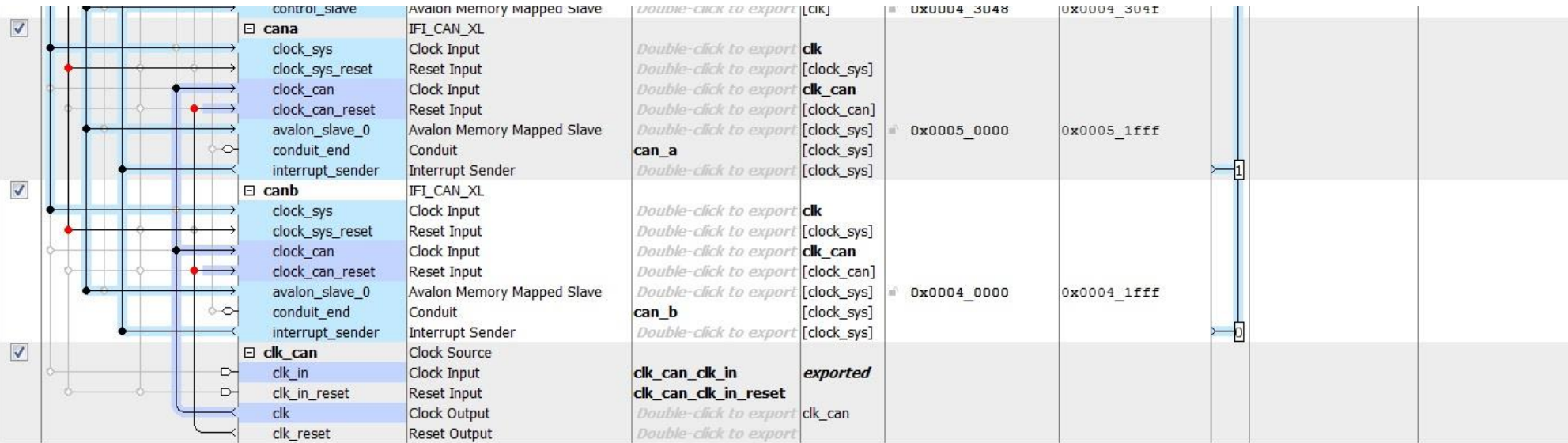
- all CAN timing parameters are startup values and should be overwritten by software
- The clock frequency of your selected clock (updated after connected in QSYS)
- The CAN timing is always a result of three parameters
 - Prescale
 - Timesegment A
 - Timesegment B
- The necessary register values of these parameters are different to the values used for computing the Baudrate and the Samplepoint
- Values for the Fast Baudrate (when selected in transmit / receive message)
- Values for XL Baudrate (when selected in transmit / receive message)

- For external CPU interfaces it is possible to select the width of the databus
- in any case: CPU has to write all 32 Bit !!
- 1*32 bit or 2*16 bit or 4*8 bit
- 64 Bits / 32Bits / 16 Bits / 8 Bits



- Clicking on Finish will add the core to your SOPC system

Adding the Core to your System



- Specify the desired instance **name**, **base address**, and **IRQ**
- connect the system clock to clock_sys Clock Input
- and connect your can-clock to clock_can when Dual_Clock is enabled
- and connect your can-reset to clock_can_reset when Dual_Clock is enabled
- export the conduit (RX, TX..)
- Complete the system generation as described in the INTEL/Altera QSYS documentation



Reference Designs

Running a Reference Design

Creating a Software Project

Running a Hardware Configuration

Running a Reference Design

- Start Quartus II, version 9.1 or a newer version
- Open one of the Quartus II projects in
 - <Core installation directory>\reference_designs\xxx\IFI_CANXL_Ref_design.qpf
- Launch QSYS inside Quartus II software and select the .qsys file
- Click "Generate" to generate the HDL and Modelsim project files
- Click "Exit" to go back to Quartus and compile the design
- Launch the IDE for the creation of software projects or Modelsim software simulation
 - For the simulation are the following lines in the Testbench included
rx_to_the_cana <= tx_from_the_cana and tx_from_the_canb;
rx_to_the_canb <= tx_from_the_cana and tx_from_the_canb;
 - This allows communication between both CAN nodes

Creating a Software Project

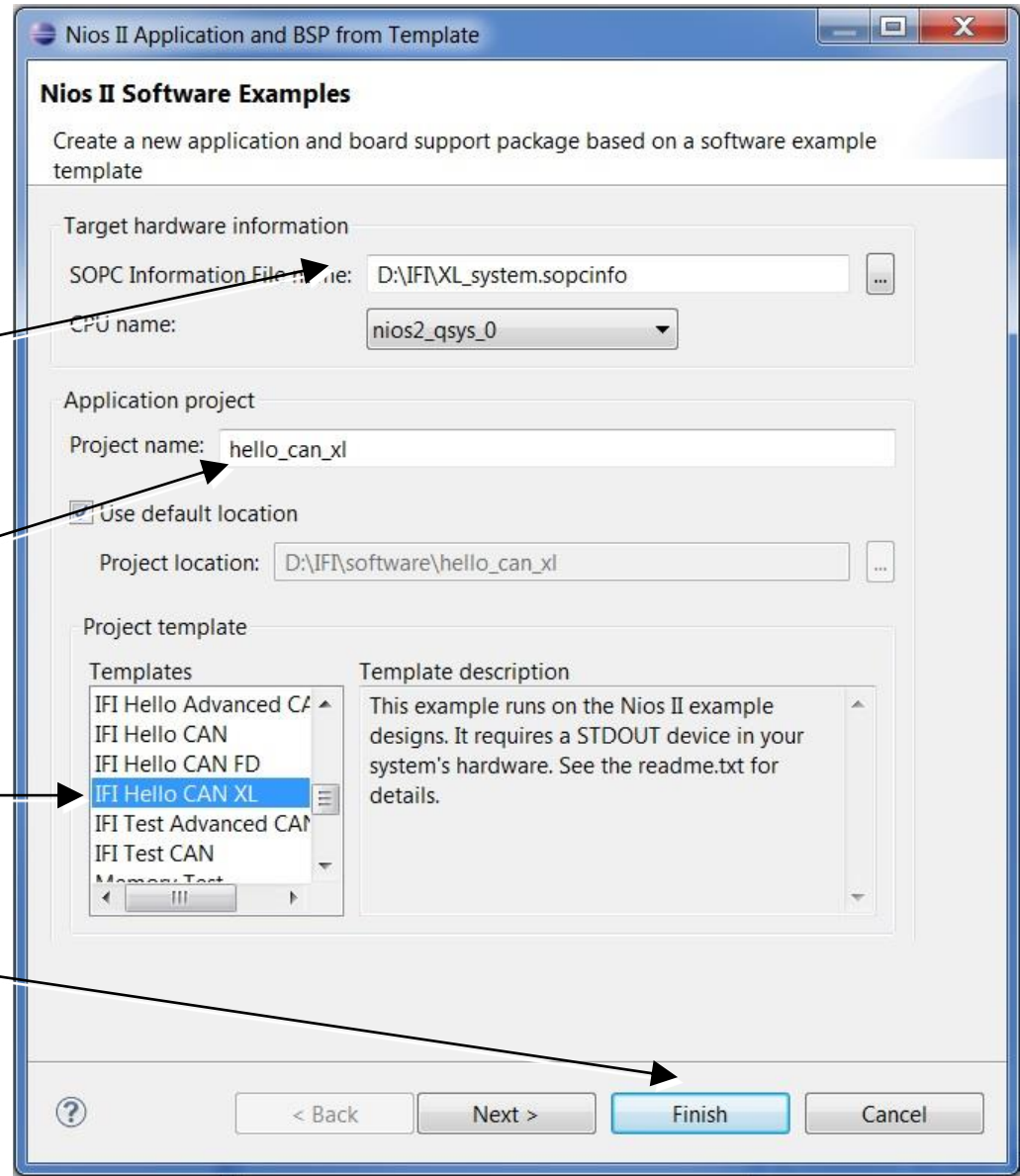
- File → New →
 - NiosII application and BSP from Template

Select the SOPCINFO of the project

Give a project name

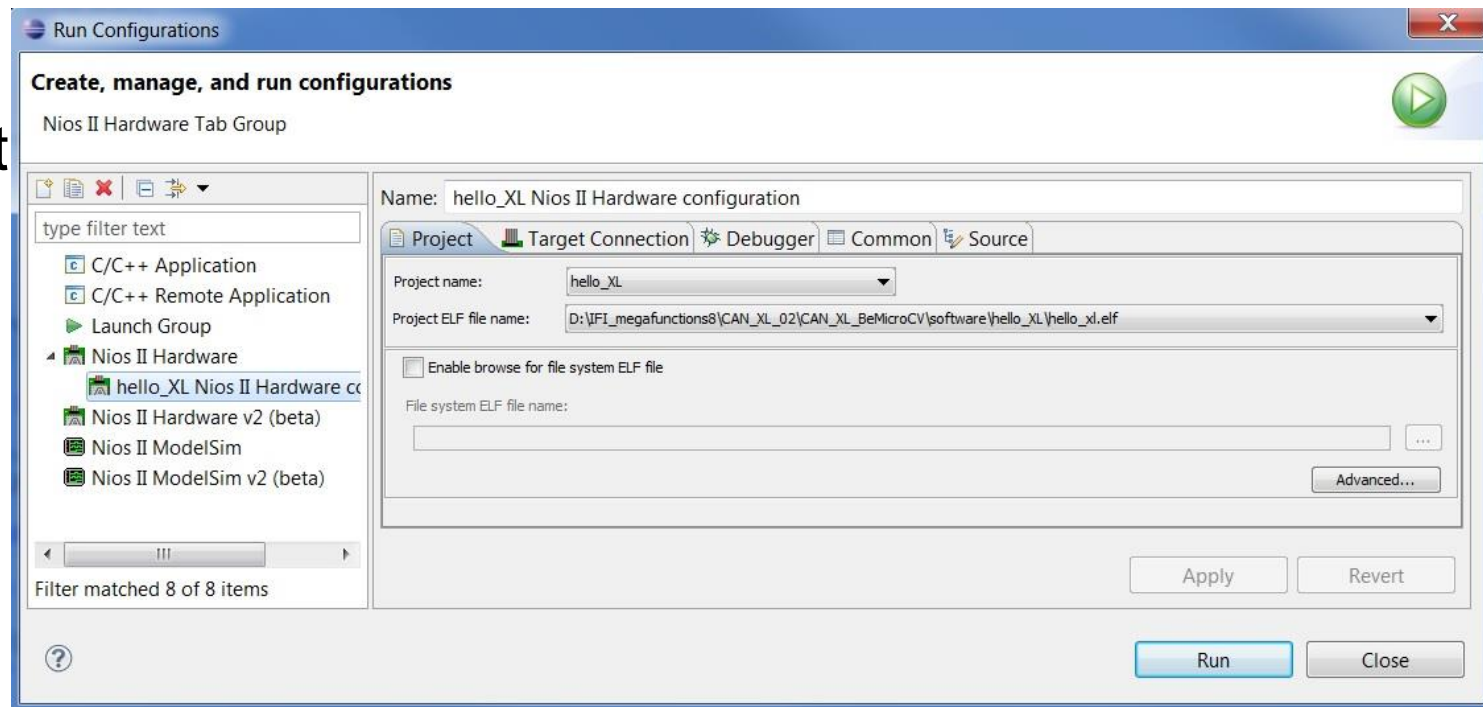
Select „IFI Hello CAN XL“

Click on Finish



Run a Hardware Configuration

- Select your project within the C/C++ Projects View
- Run → Run Configurations ...
- Select NiosII Hardware
- Click on New
- Select Project
- Click on Run

