Never stop thinking

XC2000/XE166 Family

16-bit Microcontrollers

AP16164

Flash Progamming via CAN BSL (Bootstrap Loader)

Application Note

V1.0, 2009-10

Microcontrollers

Edition 2009-10

Published by
Infineon Technologies AG
81726 Munich, Germany

LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

Device1
Revision History: V1.0 2009-10

Previous Version(s):

Page	Subjects (major changes since last revision)
–	This is the first release …

We Listen to Your Comments
Is there any information in this document that you feel is wrong, unclear or missing? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:
ipdoc@infineon.com

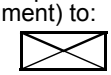

Table of Contents

1 Introduction and Requirements

The Infineon XC2000/XE166 Family of devices (Table 1) include a built-in CAN Bootstrap Loader (CAN BSL). Using the CAN BSL, it is possible to load code or data into the internal PSRAM of the device via the MultiCAN interface. What is not available however, is a built-in mechanism for internal flash programming.

This application note will demonstrate how to program the internal flash of XC2000/XE166 devices via the CAN BSL.

Table 1 XC2000/XE166XE166 Derivative with CAN BSL

Derivative	Package
XC228x, XC228x**M**	PG-LQFP-144
XC226x, XC226x**M**, XC226x***N***	PG-LQFP-100
XC238x, XC238x**A**	PG-LQFP-144
XC236x, XC236x**A**, XC236x***B***	PG-LQFP-100
XC233x**A**, XC233x***B***	PG-LQFP-64
XC2786X, XC278**5**X	PG-LQFP-144
XC2766X, XC276**5**X, XC276***4***X	PG-LQFP-100
XE167, XE167**M**	PG-LQFP-144
XE164, XE164**M**, XE164***N***	PG-LQFP-100
XE162**M**, XE162***N***	PG-LQFP-64
XE169H, XC229xH	PG-LQFP-176 (coming soon)
XE167H, XC228xH	PG-LQFP-144 (coming soon)

The example described makes use of the Tasking compiler tool. However to debug and download readable Hex files, it is possible to directly use the **Hitex** tool (**HiTOP53**) included in the Infineon **UConnect-CAN XE164** kit.

1.1 Hardware Requirements

The following hardware is required for the example described in this application note:

- Two XC2000/XE16x boards
  - If a compiler and debug tool are already installed, the application software can run in any two Infineon Easy Kit boards equipped with a target device listed in Table 1. The Infineon **UConnect-CAN XE164** comes with a free compiler and debug tool development suite (a size limited evaluation version). In the example described in this document, the **UConnect-CAN XE164** is the 'host' board, while the **XE167FM EasyKit** with active CAN BSL, acts as the 'slave' board.
- Terminal program; MTTTY for example.
  - This is used to give commands to the host and to display actual states and error messages.
- CAN bus cable
- CAN Analyzer tool (optional)

2 CAN BSL (Bootstrap Loader) in the XC2000/XE166

A built-in CAN Bootstrap Loader (BSL) is implemented in the Infineon XC2000/XE166 family of devices.

CAN BSL provides a mechanism to load program code or data via the MultiCAN module into the PSRAM of a device, and to start executing the loaded code from address E0 0000_H (address of the first transmitted byte).

The CAN BSL is an integrated mechanism that can be selected via port configuration during a system start after a Power-on Reset, or through write configuration registers and trigger then a software reset. For more details, please see **Chapter 2.1**.

The CAN BSL uses a CAN node 0 interface and configures the MultiCAN module to the baud rate of the host device.

Once communication has been established, the BSL receives a defined number of messages (the number defined in the host) for downloading the code or data. The received code or data is sequentially written to the PSRAM.

Once the download has finished, the program code that has been loaded is executed and the BSL is terminated.

The complete load sequence is based on the following three CAN standard frames:

- **Initialization Frame**
  - sent by the external host to XC2000/XE166
- **Acknowledge Frame**
  - sent by the XC2000/XE166 to the external host
- **Data Frames**
  - sent by the external host to the XC2000/XE166

The initialization frame is used in the XC2000/XE166 device for baud rate detection.

After successful baud rate detection, the XC2000/XE166 reports to the external host via the acknowledge frame.

Data frames can then be transmitted from the host to the XC2000/XE166 device. Data frames are always transmitted in 8 bytes.

