



# IFI Advanced CAN Module

## *User Guide*

Core Version: 04095A55  
Document Date: 11.2016

# Contents

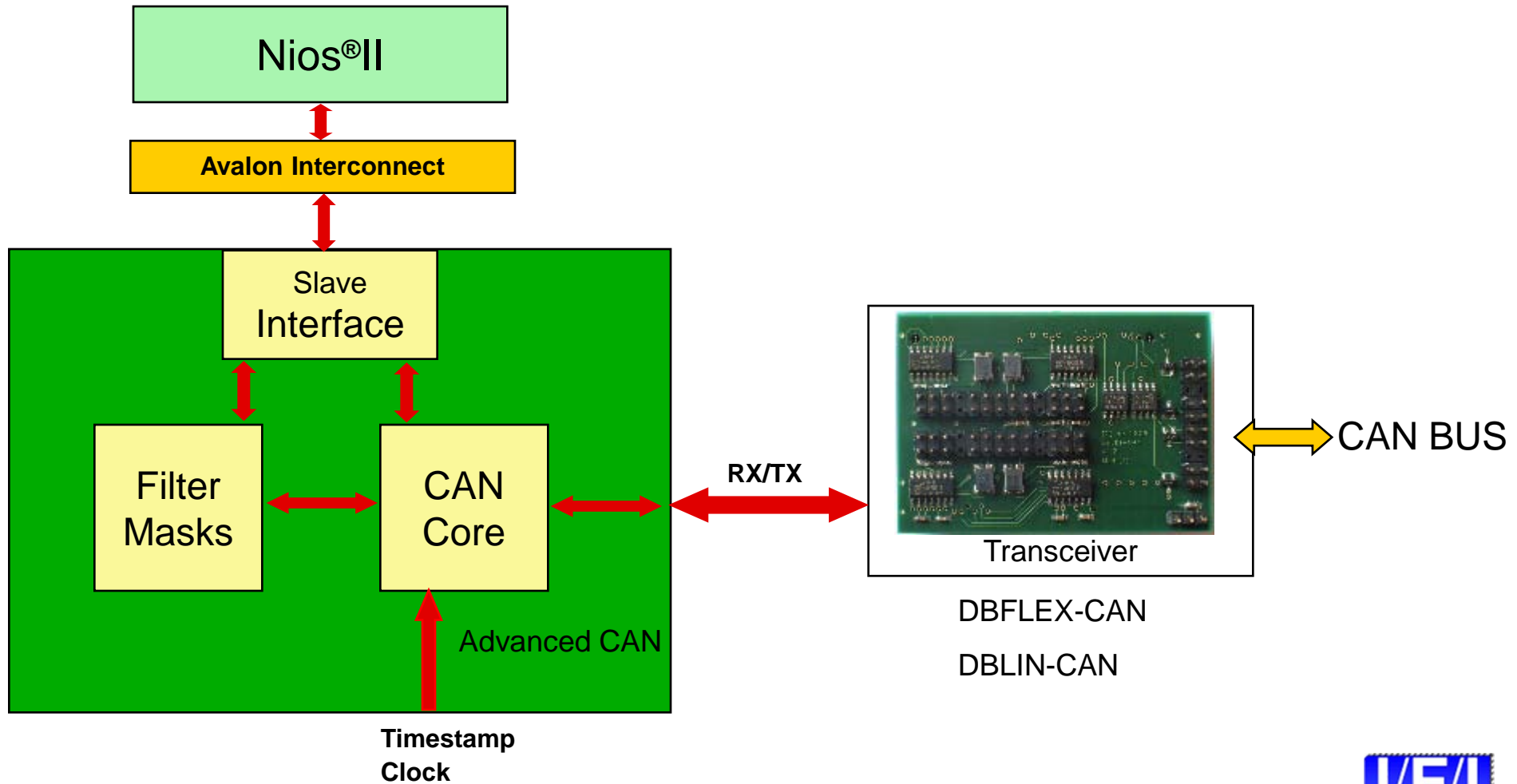
- *Overview*
- *Install*
- *Integrating the Core using SOPC Builder*
- *Integrating the Core using QSYS Interconnection*
- *Reference Designs*
- *Using the Core without Nios/SOPC/QSYS*
- *Detailed Information*
- *VHDL Testbench*
- *License Agreement*



# Overview

- *Block Diagram*
- *Feature List*
- *Altera Implementation*
- *OpenCore Plus Feature*
- *Using the Core without External Hardware*
- *Using Signal Tap II*
- *Options*
- *Pricing*
- *References*
- *CAN Background*
- *Contacting Technical Support*

# Block Diagram



# IFI Advanced CAN Feature List

- CAN 2.0B
  - Standard or Extended Identifier
  - Remote Frames
  - Error-Handling
- Up to 254 Messages Transmit Buffer
  - FIFO Pointer accessible
- Up to 4096 Messages Receive Buffer
  - FIFO Pointer accessible
- Up to 256 Message Filters
  - Every Message Filter contains one MASK- and one Identifier-Register
- Avalon Interface
  - Example Software included
  - HAL Drivers for NIOS II included

# IFI Advanced CAN Feature List

- Silent Mode
- High Priority Message
- 32 Bit Timestamp:
  - for received messages
  - for transmitted messages with frame number other than 0
- For external CPU support :
  - 8,16 or 32 Bit Interface
- Read of the Compile Time Parameters possible

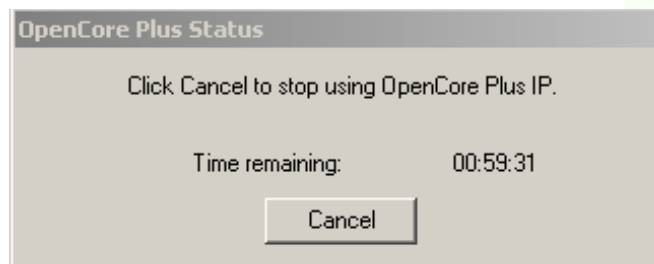
# IFI Advanced CAN Implementation

- Design Flows supported
  - QSYS/SOPC Builder
  - Megawizard
- Device Families targeted (depending on date of purchase)
  - CYCLONE
  - STRATIX
  - ARRIA
  - MAX 10
- Minimal Device Resource Utilization
  - about 2500 LE + 4 M9K for Cyclone III, Cyclone IV, MAX 10
  - about 1200 ALM + 4 M10K for Stratix, Arria, Cyclone V
- Maximum Device Resource Utilization
  - ~ 100 LEs additional
  - Number of Ram blocks depend on the user setting

# OpenCore Plus Feature

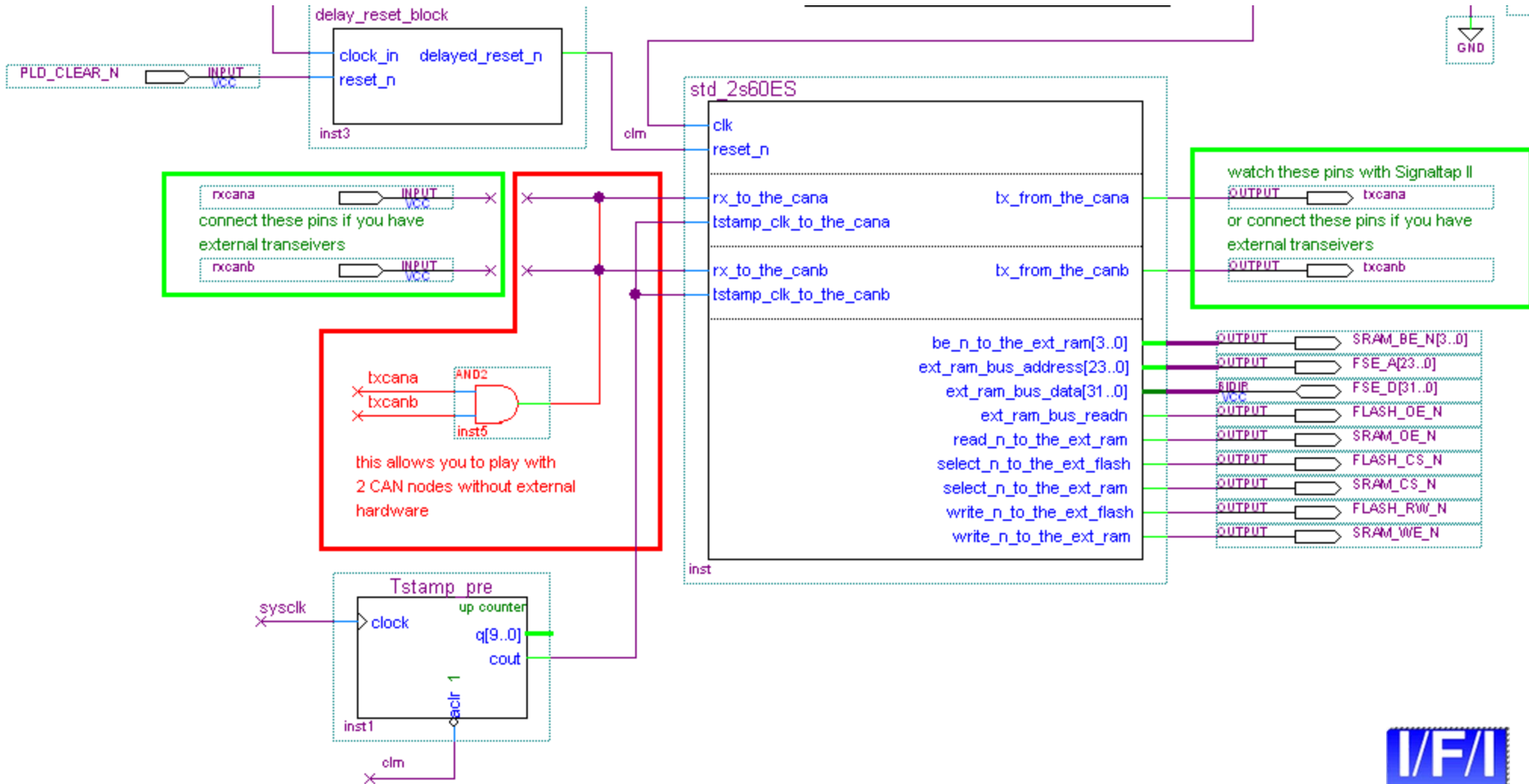
## ■ Test the CAN Module on your board

- 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.

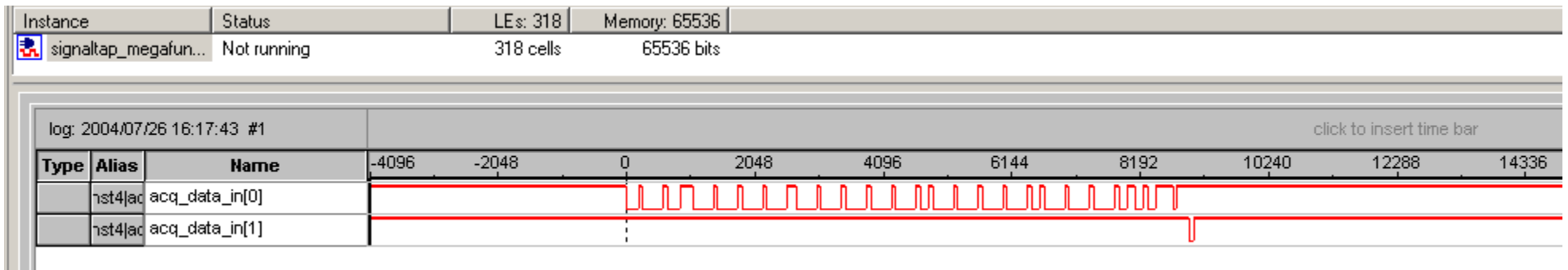




# IFI Advanced CAN Reference Design



# IFI Advanced CAN Reference Design



- Use SignalTap II to watch the TX Pins

# IFI Advanced CAN Pricing

- Node-Locked License: 7500.-- €
  - 1 year Maintenance included
  - T-Guard or NIC-ID
  - Maintenance: 750.-- €/ year
- Floating License: 9375.-- €
  - 1 year Maintenance included
  - Single or Multi Server
  - Maintenance: 937.-- €/ year
- Licensing:
  - Unlimited NIC-ID License
  - Royalty Free with IFI
  - The CAN-NETWORK-PROTOCOL-License is not included
    - Available by Bosch (Bosch charges royalties)

# IFI NIOS® II Advanced CAN

## ■ Hardware Tested on

- DBC1C12 / DBC2C20 / DBC3C40 Cyclone Development Board
- Cyclone / Cyclone II NIOS Development Kit
- Cyclone III Starter Kit
- Stratix / StratixII NIOS Development Kit
- Against
  - Each other
  - Vector CANalyzer
  - Other CANcontrollers

## ■ The IFI Advanced CAN is running successfully in many customer projects

## ■ The IFI CAN IP Core 6.8 passed the ISO CAN conformance tests.

- ISO 16845:2004 Road vehicles-Controller area network (CAN) – Conformance test plan
- C&S enhancement / corrections
- Reference: CAN CONFORMANCE TESTING Test Specification C&S Version 1.5



# CAN-Background

## ■ CAN Messages:

- Every CAN message consist of a certain number of bits that are divided into fields. There are fields like Arbitration Field, Data Field, CRC, 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).
- 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 from a complementary bit.
- Error Frame and the Overload Frame are of a fixed form and not coded with bit stuffing.

## ■ Error Detection:

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

# CAN-Background

- CRC Error
- Form Error
- ACK Error

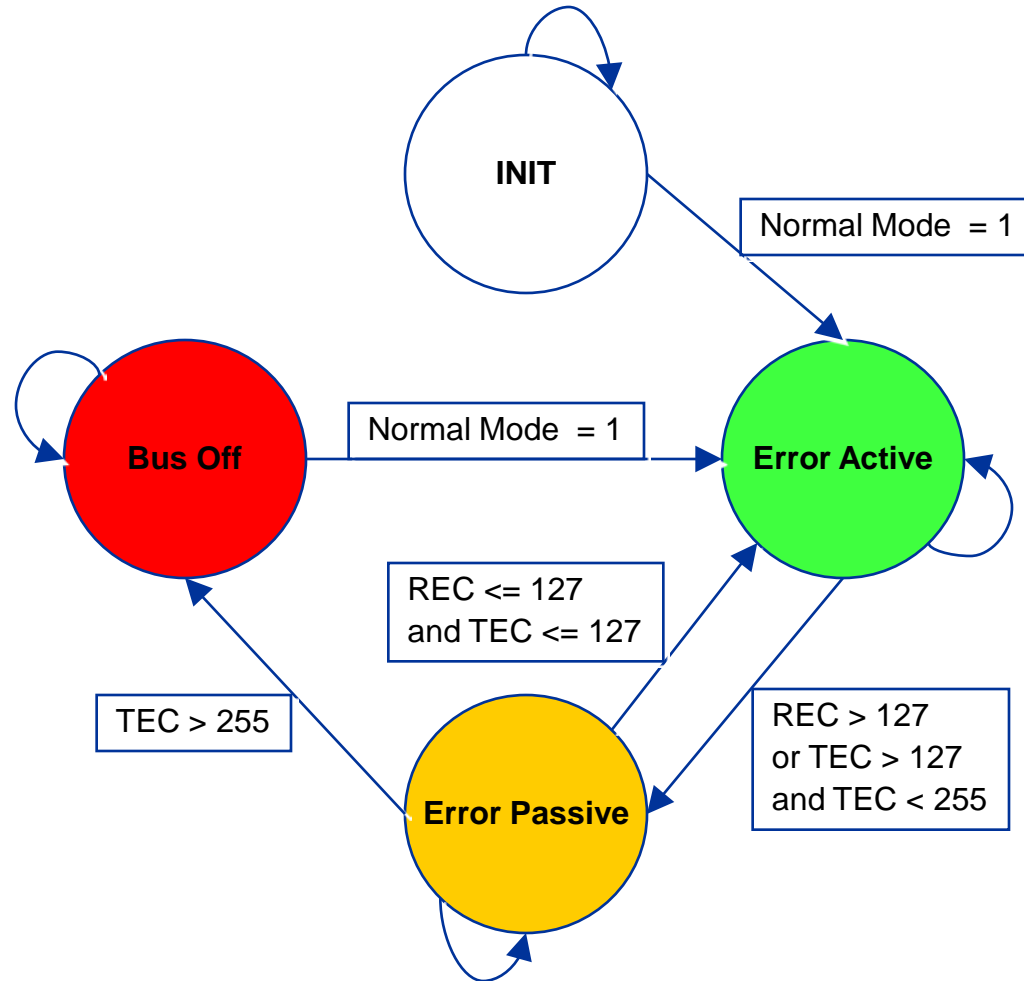
## ■ Error Handling:

- Error detected
- Transmit 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 SOPC Builder Ready 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.