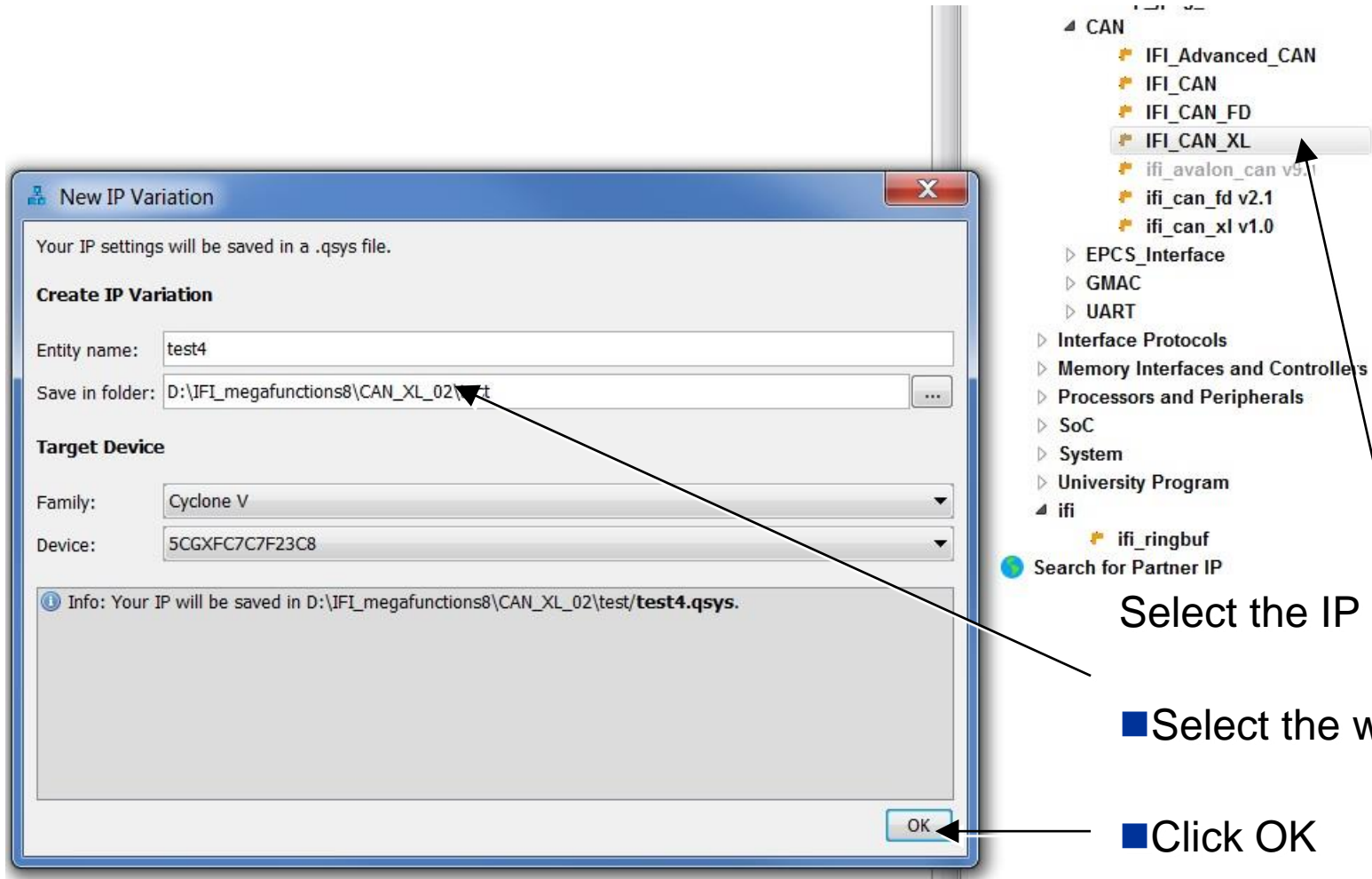




Using the Core without Nios/SOPC/QSYS

IP Catalog
Parameterize
Generate
Quartus Symbol
Port Description
ReadWrite Timing

IP Catalog



Select the IP

■ Select the wrapper name

■ Click OK

IP Parameter Editor - test4.qsys* (D:\JL megafunctions\CAN_XL_02\test4.qsys)

File Edit System Generate View Tools Help

Parameters 15

System: test4 Path: if1_can0_0

IF1_CAN_XL

F. Springer

Details

general Parameter

Timestamp: OFF

Dual_Clock: OFF

Bus_statistic: OFF

PWM for XL FAST mode: OFF

Filter: ON

ID Format: IF1_legacy

Fifo:

Size of the Receive Fifo (kByte): 4

Size of the Transmit Fifo (kByte): 4

Arbitration Baudrate:

CAN Clock in Hz: 100

Prescale: 4

Registervalue Prescale: 2

Starvalue Timesegment A: 33

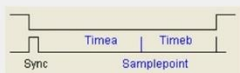
Registervalue Time_a: 32

Starvalue Timesegment B: 6

Registervalue Time_b: 4

Arbitration Baudrate Kbaud: 0

Arbitration Sample Point %: 85



FD Data Baudrate:

CAN Clock in Hz: 100

Prescale: 2

Registervalue Prescale: 0

Starvalue Timesegment A: 16

Registervalue Time_a: 15

Starvalue Timesegment B: 3

Registervalue Time_b: 1

FD Baudrate Kbaud: 0

FD Sample Point %: 85

XL Data Baudrate:

Prescale: 2

Registervalue Prescale: 0

Starvalue Timesegment A: 7

Registervalue Time_a: 6

Starvalue Timesegment B: 2

Registervalue Time_b: 0

XL Baudrate Kbaud: 0

XL Sample Point %: 80

External Interface

Datavidth: 32

Messages

Type Path Message

0 Errors, 0 Warnings

Details

IF1_CAN_XL

Name: if1_can0

Version: 1.0

Author: IF1

Description: IF1_CAN_XL - v1.0

Group: IF1CAN

Data Sheet: [file:///C:/altera/15.1/ip/ifi/if1_can_v1.0/doc/IF1_CAN0L.pdf](#)

general Parameter

Timestamp: OFF to save (about 100 LE) FPGA resources

Dual_Clock: OFF to save (about 350 LE) FPGA resources

Bus_statistic: OFF to save (about 300 LE) FPGA resources

PWM for XL FAST mode: OFF to save (about 100 LE) FPGA resources

Filter: OFF to save (about 200 LE 2 M9k) FPGA resources

ID Format: IF1_legacy (like IF1 CAN_IP, IF1 Advanced CAN IP), CANalyzer format, ID_other: TBD

Fifo:

Size of the Receive Fifo (kByte): 4, 8, 16, 32 or 64 kByte

Size of the Transmit Fifo (kByte): 4, 8, 16, 32 or 64 kByte

Arbitration Baudrate:

CAN Clock in Hz: Your CAN Clock in Hz

Prescale: 2 .. 511

Registervalue Prescale: Registervalue Prescale

Starvalue Timesegment A: 2 .. 511

Registervalue Time_a: Registervalue Time_a

Starvalue Timesegment B: 2 .. 255

Registervalue Time_b: Registervalue Time_b

Arbitration Baudrate Kbaud: Resulting Arbitration Baudrate

Arbitration Sample Point %: Resulting Sample Point in %

FD Data Baudrate:

CAN Clock in Hz: Your CAN Clock in Hz

Prescale: 2 .. 511

Registervalue Prescale: Registervalue Prescale

Starvalue Timesegment A: 2 .. 255

Registervalue Time_a: Registervalue Time_a

Starvalue Timesegment B: 2 .. 255

Registervalue Time_b: Registervalue Time_b

FD Baudrate Kbaud: Resulting FD Baudrate

FD Sample Point %: Resulting Sample Point in %

XL Data Baudrate:

Prescale: 2 .. 511

Registervalue Prescale: Registervalue Prescale

Starvalue Timesegment A: 2 .. 255

Registervalue Time_a: Registervalue Time_a

Starvalue Timesegment B: 2 .. 255

Registervalue Time_b: Registervalue Time_b

XL Baudrate Kbaud: Resulting XL Baudrate

XL Sample Point %: Resulting Sample Point in %

External Interface

Datavidth: 32 Bit, 16 Bit, 8 Bit or 64 Bit wide

Preset

Preset for if1_can0_0

Project: Click New... to create a preset.

Library: No presets for IF1_CAN_XL 1.0

Apply Update Delete New...

Generate HDL... Finish

Parameters

System: XL_system Path: cana

IFI
F. Sprenger

IFI_CAN_XL
ifi_canxl

general Parameter

Timestamp:: ON_internal

Dual_Clock:: ON

Bus_statistic:: ON

PWM for XL FAST mode: OFF

Filter:: ON

ID Format:: IFI_legacy

Fifo:

Size of the Receive Fifo (kByte): 4

Size of the Transmit Fifo (kByte): 4

Arbitration Baudrate:

CAN Clock in Hz: 160000000

Prescale: 8

Registervalue Prescale:: 6

Startvalue Timesegment A: 15

Registervalue Time_a:: 14

Startvalue Timesegment B: 4

Registervalue Time_b:: 2

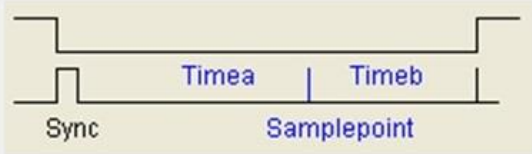
Arbitration Baudrate KBaud:: 1000

Arbitration Sample Point %:: 80

FD Data Baudrate:

CAN Clock in Hz: 160000000

Prescale: 4




- Enable or Disable the Timestamp
- Enable Dual Clock for System and CAN
- Enable Bus Statistic
- Disable PWM
- Enable Filters and Masks
- Select ID Format
- Receive Fifo-size in kByte
- Transmit Fifo-size in kByte

■ Baudrate Calculator

Arbitration Baudrate:

CAN Clock in Hz:	160000000
Prescale:	8
Registervalue Prescale::	6
Startvalue Timesegment A:	15
Registervalue Time_a::	14
Startvalue Timesegment B:	4
Registervalue Time_b::	2
Arbitration Baudrate KBaud::	1000
Arbitration Sample Point %::	80



FD Data Baudrate:

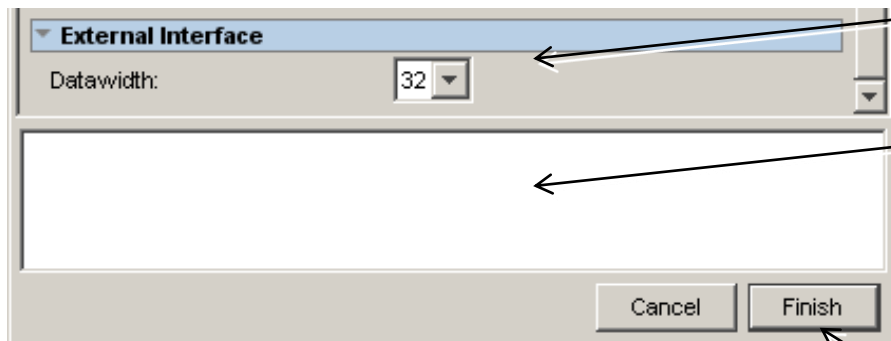
CAN Clock in Hz:	160000000
Prescale:	4
Registervalue Prescale::	2
Startvalue Timesegment A:	16
Registervalue Time_a::	15
Startvalue Timesegment B:	3
Registervalue Time_b::	1
FD Baudrate KBaud::	2000
FD Sample Point %::	85

XL Data Baudrate:

Prescale:	2
Registervalue Prescale::	0
Startvalue Timesegment A:	16
Registervalue Time_a::	15
Startvalue Timesegment B:	3
Registervalue Time_b::	1
XL Baudrate KBaud::	4000
XL Sample Point %::	85