Table 2 CAN Bootstrap Loader Frames

Frame Type	Parameter	Description
Initialization Frame	Identifier	11-bit, don't care
	DLC=8	
	Data byte 0-1	0x5555
	Data byte 2-3	acknowledge message identifier ACK_ID (use bits [13:0], bit 13=IDE)
	Data byte 4-5	Data Message Count MSG_CNT Must be <= 1FE0H for 64KBytes SPRAM, see **Chapter 2.2** For example; MSG_CNT=0x200 for 0000_H-$0FFF_H$ (4KBytes)
	Data byte 6-7	Data Message Identifier MSG_ID
Acknowledge Frame	Identifier	ACK_ID as received bytes [3:2] of the Initialization Frame
	DLC=4	
	Data byte 0-1	BTR register value in XC2000
	Data byte 0-1	Data bytes [3.2] of the Initialization Frame
Data Frame	Identifier	Data Message Identifier MSG_ID as sent by data bytes [7:6] of the Initialization Frame
	DLC=8	
	Data byte 0-7	Data Bytes, assigned to increasing destination (PSRAM) addresses

2.1 Entering the CAN BSL

The XC2000/XE166 enters CAN BSL mode, when the CAN BSL mode is selected within the startup configuration upon a reset signal. Bit field HWCFG=xxx101_b in register STSTAT indicates the CAN BSL mode.

Bit field HWCFG is updated either by:

- Hardware: with a pull-up/down resistor when the configuration pins P10.2-0 has the value 101_b at Power-On, or after an Internal Application Reset.
- Software: when 000xx101_b is written into bit field SWRSTCON.SWCFG and an Application Reset is triggered with SWRSTCON.SWBOOT=1 (Software Boot configuration selection).

Note: To directly generate a software reset request trigger, either use the instruction SRST, or set bit SWRSTCON.SWRSTREQ = 1.

Notes

1. *Because of a hardware bug (see **Chapter 8.1** '**RESET_X.002**') the bit field HWCFG in the register STSTAT is unable to indicate the actual start-up mode selected in early XC2000/XE166 devices. This bug is fixed in the **M-, N-, 5-, H- series** of devices (see **Table 1**).*
2. *The type of software reset should be configured first (RSTCON0.SW = $1x_b$)*
3. *The type of instruction (**SRST**) is defined via bit field RSTCON0.CPU.*
4. *01_b is reserved. It can not be configured for RSTCON0.CPU or RSTCON0.SW.*
5. *For example code, please see **Chapter 8.2**.*

The CAN BSL initializes the MultiCAN as follows:

- CAN node 0 of the MultiCAN module is used.
- Message Object 0 (MO0) is configured as Receive Object to receive the initialization and data frame.
- Message Object 1 (MO1) is configured as Transmit Object to transmit the acknowledge frame.
- CAN input/output pins: P2.6=RxDC0 and P2.5=TxDC0. RXSEL=0x0003.
- In CAN BSL the system clock is provided by OSC_HP circuitry (supplied by a external oscillator or an external clock source). f_{SYS}:=f_{OSC} is selected by SYSCON0.CLKSEL=01_b.

During the initialization phase:

- The Bit Timing Mode is used for baud rate detection and analysis of the bit timing.
- The Analyzer Mode is used to monitor the CAN frame without active participation.
- All Acceptance Mask bits of the receive object MO0 are OFF.

After the initialization phase:

- The Analyzer Mode is switched off. The bit timing mode for CAN node 0 does not change.
- BTR is set with the calculated value and SJW=01_b.
- All Acceptance Mask bits of the receive object MO0 are ON: CAN_MOAMR0=01FFF,$FFFF_H$.

2.2 Loading the User Code

The sequence for loading the user code is as follows:

- The code is downloaded starting at address $0E0,0000_H$. Consecutive data bytes are stored at incrementing addresses.
- After receipt of the last CAN data frame, the Bootstrap Loader sequence is finished and executes a jump to address $0E0,0000_H$.
- XC2000/XE166 has 64K PSRAM ($0E0,0000_H$ - $0E0,FFFF_H$), but the last 255 bytes are dedicated to the on-chip firmware; i.e. If code or data is downloaded to that area it will be overwritten afterwards by the on-chip firmware. Therefore the maximum CAN message count (MSG_CNT) in the initialization frame is 8192-32.

2.3 Exiting CAN BSL

After the Bootstrap Loader has been activated, the Watchdog Timer and the Debug system are disabled.

The Debug system and Watchdog Timer are released automatically when the BSL terminates after having received a set number of messages from the host (the number of messages is defined via a variable in the host).

The CAN BSL is also aborted after a Power On Reset, if the non-BSL port configuration has been used (see **Chapter 2.1**).

3 General System Description

As host device, the **UConnect-CAN XE164** is connected to a PC via a JTAG/UART interface. The XE164 should be in Standard Start mode.