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

IFI Ingenieurbüro **F**ür **I**c-Technologie  
F. Sprenger  
Kleiner Weg 3 -- 97877 Wertheim -- Germany  
Phone: (+49)9342/96080  
E-Mail: [ifi@ifi-pld.de](mailto:ifi@ifi-pld.de)  
<http://www.ifi-pld.de>





# Install

- *How to install*
- *QSYS/SOPC Builder Ready OpenCore Package*
- *Licensing*
- *Set up Licensing*



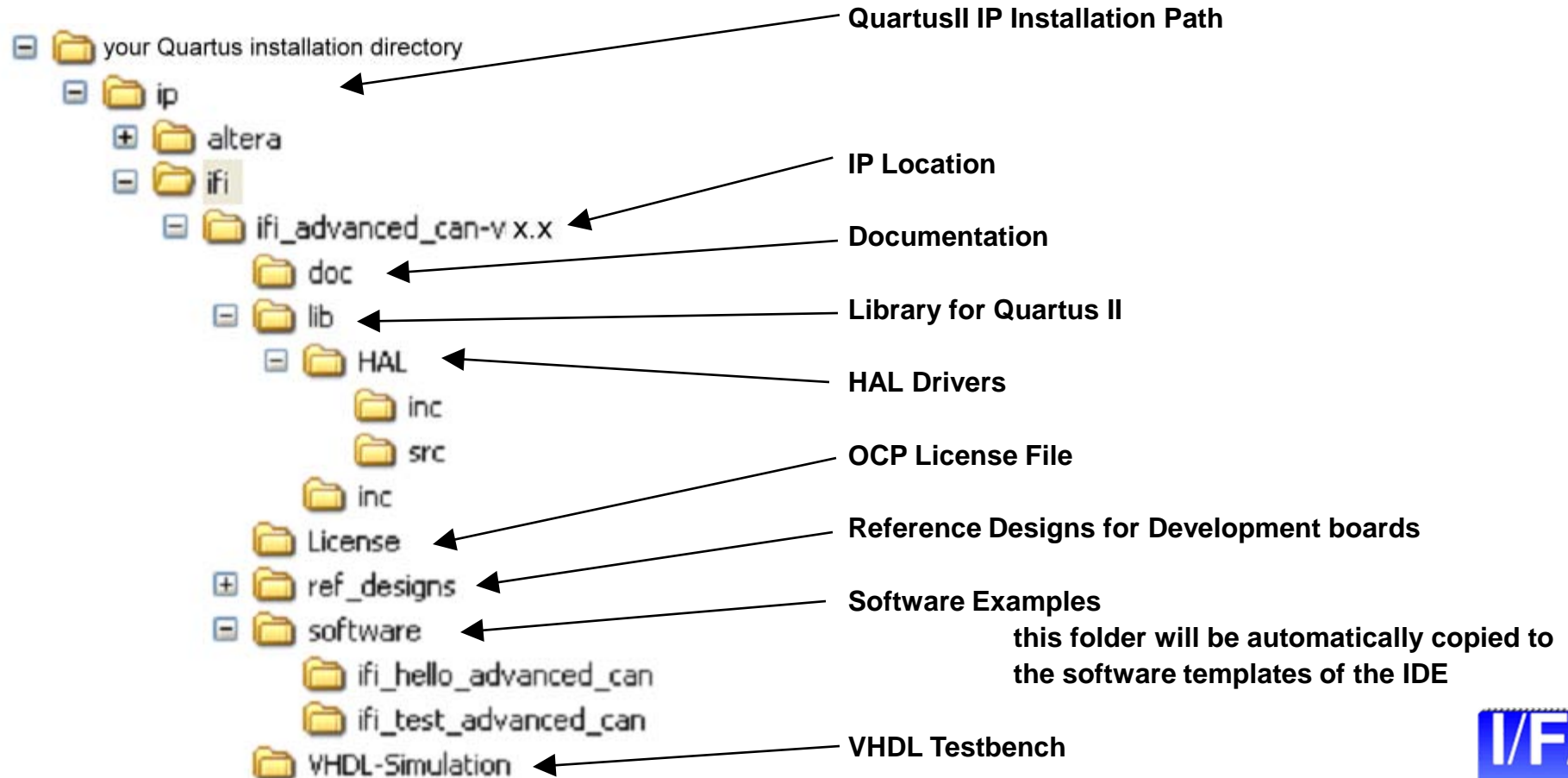
# Install the IFI Advanced CAN

- Before you can start using IFI Advanced CAN functions, you must install the IFI Advanced CAN files on your computer. The following instructions describe this process.
- **Close QuartusII and IDE.**
- **The installed QuartusII version must be 9.1 or newer**
- **Install the IFI Advanced CAN Files**
  - The following instructions describe how you install IFI Advanced CAN on computers running the Windows, Linux, or Solaris operating systems.
  - IF you don't change the installation path, the QSYS/SOPC Builder and the Megawizard will find the IP automatically
  - Windows
    - Follow these steps to install the IFI Advanced CAN 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 Advanced CAN function and <filename> is the filename of the IFI Advanced CAN function.
    - Click OK. The IFI Advanced CAN Installation dialog box appears. Follow the on-screen instructions to finish installation.
  - Solaris & Linux
    - Follow these steps to install the IFI Advanced CAN 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 Advanced CAN function.
    - Extract the package by typing the following command:
      - `tar xvf <filename>.tar`



# SOPC/QSYS Ready OpenCore Package

The SOPC Builder Ready OpenCore Package contains all files required for plug-and-play integration of this core into SOPC and QSYS tools, allowing the user to easily evaluate the core within his Avalon-based system.



# CANopen Protocol Software



- There is an additional folder included
  - CANopen\_slave\_IFI\_Advanced\_CAN
- In this folder you can find a software demonstration from the IXXAT CANopen slave, adapted to the I/F/I Advanced Can.
- Further documentation you will find in the doc folder as:
  - WP112-0010\_CANopen\_Protocol\_Software\_Quick\_Start\_Guide\_IFI\_CAN.pdf



# Licensing

## ■ OpenCorePlus License

This package is shipped with a OpenCorePlus license or the license is attached to email <Core installation directory>\license\license\_ocp.dat.

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

The license permit generation of <revision\_name>\_time\_limited.sof files.

The hardware evaluation feature will run during you have an established connection between your board and the QuartusII programmer. 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 VHD file can be read into Quartus II and place and route can be performed. The license permit 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 Advanced CAN 's **license\_ocp.dat** file in the Quartus II software.
  - Before you set up licensing for the IFI Advanced CAN , you must already have the Quartus II software installed on your computer with licensing set up.
- **Append the License to Your license.dat File**
  - To append the license, follow these steps:
  - Open the IFI Advanced CAN 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 Advanced CAN 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 Advanced CAN 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 SOPC Builder

- *Adding the Core to your System*
- *About*
- *Documentation*
- *Parameterize*

# Adding the Core to your System

- This walkthrough involves the following steps:
  - Create a New Quartus II Project
  - Create a New SOPC Builder Design
  - Launch IP Toolbench
    - Step 1: Parameterize
    - Step 2: Generate
- **Create a New Quartus II Project**
  - Before you begin, you must create a new Quartus II project. With the New Project wizard, you specify the working directory for the project, assign the project name, and designate the name of the top-level design entity. You will also specify the IFI Advanced CAN IP user library. To create a new project, follow these steps:
    - Choose **Programs > Altera > Quartus II <version>** (Windows Start menu) to run the Quartus II software or
    - Choose **Programs > intelFPGA > Quartus II <version>** (Windows Start menu) to run the Quartus II software
    - Choose **New Project Wizard** (File menu).
    - Click **Next** in the introduction (the introduction will not display if you turned it off previously).
    - Specify the working directory for your project. This walkthrough uses the directory **c:\qdesigns\myproject**.
    - Specify the name of the project. This walkthrough uses **myproject**.
    - Click **Next**.
    - Click **User Libraries...**
    - Type **<path>\ifi\_advanced\_can-v9.1<version>\lib\** into the **Library name** box, where **<path>** is the directory in which you installed the IFI CAN IP.
    - Click **Add**.
    - Click **OK**.
    - Click **Next**.
    - Choose the target device family in the **Family** list.
    - Click **Finish**.
    - You have finished creating your new Quartus II project.

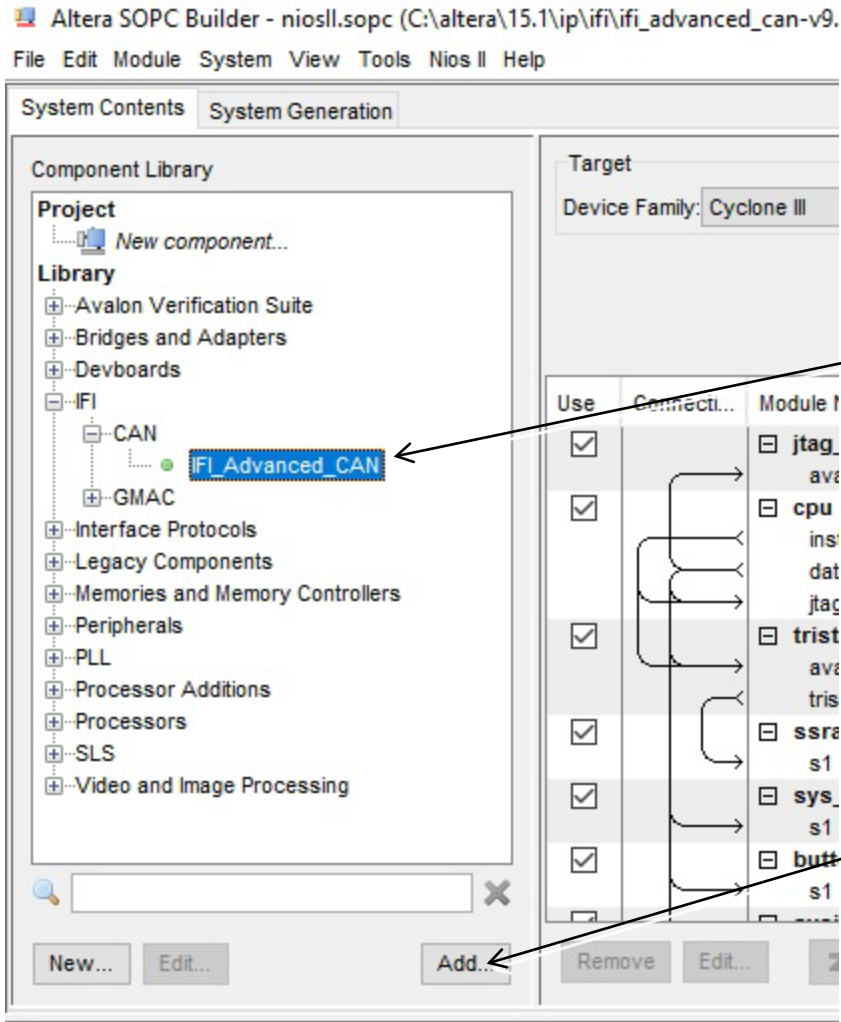




# Integrating the Core with your System using SOPC Builder

- This section contains instructions on the following:
  - Adding the Core to your System
  - Using ModelSim to Simulate the Core within your System
  - Running the Reference Design
- These instructions assume that the user is familiar with the
  - OpenCore evaluation process,
  - Quartus II development software,
  - and the SOPC Builder tool.
- For more information on these prerequisites, please visit *[www.altera.com](http://www.altera.com)*.

# Adding the Core to your SOPC System



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

IFI\_Advanced\_CAN - cana

**IFI**  
IFI\_Advanced\_CAN  
ifi\_avalon\_can\_advanced

Info

**Block Diagram**

**Timestamp**  
Timestamp:: Generate\_Timestamp\_ON

**Filter and Masks**  
Number of Filters and Masks used: 64

**Buffer:**  
Size of the Receive Fifo in Messages: 32  
Size of the Transmit Fifo in Messages: 30

**Baudrate:**

System Clock in Hz:	64000000
Startvalue Prescale:	2
Registervalue Prescale::	0
Startvalue Timesegment A:	25
Registervalue Time_a::	24
Startvalue Timesegment B:	6
Registervalue Time_b::	4
Baudrate KBaud::	1000
Sample Point %::	81

**External Interface**  
Datawidth: 32

**Endian-ness**  
Endian-ness:: Little\_Endian

Info: cana: Timestampcounter enabled  
Info: cana: 32 Bit Data Interface

Cancel Finish


Info + Documentation

Parameters

Information



IFI\_Advanced\_CAN - cana




**IFI\_Advanced\_CAN**  
ifi\_avalon\_can\_advanced

F. Sprenger

Info

**Block Diagram**



**Timestamp**

Timestamp::

**Filter and Masks**

Number of Filters and Masks used:

**Buffer:**

Size of the Receive Fifo in Messages:

Size of the Transmit Fifo in Messages:

**Baudrate:**

System Clock in Hz:

Startvalue Prescale:

Registervalue Prescale::

Startvalue Timesegment A:

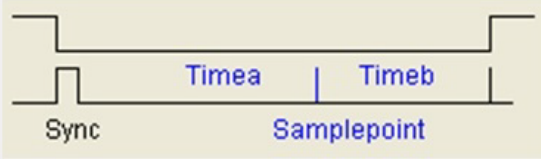
Registervalue Time\_a::

Startvalue Timesegment B:

Registervalue Time\_b::

Baudrate KBAud::

Sample Point %::



**External Interface**

Datawidth:

**Endian-ness**

Endian-ness::

Info: cana: Timestampcounter enabled

Info: cana: 32 Bit Data Interface

Cancel Finish

IFI\_Advanced\_CAN Info

## IFI\_Advanced\_CAN

**Class Name** ifi\_avalon\_can\_advanced

**Version** 9.1

**Author** IFI

**Description** IFI\_Advanced\_CAN - v9.1

**Group** IFI/CAN

**Data Sheet** [file:///C:/altera/15.1/ip/ifi/ifi\\_advanced\\_can-v9.1/doc/IFI\\_Advanced\\_CAN\\_docu.pdf](file:///C:/altera/15.1/ip/ifi/ifi_advanced_can-v9.1/doc/IFI_Advanced_CAN_docu.pdf)

**Timestamp**

**Timestamp:** ON or OFF

**Filter and Masks**

**Number of Filters and Masks used** 64, 128 or 256

**Buffer:**

**Size of the Receive Fifo in Messages** 32, 64, 128, 256, 512, 1024, 2048 or 4096

**Size of the Transmit Fifo in Messages** 30, 62, 126 or 254

**Baudrate:**

<b>System Clock in Hz</b>	Your System Clock in Hz
<b>Startvalue Prescale</b>	2 .. 255
<b>Registervalue Prescale:</b>	Registervalue Prescale
<b>Startvalue Timesegment A</b>	0 .. 31
<b>Registervalue Time_a:</b>	Registervalue Time_a
<b>Startvalue Timesegment B</b>	2 .. 31
<b>Registervalue Time_b:</b>	Registervalue Time_b
<b>Baudrate KBAud:</b>	Resulting Baudrate
<b>Sample Point %:</b>	Resulting Sample Point in %

**External Interface**


**Datawidth** 32 Bit, 16Bit or 8 Bit wide

**Endian-ness**

**Endian-ness:** Little\_Endian or Big\_Endian



IFI\_Advanced\_CAN - cana



**IFI\_Advanced\_CAN**  
ifi\_avalon\_can\_advanced

F. Sprenger Info

**Block Diagram**

```

    graph LR
      clock --> can_slave_clock
      conduit --> can_slave_export
      avalon --> can_slave
      interrupt --> slave_irq
      subgraph cana
        can_slave_clock
        can_slave_export
        can_slave
        slave_irq
      end
  
```

**Timestamp**

Timestamp::

**Filter and Masks**

Number of Filters and Masks used:

**Buffer:**

Size of the Receive Fifo in Messages:

Size of the Transmit Fifo in Messages:

**Baudrate:**

System Clock in Hz:	64000000
Startvalue Prescale:	2
Registervalue Prescale::	0
Startvalue Timesegment A:	25
Registervalue Time_a::	24
Startvalue Timesegment B:	6
Registervalue Time_b::	4
Baudrate KBaud::	1000
Sample Point %::	81

- Enable or Disable the Timestamp
- Select the number of Filters and Masks
- Receive buffer-size in messages
- Transmit buffer-size in messages
- Baudrate Calculator



**Block Diagram**

```

    graph LR
      clock --- can_slave_clock
      conduit --- can_slave_export
      avalon --- can_slave
      interrupt --- slave_irq
      subgraph cana
        can_slave_clock
        can_slave_export
        can_slave
        slave_irq
      end
  
```

**Timestamp**

Timestamp::

**Filter and Masks**

Number of Filters and Masks used:

**Buffer:**

Size of the Receive Fifo in Messages:

Size of the Transmit Fifo in Messages:

**Baudrate:**

System Clock in Hz:

Startvalue Prescale:

Registervalue Prescale::

Startvalue Timesegment A:

Registervalue Time\_a::

Startvalue Timesegment B:

Registervalue Time\_b::

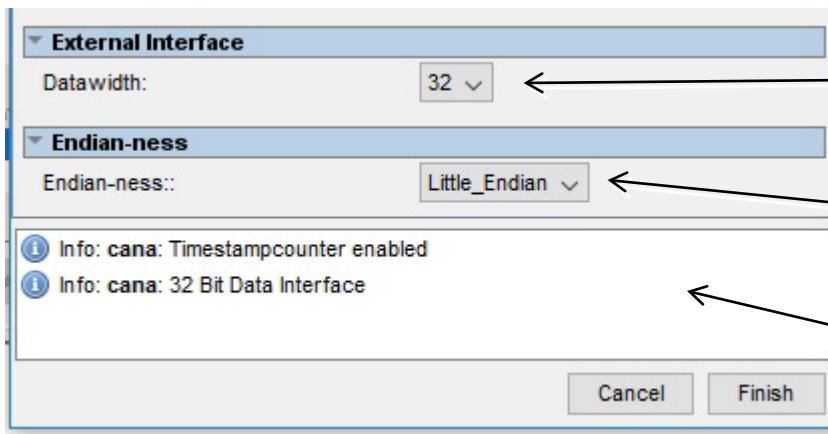
Baudrate KBaud::

Sample Point %::

- The CAN timing parameters are startup values and could be overwritten by software
- The clock frequency of your selected clock.
- The CAN timing is always a result of 3 parameters
  - Prescale
  - Timesegment A
  - Timesegment B
- The necessary registervalue of these parameters are different to the values used for computation of the
- Baudrate and the
- Samplepoint.