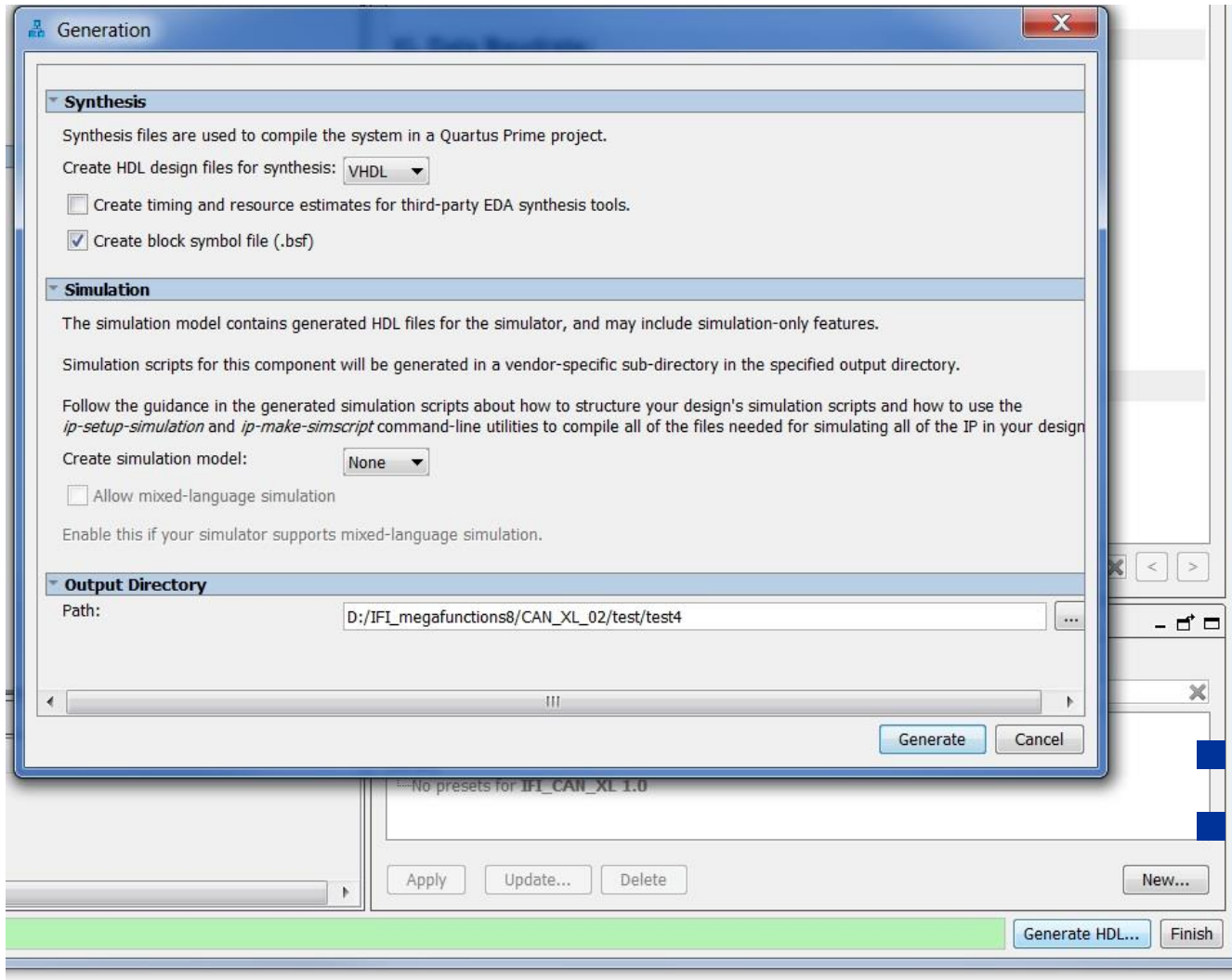
- Both CAN timing parameters are startup values and should be overwritten by software
- The clock frequency of your clock
 - With Megawizzard this is not updated by the IP
 - It is used for this GUI only, so ignore the computation here
- The CAN timing is always a result of three parameters
 - Prescale
 - Timesegment A
 - Timesegment B
- The necessary register values of these parameters are different to the values used for computing the
 - Baudrate and the
 - Samplepoint
- Values for the Fast Baudrate (when selected in transmit / receive message)

- For external CPU interfaces it is possible to select the width of the databus
- in any case CPU has to write all 32 Bit !
4*8 bit or 2*16 bit



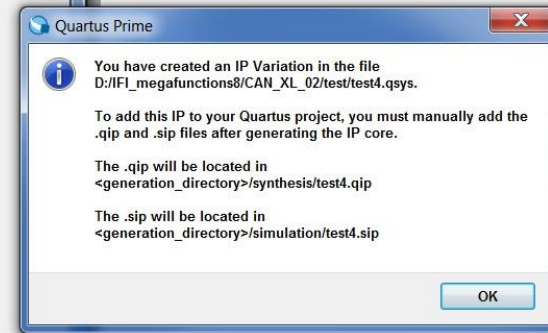
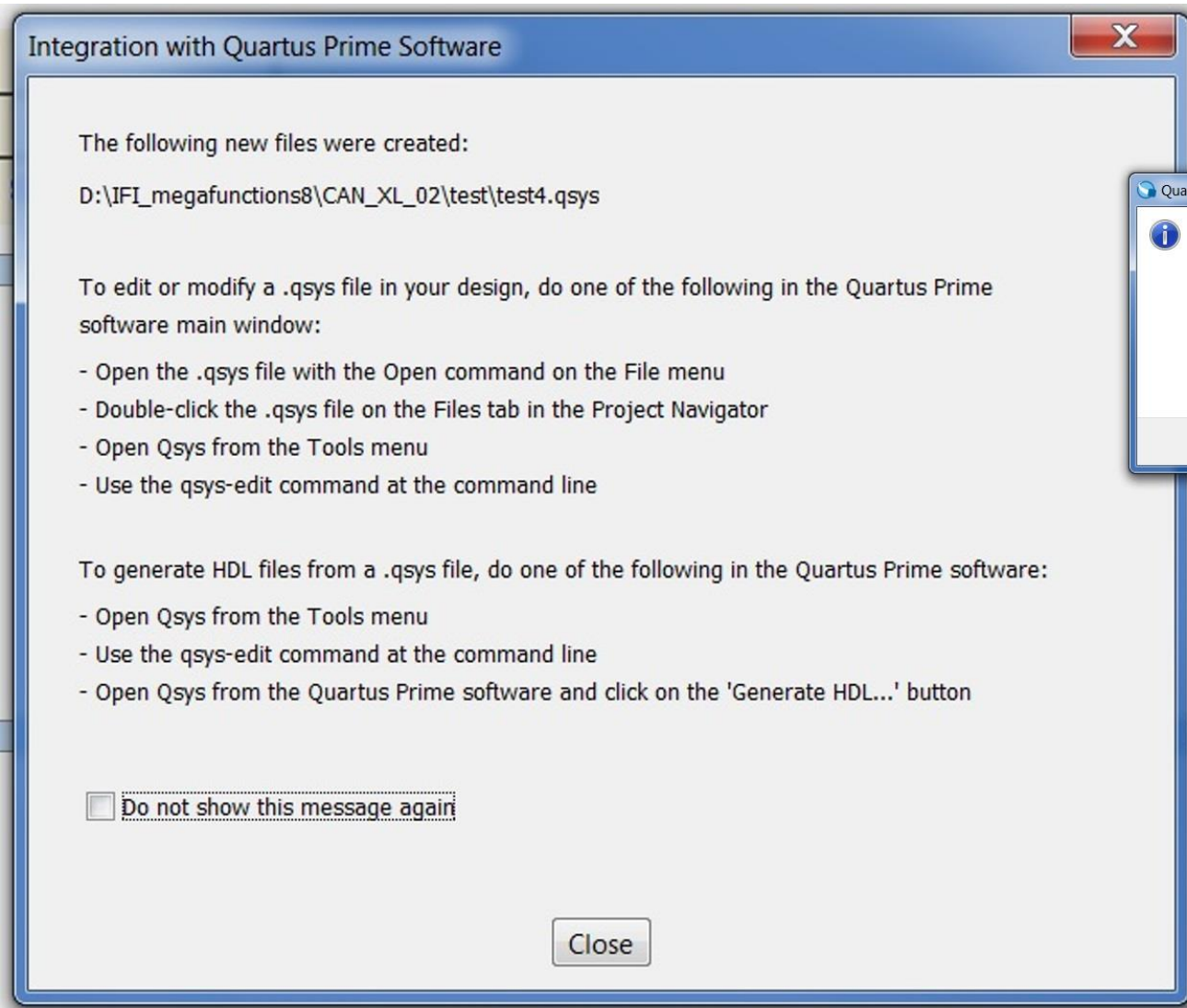
- 64 Bits / 32Bits / 16 Bits / 8 Bits
- In the information field you will find information, warnings and errors
- Click on Finish will add the core to your system