As slave device, the **XE167FM EasyKit** is connected to the host via a CAN interface. The XE167FM should be in active CAN BSL mode.

This application note describes software for the host board. The host device takes control of the slave device via the CAN bus, while commands are received and process status changes are reported via the JTAG/UART interface to the PC COM port.

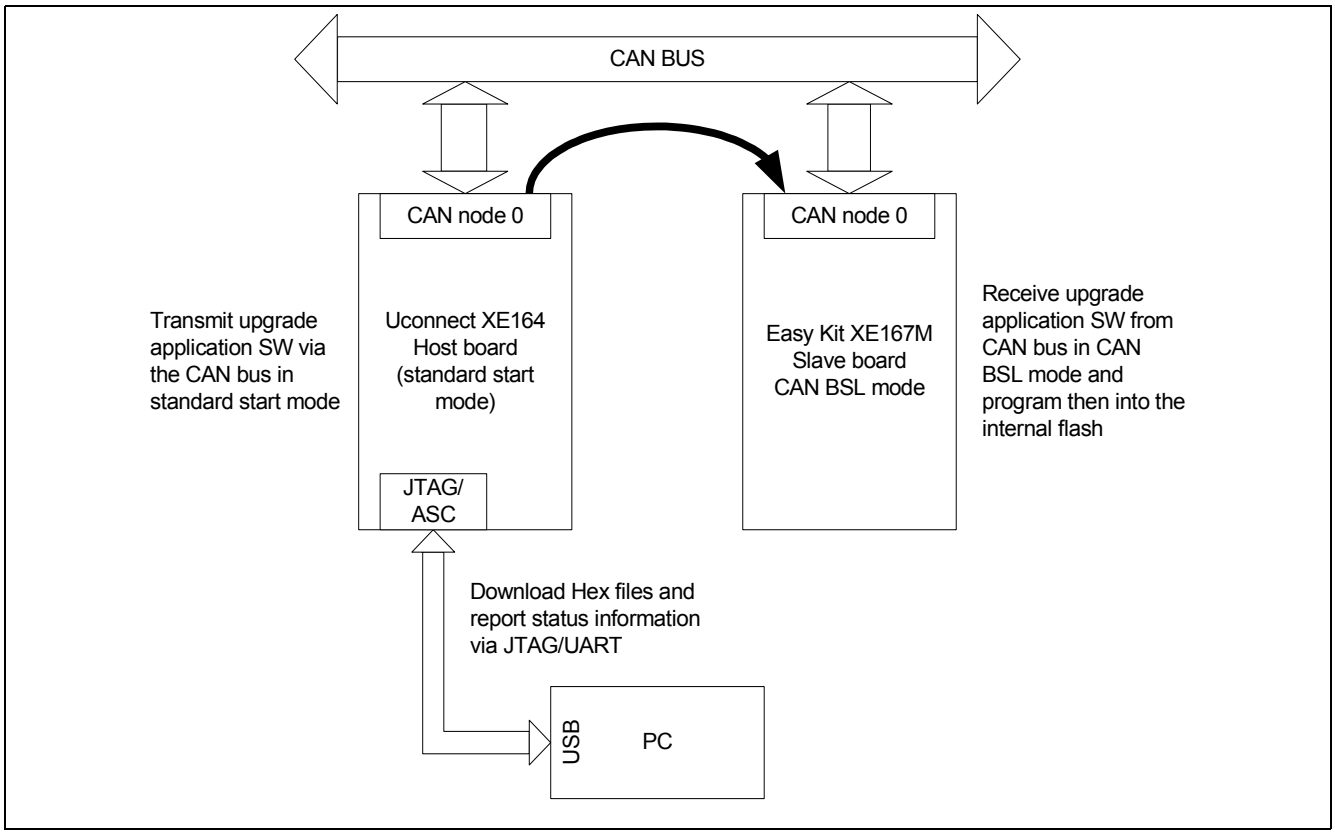

Figure 1 System Overview

There are three different pieces of code (three Tasking projects) for the host device associated with this application note:

- **Transmitter Program: CAN_BSLMaster**
  - Initializes the MultiCAN module in the host device (including the CAN transmit pins and configuring the initialization frame) to establish the communication with the slave.
  - Defines a port pin to signal the process of starting, data transmission, and termination of the download sequence.
  - Initializes the USIC module and configures U0C0 as the ASC interface to communicate with the PC COM port.
  - Code should be compiled for the location of the internal flash at address $0C1,0000_H$.
- **Receiver Program: CAN_Loader**
  - Must first be downloaded to the slave device in active CAN BSL mode. After a successful download, this code will be executed.
  - It contains commands for a ‘handshake’ between the host and slave devices.
  - Defines a port pin to signal the process status on the slave board.
  - Code should be compiled for the location of the internal PSRAM, at address $0E0,0000_H$.