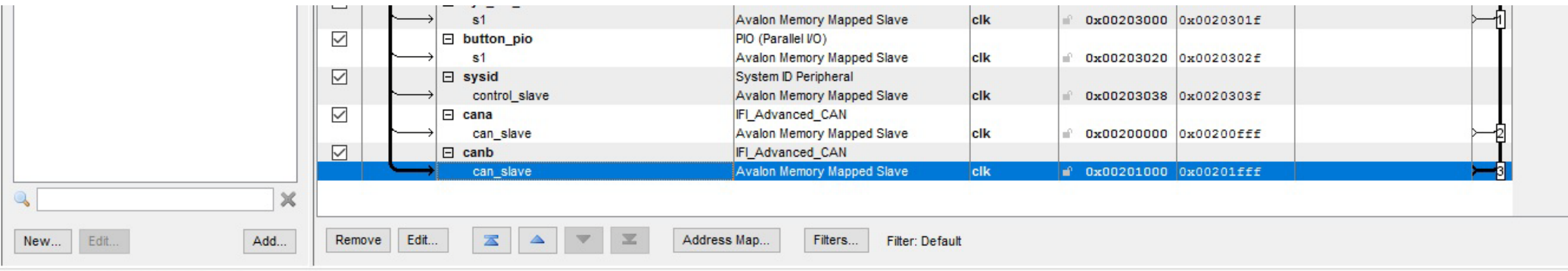


- For external CPU interfaces it's possible to select the width of the databus and the endian-ness of the bus. The NIOSII CPU is Little\_Endian



- 32Bits / 16 Bits / 8 Bits
- Little\_Endian / Big\_Endian
- In the information field you will find informations, warnings and errors
- Click on Finish will add the core to your SOPC system

# Adding the Core to your System



<input checked="" type="checkbox"/>	s1	Avalon Memory Mapped Slave	clk	0x00203000	0x0020301f	
<input checked="" type="checkbox"/>	button_pio	PIO (Parallel I/O)				
<input checked="" type="checkbox"/>	s1	Avalon Memory Mapped Slave	clk	0x00203020	0x0020302f	
<input checked="" type="checkbox"/>	sysid	System ID Peripheral				
<input checked="" type="checkbox"/>	control_slave	Avalon Memory Mapped Slave	clk	0x00203038	0x0020303f	
<input checked="" type="checkbox"/>	cana	IFI_Advanced_CAN				
<input checked="" type="checkbox"/>	can_slave	Avalon Memory Mapped Slave	clk	0x00200000	0x00200fff	
<input checked="" type="checkbox"/>	canb	IFI_Advanced_CAN				
<input checked="" type="checkbox"/>	can_slave	Avalon Memory Mapped Slave	clk	0x00201000	0x00201fff	

- Info: cana: Timestampcounter enabled
- Info: cana: 32 Bit Data Interface
- Info: canb: Timestampcounter enabled
- Info: canb: 32 Bit Data Interface

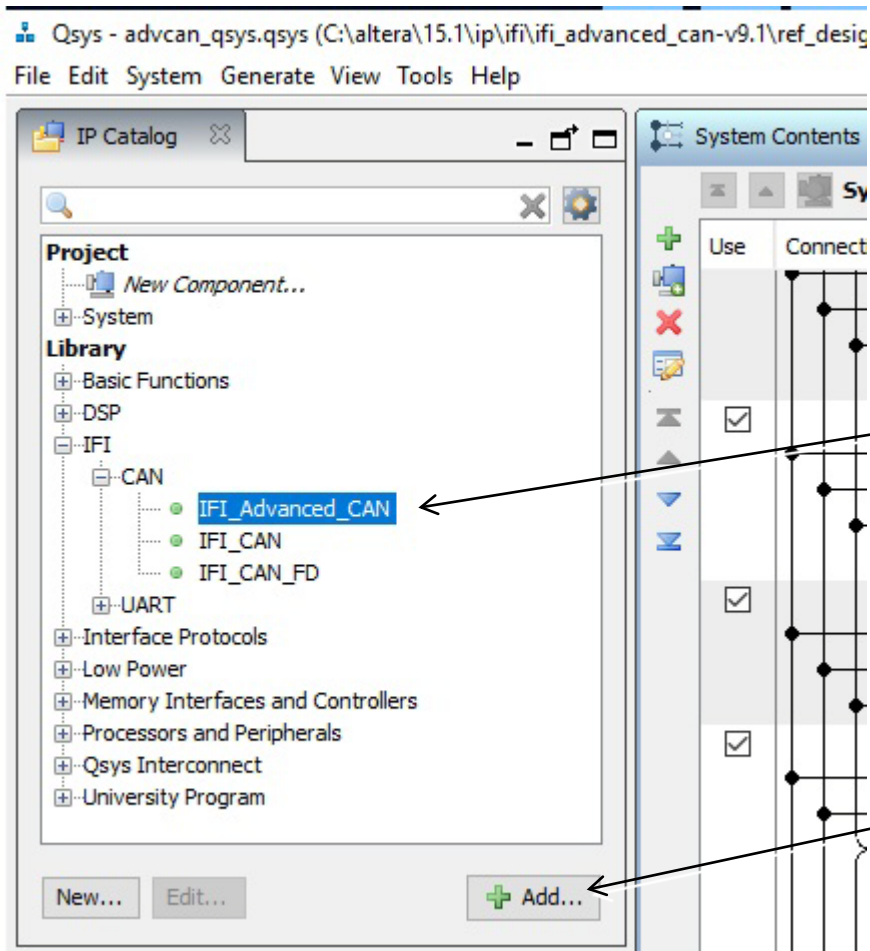
- Specify desired instance name, base address, and IRQ.
- Complete system generation as described in the SOPC Builder documentation.



# 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
  - OpenCore evaluation process,
  - Quartus II development software,
  - and the QSYS Interconnect tool
- For more information on these prerequisites, please visit *[www.altera.com](http://www.altera.com)*

# Adding the Core to your QSYS System



- Launch the 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.



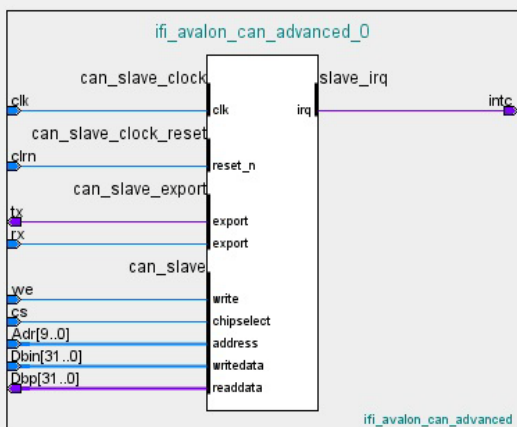
IFI\_Advanced\_CAN  
ifi\_avalon\_can\_advanced

Documentation

Info +  
Documentation

Block Diagram

Show signals



Timestamp

Timestamp:: Generate\_Timestamp\_OFF

Filter and Masks

Number of Filters and Masks used: 64

Buffer

Size of the Receive Fifo in Messages: 32

Size of the Transmit Fifo in Messages: 30

Baudrate

System Clock in Hz: 50000000

Startvalue Prescale: 2

Registervalue Prescale: 0

Startvalue Timesegment A: 20

Registervalue Time\_a: 19

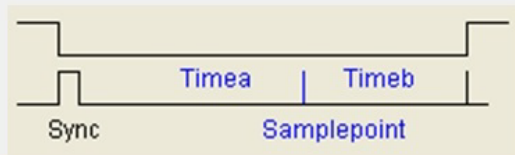
Startvalue Timesegment B: 4

Registervalue Time\_b: 2

Baudrate KBaud: 1000

Sample Point %: 84

Parameters



External Interface

Datawidth: 32

Endian-ness

Endian-ness: Little\_Endian

Information

Warning: ifi\_avalon\_can\_advanced: add\_file: C:/altera/15.1/ip/ifi/ifi\_advanced\_can-v9.1/lib/nioscan\_advanced\_top.vhd is encrypted; it cannot be used for simulation

Warning: ifi\_avalon\_can\_advanced: add\_file: C:/altera/15.1/ip/ifi/ifi\_advanced\_can-v9.1/lib/nioscan\_advanced.vhd is encrypted; it cannot be used for simulation

Warning: ifi\_avalon\_can\_advanced: add\_file: C:/altera/15.1/ip/ifi/ifi\_advanced\_can-v9.1/lib/ificancore.vhd is encrypted; it cannot be used for simulation

Cancel Finish



F. Sprenger

**IFI\_Advanced\_CAN**  
ifi\_avalon\_can\_advanced

F. Sprenger

Documentation

---

**Block Diagram**

Show signals

**Timestamp**

Timestamp:: Generate\_Timestamp\_OFF

---

**Filter and Masks**

Number of Filters and Masks used: 64

---

**Buffer:**

Size of the Receive Fifo in Messages: 32

Size of the Transmit Fifo in Messages: 30

---

**Baudrate:**

System Clock in Hz: 50000000

Startvalue Prescale: 2

Registervalue Prescale:: 0

Startvalue Timesegment A: 20

Registervalue Time\_a:: 19

Startvalue Timesegment B: 4

Registervalue Time\_b:: 2

Baudrate Kbaud:: 1000

Sample Point %:: 84

---

**External Interface**

Datawidth: 32

---

**Endian-ness**

Endian-ness:: Little\_Endian

---

Warning: ifi\_avalon\_can\_advanced: add\_file: C:/altera/15.1/ip/ifi/ifi\_advanced\_can-v9.1/lib/nioscan\_advanced\_top.vhd is encrypted; it cannot be used for simulation

Warning: ifi\_avalon\_can\_advanced: add\_file: C:/altera/15.1/ip/ifi/ifi\_advanced\_can-v9.1/lib/nioscan\_advanced.vhd is encrypted; it cannot be used for simulation

Warning: ifi\_avalon\_can\_advanced: add\_file: C:/altera/15.1/ip/ifi/ifi\_advanced\_can-v9.1/lib/ificancore.vhd is encrypted; it cannot be used for simulation

Cancel Finish

## IFI\_Advanced\_CAN

<b>Name</b>	ifi_avalon_can_advanced
<b>Version</b>	9.1
<b>Author</b>	IFI
<b>Description</b>	IFI_Advanced_CAN - v9.1
<b>Group</b>	IFI/CAN
<b>Data Sheet</b>	<a href="file:///C:/altera/15.1/ip/ifi/ifi_advanced_can-v9.1/doc/IFI_Advanced_CAN_docu.pdf">file:///C:/altera/15.1/ip/ifi/ifi_advanced_can-v9.1/doc/IFI_Advanced_CAN_docu.pdf</a>

### Timestamp

**Timestamp:** ON or OFF

### Filter and Masks

**Number of Filters and Masks used** 64, 128 or 256

### Buffer:

**Size of the Receive Fifo in Messages** 32, 64, 128, 256, 512, 1024, 2048 or 4096

**Size of the Transmit Fifo in Messages** 30, 62, 126 or 254

### Baudrate:

<b>System Clock in Hz</b>	Your System Clock in Hz
<b>Startvalue Prescale</b>	2 .. 255
<b>Registervalue Prescale:</b>	Registervalue Prescale
<b>Startvalue Timesegment A</b>	0 .. 31
<b>Registervalue Time_a:</b>	Registervalue Time_a
<b>Startvalue Timesegment B</b>	2 .. 31
<b>Registervalue Time_b:</b>	Registervalue Time_b
<b>Baudrate Kbaud:</b>	Resulting Baudrate
<b>Sample Point %:</b>	Resulting Sample Point in %

### External Interface

**Datawidth** 32 Bit, 16Bit or 8 Bit wide

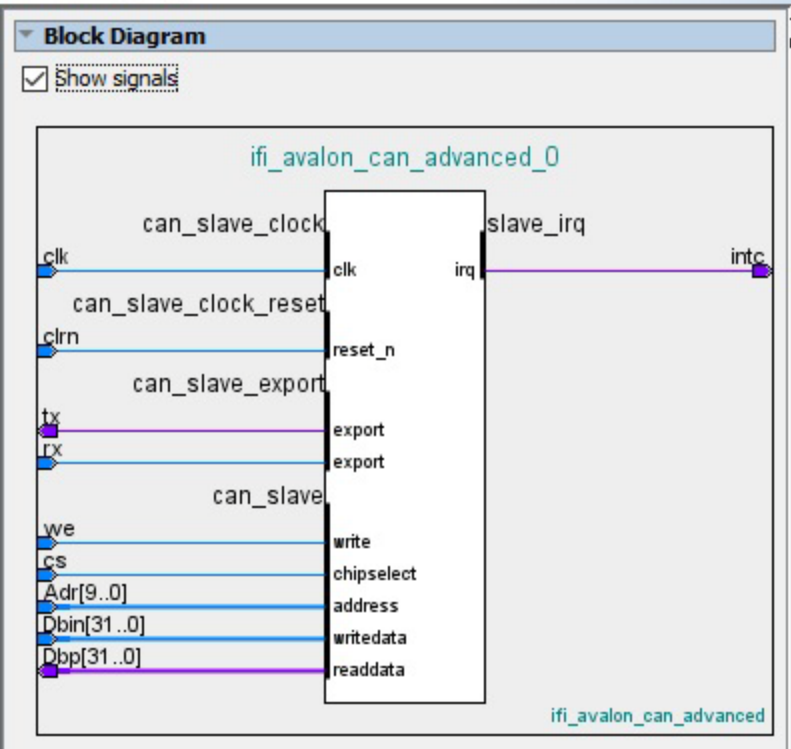
### Endian-ness

**Endian-ness:** Little\_Endian or Big\_Endian



# IFI\_Advanced\_CAN

ifi\_avalon\_can\_advanced



**Timestamp**  
Timestamp:: Generate\_Timestamp\_OFF

**Filter and Masks**  
Number of Filters and Masks used: 64

**Buffer:**  
Size of the Receive Fifo in Messages: 32  
Size of the Transmit Fifo in Messages: 30

**Baudrate:**  
System Clock in Hz: 50000000  
Startvalue Prescale: 2  
Registervalue Prescale:: 0  
Startvalue Timesegment A: 20  
Registervalue Time\_a:: 19  
Startvalue Timesegment B: 4  
Registervalue Time\_b:: 2  
Baudrate KBaud:: 1000

- Enable or Disable the Timestamp
- Select the number of Filters and Masks
- Receive buffer-size in messages
- Transmit buffer-size in messages
- Baudrate Calculator





- The CAN timing parameters are startup values and could be overwritten by software

- The clock frequency of your selected clock.

- The CAN timing is always a result of 3 parameters

- Prescale

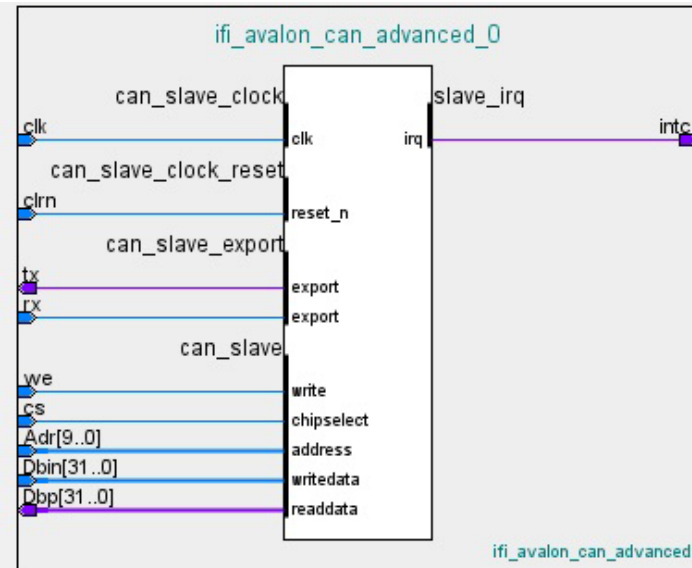
- Timesegment A

- Timesegment B

- The necessary registervalue of these parameters are different to the values used for computation of the

- Baudrate and the

- Samplepoint.