Generation of the Core Variation

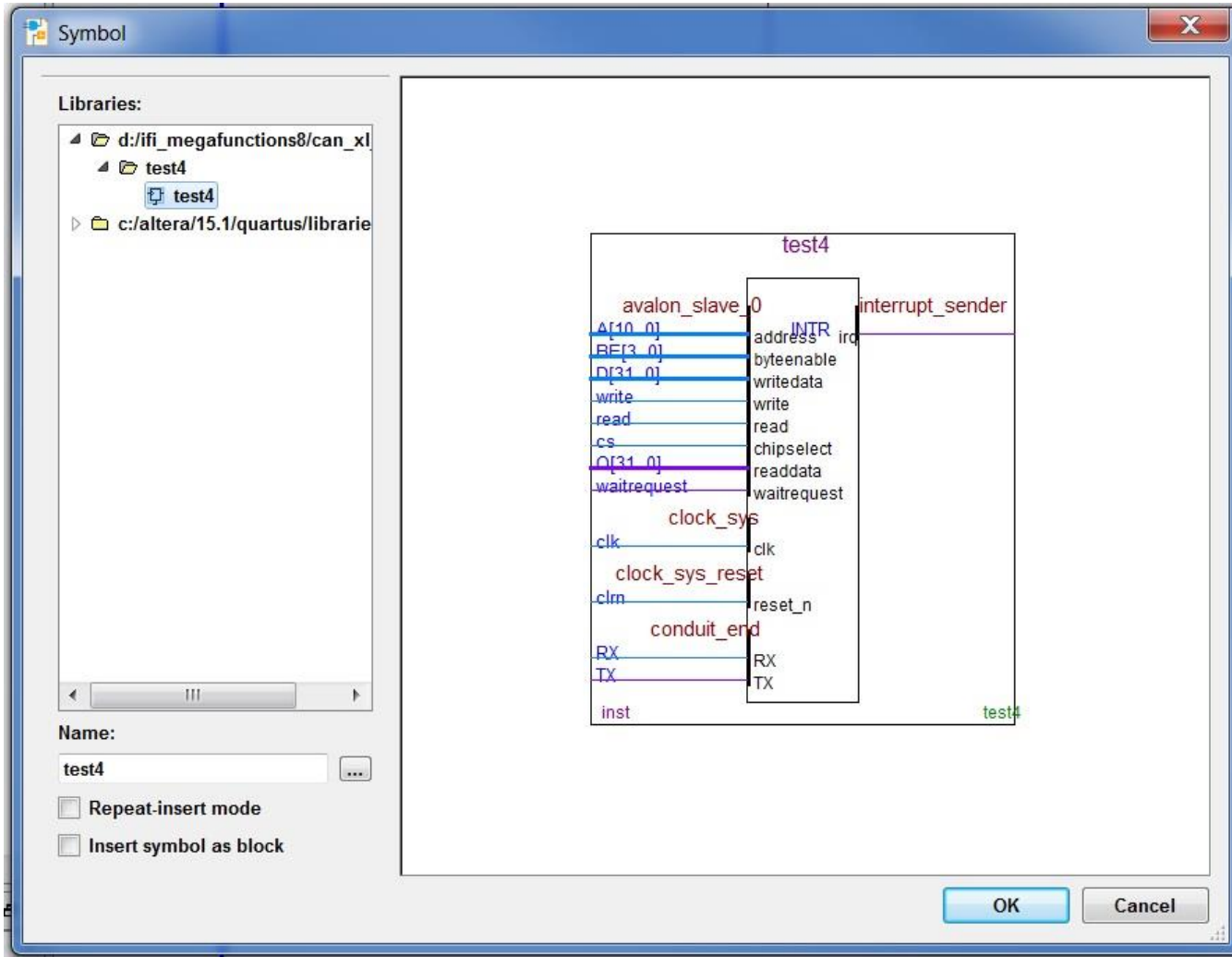


■ generate in VHDL
■ and make BSF

Use the QuartusII IP File



Your new QuartusII IP Symbol



- use Symbol for BlockDesign
- or use test4_inst.vhd for VHDL design

default Port description

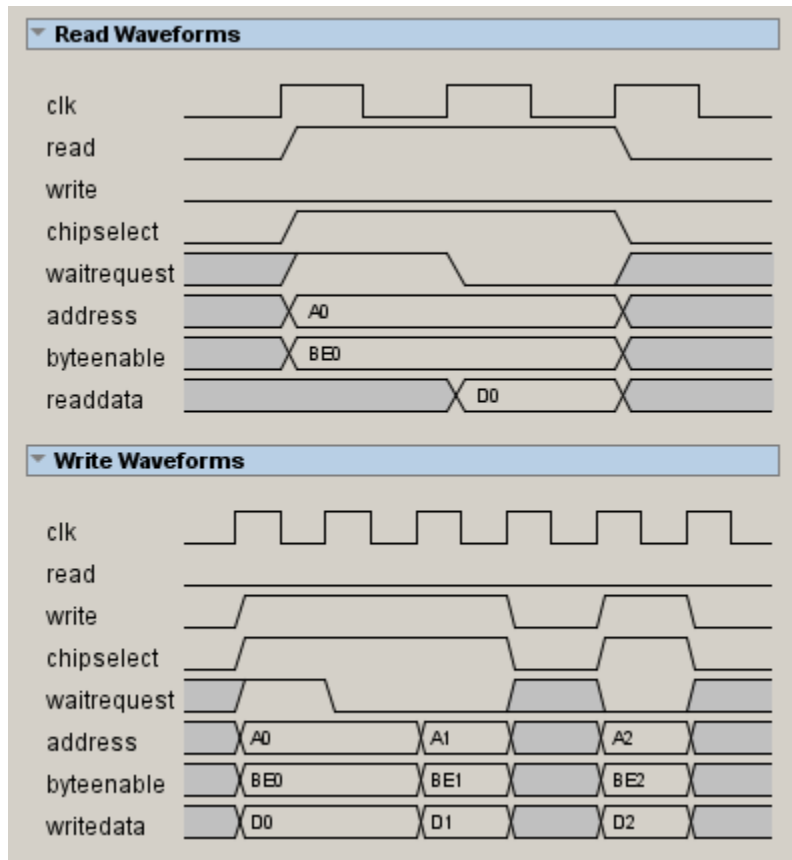
Portname	Direction	Usage	Description
TX	output	external	Transmit data from CAN_XL IP-core
RX	input	external	Receive data in CAN_XL IP-core
clk	input	internal	System clock
clrn	input	internal	System reset (low active)
read	input	internal	Read request
write	input	internal	Write request
cs	input	internal	Chip select
waitrequest	output	internal	Waitrequest (identical to Avalon Memory Mapped Slave)
A[addresswidth-1..0]	input	internal	Address for read/write requests
D[datawidth-1..0]	input	internal	Write data bus
Q[datawidth-1..0]	output	internal	Read data bus
BE[byteenables-1..0]	input	internal	Byteenable
INTR	output	internal	Interrupt request

additional Port description

Portname	Direction	Usage	Description
TimeStamp_clock	input	external	Clock for TimeStamp
TimeStamp_reset	input	external	Reset for TimeStamp (high active)
SOF	output	external	Pulse when SOF (Start Of Frame) is detected
RX_ack	output	external	Pulse when a received frame was accepted
TX_ack	output	external	Puls when a transmitted frame was accepted

- Select TimeStamp_ExternalControl to get these additional inputs and outputs for your own TimeStamp design

ReadWrite Timing



- The IP Catalog implementation
 - use the same timing as with the avalon memory mapped slave



License Agreement

License Agreement

PLEASE REVIEW THE FOLLOWING TERMS AND CONDITIONS CAREFULLY BEFORE USING THE IFI IP-MODULE. BY USING THE IFI IP-MODULE AND/OR PAYING A LICENSE FEE, YOU INDICATE YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS, WHICH CONSTITUTE THE LICENSE AGREEMENT (the "AGREEMENT") BETWEEN YOU AND IFI. IN THE EVENT THAT YOU DO NOT AGREE WITH ANY OF THESE TERMS AND CONDITIONS, DO NOT USE THE IFI IP-MODULE AND WE ASK YOU TO DESTROY ANY COPIES YOU HAVE MADE IMMEDIATELY.

DEFINITIONS:

"Party" means either IFI or YOU.

"Specification" means IFI's technical description for the IFI IP-MODULE covered by this Agreement to the extent such technical description relates to the operation, performance and other material attributes of the IFI IP-MODULE.

1. License to the IFI IP-MODULE:

- 1.1 Subject to the terms and conditions of this Agreement (including but not limited to YOUR payment of the license fee set forth in Paragraph 4.0), IFI grants to YOU a single-user, non-transferable, non-exclusive and (except as specified by IFI) perpetual license to use the IFI IP-MODULE as follows. YOU may:
 - (a) design with, parameterize, compile and route the IFI IP-MODULE;
 - (b) program Altera devices with the IFI IP-MODULE;
 - (c) use the IFI IP-MODULE on a single computer only; and
 - (d) except as otherwise provided in Paragraph 1.2, YOU may use, distribute, sell and/or otherwise market products containing licensed products to any third party in perpetuity. YOU may also sublicense YOUR right to use and distribute products containing licensed products as necessary to permit YOUR distributors to distribute and YOUR customers to use products containing licensed products. YOU are expressly prohibited from using the IFI IP-MODULE to design, develop or program Non-Altera Devices .
- 1.2 YOU may make only one copy of the IFI IP-MODULE for back-up purposes only. The IFI IP-MODULE may not be copied to, installed on or used with any other computer, or accessed or otherwise used over any network, without prior written approval from IFI.
- 1.3 Any copies of the IFI IP-MODULE made by or for YOU shall include all intellectual property notices, including copyright and proprietary rights notices, appearing on such IFI IP-MODULE. Any copy or portion of the IFI IP-MODULE, including any portion merged into a design and any design or product that incorporates any portion of the IFI IP-MODULE, will continue to be subject to the terms and conditions of this Agreement.
- 1.4 The source code of the IFI IP-MODULE, and algorithms, concepts, techniques, methods and processes embodied therein, constitute trade secrets and confidential and proprietary information of IFI and its licensors and LICENSEE shall not access or use such trade secrets and information in any manner, except to the extent expressly permitted herein. IFI and its licensors retain all rights with respect to the IFI IP-MODULE, including any copyright, patent, trade secret and other proprietary rights, not expressly granted herein.

2. License Restrictions:

YOU MAY NOT USE THE IFI IP-MODULE EXCEPT AS EXPRESSLY PROVIDED FOR IN THIS AGREEMENT OR SUBLICENSE OR TRANSFER THE IFI IP-MODULE OR RIGHTS WITH RESPECT THERETO. YOU MAY NOT DECOMPILE, DISASSEMBLE, OR OTHERWISE REVERSE ENGINEER THE IFI IP-MODULE OR ATTEMPT TO ACCESS OR DERIVE THE SOURCE CODE OF THE IFI IP-MODULE OR ANY ALGORITHMS, CONCEPTS, TECHNIQUES, METHODS OR PROCESSES EMBODIED THEREIN; PROVIDED, HOWEVER, THAT IF YOU ARE LOCATED IN A MEMBER NATION OF THE EUROPEAN UNION OR OTHER NATION THAT PERMITS LIMITED REVERSE ENGINEERING NOTWITHSTANDING A CONTRACTUAL PROHIBITION TO THE CONTRARY, YOU MAY PERFORM LIMITED REVERSE ENGINEERING, BUT ONLY AFTER GIVING NOTICE TO IFI AND ONLY TO THE EXTENT PERMITTED BY THE APPLICABLE LAW IMPLEMENTING THE EU SOFTWARE DIRECTIVE OR OTHER APPLICABLE LAW NOTWITHSTANDING A CONTRACTUAL PROHIBITION TO THE CONTRARY.

3. Term:

This Agreement is effective until terminated. YOU may terminate it at any time by destroying the IFI IP-MODULE together with all copies and portions thereof in any form (except as provided below). It will also terminate immediately if YOU breach any term of this Agreement and upon conditions set forth elsewhere in this Agreement. Upon any termination of this Agreement, YOU shall destroy the IFI IP-MODULE, including all copies and portions thereof in any form (whether or not merged into a design or Licensed Product), and YOUR license and rights under this Agreement shall terminate except that YOU and YOUR customers may continue to sell and use Licensed Products which have been developed in accordance with this Agreement and shipped prior to the termination. In no event may any portions of the IFI IP-MODULE be used in development after termination. In the event of termination for any reason, the rights, obligations and restrictions under Paragraphs 2, 4, 9, and 10 shall survive termination of this Agreement.

4. Payment:

In consideration of the license granted by IFI under Paragraph 1.1 and other rights granted under this Agreement, YOU shall pay the license fee for the IFI IP-MODULE that has been specified by IFI. Such payment shall, as directed by IFI, be made directly to IFI. YOU shall pay all taxes and duties associated with this Agreement, other than taxes based on IFI's income.

5. Maintenance and Support:

IFI shall, but only until the date, in the format YYYY.MM, provided in the license file for a IFI IP-MODULE ("Maintenance Expiration Date"):

- 5.1 use commercially reasonable efforts to provide YOU with fixes to defects in the IFI IP-MODULE that cause the IFI IP-MODULE not to conform substantially to the Specifications and that are diagnosed as such and replicated by IFI;
- 5.2 provide YOU with fixes and other updates to the IFI IP-MODULE that IFI chooses to make generally available to its customers without a separate charge; and
- 5.3 respond by telephone or email to inquiries from YOU.

6. Limited Warranties and Remedies:

- 6.1 IFI represents and warrants that, until the Maintenance Expiration Date ("Warranty Period"), the IFI IP-MODULE will substantially conform to the Specifications. YOUR sole remedy, and IFI's sole obligation, for a breach of this warranty shall be (a) for IFI to use commercially reasonable efforts to remedy the non-conformance or (b) if IFI is unable substantially to remedy the non-conformance, for YOU to receive a refund of license fees paid during the previous one (1) year for the defective IFI IP-MODULE. If YOU receive such a refund, YOU agree that YOUR license and rights under this Agreement for the defective IFI IP-MODULE shall immediately terminate and YOU agree to destroy the defective IFI IP-MODULE, including all copies thereof in any form and any portions thereof merged into a design or product, and to certify the same to IFI.
- 6.2 The foregoing warranties apply only to IFI IP-MODULEs delivered by IFI. The warranties are provided only to YOU, and may not be transferred or extended to any third party, and apply only during the Warranty Period for claims of breach reported (together with evidence thereof) during the Warranty Period. YOU shall provide IFI with such evidence of alleged non-conformities or defects as IFI may request, and IFI shall have no obligation to remedy any non-conformance or defect it cannot replicate. The warranties do not extend to any IFI IP-MODULE which have been modified by anyone other than IFI.

7. Representation:

Each party represents that it has the right to enter into this Agreement and to perform its obligations hereunder.

8. Indemnification:

- 8.1 Expressly subject to Section 9, IFI shall defend YOU against any proceeding brought by a third party to the extent based on a claim that the IFI IP-MODULE, as delivered by IFI and as used in accordance with this Agreement, infringes a third party's copyright, trade secret, patent, or any other intellectual property right ("IP right"), and pay any damages awarded in the proceeding as a result of the claim (or pay any amount agreed to by IFI as part of a settlement of the claim), provided that IFI shall have no liability hereunder unless YOU notify IFI promptly in writing of any such proceeding or claim, give IFI sole and complete authority to control the defence and settlement of the proceeding or claim, and provide IFI with any information, materials, and other assistance requested by IFI.
- 8.2 In the event of any such claim or proceeding or threat thereof, IFI may (and, in the event any such claim or proceeding results in the issuance of an injunction by a court of competent jurisdiction prohibiting YOU from using the IFI IP-MODULE, IFI shall), at its option and expense and subject to the limitations of Paragraph 9, seek a license to permit the continued use of the affected IFI IP-MODULE or use commercially reasonable efforts to replace or modify the IFI IP-MODULE so that the replacement or modified version is non-infringing or has a reduced likelihood of infringement, provided that the replacement or modified version has functionality comparable to that of the original. If IFI is unable reasonably to obtain such license or provide such replacement or modification, IFI may terminate YOUR license and rights with respect to the affected IFI IP-MODULE, in which event YOU shall return to IFI the affected IFI IP-MODULE, including all copies and portions thereof in any form (including any portions thereof merged into a design or product), and certify the same to IFI, and IFI shall refund the license fee paid by YOU for the affected IFI IP-MODULE.
- IFI shall have no liability or obligation to YOU hereunder for any infringement or claim based on or resulting from (a) the combination or use of the IFI IP-MODULE with other products or components; (b) modification of the IFI IP-MODULE by anyone other than IFI, (c) the use of other than the most recent version of the IFI IP-MODULE if the infringement or claim would have been avoided (or the likelihood thereof reduced) by use of the most recent version; (d) requirements specified by YOU; (e) use of the IFI IP-MODULE in any way not contemplated under this Agreement; or (f) any use of the IFI IP-MODULE, to the extent that IFI has indicated in the applicable Specification that third-party licenses 8.3a may be required to use such IFI IP-MODULE if LICENSEE has not obtained the necessary third-party licenses.
- 8.3a The license does not include the CAN-Network license (Bosch).
- 8.4 The provisions of this Paragraph 8 state the entire liability and obligations of IFI, and YOUR sole and exclusive rights and remedies, with respect to any proceeding or claim relating to infringement of copyright, trade secret, patent, or any other intellectual property right.