- **Application Software: MainApplication_XE16x**
  - This is the upgrade software.
  - In this application note we create a MultiCAN module application. CAN objects are configured in order that the slave can receive messages from the host and also execute commands.
  - Code should be compiled for the location of the internal flash, at address $0C0,0000_H$.

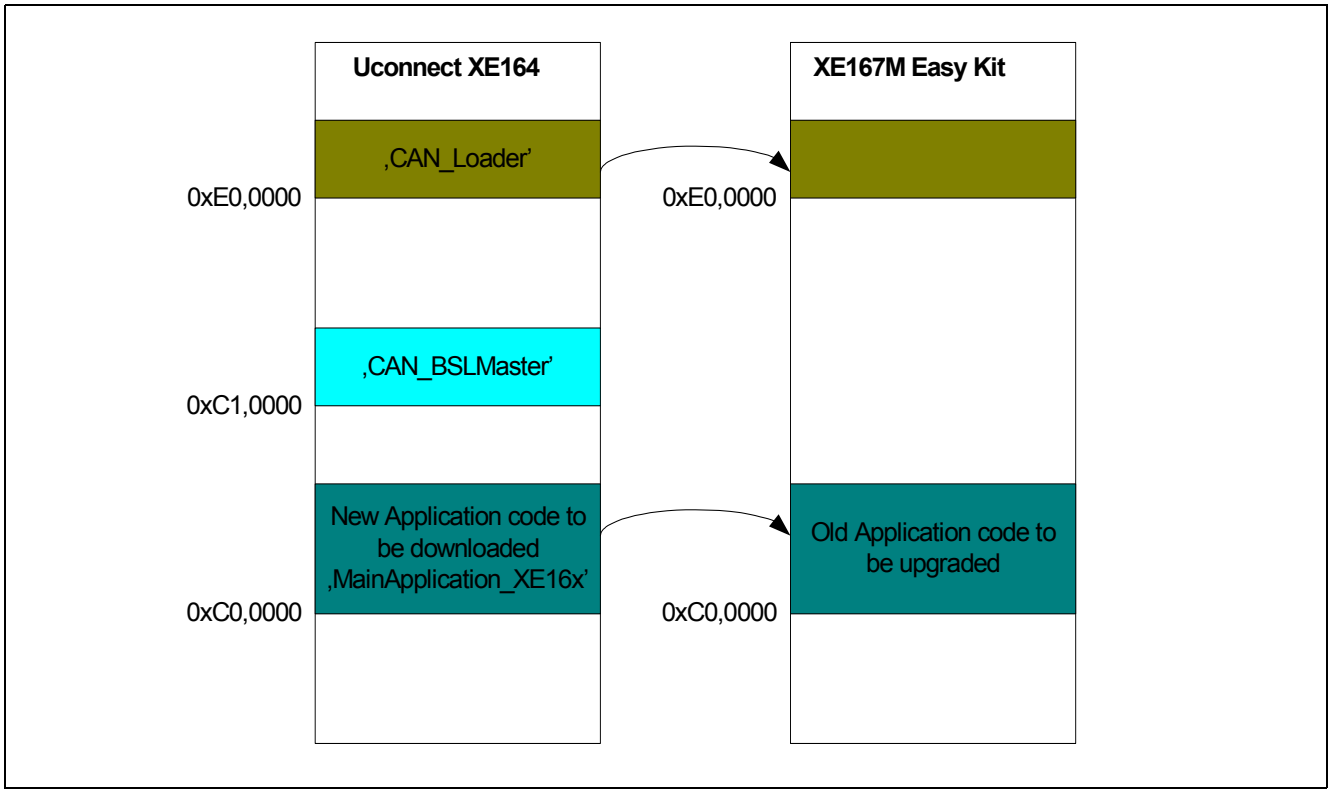

Figure 2 Memory Map

3.1 Master Board

One method of connecting a PC to a microcontroller is to use the PC COM port to download a generated hex file into the microcontroller. In this case the microcontroller must be in active ASC BSL mode. Infineon provides freeware tools, such as **Memtool** and **Minimon**, to support on-chip flash programming.

Another method is to use the JTAG interface. In this application example, **UConnect-CAN XE164** is used as a 'USB-to-CAN bridge' to download and program the XC2000/XE166 internal flash from the PC.