**Filter and Masks**

Number of Filters and Masks used: 64

**Buffer:**

Size of the Receive Fifo in Messages: 32

Size of the Transmit Fifo in Messages: 30

**Baudrate:**

System Clock in Hz: 50000000

Startvalue Prescale: 2

Registervalue Prescale:: 0

Startvalue Timesegment A: 20

Registervalue Time\_a:: 19

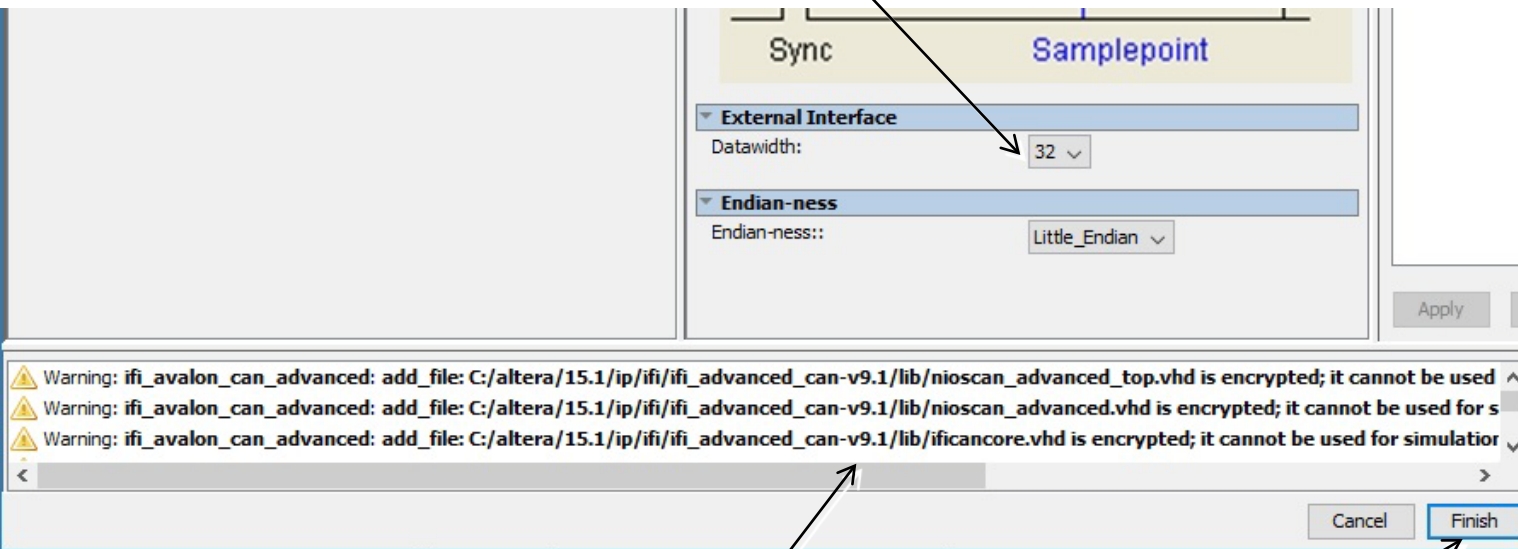
Startvalue Timesegment B: 4

Registervalue Time\_b:: 2

Baudrate KBAud:: 1000

Sample Point %:: 84

- For external CPU interfaces it is possible to select the width of the databus.
- 64 Bits / 32Bits / 16 Bits / 8 Bits



- In the information field you will find informations, warnings and errors
- Clicking on Finish will add the core to your QSYS system

# Adding the Core to your System

Component	Description	Export	Signal	Base Address	End Address	IRQ
control_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0004_3040	0x0004_3047	
nios2_gen2_0	Nios II Processor					
clk	Clock Input	Double-click to export	clk [clk]			
reset	Reset Input	Double-click to export				
data_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
irq	Interrupt Receiver	Double-click to export	[clk]			IRQ 0
debug_reset_request	Reset Output	Double-click to export	[clk]			IRQ 31
debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0004_2800	0x0004_2fff	
custom_instruction_m...	Custom Instruction Master	Double-click to export				
cana	IFI_Advanced_CAN					
can_slave_dock	Clock Input	Double-click to export	clk [can_slave_...]			
can_slave_dock_reset	Reset Input	Double-click to export				
can_slave_export	Conduit		can_a [can_slave_...]			
can_slave	Avalon Memory Mapped Slave	Double-click to export	[can_slave_...]	0x0004_1000	0x0004_1fff	
slave_irq	Interrupt Sender	Double-click to export	[can_slave_...]			
canb	IFI_Advanced_CAN					
can_slave_dock	Clock Input	Double-click to export	clk [can_slave_...]			
can_slave_dock_reset	Reset Input	Double-click to export				
can_slave_export	Conduit		can_b [can_slave_...]			
can_slave	Avalon Memory Mapped Slave	Double-click to export	[can_slave_...]	0x0004_0000	0x0004_0fff	
slave_irq	Interrupt Sender	Double-click to export	[can_slave_...]			

Current filter:

Messages: 8 Warnings

Warning: advanced\_core\_ifi\_avalon\_can\_advanced... add file C:\altera\15.1\ip\if\_i\_advanced\_core\_v0.1\lib\nioscan\_advanced\_top.uhd is deprecated; it cannot be used for simulation

- Specify the desired instance **name**, **base address**, and **IRQ**
- connect the system clock to can\_slave\_clock Clock Input
- Complete the system generation as described in the QSYS documentation



# Reference Designs

- *Running a Reference Design*
- *Creating a Software Project*
- *Run a Hardware Configuration*
- *Using Modelsim to Simulate the Core within your System*

# Running a Reference Design

- Start Quartus II, version 9.1 or higher.
- Open the Quartus II project <Core installation directory>\reference\_designs\xxx\Reference\_design.qpf
- Launch SOPC/QSYS Builder from Quartus II (Tools menu).
- 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 creation of software projects, InstructionSetSimulation 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

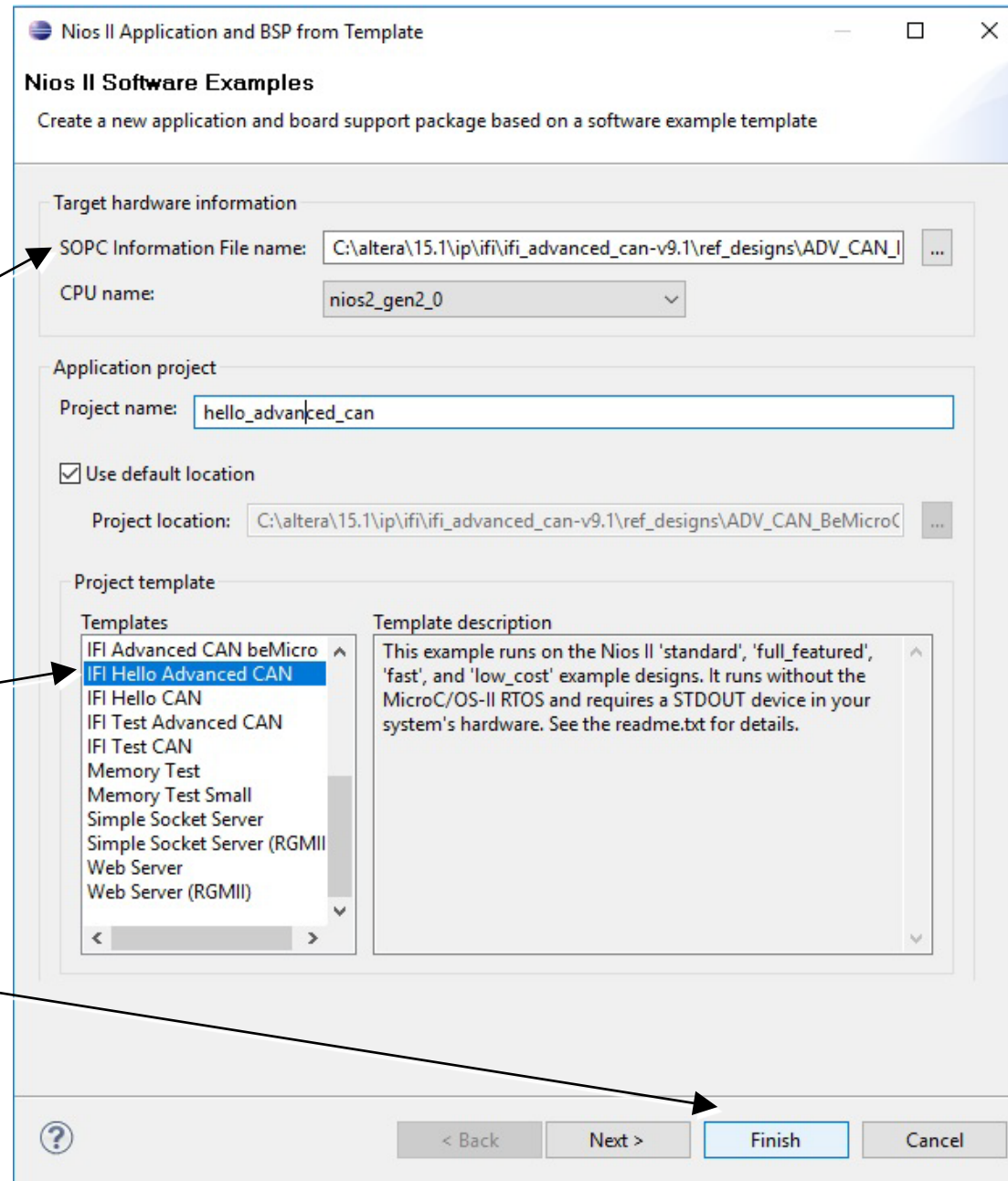
## ■ File → New →

- NIOS II Application and BSP from Template
- Click Next

Select the SOPC Information File of your project with project name

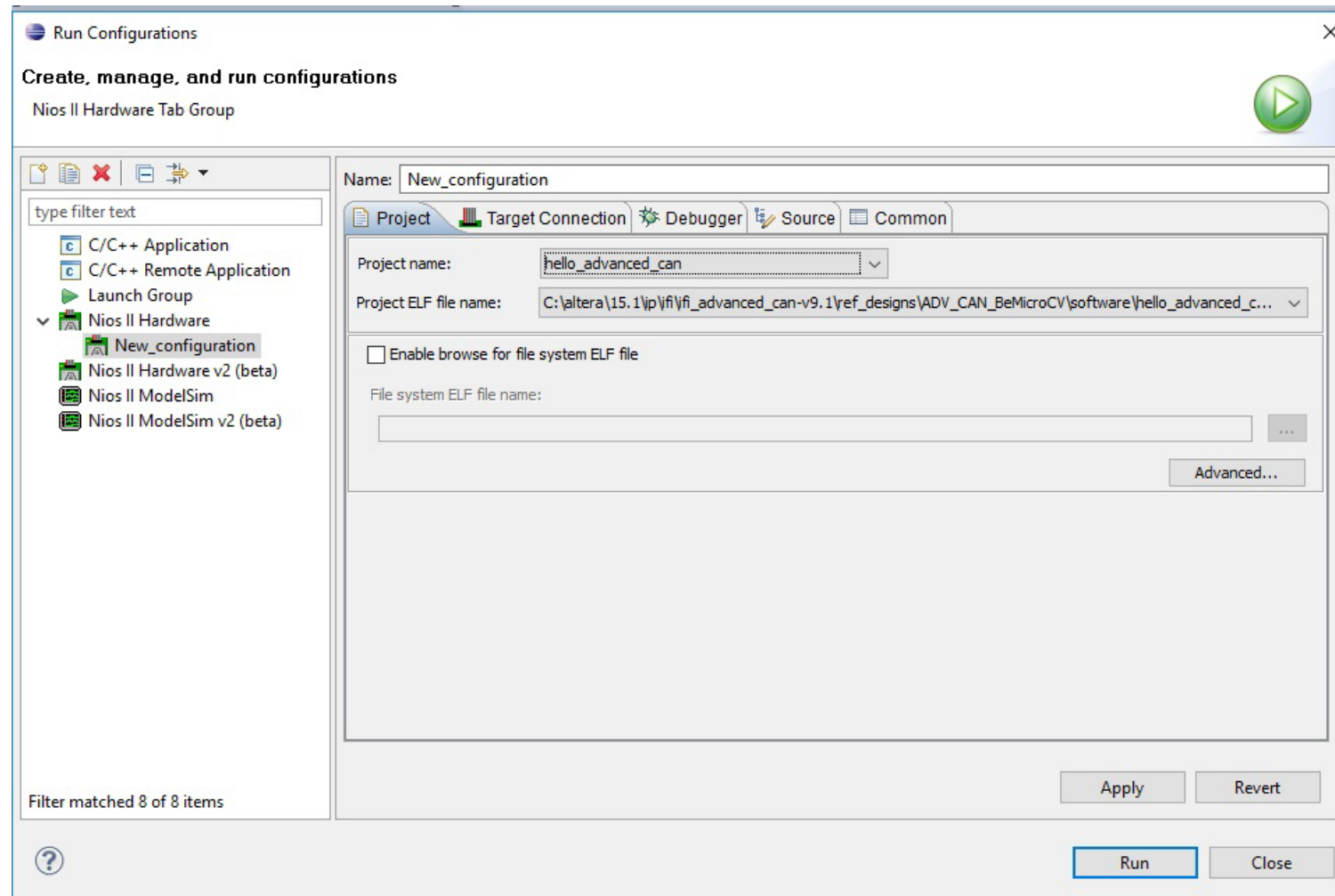
Select „Hello Advanced CAN“

Click on Finish



# Run a Hardware Configuration

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



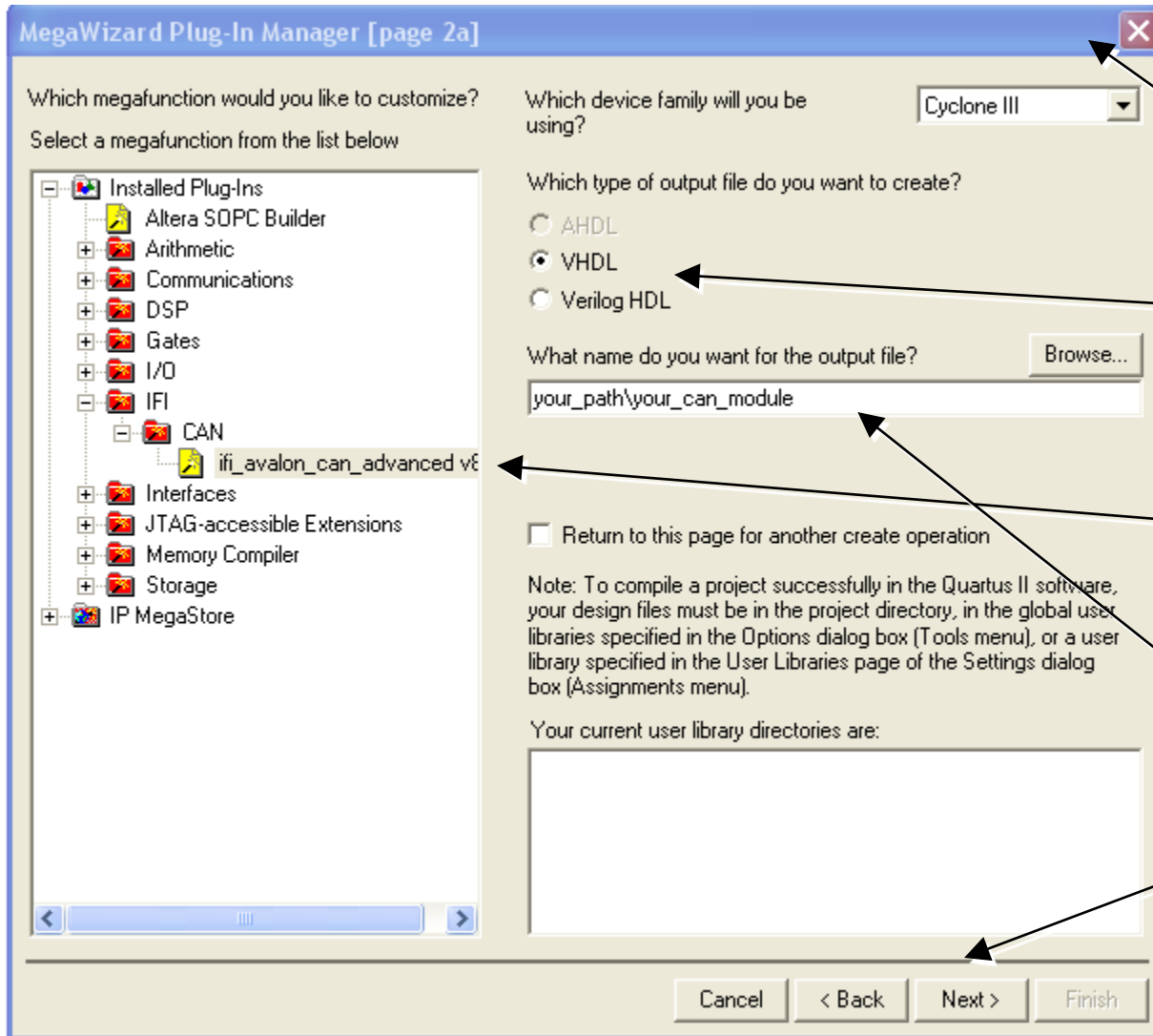
# Using ModelSim to Simulate the Core within your System