LIMITATIONS OF LIABILITY

- 9.1 In no event shall the aggregate liability of IFI relating to this Agreement or the subject matter hereof under any legal theory (whether in tort, contract or otherwise), including any liability under Paragraph 8 or for any loss or damages directly or indirectly suffered by YOU relating to the IFI IP-MODULE, exceed the aggregate amount of the license fees paid by YOU in the previous one (1) year under this Agreement.
- 9.2 IN NO EVENT SHALL IFI BE LIABLE UNDER ANY LEGAL THEORY, WHETHER IN TORT, CONTRACT OR OTHERWISE (a) FOR ANY LOST PROFITS, LOST REVENUE OR LOST BUSINESS, (b) FOR ANY LOSS OF OR DAMAGES TO OTHER SOFTWARE OR DATA, OR (c) FOR ANY INCIDENTAL, INDIRECT, CONSEQUENTIAL OR SPECIAL DAMAGES RELATING TO THIS AGREEMENT OR THE SUBJECT MATTER HEREOF, INCLUDING BUT NOT LIMITED TO THE DELIVERY, USE, SUPPORT, OPERATION OR FAILURE OF THE MEGACORE LOGIC IFI IP-MODULE, EVEN IF IFI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LIABILITY.

10. General:

- 10.1 YOU may not sublicense, assign, or transfer this license, or disclose any trade secrets embodied in the IFI IP-MODULE, except as expressly provided in this Agreement. Any attempt to sublicense, assign, or otherwise transfer without prior written approval of the other party any of the rights, duties, or obligations hereunder is void.
- 10.2 This Agreement is entered into for the benefit of IFI and its licensors and all rights granted to YOU and all obligations owed to IFI shall be enforceable by IFI.
- 10.3 If YOU have any questions concerning this Agreement, including software maintenance or warranty service, YOU should contact IFI Ing.Büro Für Ic-Technologie, Franz Sprenger, Am Tannenberg 28, 97877 Wertheim, Germany.
- 10.4 YOU agree that the validity and construction of this Agreement and performance hereunder, shall be governed by the laws of German jurisdictions, without reference to conflicts of law principles. YOU agree to submit to the exclusive jurisdiction of the courts in Germany, for the resolution of any dispute or claim arising out of or relating to this Agreement. The Parties hereby agree that the Party who does not prevail with respect to any dispute, claim, or controversy relating to this Agreement shall pay the costs actually incurred by the prevailing Party, including any attorneys' fees.
- 10.5 No amendment to this Agreement shall be effective unless it is in writing signed by a duly authorized representative of both Parties. The waiver of any breach or default shall not constitute a waiver of any other right hereunder.**
- 10.6 In the event that any provision of this Agreement is held by a court of competent jurisdiction to be legally ineffective or unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable and the validity of the remaining provisions shall not be affected.
- 10.7 The article headings throughout this Agreement are for reference purposes only and the words contained therein shall not be construed as a substantial part of this Agreement and shall in no way be held to explain, modify, amplify, or aid in the interpretation, construction or meaning of the provisions of this Agreement.
- 10.8 BY USING THE IFI IP-MODULE, YOU AND IFI ACKNOWLEDGE THAT YOU AND IFI HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU AND IFI FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN YOU AND IFI, WHICH SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN YOU AND IFI RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT, UNLESS YOU HAVE A SEPARATE LICENSE SIGNED BY AN AUTHORIZED IFI REPRESENTATIVE.



IFI CAN_XL IP

Register Map

Detailed Information

- *Address map*
- *Registers*
- *CAN Timing*
- *Status Informations*
- *Mask and Filter Handling*
- *HAL Drivers*
- *Driver Routines*
- *Structures*
- *Software Examples*

Addressmap CAN_XL

	StartAddress ByteAddress	EndAddress	
Status and Command	0x0000	0x007C	Read and Write
receive FIFO	0x0600	0x0E1C	Read only (32+2048 Bytes)
transmit FIFO	0x0F00	0x171C	Write only (32+2048 Bytes)
Filter-Mask	0x1800	0x1FFC	Read and Write (2048 Bytes)

■ Register access types

- w software can write (and read in most cases)
- r read only
- h hardware updated periodically or on event
- s software write 1 to Set Bit, hardware clears Bit, write 0 no operation
- c software write 1 to Clear Bit, hardware sets Bit, write 0 no operation

Status and Command all

Status and Command	Byte-Address	Word-Address
Main Status and Command	0x0000	0
Receiver Status and Command	0x0004	1
Transmitter Status and Command	0x0008	2
Interrupt pending	0x000C	3
Interrupt Mask	0x0010	4
Timing for slow Baudrate	0x0014	5
Timing for fast FD Baudrate	0x0018	6
TransmitterDelay FD	0x001C	7
ErrorCounter	0x0020	8
Additional ErrorCounter	0x0024	9
Transmission suspend	0x0028	10
Transmission repeatcount	0x002C	11
Bus Traffic Rate	0x0030	12
Timestamp Controll	0x0034	13

Status and Command	Byte-Address	Word-Address
Actual running TimestampCounter	0x0038	14
Last Timestamp	0x003C	15
reserved	0x0040	16
Error Controller	0x0044	17
Compiler Parameter	0x0048	18
CAN_Clock	0x004C	19
System Clock	0x0050	20
Revision	0x0054	21
IP_core ID	0x0058	22
Testregister	0x005C	23

Status and Command all

Status and Command	Byte-Address	Word-Address
Timing for fast XL Baudrate	0x0060	24
TransmitterDelay XL	0x0064	25
PWM timing	0x0068	26
reserved2	0x006C	27
XL Filter Valid - Mask	0x0070	28
XL Filter Bits	0x0074	29
AF Filter Mask	0x0078	30
AF Filter Bits	0x007C	31

Status and Command, Interrupt

Status and Command Interrupt	ByteAddress	WordAddress	
Main Status and Command	0x0000	0	See next pages
Receiver Status and Command	0x0004	1	See next pages
Transmitter Status and Command	0x0008	2	See next pages
Interrupt pending	0x000C	3	See next pages
Interrupt Mask	0x0010	4	See next pages

Main Status and Command

Byte 3	Bit	31	30	29	28	27	26	25	24
	read		HardReset		\$presume_ack	CAN_XL disabled		§ ISO enable	CAN_FD disabled
	write	NormalMode Start_puls		#reset ErrorcounterII	\$presume_ack	CAN_XL disable		§ ISO enable 0: non-ISO	CAN_FD disable
Byte 2	Bit	23	22	21	20	19	18	17	16
	read		Enabled 64Bit extTimeStamp	§ Exception enabled	§ EdgeFilter enabled	Use_PWM	ErrorSignaling disabled	Restricted 3.3.2	BusMonitor 3.3.1
	write		Enable 64Bit extTimeStamp	§ Exception enable	§ EdgeFilter enable	Use_PWM	ErrorSignaling disabled	Restricted 3.3.2	BusMonitor 3.3.1
Byte 1	Bit	15	14	13	12	11	10	9	8
	read								
	write								
Byte 0	Bit	7	6	5	4	3	2	1	0
	read			#Warning	BusOff	Error Passive	Error Active		Enable
	write								Enable

- Writing 0xDEADCAFD to this register set the CAN-Core in a Hardreset Mode, all action on CAN-BUS is stopped immediately, this may generate errorframes when core was transmitting
- Writing something else releases the Hardreset Mode

Status and Command Register (1)

- § Enable ISO standard
 - w setting this bit to 1 enables ISO standard protocol handling, 0: use non-ISO, BOSCH specification 1.0 April 2012
- Disable CAN-FD protocol
 - w setting this bit to 1 disables all CAN-FD protocol handling, IP-core works in CAN-basic modes only
- **Disable CAN-XL protocol**
 - w setting this bit to 1 disables all CAN-XL protocol handling, IP-core works in CAN-basic or CAN-FD modes only
- Hardreset
 - r this bit shows if the IP-core is in Hardreset mode, all actions an CAN-BUS stopped
 - writing 0xDEADCAFD set the IP-core in hardreset mode, do not use this during normal CAN-BUS activity, this would lead to bus-errors, use limited to exceptional situations
- #reset ErrorcounterII
 - Additional reset bit
- \$presume_ack
 - w setting this bit to 1 enables absent acknowledgement to be ignored
- Normal Mode, Start_puls
 - s writing 1 to this bit starts the CAN-BUS handling, set this after all other configuration registers (timing...) are set

Status and Command Register (2)

- Bus monitoring mode, Silent Mode 3.3.1
 - w set this Bit to 1 to enable that mode
- Restricted operation mode 3.3.2
 - w set this Bit to 1 to enable that mode
- XL has a special mode of ErrorSignaling disabled
 - w set this bit to 1 to enable this Mode
- Use_PWM for CAN XL SIC transceivers
 - w set this bit to 1 to enable PWM (usually with ErrorSignaling disabled)
- Edgefilter enable
- Exception enable
- Enable external 64Bit TimeStamp in RXfifo
- Enable
 - w set this bit to 1 to enable the receive and transmit controllers for the Fifos
- Error Active
 - r The CAN node works properly
- Error Passive
 - r The CAN sends recessive errorframes and suspends the transmission for 8 bittimes
- BusOff
 - r The CAN node has stopped all activities
- # Warning
 - r CAN node has some errors but is still in error active

Receiver Fifo Status and Command

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	RFcnt 15	RFcnt 14	RFcnt 13	RFcnt 12	RFcnt 11	RFcnt 10	RFcnt 9	RFcnt 8
	write								
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	RFcnt 7	RFcnt 6	RFcnt 5	RFcnt 4	RFcnt 3	RFcnt 2	RFcnt 1	RFcnt 0
	write								
Byte 1	Bit	15	14	13	12	11	10	9	8
	read			Overflow	Full	More than 7/8 full	More ¾ full	More than half full	empty
	write								
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	Reset FIFO							
	write	Reset FIFO Clear Bit to enable FIFO							Write 1 to Remove Message

- RFcnt [15..0] : Receive Fifo number of stored Messages/Frames

Transmitter Fifo Status and Command

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	TFcnt 15	TFcnt 14	TFcnt 13	TFcnt 12	TFcnt 11	TFcnt 10	TFcnt 9	TFcnt 8
	write								
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	TFcnt 7	TFcnt 6	TFcnt 5	TFcnt 4	TFcnt 3	TFcnt 2	TFcnt 1	TFcnt 0
	write								
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	Highpriority pending	Invalid access	Overflow	Full	Less than half full	Less than ¼ full	Less than 1 / 8 full	empty
	write								
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	Reset FIFO							
	write	Reset FIFO Clear Bit to enable FIFO	Remove pending message					Announce highpriority	Write 1 to add Message

- TFcnt [15..0] : Transmit Fifo number of stored Messages/Frame

Transmitter Fifo Status and Command

- Add Message
 - s set this bit to 1 after a frame is completely written into the transmitter fifo, so that frame can be transmitted and the Fifo advances to next place
- Announce High priority Message (**Classic, FD**)
 - s set this bit to 1 before a high priority message is written to the transmit fifo
 - for storing a number of messages for transmit we use a Fifo (First In First out), because the CAN-BUS has an arbitration using the Message-ID bits, it is possible that a transmit message is delayed because other external CAN-Nodes have a higher priority
 - the Transmitter Fifo has the option to store one high priority message out of the fifo order, so at next possible transmission this high priority message is transmitted when its message-ID is the highest on the CAN-BUS
 - correct sequence for a high priority message
 - 1 do not interrupt loading of a „normal“ message
 - 2 test if HighPriority pending is cleared, no more than one highpriority Message is supported
 - 3 set Announce High Priority Bit
 - 4 write complete message to transmit fifo
 - 5 set Add Message Bit
 - note: a high priority message restarts the repeatcount of the interrupted message
- Remove pending message
 - s set this bit to remove actual transmit message from CAN-Bus if possible, do not use when a high priority message is loaded
- Invalid Acces
 - r shows that more DataBytes were written to the fifo than the DLC announced
- HighPriority pending
 - r shows that the high priority message is not transmitted yet