The **UConnect-CAN XE164** is a low cost USB stick providing full evaluation capability for the new XE164 microcontroller. For detailed information about the XE164 UConnect stick, please refer to www.infineon.com.

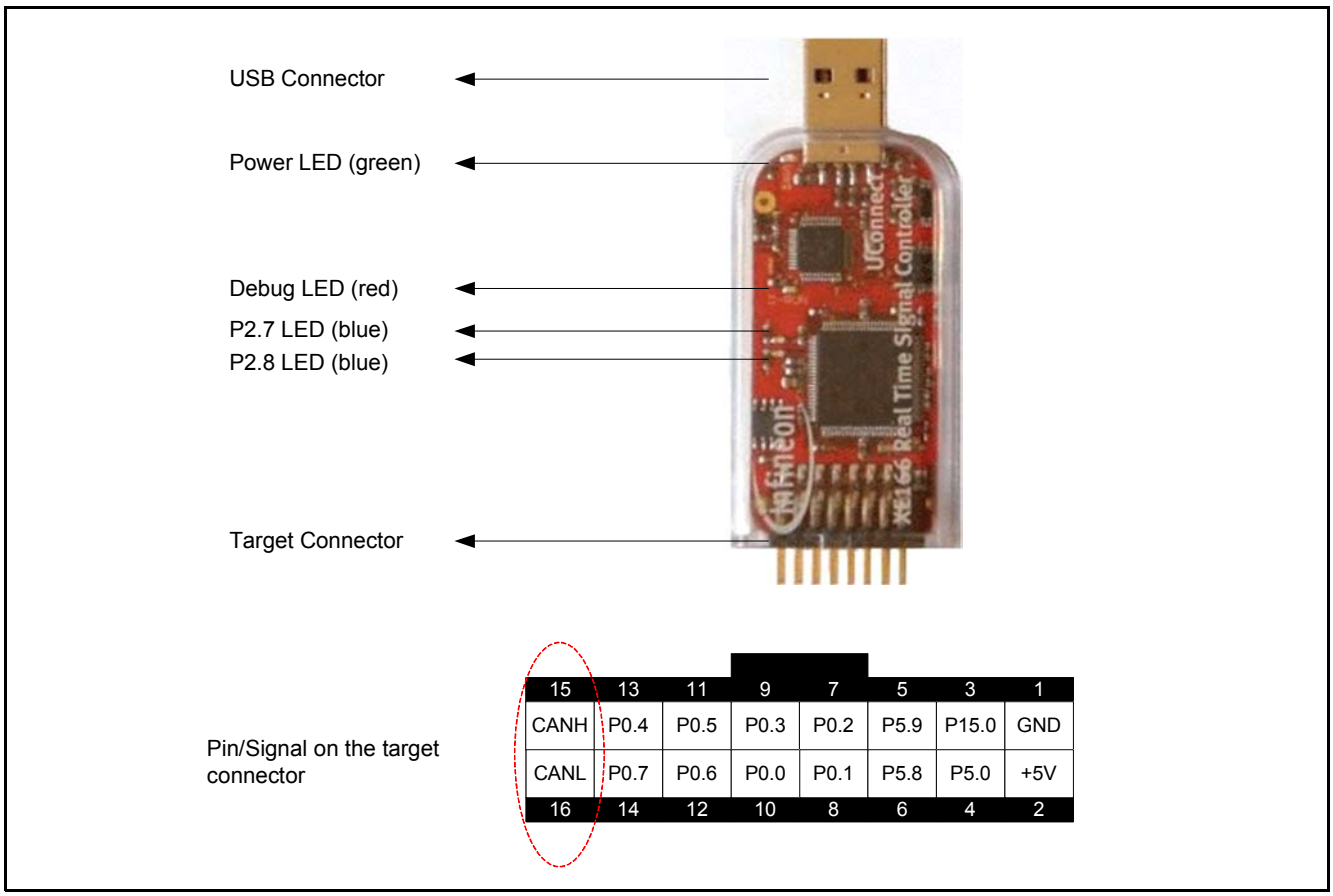

Figure 3 UConnect_CAN XE164

At one end of the **UConnect-CAN XE164** is the USB connector. Inside there is an FT2232 chip with two USB channels. One channel is for the JTAG interface supporting the OCDS feature, while the other channel acts as the virtual COM port. This serial channel is connected to pins P7.3 and P7.4 of the XE164 USIC module (Tx and Rx of the U0C0 ASC interface).

To start the serial interface either Windows HyperTerminal software, or freeware like MTTTY, can be used.