- Generate your system using the SOPC Builder.
- Launch the IDE and launch ModelSim or ModelSim Altera/Intel Edition, via the
  - Run--Run...--New Modelsim Configuration in IDE.
- Type 's' to load the design files.
- Type 'w' to add the appropriate waveforms to the wave window.
- Type 'run 5 ms' to start the simulation.
- For more details on simulation, please see Altera Application Note 351:
  - Simulating NiosII Embedded Processor Designs. (<SOPC Builder installation directory>\documents\AN351.pdf)

# Using the Core without SOPC/QSYS

- *MegaWizard Plug-In Manager*
- *IP Catalog*
- *Parameterize*
- *Generate*
- *Quartus Symbol*
- *Reopening of the Modul*
- *Port Description*
- *Read Timing*
- *Write Timing*

# MegaWizard Plug-In Manager for older Quartus versions



Start MegaWizard Plug-In Manager

Select the wrapper and simulation language

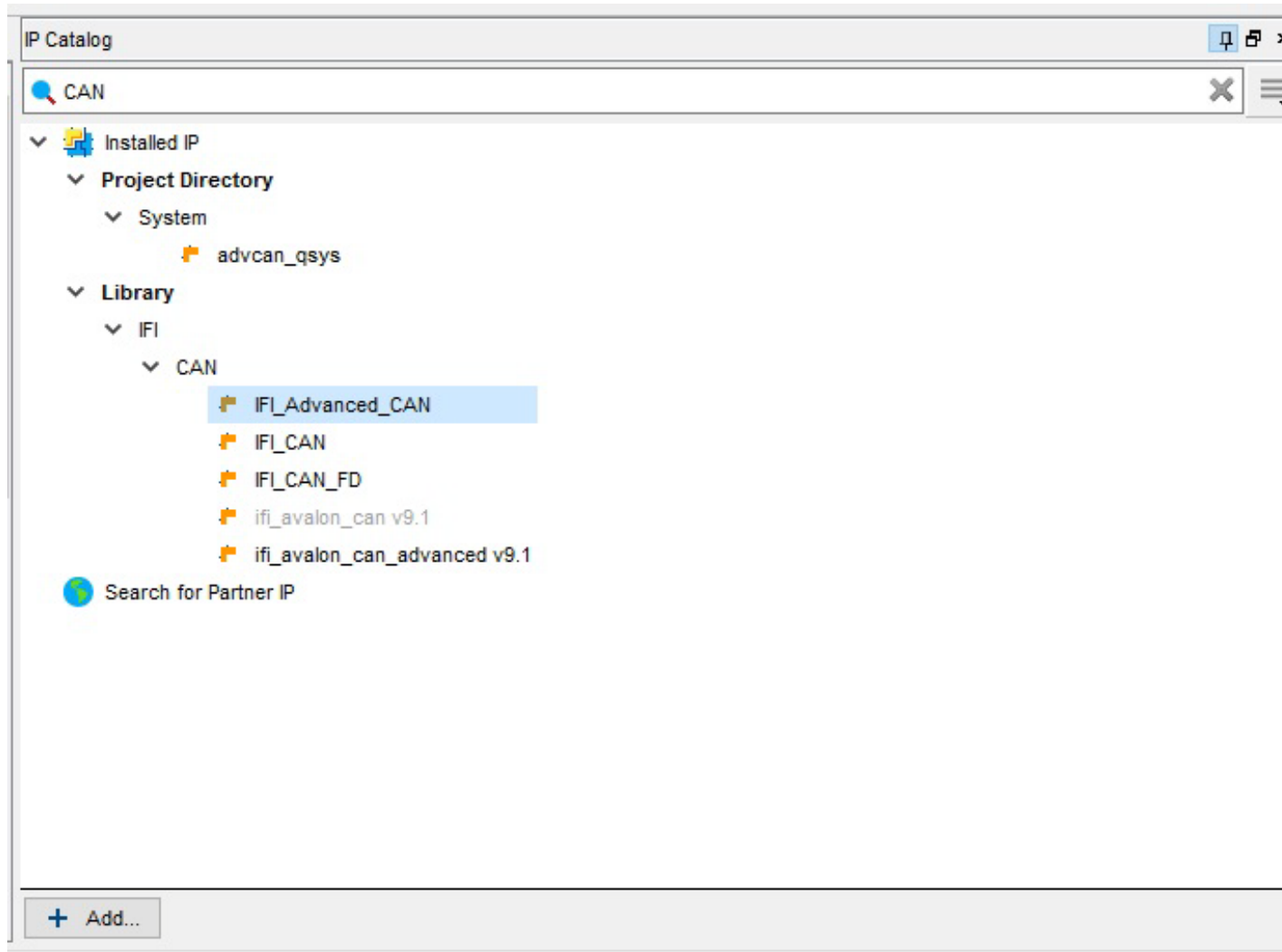
Select the IP Core

Type in the name

Click Next



# Use IP Catalog and Advanced CAN IP





IFI\_Advanced\_CAN - cana

**IFI**  
F. Sprenger

**IFI\_Advanced\_CAN**  
ifi\_avalon\_can\_advanced

Info

**Block Diagram**

```

    graph TD
      cana[can]
      clock --> cana
      conduit --> cana
      avalon --> cana
      interrupt --> cana
      cana --> can_slave_clock[can_slave_clock]
      cana --> can_slave_export[can_slave_export]
      cana --> can_slave[can_slave]
      cana --> slave_irq[slave_irq]
  
```

**Timestamp**

Timestamp:: Generate\_Timestamp\_ON

**Filter and Masks**

Number of Filters and Masks used: 64

**Buffer:**

Size of the Receive Fifo in Messages: 32

Size of the Transmit Fifo in Messages: 30

**Baudrate:**

System Clock in Hz: 64000000

Startvalue Prescale: 2

Registervalue Prescale:: 0

Startvalue Timesegment A: 25

Registervalue Time\_a:: 24

Startvalue Timesegment B: 6

Registervalue Time\_b:: 4

Baudrate KBAud:: 1000

Sample Point %:: 81

**External Interface**

Datawidth: 32

**Endian-ness**

Endian-ness:: Little\_Endian

Info: cana: Timestampcounter enabled

Info: cana: 32 Bit Data Interface

Cancel Finish


Info + Documentation

Parameters

Information



IFI\_Advanced\_CAN - cana

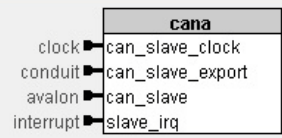


**IFI\_Advanced\_CAN**  
ifi\_avalon\_can\_advanced

F. Sprenger

Info

**Block Diagram**



**Timestamp**

Timestamp:

**Filter and Masks**

Number of Filters and Masks used:

**Buffer:**

Size of the Receive Fifo in Messages:

Size of the Transmit Fifo in Messages:

**Baudrate:**

System Clock in Hz:

Startvalue Prescale:

Registervalue Prescale:

Startvalue Timesegment A:

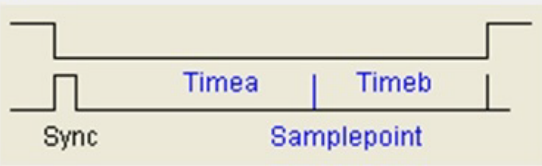
Registervalue Time\_a:

Startvalue Timesegment B:

Registervalue Time\_b:

Baudrate KBaud:

Sample Point %:



**External Interface**

Datawidth:

**Endian-ness**

Endian-ness:

Info: cana: Timestampcounter enabled

Info: cana: 32 Bit Data Interface

Cancel Finish

IFI\_Advanced\_CAN Info

**IFI\_Advanced\_CAN**

**Class Name** ifi\_avalon\_can\_advanced

**Version** 9.1

**Author** IFI

**Description** IFI\_Advanced\_CAN - v9.1

**Group** IFI/CAN

**Data Sheet** [file:///C:/altera/15.1/ip/ifi/ifi\\_advanced\\_can-v9.1/doc/IFI\\_Advanced\\_CAN\\_docu.pdf](file:///C:/altera/15.1/ip/ifi/ifi_advanced_can-v9.1/doc/IFI_Advanced_CAN_docu.pdf)

**Timestamp**

**Timestamp:** ON or OFF

**Filter and Masks**

**Number of Filters and Masks used** 64, 128 or 256

**Buffer:**

**Size of the Receive Fifo in Messages** 32, 64, 128, 256, 512, 1024, 2048 or 4096

**Size of the Transmit Fifo in Messages** 30, 62, 126 or 254

**Baudrate:**

**System Clock in Hz** Your System Clock in Hz

**Startvalue Prescale** 2 .. 255

**Registervalue Prescale:** Registervalue Prescale

**Startvalue Timesegment A** 0 .. 31

**Registervalue Time\_a:** Registervalue Time\_a

**Startvalue Timesegment B** 2 .. 31

**Registervalue Time\_b:** Registervalue Time\_b

**Baudrate KBaud:** Resulting Baudrate


**Sample Point %:** Resulting Sample Point in %

**External Interface**

**Datawidth** 32 Bit, 16Bit or 8 Bit wide

**Endian-ness**

**Endian-ness:** Little\_Endian or Big\_Endian



F. Sprenger

- Enable or Disable the Timestamp
- Select the number of Filters and Masks
- Receive buffer-size in messages
- Transmit buffer-size in messages
- Baudrate Calculator



**Block Diagram**

clock → can\_slave\_clock  
 conduit → can\_slave\_export  
 avalon → can\_slave  
 interrupt → slave\_irq

**Timestamp**

Timestamp::

**Filter and Masks**

Number of Filters and Masks used:

**Buffer:**

Size of the Receive Fifo in Messages:

Size of the Transmit Fifo in Messages:

**Baudrate:**

System Clock in Hz:

Startvalue Prescale:

Registervalue Prescale:

Startvalue Timesegment A:

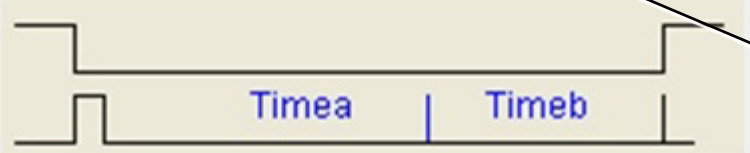
Registervalue Time\_a:

Startvalue Timesegment B:

Registervalue Time\_b:

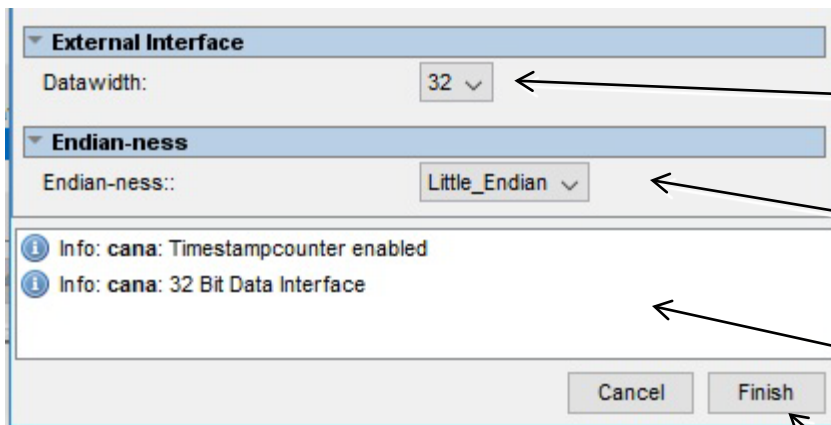
Baudrate KBAud:

Sample Point %:



- The CAN timing parameters are startup values and could be overwritten by software
- The clock frequency of your selected clock.
- The CAN timing is always a result of 3 parameters
  - Prescale
  - Timesegment A
  - Timesegment B
- The necessary registervalue of these parameters are different to the values used for computation of the
  - Baudrate and the
  - Samplepoint.

- For external CPU interfaces it's possible to select the width of the databus and the endian-ness of the bus. The NIOSII CPU is Little\_Endian



- 32Bits / 16 Bits / 8 Bits
- Little\_Endian / Big\_Endian
- In the information field you will find informations, warnings and errors
- Click on Finish



# Generation of the Core Variation

Parameters

System: canc Path: ifi\_avalon\_can\_advanced\_0

**IFI\_Advanced\_CAN**  
ifi\_avalon\_can\_advanced

Details

**Timestamp**  
Timestamp:: Generate\_Timestamp\_OFF

**Filter and Masks**  
Number of Filters and Masks used: 64

**Buffer:**  
Size of the Receive Fifo in Messages: 32  
Size of the Transmit Fifo in Messages: 30

**Baudrate:**  
System Clock in Hz: 50000000  
Startvalue Prescale: 2  
Registervalue Prescale:: 0  
Startvalue Timesegment A: 20  
Registervalue Time\_a:: 19  
Startvalue Timesegment B: 4  
Registervalue Time\_b:: 2  
Baudrate KBAud:: 1000  
Sample Point %:: 84

**External Interface**  
Datawidth: 32

**Integration with Quartus Prime Software**

The following new files were created:  
C:\altera\...advanced\_can-v9.1\ref\_designs\ADV\_CAN\_BeMicroCV\canc.qsys

To edit or modify a .qsys file in your design, do one of the following in the Quartus Prime software main window:

- Open the .qsys file with the Open command on the File menu
- Double-click the .qsys file on the Files tab in the Project Navigator
- Open Qsys from the Tools menu
- Use the qsys-edit command at the command line

To generate HDL files from a .qsys file, do one of the following in the Quartus Prime software:

- Open Qsys from the Tools menu
- Use the qsys-edit command at the command line
- Open Qsys from the Quartus Prime software and click on the 'Generate HDL...' button