Interrupt pending

Byte 3 Receive- FIFO	Bit	31	30	29	28	27	26	25	24
	read		ReceiveFifo overflow	ReceiveFifo full	ReceiveFifo more 7/8 full	ReceiveFifo more 3/4 full	ReceiveFifo more half full	ReceiveFifo notempty per	ReceiveFifo notempty
	write	Set IRQpend by Software	ReceiveFifo overflow	ReceiveFifo full	ReceiveFifo more 7/8 full	ReceiveFifo more 3/4 full	ReceiveFifo more half full	ReceiveFifo notempty per	ReceiveFifo notempty
Byte 2 Transmitt- FIFO	Bit	23	22	21	20	19	18	17	16
	read		TransmitFifo 1 msg removed	TransmitFifo overflow	TransmitFifo full	TransmitFifo less half full	TransmitFifo less ¼ full	TransmitFifo less 1/8 full	TransmitFifo empty
	write		TransmitFifo 1 msg removed	TransmitFifo overflow	TransmitFifo full	TransmitFifo less half full	TransmitFifo less 1/4 full	TransmitFifo less 1/8 full	TransmitFifo empty
Byte 1	Bit	15	14	13	12	11	10	9	8
	read				§RMPProtocol Exception	§FDProtocol Exception	reserved	ErrorCounter II Overrun	Timestamp Overrun
	write				§RMPProtocol Exception	§FDProtocol Exception	reserved	ErrorCounter II Overrun	Timestamp Overrun
Byte 0	Bit	7	6	5	4	3	2	1	0
	read					#REC_TEC Increment	#Error State Change	Error warn	Busoff
	write					#REC_TEC Increment	#Error State Change	Error Warn	Busoff

- all possible IRQ-Bits can be read here (independent of the IRQ-Mask)
- writing 1 to a bit reset the corresponding IRQ-Pending Bit (do not set bit 31 !)
- if any IRQ-Pending bit and its corresponding IRQ-Mask bit is set the IRQ is set
- writing 1 to bit 31 allows Software to load IRQpending register (for debug purposes)

Interrupt mask

Byte 3 Receive-FIFO	Bit	31	30	29	28	27	26	25	24
	read		ReceiveFifo overflow	ReceiveFifo full	ReceiveFifo more 7/8 full	ReceiveFifo more 3/4 full	ReceiveFifo more half full	ReceiveFifo notempty per	ReceiveFifo notempty
	write	Set Receive IRQ-Mask	ReceiveFifo overflow	ReceiveFifo full	ReceiveFifo more 7/8 full	ReceiveFifo more 3/4 full	ReceiveFifo more half full	ReceiveFifo notempty per	ReceiveFifo notempty
Byte 2 Transmitt-FIFO	Bit	23	22	21	20	19	18	17	16
	read		TransmitFifo 1 msg removed	TransmitFifo overflow	TransmitFifo full	TransmitFifo less half full	TransmitFifo less ¼ full	TransmitFifo less 1/8 full	TransmitFifo empty
	write	Set Transmit IRQ-Mask	TransmitFifo 1 msg removed	TransmitFifo overflow	TransmitFifo full	TransmitFifo less half full	TransmitFifo less 1/4 full	TransmitFifo less 1/8 full	TransmitFifo empty
Byte 1	Bit	15	14	13	12	11	10	9	8
	read				§RMPProtocol Exception	§FDProtocol Exception	reserved	ErrorCounter II Overrun	Timestamp Overrun
	write	Set TimeStamp IRQ-Mask			§RMPProtocol Exception	§FDProtocol Exception	reserved	ErrorCounter II Overrun	Timestamp Overrun
Byte 0	Bit	7	6	5	4	3	2	1	0
	read					#REC_TEC Increment	#Error State Change	Error warn	Busoff
	write	Set Error IRQ-Mask				#REC_TEC Increment	#Error State Change	Error Warn	Busoff

- Set receive IRQ-Mask : 1 => write enable for Receiver IRQ-Mask
- Set Transmit IRQ-Mask : 1 => write enable for Transmitter IRQ-Mask
- Set TimeStamp IRQ-Mask : 1 => write enable for Timestamp interrupt mask
- Set Error IRQ-Mask : 1 => write enable for error interrupt mask

Interrupt Pending Register Receive

- ReceiveFifo overflow
 - Received messages are lost, because the receive fifo was already full
- ReceiveFifo full
 - the fifo is full
- ReceiveFifo more than 7/8 full
 - more than 7/8 of the fifo is full, counted in Bytes not in messages
- ReceiveFifo more than 3/4 full
 - more than 3/4 of the fifo is full, counted in Bytes not in messages
- ReceiveFifo more than half full
 - more than half of the fifo is full, counted in Bytes not in messages
- ReceiveFifo notempty permanent
 - at least one message is in the fifo
- ReceiveFifo notempty
 - The receive fifo was empty and the first message was written into the fifo

Interrupt Pending Register Transmit

- TransmitFifo 1 msg removed
 - one transmit message was removed from transmit fifo, because the message was successfully send or repeat count reached limit
- TransmitFifo overflow
 - transmit message lost, because the message was written to already full fifo
- TransmitFifo full
 - the fifo is full
- TransmitFifo less than half full
 - less than half of the fifo is full, counted in Bytes not in messages
- TransmitFifo less than 1/4 full
 - less than 1/4 of the fifo is full, counted in Bytes not in messages
- TransmitFifo less than 1/8 full
 - less than 1/8 of the fifo is full, counted in Bytes not in messages
- TransmitFifo empty
 - The fifo is empty
- ErrorCounterII Overrun
 - there was a overrun in one of the additional Error Counters
- TimeStamp Overrun
 - there was a overrun in the Timestamp counter
- § Protocol Exception
 - there was a protocol exception
- #REC_TEC increment
 - REC receive error counter or TEC transmit error counter was incremented (by an error)
- #Error State change
 - The can error state changed (error active, error passive, busoff), warning is not part of the official error state and has ist own IRQ see ErrorWarn
- ErrorWarn
 - CAN-Errorcounter reached the warning level
- BusOff
 - because of too many errors the CAN-Node switched off from the CAN-BUS

Timing slow and fast

Timing and Transmission Control	ByteAddress	Access Type	
Timing for Arbitration Baudrate	0x0014	w	See next page
Timing for FD Data Baudrate	0x0018	w	See next page
Timing for XL Data Baudrate	0x0060	w	See next page
TransmitterDelay XL	0x0064	w	See below
TransmitterDelay FD	0x001C	w	See below

- TransmitterDelay[31..16] : Transmitter Delay in can_clocks
- for ISO and for non-ISO
 - § TransmitterDelay [15] : enable Transmitter Delay (set to 1 when using BRS)
 - § TransmitterDelay [14] : absolute Transmitter Delay (do not add to measured Transmitter Delay)
 - § TransmitterDelay [12..0] : TXdelay in can_clock ticks, set to SSP of fast bit

Timing Arbit, FD and XL

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	SJW 6	SJW 5	SJW 4	SJW 3	SJW 2	SJW 1	SJW 0	Prescale 7
	write	SJW 6	SJW 5	SJW 4	SJW 3	SJW 2	SJW 1	SJW 0	Prescale 7
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	Prescale 6	Prescale 5	Prescale 4	Prescale 3	Prescale 2	Prescale 1	Prescale 0	Timea 8
	write	Prescale 6	Prescale 5	Prescale 4	Prescale 3	Prescale 2	Prescale 1	Prescale 0	Timea 8
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	Timea 7	Timea 6	Timea 5	Timea 4	Timea 3	Timea 2	Timea 1	Timea 0
	write	Timea 7	Timea 6	Timea 5	Timea 4	Timea 3	Timea 2	Timea 1	Timea 0
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	Timeb 7	Timeb 6	Timeb 5	Timeb 4	Timeb 3	Timeb 2	Timeb 1	Timeb 0
	write	Timeb 7	Timeb 6	Timeb 5	Timeb 4	Timeb 3	Timeb 2	Timeb 1	Timeb 0

- SJW [6..0] : SynchronisationJumpWidth Register 0..127 => 1..128 tq
- **Prescale [7..0]** : Prescale Counter Register 0..255 => 2..257
- **Timea [8..0]** : Timing Segment A Register 0..511 => 1..512 tq (Arbit only)
0..255 => 1..256 tq (FD and XL)
- Timeb [7..0] : Timing Segment B Register 0..254 => 2..256 tq
(-1) 255 => 1 tq

Timing Settings for CAN

- The prescale counter divides your clock frequency
 - The number you fill in is incremented by 2
 - Example: the clock frequency is 50 MHz → 20 ns for each clock period. If you write a 2 to the prescale → division factor = 4 → 80 ns for each time segment
- The bit length for the CAN transmission rate is
 - 1 time segment for Sync
 - $(\text{Timea}+1) \cdot \text{time segment before samplepoint}$
 - $(\text{Timeb}+2) \cdot \text{time segment after samplepoint}$

Sync Segment	Timeslot before Samplepoint	Timeslot after Samplepoint	
		Timeb – SJW + 1	SJW + 1
1	Timea + 1	Timeb + 2	
Samplepoint →			
←		Total Bit Time	
		→	

Timing Settings for CAN

■ Example:

- You want to transmit with 500k baudrate
- $1 / 500k \rightarrow 2000$ ns total bit time
- If the length of one time segment is 80 ns
 - Divide $2000 / 80 \rightarrow 25$ segments are necessary
- $25 - 1$ segments for sync $\rightarrow 24$ segments
- The relationship between timea and timeb is responsible for the samplepoint
- If you chose 14 for (timea +1) and 10 for (timeb +2)
 - Your samplepoint will be at 60% of the bit time
- Timea has to be written with the value 13
- Timeb has to be written with the value 8

■ Synchronisation Jump Width

- Default value 0 \rightarrow Resynchronisation Jump is 1 Time segment
- value 3 \rightarrow Resynchronisation Jump is 4 Time segments
- Timeb includes the Synchronisation Jump Width

PWM timing

Timing and Transmission Control	ByteAddress	Access Type	
PWM	0x0068	w	PWM

- TransmitterDelay[31..16] : Transmitter Delay in can_clocks
- for ISO and for non-ISO
 - § TransmitterDelay [15] : enable Transmitter Delay (set to 1 when using BRS)
 - § TransmitterDelay [14] : absolute Transmitter Delay (do not add to measured Transmitter Delay)
 - § TransmitterDelay [12..0] : TXdelay in can_clock ticks, set to SSP of fast bit

Status & Command Error & transmission

Status & Command for ErrorCounter and Transmission Controll	ByteAddress	Access Type	
ErrorCounter	0x0020	h	See below
Additional ErrorCounter	0x0024	h	See below
Transmission suspend	0x0028	w	[23..0] default transmission suspend in us
Transmission repeatcount	0x002C	w	[15..0] default transmission repeatcount

- ErrorCounter [23..16] : receive error counter using CAN rules
- ErrorCounter [8..0] : transmit error counter using CAN rules

- ErrorCounterII [31..24] : count errors during dataphase if transmitter (fast Baudrate) [1]
- ErrorCounterII [23..16] : count errors during arbitrationphase if transmitter (slow Baudrate) [1]
- ErrorCounterII [15.. 8] : count errors during dataphase if receiver (fast Baudrate) [1]
- ErrorCounterII [7.. 0] : count errors during dataphase if receiver (slow Baudrate) [1]
- these 4 ErrorCounterII are resetted with setting the #reset errorcounter II bit in main status and control or setting normal-Bit

- programmable Transmission RepeatCount
 - 0 => endless repeat of transmit message until successful transmit (default)
 - 1 => just one try => single shot mode, no repeat on error (used for MilCAN for example)
 - 2 => repeat ones, when first try was not successful
 - <value> => make maximum <value> attempts to transmit message global value or per individual message