The **HiTOP53-UConnect-XE164** can be used for debugging.

Tools can be directly installed from the **UConnect-CAN XE164** kit's CD.

On the other side of the UConnect stick is the Target Connector (**Figure 3**). XE164 port pin P2.6 (CAN0_Rx) and P2.5 (CAN0_Tx) are connected to CAN transceiver.

To start this application SW pin CANL and pin CANH on host board **UConnect-CAN XE164** and slave board **XE167FM EasyKit** (see **Figure 5**) must be connected together.

3.1.1 Transmitter Program (Project File: CAN_BSLMaster)

The Transmitter program is the main program for the host board. It controls the interface between the development environment (the PC) and the CAN bus.

The **UConnect-CAN XE164** board has an 8MHz quartz crystal for the input oscillator circuit. At startup, through a function call, the system frequency is changed to 40MHz via the internal PLL configuration and Pin 2.8 is defined for system clock output.

The Transmitter program uses the communication peripherals of the microcontroller:

- The USIC0 channel 0 is configured as UART (19200Baud) to communicate serially to the HyperTerminal on the PC.
- The MultiCAN module CAN node 0 (Rx=P2.6, Tx=P2.5) is configured with 200Kbaud to communicate to the slave.
- Message Object 0 is defined as Receive object.
- Message Object 1 is defined as Transmit object.

In the CAN initialization routine the Initialization Frame is configured as per the description in **Table 2**. No interrupt is defined in the CAN or USIC module (polling system).

After initialization, the download from host to slave begins.