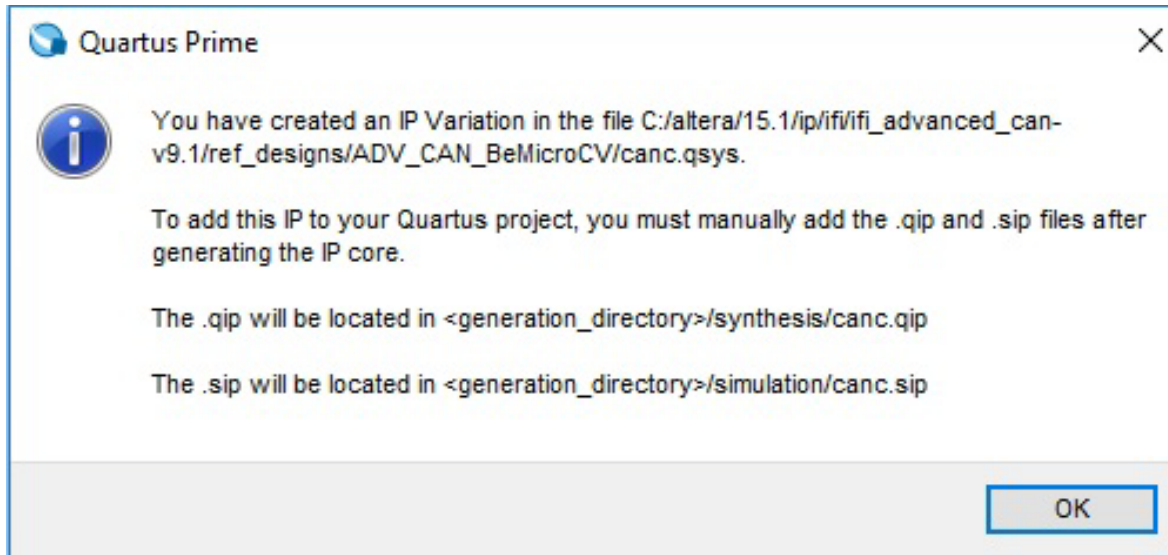
Do not show this message again

Close

Messages

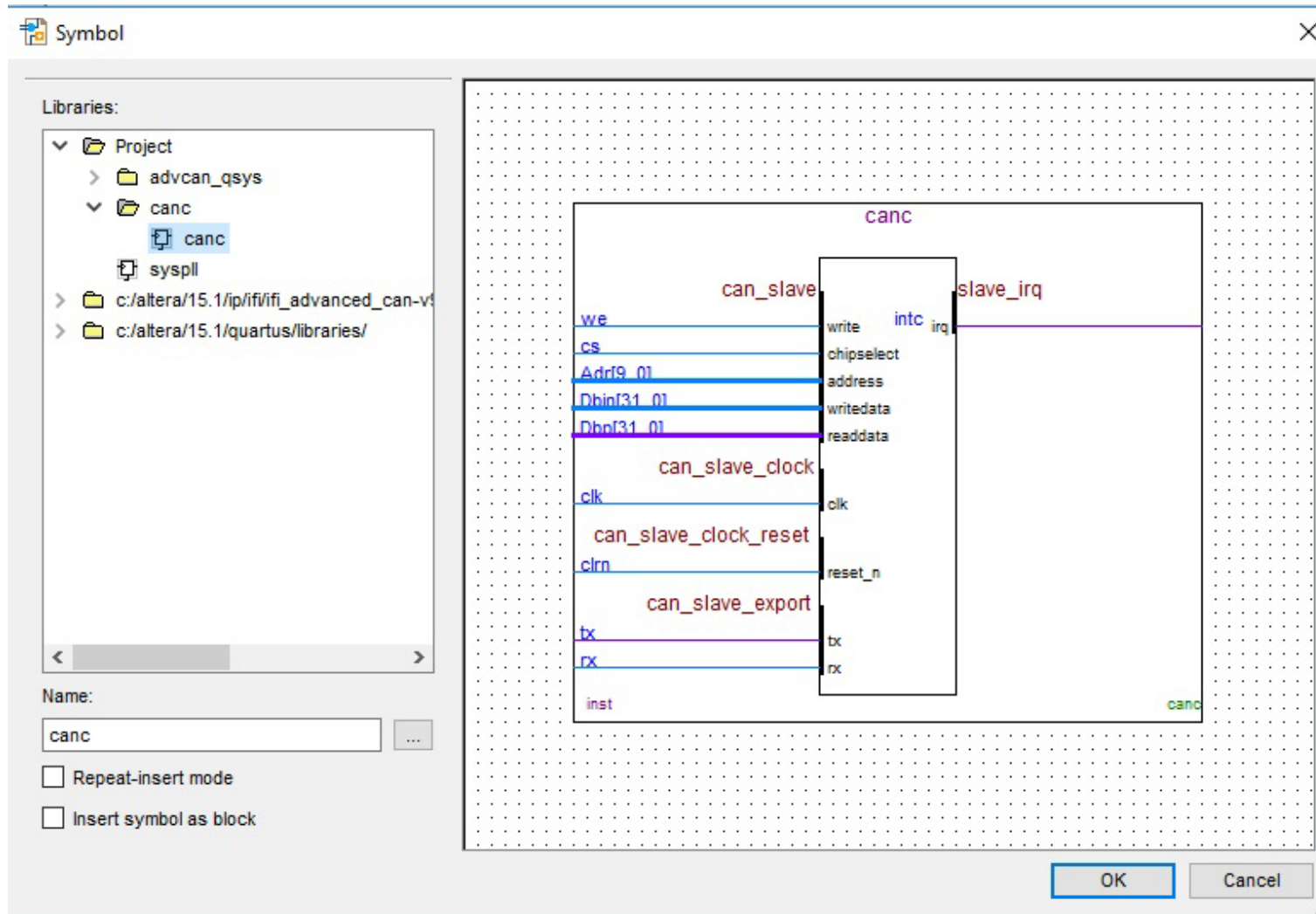
Type	Path	Message
Warning	4 Warnings	
Warning	canc.ifi_avalon_can_advanced	add file C:/altera/15.1/ip/ifi_advanced_can-v9.1/lib/nioscan_advanced_top.vhd is encrypted; it cannot be used for simulation

# Use the QuartusII IP File



- File open
- Search for \*.qip
- Select file
- Project / Add current file to project

# Your new QuartusII IP Symbol

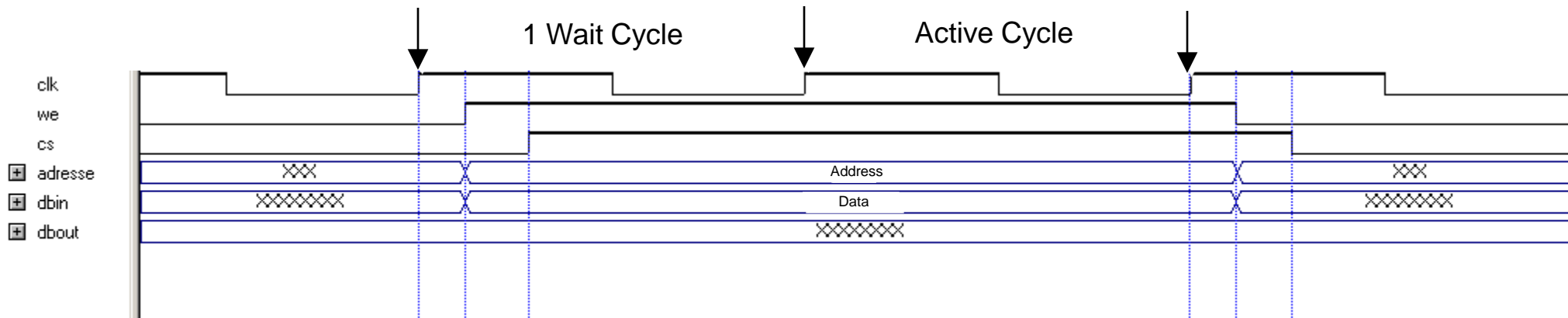




# Port description

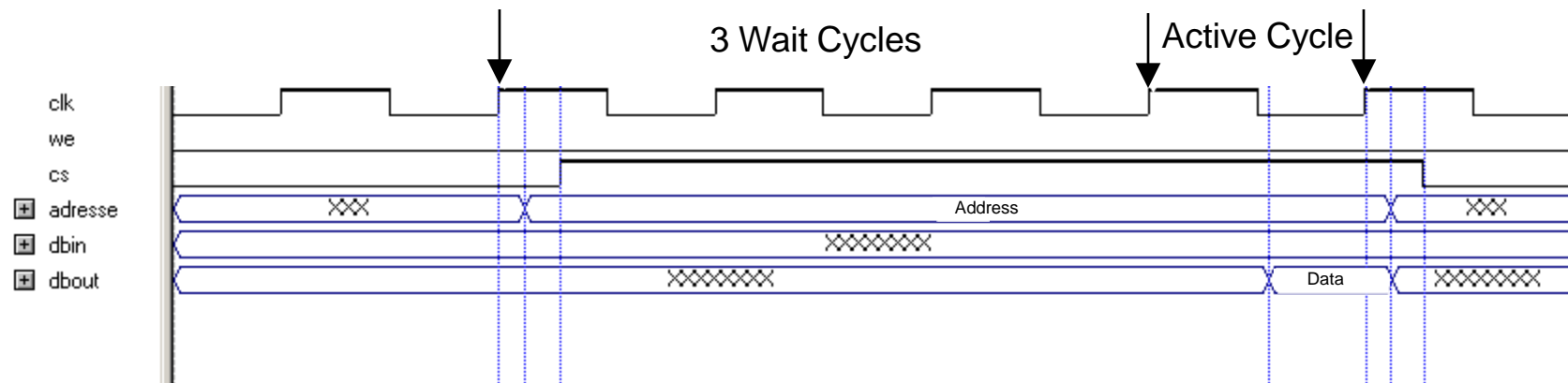
Portname	Direction	Usage	Description
tx	output	External	Transmit from CAN
rx	input	External	Receive to CAN
tstamp_clk	input	External	Timestamp Clock Input
clk	input	Internal	System clock
clrn	input	Internal	System reset (low active)
we	input	Internal	Write request (high active)
cs	input	Internal	Chip select (high active)
Adr[x..0]	input	Internal	Address for read/write requests
Dbn[x..0]	input	Internal	Write data bus
Dbp[x..0]	output	Internal	Read data bus
intc	output	Internal	Interrupt request

# Write Timing for the 32 Bit Interface



**For the 8 and 16 Bit Interface no wait is necessary**

# Read Timing



# Detailed Information

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

# Addressmap

- Address 0 to 3 are acting like a FIFO.
  - Writing: Transmitmessage FIFO
  - Reading: Receivemessage FIFO
- You can write up to 30..254 Transmitobjects into this FIFO.
- Address 1,2 and 3 can be written in any sequenz
- Writing address 0 increment the fifo pointer and the next 4 values can be written
- You can read received objects by reading address 0 to 3
- If the timestamp is used, you can read it by reading address 10
- Writing 1 to Rec Fifo read next value in the status register set the read pointer to the next received object and also the next timestampvalue (if used).

# Data length code

- Read => Received message
- Write => Transmit message

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
	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
	read								Obj 8
	write								
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
	write								
Byte 0	Bit	7	6	5	4	3	2	1	0
	read				RTR	DLC 3	DLC 2	DLC 1	DLC 0
	write				RTR	DLC 3	DLC 2	DLC 1	DLC 0

- Frn [7.. 0] : Frame number for Transmit Timestamp
- Obj [8..0] : Filter Object number of a received message
- RTR : Remote transmit
- DLC [3..0] : Data length code (0 .. 8 DataBytes)

## Base Address Offset 0

# Frame Number for Transmit Timestamp

- If this number is set to 00x, the feature is deactivated.
- Writing any number different to 00x activates this feature.
- If there is a number between 01x and FFx, the successful transmit of the message will be timestamped and flagged like a normal receive. Within the receive buffer the Data length code register contains this number and the object number will be 00x. This information is not depending from a filter setting.
- Reading a number greater 00x means the message is not a message coming from another transmitter, it's his own transmitted message.

# Identifizier

- Read => Received message
- Write => Transmit message

Byte 3	Bit	31	30	29	28	27	26	25	24
	read			IDE	IDX 28	IDX 27	IDX 26	IDX 25	IDX 24
	write			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

- IDE : 1 => Use Extended Identifier
- IDX [28..11] + ID [10..0] : Extended Identifier
- ID [10..0] : Standard Identifier

## Base Address Offset 1



# Data 1 - 4

- Read => Received message
- Write => Transmit message

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	Data4 7	Data4 6	Data4 5	Data4 4	Data4 3	Data4 2	Data4 1	Data4 0
	write	Data4 7	Data4 6	Data4 5	Data4 4	Data4 3	Data4 2	Data4 1	Data4 0
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	Data3 7	Data3 6	Data3 5	Data3 4	Data3 3	Data3 2	Data3 1	Data3 0
	write	Data3 7	Data3 6	Data3 5	Data3 4	Data3 3	Data3 2	Data3 1	Data3 0
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	Data2 7	Data2 6	Data2 5	Data2 4	Data2 3	Data2 2	Data2 1	Data2 0
	write	Data2 7	Data2 6	Data2 5	Data2 4	Data2 3	Data2 2	Data2 1	Data2 0
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	Data1 7	Data1 6	Data1 5	Data1 4	Data1 3	Data1 2	Data1 1	Data1 0
	write	Data1 7	Data1 6	Data1 5	Data1 4	Data1 3	Data1 2	Data1 1	Data1 0

- Data1 [7..0] : Databyte 1
- Data2 [7..0] : Databyte 2
- Data3 [7..0] : Databyte 3
- Data4 [7..0] : Databyte 4

## Base Address Offset 2

# Data 5 - 8

- Read => Received message
- Write => Transmit message

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	Data8 7	Data8 6	Data8 5	Data8 4	Data8 3	Data8 2	Data8 1	Data8 0
	write	Data8 7	Data8 6	Data8 5	Data8 4	Data8 3	Data8 2	Data8 1	Data8 0
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	Data7 7	Data7 6	Data7 5	Data7 4	Data7 3	Data7 2	Data7 1	Data7 0
	write	Data7 7	Data7 6	Data7 5	Data7 4	Data7 3	Data7 2	Data7 1	Data7 0
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	Data6 7	Data6 6	Data6 5	Data6 4	Data6 3	Data6 2	Data6 1	Data6 0
	write	Data6 7	Data6 6	Data6 5	Data6 4	Data6 3	Data6 2	Data6 1	Data6 0
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	Data5 7	Data5 6	Data5 5	Data5 4	Data5 3	Data5 2	Data5 1	Data5 0
	write	Data5 7	Data5 6	Data5 5	Data5 4	Data5 3	Data5 2	Data5 1	Data5 0

- Data5 [7..0] : Databyte 5
- Data6 [7..0] : Databyte 6
- Data7 [7..0] : Databyte 7
- Data8 [7..0] : Databyte 8

## Base Address Offset 3

# Timing and Control

Byte 3	Bit	31	30	29	28	27	26	25	24
	read						SJW 1	SJW 0	
	write	Set prescale	Set Silent Mode	Remove pending Message	Silent Mode	Timestamp-counter reset	High Priority Message		Normal mode
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	Prescale 7	Prescale 6	Prescale 5	Prescale 4	Prescale 3	Prescale 2	Prescale 1	Prescale 0
	write	Prescale 7	Prescale 6	Prescale 5	Prescale 4	Prescale 3	Prescale 2	Prescale 1	Prescale 0
Byte 1	Bit	15	14	13	12	11	10	9	8
	read				Timea 4	Timea 3	Timea 2	Timea 1	Timea 0
	write	Set timea	Set SJW		Timea 4	Timea 3	Timea 2	Timea 1	Timea 0
Byte 0	Bit	7	6	5	4	3	2	1	0
	read				Timeb 4	Timeb 3	Timeb 2	Timeb 1	Timeb 0
	write	Set timeb	SJW 1	SJW 0	Timeb 4	Timeb 3	Timeb 2	Timeb 1	Timeb 0

Base Address Offset 4

# Timing and Control

- Timea [4..0] : Timing Segment B
- Timeb [4..0] : Timing Segment A
- Prescale [7..0] : Prescale Counter
- SJW [1..0] : Synchronisation Jump Width
- Normal Mode:
  - writing 1, start the CAN Node or restart after busoff
  - writing 0, not used
- Silent Mode:
  - writing 1, the CAN Node receive messages without Acknowledgement
  - writing 0, the CAN Node receive messages with Acknowledgement
- Timestamp Counter reset:
  - writing 1, reset the optional Timestampcounter
  - writing 0, not used
- High Priority Message:
  - writing 1, flags that the next written message is a High Priority Message
  - writing 0, not used
- Remove pending Message
  - writing 1, the CAN Node removes the ongoing message from the transmit buffer

# Timing Settings for CAN

- The prescale counter divide 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
  - $(\text{Timea}+1)$  \* time segment before samplepoint
  - $(\text{Timeb}+2)$  \* time segment after samplepoint

Sync Segment	Timeslot before Samplepoint	Timeslot after Samplepoint	
		$\text{Timeb} - \text{SJW} + 1$	$\text{SJW} + 1$
1	$\text{Timea} + 1$	$\text{Timeb} + 2$	
Samplepoint →			
←		Total Bit Time	
		→	



# Timing Settings for CAN

## ■ Example:

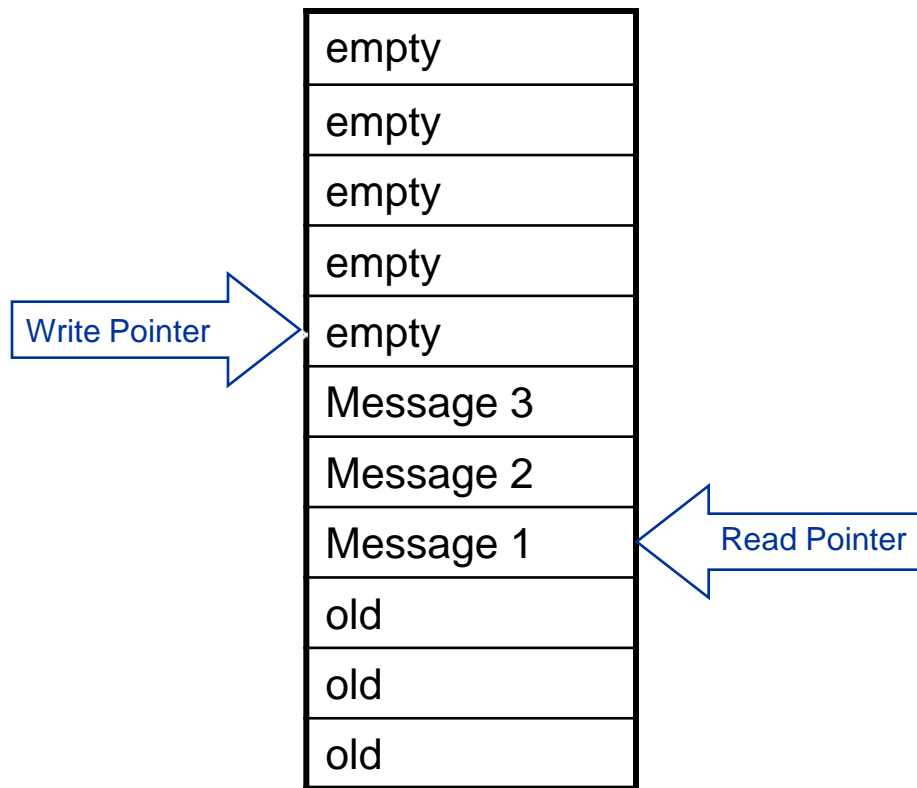
- You want to transmit with 500k baudrate
- $1 / 500k \rightarrow 2000$  ns total bit time
- If the length of one time segment is 80 ns
  - Divide  $2000 / 80 \rightarrow 25$  segments are necessary
- $25 - 1$  segments for sync  $\rightarrow 24$  segments
- The relationship between timea and timeb is 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
- Max value 3  $\rightarrow$  Resynchronisation Jump is 4 Time segments
- Timeb includes the Synchronisation Jump Width