Status & Command Traffic & Timestamp

Status & Command for Bus Traffic and Timestamp	ByteAddress	Access Type	
Bus Traffic Rate	0x0030		See below
Timestamp Controll	0x0034		See below
Actual running TimestampCounter	0x0038	r	Actual value of Timestamp Counter (32 Bit)
Last Timestamp	0x003C	r	Timestamp of last SOF / ACK (32 Bit)

- BusTraffic [31] : s restart bus traffic rate by writing 1
- BusTraffic [30] : w 1 => count frames (SOF), else measure Rate
- BusTraffic [15.. 0] : h Traffic rate in % 0xFFFF => 100 %
- BusTraffic [29 .. 0] : h Frames since restart when BusTraffic[30] set

- TimestampControll [31] : s TimestampCounter Overrun into Bit32, clear by writing 1
- TimestampControll [3] : w 1 => select SOF as trigger for TimeStamp, else use ACK as Trigger
- TimestampControll [2] : w use TimeStamp_Clock input as Timestamp clock, else 1 us is used
- TimestampControll [1] : w use TimeStamp_reset input as Timestamp reset
- TimestampControll [0] : s Reset Timestamp Counter by writing

Compiler Parameter (read-only)

Byte 3	Bit	31	30	29	28	27	26	25	24
	read								
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	no PWM	Timestamp external	ID_format 1	ID_format 0	no Busstatistic	no Filter	no Timestamp	Use Single Clock
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	RFifo 7	RFifo 6	RFifo 5	RFifo 4	RFifo 3	RFifo 2	RFifo 1	Rfifo 0
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	TFifo 7	TFifo 6	TFifo 5	TFifo 4	TFifo 3	TFifo 2	TFifo 1	TFifo 0

- ID_format [1.. 0] : 00 IFI_legacy
: 01 CANalyzer
: 10 ID_other
- no Busstatistic when 1, else Busstatistic enabled
- no Timestamp when 1, else Timestamp enabled
- no PWM when 1, else PWM for XL fast mode is implemented
- use Single Clock when 1, else Dual_Clock enabled
- RFifo [7..0] receive Fifo size in kByte
- TFifo [7..0] transmit Fifo size in kByte
- Timestamp-values (64 Bit) from external when 1, else internal 32 counter used

more Compiler Parameter

	ByteAddress	Access Type	
More Parameter	0x004C	r	Read only
CAN_Clock	0x004C	r	Frequency of connected CAN_Clock in Hz
System Clock	0x0050	r	Frequency of connected System_Clock in Hz
Revision	0x0054	r	See below
IP-core ID	0x0058	r	0xD073CA58 for IFI CAN_XL IP-core
Testregister	0x005C	w	Write bit invers Resetvalue 0xAFFEDEAD

- Revision [31.. 24] : Month of IP-core Revision 1 .. 12
 - Revision [23 .. 16] : Year of IP-core Revision 20 ..
 - Revision [15 .. 8] : Minimum Version of QuartusII 91 .. for 9.1 ..
 - Revision [7 .. 0] : Core-Revision 10 .. for 1.0 ..
- 0x 0C 1b 5b 0A for December 2021 Quartus 9.1 Core 1.0

Status and Command, XL Filter

Status and Command XL Filter	ByteAddress	WordAddress	
XL Filter Enable + Masks	0x0060	24	Enable and Masks
XL Filter Bits	0x0064	25	[26..16] ID, [15..8] SDT bits, [7..0] VCID bits
AF Filter Mask	0x0068	26	Mask for Acceptance Field
AF Filter Bits	0x006C	27	Acceptance Field

■ XL Filter Valid – Mask

- [31] ID filter enable : 1 Priority ID Filter must match
- [30] SDT filter enable : 1 SDT Filter must match all enabled Filters must be valid (AND)
- [29] VCID filter enable : 1 VCID Filter must match
- [28] AF filter enable : 1 AF Filter must match

- [27] SEC Mask SEC 0: bit don't care 1: bit must match
- [26..16] ID Mask Priority Identifier 0: bit don't care 1: bit must match
- [15..8] SDT Mask Service Data Type 0: bit don't care 1: bit must match
- [7..0] VCID Mask Virtual CAN Network ID 0: bit don't care 1: bit must match

■ [31..0] Mask for the AF Filter Bits 0: bit don't care 1: bit must match

■ [31..0] AF Filter Bits Acceptance Field bits

Receive FIFO frame

	ByteAddress	EndAddress Access Type	
Receive FIFO	0x0600	0x0E1C	Read only
Statistik used repeatcount or TimeStamp bit 63..32	0x0600	r	Only when frame number > 0 to get used repeatcount External Timestamp bits 63..32 when enabled
TimeStamp bit 31..0	0x0604	r	
FrameNumber_Object	0x0608	r	[31..24] FrameNumber, [15..8] Object
Identifier	0x060C	r	See Receive FIFO ID
SDT_VCID_DLC	0x0610	r	[31..24] SDT, [23..16] VCID,... [10..0] DLC
AF	0x0614	r	[31..0] AF
PCRC	0x0618	r	[12..0] PCRC
FCRC	0x061C	r	[31..0] FCRC with XL or CRC21 or CRC17 or CRC15
DataBytes 4,3,2,1	0x0620	r	[31..24] Byte4, [23..16] Byte3, [15..8] Byte2, [7..0] Byte1
DataBytes 8,7,6,5	0x0624	r	..
Up to
DataBytes 64,63,62,61	0x065C	r	..
Up to	..		
DataBytes 2048,2047,2046,2045	0x0E1C	r	..

Receive Fifo FrameNumber_Object

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	Frn 7	Frn 6	Frn 5	Frn 4	Frn 3	Frn 2	Frn 1	Frn 0
Byte 2	Bit	23	22	21	20	19	18	17	16
	read								Obj 8
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	Obj 7	Obj 6	Obj 5	Obj 4	Obj 3	Obj 2	Obj 1	Obj 0
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	ESI	BRS	FDF (EDL)	RTR	DLC 3	DLC 2	DLC 1	DLC 0

- Frn [7.. 0] : Frame number for Transmit Timestamp
: > 0 => message received from own transmitter (to get the Timestamp)
: 0 => message received from external CAN-Node
- Obj[8..0] : Filter Object number of a received message
- ESI : Error State Indicator only with FDF
- BRS : Bit Rate Switch only with FDF
- FDF : FD Frame (EDL Extended Data Length) CAN-FD
- RTR : Remote Transmission Request not with FDF
- **DLC [3..0]** : **Data length code** **moved to SDT_VCID_DLC**, do not use DLC here

Receive Fifo SDT_VCID_DLC

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	SDT 7	SDT 6	SDT 5	SDT 4	SDT 3	SDT 2	SDT 1	SDT 0
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	VCID 7	VCID 6	VCID 5	VCID 4	VCID 3	VCID 2	VCID 1	VCID 0
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	XLF	SEC	BRS	FDF	RTR/RRS	DLC 10	DLC 9	DLC 8
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	DLC 7	DLC 6	DLC 5	DLC 4	DLC 3	DLC 2	DLC 1	DLC 0

- SDT [7.. 0] : Service Device Type
- VCID[7..0] : Virtual CAN Network ID
- **XLF** : **XL Frame**
- **SEC** : **Security protocol bit**
- BRS : Bit Rate Switch only with FDF
- FDF : FD Frame (EDL Extended Data Length) CAN-FD
- RTR/**RRS** : Remote Transmission Request not with FDF
- **RRS with XL**
- **DLC [10..0]** : **Data length code (0..2047 => 1 .. 2048 DataBytes) when XL Frame**
- **DLC [3..0]** : **Data length code (0 .. 8 DataBytes) or (0..64 DataBytes) when FDF set**

Identifier usage see selected ID-Format

Byte 3	Bit	31	30	29	28	27	26	25	24
	read			IDE	IDX 28	IDX 27	IDX 26	IDX 25	IDX 24
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	IDX 23	IDX 22	IDX 21	IDX 20	IDX 19	IDX 18	IDX 17	IDX 16
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	IDX 15	IDX 14	IDX 13	IDX 12	IDX 11	ID 10	ID 9	ID 8
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	ID 7	ID 6	ID 5	ID 4	ID 3	ID 2	ID 1	ID 0

- IDE : 1 => Use Extended Identifier
- IDX [28..11] + ID [10..0] : Extended Identifier
- ID [10..0] : Standard Identifier (Base) or (Priority with XL)

Transmit FIFO frame

	ByteAddress	EndAddress	
Transmit FIFO	0x0F00	0x171C	Write only
Transmit suspend in us	0x0F00	w	No write or -1 use default value
RepeatCount	0x0F04	w	No write or -1 use default value
FrameNumber for Timestamp	0x0F08	w	FrameNumber
Identifier	0x0F0C	w	See Transmit FIFO ID
SDT_VCID_DLC	0x0F10	w	[31..24] SDT, [23..16] VCID,... [10..0] DLC
AF	0x0F14	w	[31..0] AF
res1	0x0F18	w	Reserved 1
res2	0x0F1C	w	Reserved 2
DataBytes 4,3,2,1	0x0F20	w	[31..24] Byte4, [23..16] Byte3, [15..8] Byte2, [7..0] Byte1
DataBytes 8,7,6,5	0x0F24	w	..
Up to	...	w	..
DataBytes 64,63,62,61	0x0F5C	w	..
Up to	...	w	..
DataBytes 96,95,94,93	0x0F7C	w	Maximum with HighPriority
Up to	...	w	..
DataBytes 2048,2047,2046,2045	0x171C	w	

Transmit Fifo FrameNumber for Timestamp

Byte 3	Bit	31	30	29	28	27	26	25	24
	write	Frn 7	Frn 6	Frn 5	Frn 4	Frn 3	Frn 2	Frn 1	Frn 0
Byte 2	Bit	23	22	21	20	19	18	17	16
	write								
Byte 1	Bit	15	14	13	12	11	10	9	8
	write								
Byte 0	Bit	7	6	5	4	3	2	1	0
	write								

- Frn [7.. 0] : Frame number for Transmit Timestamp
 - : > 0 => enable Receiver without Filter to get Timestamp for own transmitted Frames
 - : 0 => no Timestamp for Transmission (default)

Transmit Fifo SDT_VCID_DLC

Byte 3	Bit	31	30	29	28	27	26	25	24
	write	SDT 7	SDT 6	SDT 5	SDT 4	SDT 3	SDT 2	SDT 1	SDT 0
Byte 2	Bit	23	22	21	20	19	18	17	16
	write	VCID 7	VCID 6	VCID 5	VCID 4	VCID 3	VCID 2	VCID 1	VCID 0
Byte 1	Bit	15	14	13	12	11	10	9	8
	write	XLF	SEC	BRS	FDF	RTR/RRS	DLC 10	DLC 9	DLC 8
Byte 0	Bit	7	6	5	4	3	2	1	0
	write	DLC 7	DLC 6	DLC 5	DLC 4	DLC 3	DLC 2	DLC 1	DLC 0

- SDT [7.. 0] : Service Device Type
- VCID[7..0] : Virtual CAN Network ID
- **XLF** : **XL Format**
- **SEC** : **Security Protocol**
- BRS : Bit Rate Switch only with FDF
- FDF (EDL) : FD Format (Extended Data Length) CAN-FD
- RTR /**RRS** : Remote Transmission Request not with FDF
- **RRS with XL**
- DLC [10..0] : Data length code (0..2047 => 1 .. 2048 DataBytes) with XL
- DLC [3..0] : Data length code (0 .. 8 DataBytes) or (0..64 DataBytes) when FDF set

Identifier usage see selected ID-Format

Byte 3	Bit	31	30	29	28	27	26	25	24
	write			IDE	IDX 28	IDX 27	IDX 26	IDX 25	IDX 24
Byte 2	Bit	23	22	21	20	19	18	17	16
	write	IDX 23	IDX 22	IDX 21	IDX 20	IDX 19	IDX 18	IDX 17	IDX 16
Byte 1	Bit	15	14	13	12	11	10	9	8
	write	IDX 15	IDX 14	IDX 13	IDX 12	IDX 11	ID 10	ID 9	ID 8
Byte 0	Bit	7	6	5	4	3	2	1	0
	write	ID 7	ID 6	ID 5	ID 4	ID 3	ID 2	ID 1	ID 0

- IDE : 1 => Use Extended Identifier
- IDX [28..11] + ID [10..0] : Extended Identifier
- ID [10..0] : Standard Identifier (Base) or (Priority with XL)

Frame Number for Transmit Timestamp

- If this number is set to 0x00, the feature is deactivated.
- Writing any number different to 0x00 activates this feature.
- If there is a number between 0x01 and 0xFF, the successful transmit of the message will be timestamped and flagged like a normal receive. Within the receive buffer the register contains this number and the object number will be 0x00. This information is not depending from a filter setting.
- Reading a number greater 0x00 means the message is not a message coming from another transmitter, it's his own transmitted message.