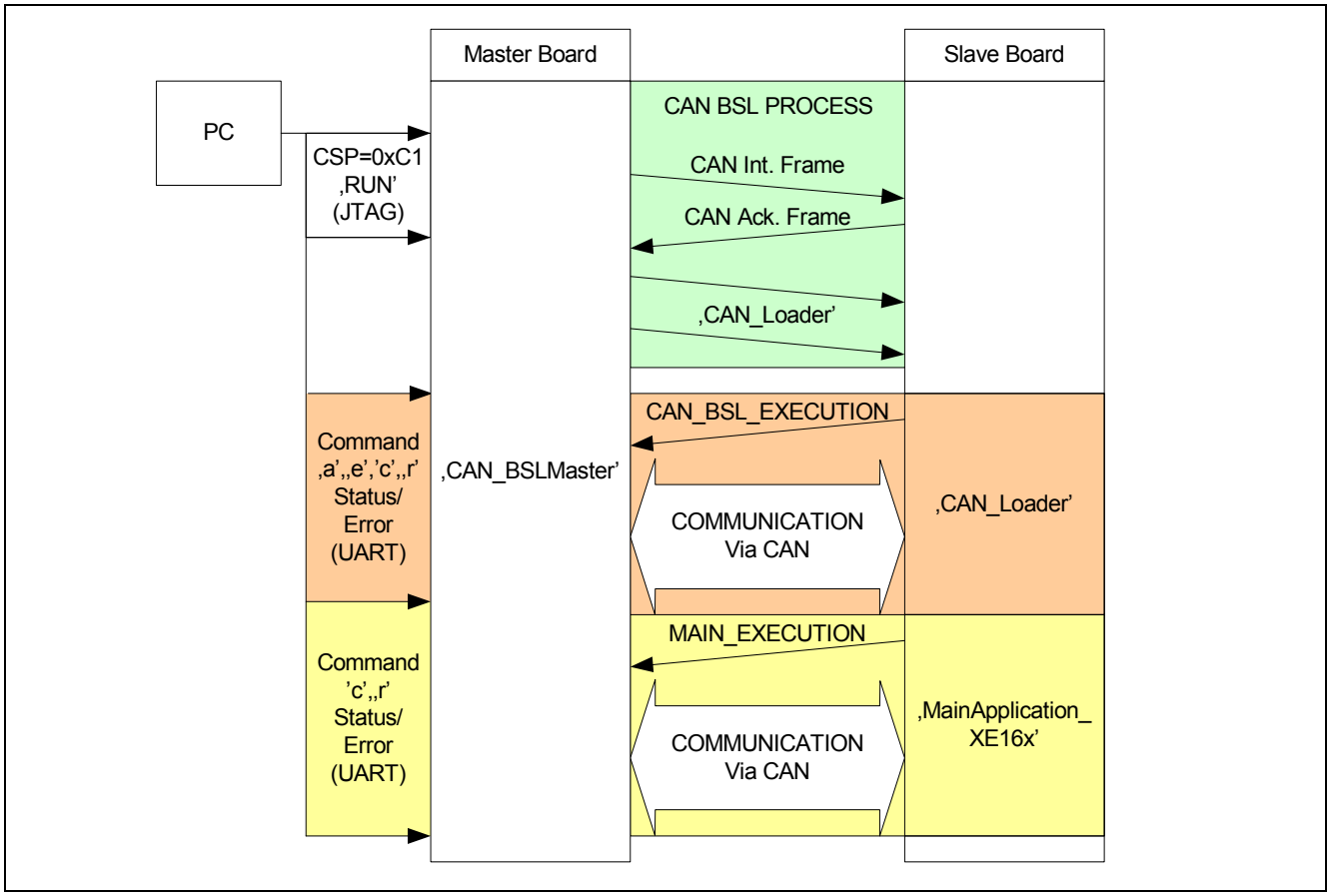

Figure 4 Data Communication on the CAN Bus

After a successfully download sequence, which can be seen with reporting on the terminal program communicated via the PC COM port, several commands can be selected to control the handshake between the host device and the slave (see **Chapter 5** for details).

3.1.2 Receiver Program (Project File: CAN_Loader)

The Receiver program is a piece of code for the slave device. It's role is to program the application program (received via the CAN bus) into the internal flash. The Receiver program must be downloaded into SPRAM in CAN BSL mode.

The built-in CAN BSL only provides a mechanism to load code via the CAN bus into the PSRAM. To program the internal flash of XC2000/XE166 devices, the flash driver (software code of flash command sequences, such as 'erase' and 'program' for example), must be implemented and included in the **CAN_Loader** project file and be downloaded to the device.

The Receiver program is composed of two main parts:

- Flash Driver
  - all the flash programming routines used to program the flash memory.
- Communication Sequence
  - between the host and the slave, including signal handling (the commands of the CAN message object) for the handshake.

Port 10 is defined as GPIO output, to signal a successful download to the slave board.

Once in CAN BSL mode on the slave board (i.e. after a Power-on Reset), the 'CAN_Loader.hex' code can be downloaded to the slave PSRAM by executing the master Transmitter program 'CAN_BSLMaster.hex', controlled by the debug tool **HiTOP53-UConnect-XE164**.

4Kbytes are currently defined for the Receiver program 'CAN_Loader.hex' in this application ($0E0,0000_H$ - $0E0,0FFF_H$).

Blinking LEDs on the slave board indicate that the CAN BSL mode has been terminated and that the code downloaded to PSRAM is executing. The software then checks the receive information of the CAN frame in cycle polling (the MultiCAN module and system clock are not reconfigured and the Tx/Rx message object and the baud rate parameter are not changed), while the software waits for a CAN message command from the host.

3.1.3 Application Software (Project File: MainApplication_XE16x)

An application program is required to upgrade the internal flash of the device. This application note includes a simple application program (**MainAplication_XE16x**), which provides the functionality to support communication with a host board via the CAN interface.

The example code is intended to be straightforward. Pin P10.2 is defined as GPIO output pin and starting the application program lights up its LED. The code also sets up the MultiCAN peripheral so that it can receive CAN messages.

In the current version only two commands are implemented:

- A command to generate an Application Reset for Internal Standard Start (start address $0C0,0000_H$).
- A command to generate an Application Reset for the CAN BSL mode.

3.2 Slave Board

Any Infineon Easy Kit equipped with a target device listed in **Table 1**, can be used as a slave board.

The following figure shows the layout overview of the **XE167FM EasyKit**. For further details, please refer to the Board User's Manual.