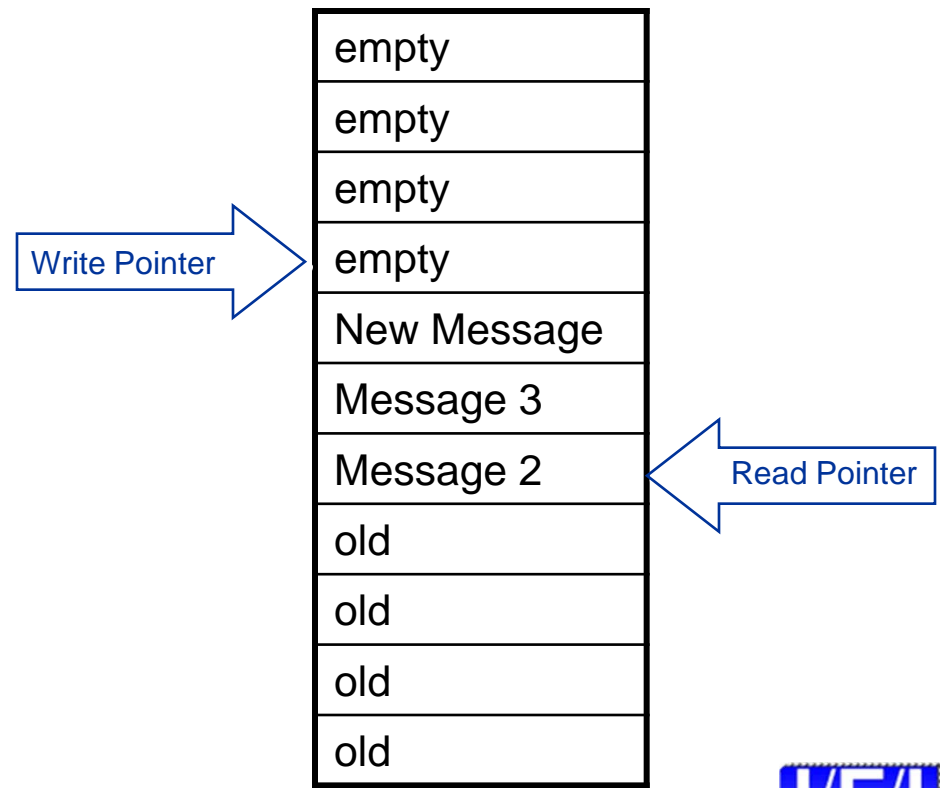
# High Priority == 0

- Normal Transmit Fifo usage
  - First in First out



External write

Internal read

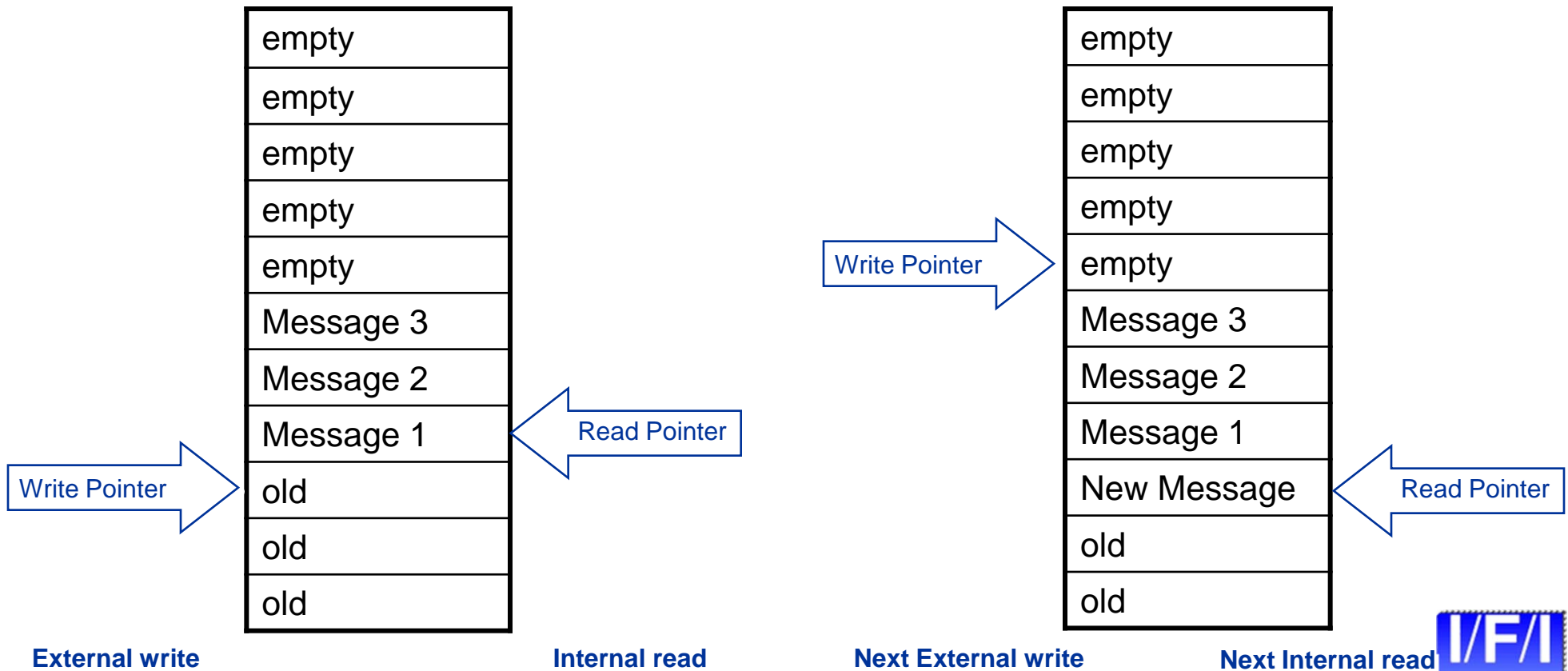


Next External write

Next Internal read

# High Priority == 1

- High Priority Transmit Fifo usage
  - First in Last out





# Interrupt mask

Byte 3	Bit	31	30	29	28	27	26	25	24
	read			Rec Buffer full	Rec Buffer overflow	Rec Buffer not empty	Tra Buffer full	Tra Buffer overflow	Tra Buffer empty
	write	Set Buffer Int Mask		Rec Buffer full	Rec Buffer overflow	Rec Buffer not empty	Tra Buffer full	Tra Buffer overflow	Tra Buffer empty
Byte 2	Bit	23	22	21	20	19	18	17	16
	read								
	write								
Byte 1	Bit	15	14	13	12	11	10	9	8
	read		Transmit ok						
	write	Set Transmit Ok Int Mask	Transmit ok						
Byte 0	Bit	7	6	5	4	3	2	1	0
	read							Error warn	Busoff
	write	Set Error Int Mask						Busoff	Error warn

- Interrupt Mask settings : 1 → Interrupt enabled
- Set Buffer Int Mask : 1 → write only interrupt mask for the receive and transmit buffer
- Set Error Int Mask : 1 → write only interrupt mask for the error flags
- Set Transmit ok Int Mask : 1 → write only interrupt mask for the Transmit ok flag

Document changed



## Base Address Offset 5

# Status

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	Receive busy	Transmit busy	Rec Buffer full INT	Rec Buffer overflow INT	Rec Buffer not empty INT	Tra Buffer full INT	Tra Buffer overflow INT	Tra Buffer empty INT
	write			R B full INT reset	R B overflow INT reset	R B not empty INT reset	T B full INT reset	T B overflow INT reset	T B empty INT reset
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	Receive Buffer full	Transmit Buffer full						
	write								
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	Message in Transmit	Transmit ok INT						
	write		Transmit ok INT reset						
Byte 0	Bit	7	6	5	4	3	2	1	0
	read				Silent mode	Error passive	Error active	Error warn	busoff
	write	RecFifo read next value						Error warn INT reset	busoff INT reset

- Status of the CAN node

Base Address Offset 6



# Status Register

- Busoff
  - The CAN node has stopped all activities
  - Writing 1 reset the interrupt
- Error warn
  - The number of errors is nearly before switching to Error passive
- Error active
  - The CAN node works properly
- Error passive
  - The CAN send recessive errorframes and suspend the transmission for 8 bittimes
- Silent Mode
  - Reading 1 → Silent Mode active
- Rec Fifo read next value
  - Writing 1 to the bit increment the receive buffer read pointer to the next message and decrement the read buffer counter
- Transmit buffer full
  - Status of the transmit buffer
- Receive buffer full
  - Status of the receive buffer

# Status Register

- **Tra buffer empty INT**
  - The buffer was not empty and the last message is on transmit
  - Writing 1 reset the interrupt
- **Tra buffer overflow INT**
  - The messages written into the transmit buffer are lost, because it was already full
  - Writing 1 reset the interrupt
- **Tra buffer full INT**
  - The transmit buffer is full
  - Writing 1 reset the interrupt
- **Rec buffer not empty INT**
  - The buffer was empty and the first message was written in the receive buffer
  - Writing 1 reset the interrupt
- **Rec buffer overflow INT**
  - Received messages are lost, because the receive buffer was already full
  - Writing 1 reset the interrupt
- **Rec buffer full INT**
  - The receive buffer is full
  - Writing 1 reset the interrupt
- **Transmit busy**
  - Messages are waiting in buffer for transmit
- **Message in Transmit**
  - Message is in transmit
- **Receive busy**
  - Messages are waiting for read
- **Transmit ok INT**
  - The actual message was successful transmitted
  - Writing 1 reset the interrupt

# Error counter

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

- Rec Error [7..0] : Number of the receive errors
- Tra Error [8..0] : Number of the transmit errors

**Base Address Offset 7**

# Version

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

■ month - year - quartus - core revision

## Base Address Offset 8



# Fifo Pointer

Byte 3	Bit	31	30	29	28	27	26	25	24
	read				Rec Buffer Counter 12	Rec Buffer Counter 11	Rec Buffer Counter 10	Rec Buffer Counter 9	Rec Buffer Counter 8
	write	Reset rec Counter							
Byte 2	Bit	23	22	21	20	19	18	17	16
	read	Rec Buffer Counter 7	Rec Buffer Counter 6	Rec Buffer Counter 5	Rec Buffer Counter 4	Rec Buffer Counter 3	Rec Buffer Counter 2	Rec Buffer Counter 1	Rec Buffer Counter 0
	write								
Byte 1	Bit	15	14	13	12	11	10	9	8
	read								Trans Buffer Counter 8
	write	Reset trans Counter							
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	Trans Buffer Counter 7	Trans Buffer Counter 6	Trans Buffer Counter 5	Trans Buffer Counter 4	Trans Buffer Counter 3	Trans Buffer Counter 2	Trans Buffer Counter 1	Trans Buffer Counter 0
	write								

- Status of the Receive and Transmit Buffer Fifos

**Base Address Offset 9**



# Fifo Pointer

- Trans buffer counter [8..0]
  - The number of messages waiting for transmit
- Reset trans counter
  - Writing 1 to the bit set the transmit buffer counter to 0 until writing 0
  - All messages in the transmit fifo are lost
- Rec buffer counter [12..0]
  - The number of received messages waiting for read
- Reset rec counter
  - Writing 1 to the bit set the receive buffer counter to 0 until writing 0
  - All messages in the receive fifo are lost



# Timestamp Register (if used)

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

- Tstamp[31..0] : Timestamp value of the received message

**Base Address Offset 10**

# Timestamp Register (if used)

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

- Tstamp[31..0] : Current Timestamp counter value

# Parameter Register (read-only)

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	Clock 7	Clock 6	Clock 5	Clock 4	Clock 3	Clock 2	Clock 1	Clock 0
	write								
Byte 2	Bit	23	22	21	20	19	18	17	16
	read				MHz/ns	Databus-width 3	Databus-width 2	Databus-width 1	Databus-width 0
	write								
Byte 1	Bit	15	14	13	12	11	10	9	8
	read	Receive-fifo size 3	Receive-fifo size 2	Receive-fifo size 1	Receive-fifo size 0	Transmit-fifo size 3	Transmit-fifo size 2	Transmit-fifo size 1	Transmit-fifo size 0
	write								
Byte 0	Bit	7	6	5	4	3	2	1	0
	read	Filter number 3	Filter number 2	Filter number 1	Filter number 0				Timestamp ON/OFF
	write								

- Compile time parameter

Base Address Offset 12

# Compile time parameter

- Timestamp ON/OFF
  - 1 → Timestamp feature is used
  - 0 → Timestamp feature is unused
- Filter number [3..0]
  - 8 → 64 Filters and Masks
  - 9 → 128 Filters and Masks
  - 10 → 256 Filters and Masks
- Transmit Fifo Size [3..0]
  - 5 → 30 Messages
  - 6 → 62 Messages
  - 7 → 126 Messages
  - 8 → 254 Messages
- Receive Fifo Size [3..0]
  - 5 → 32 Messages
  - 6 → 64 Messages
  - 7 → 128 Messages
  - 8 → 256 Messages
  - 9 → 512 Messages
  - 10 → 1024 Messages
  - 11 → 2048 Messages
  - 12 → 4096 Messages
- Databus width [3..0]
  - 10 → 32 Bit
  - 11 → 16 Bit
  - 12 → 8 Bit
- MHz/ns
  - 1 → Clock value in ns
  - 0 → Clock value in MHz
- Clock [7..0]
  - 0.. 255 → Clock value

# Filter mask

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	valid		Mask extd	Mask 28	Mask 27	Mask 26	Mask 25	Mask 24
	write	valid		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

**Base Address Offset 512...**



# Filter Identifier

Byte 3	Bit	31	30	29	28	27	26	25	24
	read	valid		IDE	IDX 28	IDX 27	IDX 26	IDX 25	IDX 24
	write	valid		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
- IDX [28..11] + ID [10..0] : Extended Identifier

**Base Address Offset 513...**



**Base Address Offset 512 .. 1023 for 256 masks**

**Base Address Offset 512 .. 767 for 128 masks**

**Base Address Offset 512 .. 639 for 64 masks**

Address	Contents		Object	
512	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 on address 0
513	Valid	ID0		
514	Valid	Mask1	2	
515	Valid	ID1		
		.		
		.		
		.		
1022	Valid	Mask511	512	
1023	Valid	ID511		

# Mask and Filter

## Filter standard / extended message

Bit Mask extd	0	1	1	1	1
Bit 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

maskbit	0	1	1	1	1
ldbit-filter	X	0	1	0	1
ldbit-received message	X	0	1	1	0
result	match	match	match	No match	No match