Error Controller

Byte 3	Bit	31	30	29	28	27	26	25	24
	read		Bitposition 14	Bitposition 13	Bitposition 12	Bitposition 11	Bitposition 10	Bitposition 9	Bitposition 8
	write	ER enable	ER reset						
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	Bitposition 7	Bitposition 6	Bitposition 5	Bitposition 4	Bitposition 3	Bitposition 2	Bitposition 1	Bitposition 0
	write								
Byte 1	Bit	15	14	13	12	11	10	9	8
	read		Form Error all	CRC Error all	Stuff Error all	Bit 1 Error all	Bit 0 Error all	Ack Error all	Overload all
	write								
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	highbaud	Form Error first	CRC Error first	Stuff Error first	Bit 1 Error first	Bit 0 Error first	Ack Error first	Overload first
	write								

- Bitposition [14..0] : Number of the Bit, when the first error occurred

Error Controller Usage

- **ER enable**
 - Writing 1 → Feature enabled
 - Writing 0 → Feature disabled
- **ER reset**
 - Writing 1 → Reset the Error_Bits and the Bitposition to 0
- **Bitposition(14 downto 0)**
 - The number of the Bit in the ongoing frame where the first error occurred
- **XXX Error all**
 - The types of errors which occurred since the last reset
- **XXX Error first**
 - The type of error which occurred as the first error since the last reset

ID-Filter and Masks

ByteAddress	WordAddress	Contents		Object	
0x1800	1536	Valid	Mask0	1	Valid can switch on or off any object. The object number, with a match between the filter and a received message, is readable in receive fifo
0x1804	1537	Valid	ID0		
0x1808	1537	Valid	Mask1	2	
0x180C	1538	Valid	ID1		
0x1810			.		
			.		
			.		
0x1FF8	2046	Valid	Mask255	256	
0x1FFC	2047	Valid	ID255		

Filter mask

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	valid	Mask EDL	Mask extd	Mask 28	Mask 27	Mask 26	Mask 25	Mask 24
	write	valid	Mask EDL	Mask extd	Mask 28	Mask 27	Mask 26	Mask 25	Mask 24
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	Mask 23	Mask 22	Mask 21	Mask 20	Mask 19	Mask 18	Mask 17	Mask 16
	write	Mask 23	Mask 22	Mask 21	Mask 20	Mask 19	Mask 18	Mask 17	Mask 16
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	Mask 15	Mask 14	Mask 13	Mask 12	Mask 11	Mask 10	Mask 9	Mask 8
	write	Mask 15	Mask 14	Mask 13	Mask 12	Mask 11	Mask 10	Mask 9	Mask 8
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	Mask 7	Mask 6	Mask 5	Mask 4	Mask 3	Mask 2	Mask 1	Mask 0
	write	Mask 7	Mask 6	Mask 5	Mask 4	Mask 3	Mask 2	Mask 1	Mask 0

■ Valid : 1 → use this mask

0 → ignore this mask

■ Mask : 1 → compare this bit

0 → ignore the value of the bit

Filter Identifier

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	valid	CANFD	IDE	IDX 28	IDX 27	IDX 26	IDX 25	IDX 24
	write	valid	CANFD	IDE	IDX 28	IDX 27	IDX 26	IDX 25	IDX 24
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	IDX 23	IDX 22	IDX 21	IDX 20	IDX 19	IDX 18	IDX 17	IDX 16
	write	IDX 23	IDX 22	IDX 21	IDX 20	IDX 19	IDX 18	IDX 17	IDX 16
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	IDX 15	IDX 14	IDX 13	IDX 12	IDX 11	ID 10	ID 9	ID 8
	write	IDX 15	IDX 14	IDX 13	IDX 12	IDX 11	ID 10	ID 9	ID 8
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	ID 7	ID 6	ID 5	ID 4	ID 3	ID 2	ID 1	ID 0
	write	ID 7	ID 6	ID 5	ID 4	ID 3	ID 2	ID 1	ID 0

- Valid : 1 → use this identifier 0 → ignore this identifier
- IDE : 1 → use Extended Identifier
- CANFD: 1 → use CANFD Frame
- IDX [28..11] + ID [10..0] : Extended Identifier

Mask and Filter

Filter standard / extended message

Bit Mask extd	0	1	1	1	1
Bit Filter IDE	X	0	1	0	1
IDE-received message	X	0	1	1	0
result	Match Standard or Extended ID	Match Standard ID	Match Extended ID	No match	No match

Filter normal / CAN-FD message

Bit Mask EDL	0	1	1	1	1
Bit Filter CANFD	X	0	1	0	1
EDL-received message	X	0	1	1	0
result	Match Normal or CAN- FD message	Match Normal message	Match CAN-FD message	No match	No match

Filter and Mask Example IFI_legacy

- For filtering on 60 to 65 you need 2 ID and mask entries
- Looking for standard IDs (not extended ID)
- For ID 60-63
 - `IOWR_IFI_NIOS_CAN_BUFFER(base, 0, 0xBFFFFFFC); //mask 60-63`
 - `IOWR_IFI_NIOS_CAN_BUFFER(base, 1, 0x8000003C); //ID 60-63`
- For ID 64-65
 - `IOWR_IFI_NIOS_CAN_BUFFER(base, 2, 0xBFFFFFFE); //mask 64-65`
 - `IOWR_IFI_NIOS_CAN_BUFFER(base, 3, 0x80000040); //ID 64-65`

ID Format Order of the ID Bits

- ID-format can be selected as Compiler Parameter when generating the hardware
 - the same format is used for
 - transmit fifo
 - receive fifo
 - filter and masks
 - MSB (left Most Significant Bit) is transmitted first
- the combination of the ID-Bits for **Normal** and **Extended** can be written in 3 different cases
- IFI_legacy (used with IFI CAN-IP and IFI Advanced CAN-IP)
 - the first transmitted ID-bits are on Bit-Positions 10 downto 0 => ID[10..0]
 - for extended Frames the Bit-Positions 28 downto 11 are transmitted last => ID[10..0],IDX[28..11]
 - ID[10] first
- CANalyzer
 - for normal Frames ID-bits from Bit-Position 10 downto 0 are transmitted => ID[10..0]
 - ID[10] first
 - for extended Frames the Bit-Positions 28 downto 0 are transmitted => ID[28..0]
 - ID[28] first
- ID_other (the 3rd used order of ID Bits)
 - the first transmitted ID-Bits are on Bit-Positions 28 downto 18 => ID[28..18]
 - for extended Frames the Bit-Positions 17 downto 0 are transmitted last => ID[28..0]
 - ID[28] first

Order of the ID Bits Convert

■ Convert extended ID from CANalyzer to IFI_legacy

- $\text{transmitID} = ((\text{value} \& 0x3FFFF) \ll 11) + ((\text{value} \& 0x1FFC0000) \gg 18);$
- $\text{transmitID} |= 0x20000000; \quad // \text{ set IDE for extended ID !!!}$

■ Convert extended IFI_legacy to CANalyzer

- $\text{value} = ((\text{receiveID} \& 0x7FF) \ll 18) + ((\text{receiveID} \& 0x1FFFF800) \gg 11);$
- $\text{value} \&= \sim 0x20000000; \quad // \text{ mask the IDE}$

Software for NIOS II and IDE

■ Files

- └─ ifi_can_xl-v1.0
 - └─ doc
 - └─ lib
 - └─ HAL
 - └─ inc
 - └─ ifi_canxl.h
 - └─ src
 - └─ ifi_canxl.c
 - └─ component.mk
 - └─ inc
 - └─ ifi_canxl_regs.h
 - └─ can_xl_wizard.lst
 - └─ License
 - └─ ref_designs
 - └─ software
 - └─ ifi_hello_canxl
 - └─ ifi_hello_canxl.c
 - └─ readme.txt
 - └─ template.xml
 - └─ ifi_canxl-v1.0.ipx



CAN driver routines (ifi_can_xl.c)

- **ifi_can_xl_open ()**
 - Initialize of the CAN node
 - Base, pointer to structure fdcanall_s with timing, interrupt mask, status, mask and filter
 - Return the version of the core
- **ifi_can_xl_read ()**
 - read a message from the buffer and increment the buffer pointer
 - Base, pointer to structure fdcanmsg_s with identifier, data[0], data[1...], dlc,timestamp
 - Return 1 for successful read, -1 for error
- **ifi_can_xl_write ()**
 - Transmit a message
 - Base, pointer to structure fdcanmsg_s with identifier, data[0], data[1...], dlc
 - Return 1 for successful read, -1 for error
- **ifi_can_xl_stat ()**
 - Read the interrupt mask register, the status register and error register
 - Base, pointer to structure fdcanstat_s with interruptmask, status, error
 - Return 0 for successful read, -1 for error
- **ifi_can_xl_wr_int ()**
 - Enable the interrupts
 - Base, Interruptregister
 - Return 0 for successful write, -1 for error
- **ifi_can_xl_wr_status ()**
 - Write the status register only
 - Base, statusregister
 - Return 0 for successful write, -1 for error
- **ifi_can_xl_wr_filter ()**
 - Write a single mask and filterpair
 - Base, filternumber, filter mask, filter identifier
 - Return 0 for successful write, -1 for error
- **ifi_can_xl_irq ()**
 - Install the interrupts
 - Base, Irq number, pointer to structure fdcapture_s with base, status, irqcount, irqdone
 - Return 0 for successful install

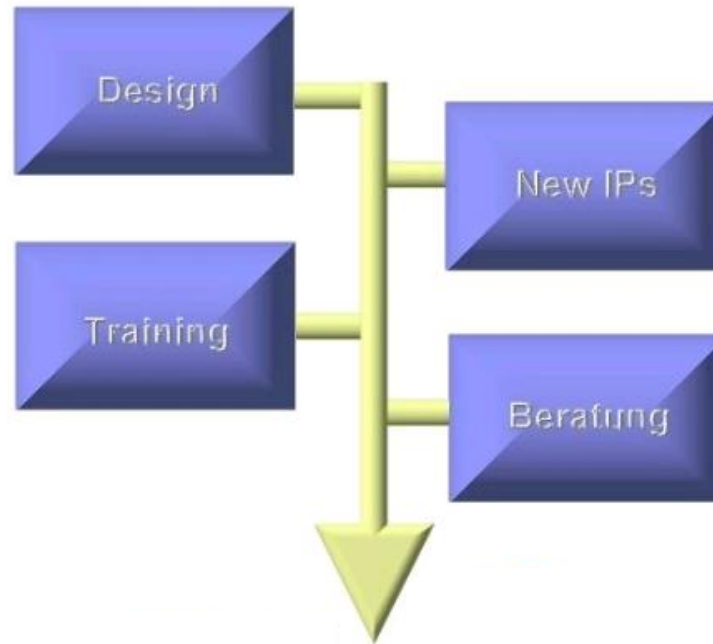
Software Examples

■ ifi_hello_can_xl.c

- A simple program which demonstrates a communication between 2 CAN nodes in XL mode
- ping / pong per interrupt

Dedicated to FPGA since 1985

- ✓ **Quartus® Training Classes**
 - QUARTUS-advanced
 - QUARTUS-Expert
- ✓ **VHDL Training Classes**
- ✓ **QSYS Training Classes**
 - NIOS®II / QSYS
- ✓ **Location**
 - Inhouse or Wertheim



- ✓ **DesignServices for FPGA**
- ✓ **IP**
 - CAN_XL Controller
 - Gigabit Ethernet MAC
 - ...
- ✓ **Consulting**

FPGA Training



INGENIEURBÜRO **F**ÜR **I**C-TECHNOLOGIE

Franz Sprenger
Am Tannenbergr 28
97877 Wertheim
Germany
Tel.: (+49) 9342 / 91091

eMail: ifi@ifi-pld.de
<http://www.ifi-pld.de>

formerly known as: Peter Riekert & Franz Sprenger