## Filter standard / extended Identifier



# Filter and Mask Example

- 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`

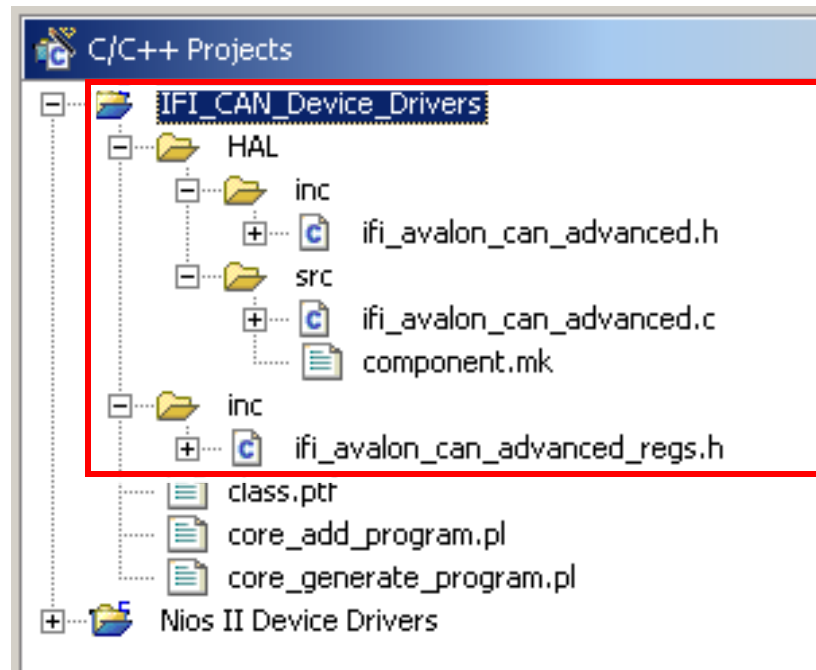
# Order of the ID Bits

- Example for transmitting, filtering and receiving the CAN ID
- ID-Format (like CANalyzer) you want to send with extended
  - value = 0x01234567;
- Convert to IFI CAN-core ID format
  - transmitID = ((value & 0x3FFFF) << 11) + ((value & 0x1FFC0000) >> 18);
  - transmitID |= 0x20000000; // set IDE for extended ID !!!
- Same for the filter
- Read receiveID
- Convert the IFI CAN-core ID format to the other ID format
  - value = ((receiveID & 0x7FF) << 18) + ((receiveID & 0x1FFFF800) >> 11);
  - value &= ~0x20000000; // mask the IDE



# Software for NIOS II and IDE

## ■ Files



# CAN driver routines (ifi\_avalon\_can\_advanced\_module.c)

- **ifi\_avalon\_can\_advanced\_module\_open ()**
  - Initialize of the CAN node
  - Base, pointer to structure canall\_s with timing, interrupt mask, status, mask and filter
  - Return the version of the core
- **ifi\_avalon\_can\_advanced\_module\_read ()**
  - read a message from the buffer and increment the buffer pointer
  - Base, pointer to structure canmsg\_s with identifier, data[0], data[1], dlc,timestamp
  - Return 1 for succesfull read, -1 for error
- **ifi\_avalon\_can\_advanced\_module\_write ()**
  - Transmit a message
  - Base, pointer to structure canmsg\_s with identifier, data[0], data[1], dlc
  - Return 1 for succesfull read, -1 for error
- **ifi\_avalon\_can\_advanced\_module\_stat ()**
  - Read the interrupt mask register, the status register and error register
  - Base, pointer to structure canstat\_s with interruptmask, status, error
  - Return 0 for for succesfull read, -1 for error
- **ifi\_avalon\_can\_advanced\_module\_wr\_int ()**
  - Enable the interrupts
  - Base, Interruptregister
  - Return 0 for for succesfull write, -1 for error
- **ifi\_avalon\_can\_advanced\_module\_wr\_status ()**
  - Write the status register only
  - Base, statusregister
  - Return 0 for for succesfull write, -1 for error
- **ifi\_avalon\_can\_advanced\_module\_wr\_filter ()**
  - Write a single mask and filterpair
  - Base, filternumber, filter mask, filter identifier
  - Return 0 for for succesfull write, -1 for error
- **ifi\_avalon\_can\_advanced\_module\_irq ()**
  - Install the interrupts
  - Base, Irq number, pointer to structure capture\_s with base, status, irqcount, irqdone
  - Return 0 for for succesfull install

# Structures (ifi\_avalon\_can\_advanced\_module.h)

```
■ struct canmsg_s
{
alt_u32 EPR_CANdlc; // data_length_code
alt_u32 EPA_CANId; // identifier
alt_u32 CANdata[2]; // 8 Byte Data
alt_u32 TIMEstamp; // timestamp if enabled
};
■ struct canall_s
{
alt_u32 EPT_CANtime; // timing
alt_u32 EPI_CANint; // interrupt mask
alt_u32 EPS_CANstatus; // status
alt_u32 EPE_CANerror; // error
alt_u32 CANbuffer[512]; // mask and filter
};
■ struct canstat_s
{
alt_u32 EPI_CANint; // interrupt mask
alt_u32 EPS_CANstatus; // status
alt_u32 EPE_CANerror; // error counters
alt_u32 EPP_CANpoint; // buffer pointers
};
■ struct capture_s
{
int base; // remember base for ISR (done by driver)
alt_u32 EPS_CANstatus; // remember Status bevor irq-reset
alt_u32 irqcnt; // counts incoming Interrupts
alt_u32 irqdone; // counts processed interrupts
};
```

# Software Examples

- There are 2 examples included
  - ifi\_hello\_advanced\_can.c
  - ifi\_test\_advanced\_can.c
- ifi\_hello\_advanced\_can.c
  - A simple program which demonstrates a communication between 2 CAN nodes
- ifi\_test\_advanced\_can.c
  - A more complex program which test all features of a communication between 2 CAN nodes

# VHDL Testbench

- *Files*
- *Usage*

# Files

## ■ In the folder VHDL Simulation

- nios\_advanced\_can\_tb.vhd
  - The Testbench
- control.in
  - ASCII file for the simulation flow
- advanced\_can.do
  - TCL script for running the simulation in Modelsim
- wave.do
  - TCL script for setup the wave window of Modelsim



# Usage

## ■ Create the CAN simulation model

- Either the generate of the SOPC Builder
- Or the finish of the MegaWizard will create the simulation model of your CAN Node(s)
- `<name_of_the_modul>.vho`

## ■ Modify the settings of the testbench

- Within the testbench you have to select the correct interface width (32,16 or 8 Bit) by commenting in and out the necessary settings
- Change the names of the CAN nodes to your names
- If you don't have timestamp selected, comment it out



# Select the interface data width

ARCHITECTURE nioscantest OF nios\_advanced\_cantest IS

--use this for 32bit Datainterface

CONSTANT AW	: Integer := 9;	}	32 Bit
CONSTANT DW	: Integer := 31;		
constant readwait	: integer := 3;		
constant writewait	: integer := 1;		

--use this for 16bit Datainterface

--CONSTANT AW	: Integer := 10;	}	16 Bit
--CONSTANT DW	: Integer := 15;		
--constant readwait	: integer := 3;		
--constant writewait	: integer := 0;		

--use this for 8bit Datainterface

--CONSTANT AW	: Integer := 11;	}	8 Bit
--CONSTANT DW	: Integer := 7;		
--constant readwait	: integer := 3;		
--constant writewait	: integer := 0;		



# Change the names

- There are two CAN module declarations
  - Change the names to your module name

---

– replace the name cana with your component name

---

```
component cana
PORT {
  tx      : OUT STD_LOGIC;
  Dbp     : OUT STD_LOGIC_VECTOR (DW DOWNT0 0);
  intc    : OUT STD_LOGIC;
  rx      : IN STD_LOGIC;
```

- There are two CAN module instantiations
  - Change the names to your module name

---

– replace the name cana with your component name

---

```
CAN_a : cana

PORT MAP (rx      => rx_Sa,
          clk     => clk_S,
          clrn    => clrn_S,
          we      => we_S,
          cs      => cs_Sa,
```

# Timestamp

- There are two CAN module declarations
  - If you don't have timestamp selected, comment it out

```
trc      : IN STD_LOGIC;  
cs       : IN STD_LOGIC;  
Adr      : IN STD_LOGIC_VECTOR (AW DOWNT0 0);  
Dbin     : IN STD_LOGIC_VECTOR (DW DOWNT0 0);  
– comment out the following 2 lines if you don't have Timestamp switched on  
;  
tstamp_clk : IN STD_LOGIC := '0'  
];  
end component;
```

- There are two CAN module instantiations
  - If you don't have timestamp selected, comment it out

```
tx => tx_Sa,  
intc => intc_Sa,  
dbp => dbp_Sa  
– comment out the following 2 lines if you don't have Timestamp switched on  
,  
tstamp_clk => tstamp_clk_S  
];
```

# More Settings

- Setup the Clock
  - Signal clkt 10ns → 50 Mhz
- Setup your busdelay for the CAN bus
  - Signal busdelay
- Setup your Timestamp Clock
  - Signal timestampt 200ns → 2,5 MHz

```
128  
129  
130 SIGNAL clkt      : time := 10 ns;  
131 SIGNAL busdelay  : time := 50 ns;  
132 SIGNAL timestampt : time := 200 ns;  
133
```

# Usage

- Define your simulation flow
  - The flow will be controlled by control.in
- There are 4 Commands implemented
  - R : read
  - W : write
  - I : wait for Interrupt
  - N : wait for cycles
- Every command starts with space and has additional parameters
  - →R→123→01AF2345

# Read Command

■ →R→A→ 123→01AF2345

Space

Command Read

Space

A or B to select the CAN node

Space

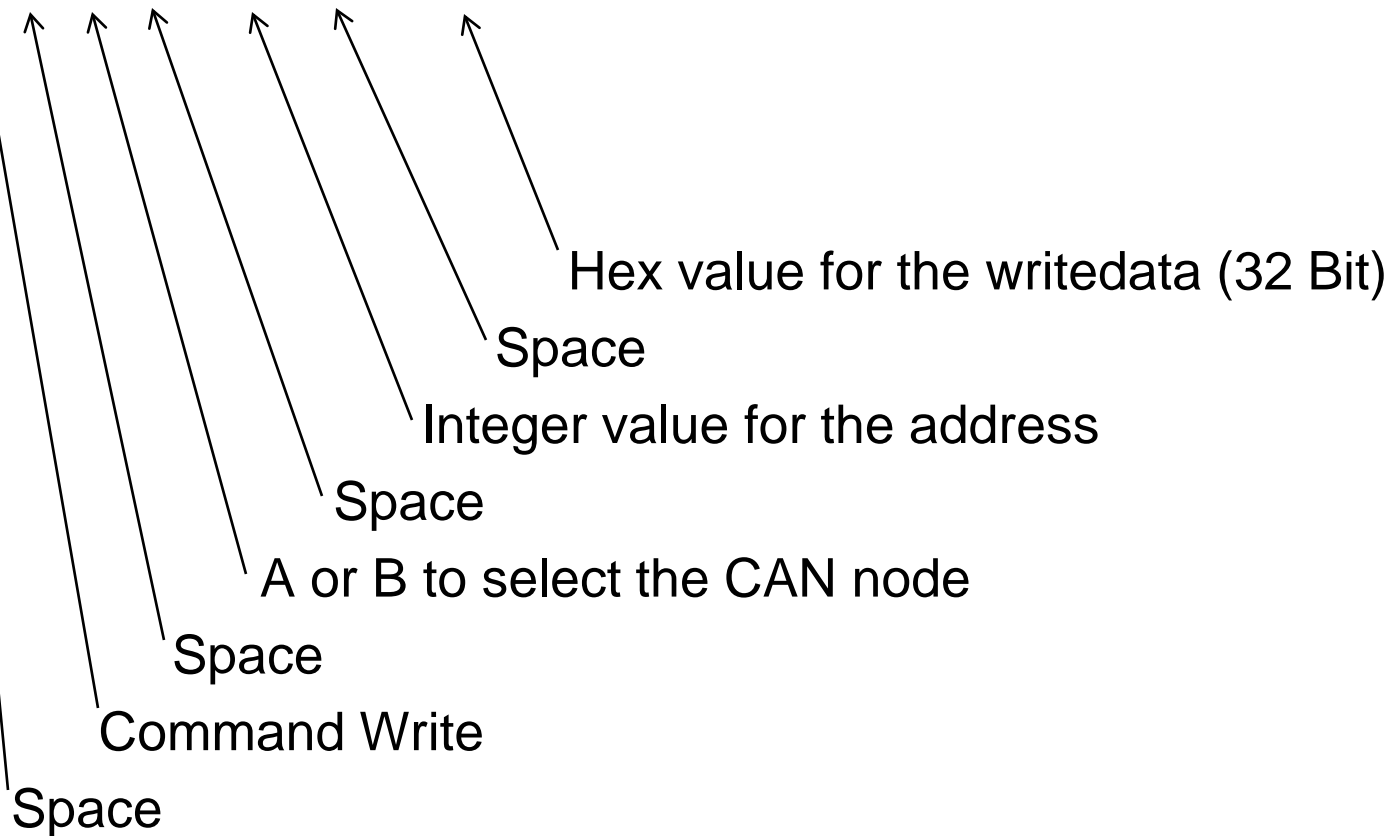
Integer value for the address

Space

Hex value for the writedata (32 Bit) not used

# Write Command

■ →W→A→ 123→01AF2345





# Wait for Interrupt Command

■ → I → A → 123 → 01AF2345

Space

Command Wait for Interrupt

Space

A or B to select the CAN node

Space

Integer value for the address not used

Space

Hex value for the writedata (32 Bit) not used



# Wait for Cycles Command

■ →N→A→ 123→01AF2345

Space

Command Wait for Cycles

Space

A or B to select the CAN node not used

Space

Integer value for the number of Cycles

Space

Hex value for the writedata (32 Bit) not used

# Control.in Example

1		
2	N A 50 00000000	-wait 50 cycles
3		
4	R A 8 00000000	-version a
5	R B 8 00000000	-version b
6		
7	N A 10 00000000	-wait 10 cycles
8		
9	W A 6 3F004003	-reset interrupts
10	W B 6 3F004003	-reset interrupts!
11		
12	W A 4 81029284	-timing a
13	W B 4 81029284	-timing b
14		
15	W A 5 BE000083	-interrupt mask a all without Transmitbuffer empty and Transmit ok
16	W B 5 BE000083	-interrupt mask b all without Transmitbuffer empty and Transmit ok
17		
18	W A 512 80000000	-set filter to allow all messages mask a
19	W A 513 80000000	-set filter to allow all messages identifier a
20		
21	W B 512 80000000	-set filter to allow all messages mask b
22	W B 513 80000000	-set filter to allow all messages identifier b
23		
24	W A 1 00000110	-send identifier a
25	W A 2 80ff00ff	-send data a
26	W A 3 ff00ff00	-send data a
27	W A 0 00000008	-send datalengthcode a
28		
29	I B 0 00000000	-wait for Interrupt at B
30		
31	R B 0 00000000	-read data length code
32	R B 1 00000000	-read identifier
33	R B 2 00000000	-read data 1 - 4
34	R B 3 00000000	-read data 5 - 8
35	W B 6 00000080	-write readfifo next value
36		

# Usage

- Use Modelsim to execute the simulation scripts
  - Copy your CAN Module .VHOs into the VHDL Simulation folder
  - Start Modelsim
  - Change the directory to VHDL Simulation
  - Start the TCL advanced\_can.do

# 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:

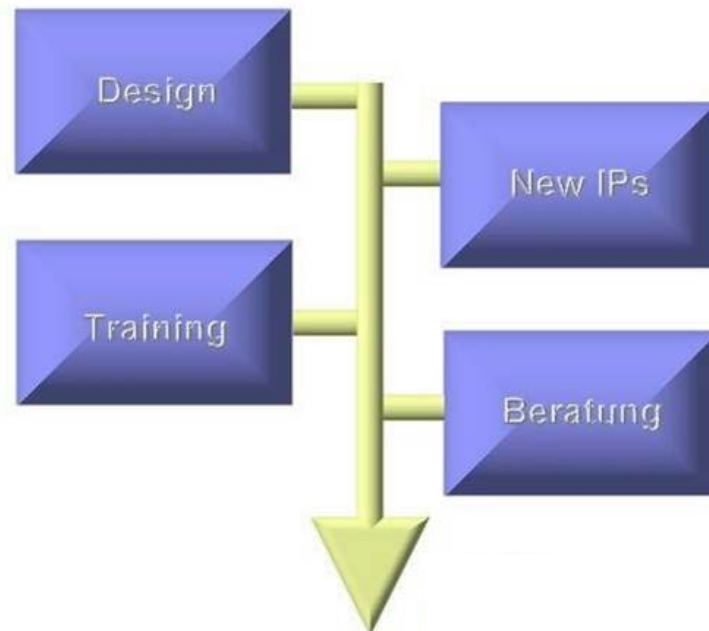
- 10.1 YOU may not sublicense, assign, or transfer this license, or disclose any trade secrets embodied in the IFI IP-MODULE, except as expressly provided in this Agreement. Any attempt to sublicense, assign, or otherwise transfer without prior written approval of the other party any of the rights, duties, 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, Kleiner Weg 3, 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.





# All about FPGA-design since 1985

more than  
30 Years  
I/F/I



## Intel® FPGA Technical Training



INGENIEURBÜRO FÜR IC-TECHNOLOGIE

Franz Sprenger  
Kleiner Weg 3  
97877 Wertheim  
Tel.: +49 (0) 9342 / 9608-0  
Fax: +49 (0) 9342 / 5381  
eMail: ifi@ifi-pld.de  
<http://www.ifi-pld.de>

- ✓ Training Classes
  - Quartus®
  - Expert, TimeQuest
  - VHDL
  - SOPC/QSYS
  - SOC
  - ...
- ✓ Design Service
- ✓ IPs
  - CAN + CAN-FD Controller
  - Gigabit Ethernet MAC
  - PCI Master/Target
  - ...
- ✓ Consulting