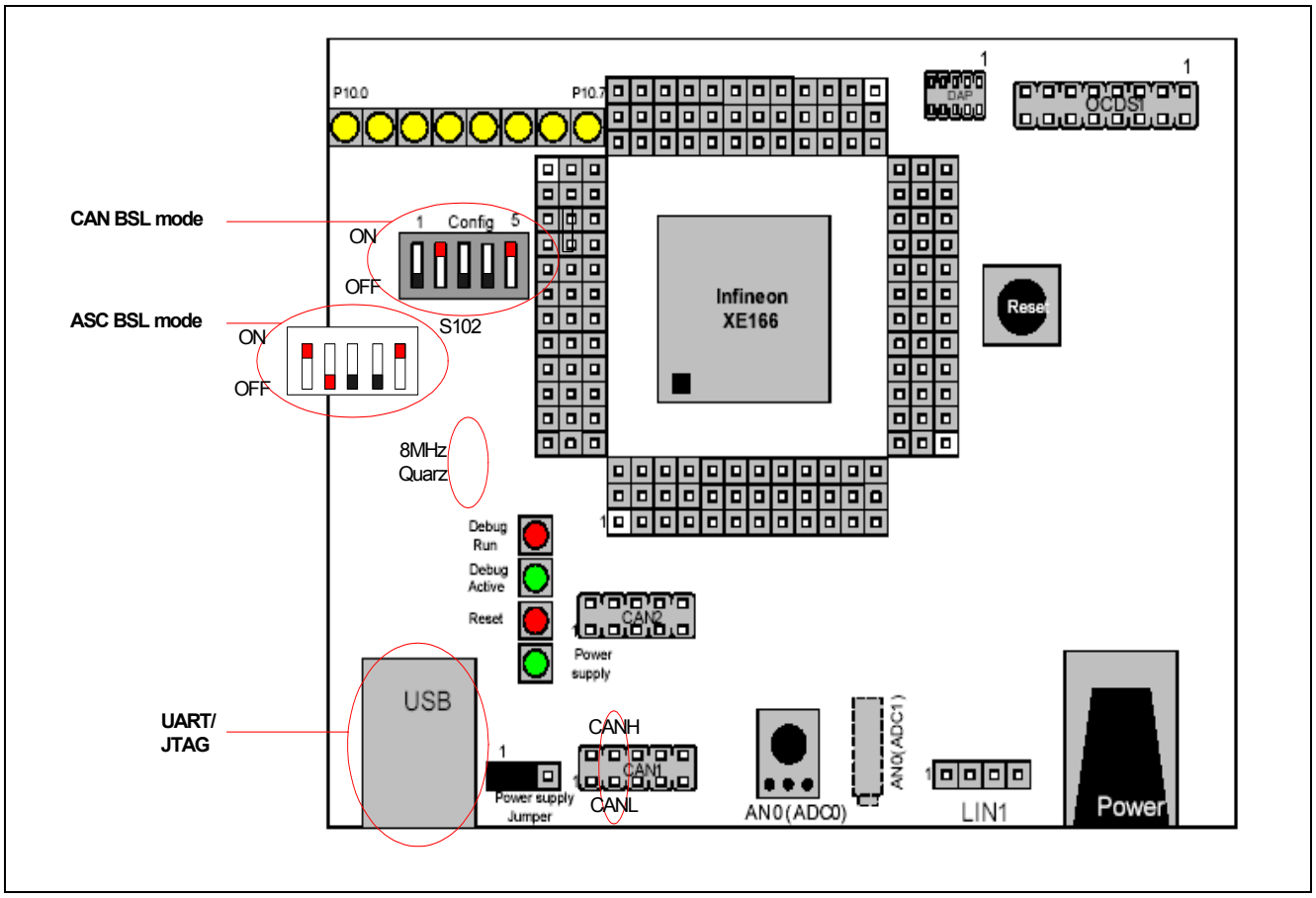

Figure 5 XE167FM Layout Overview

Notes

1. *CAN BSL can not be directly used on XC2000/XE166XE166 Easy Board (v1.1), because P2.5 and P2.6 are not connected to the CAN SUB-9 connector (see **Chapter 8.3**).*
2. *CAN BSL can not be directly started on XC2000/XE166 Easy Board (V1, V2 and V3), because the quartz crystal **NX8045GB** needs a longer rise time (ca. 1ms) than the wait-time in the built-in RoomCode (see **Chapter 8.1**).*

The CAN BSL requires a 'point-to-point' connection with the host; i.e. the XC2000/XE166 must be the only CAN node connected to the network.

A system clock with at least 4MHz is required for the CAN Bootstrap Loader operation.

The XC2000/XE167 has a 1:1 direct drive clock in the CAN BSL mode.

The following baud rates have been tested:

- With ceramic resonator (8MHz): 100Kbaud, 200Kbaud, 500Kbaud, 1Mbaud
- With internal oscillator (nominal value 10MHz): 100kbaud, 200kbaud, 400kbaud (500kbaud fails),
  - for example; CAN BSL mode is started by execution of the software workaround directly from the internal flash (see **Chapter 8.2**).

4 XC2000/XE166 Internal Flash Programming

This chapter is divided into two sections:

- The Internal Flash Memory
- Flash Driver

4.1 The Internal Flash Memory

The XC2000/XE166 M-series devices (see **Table 1**) have up to 832Kbytes of on-chip flash memory.

The on-chip flash memory consists of up to four independent flash units (maximum 256Kbtes/unit). All flash units can be used for both code and data storage.

Each unit is built up from 4-KByte physical sectors. Each sector can be separately erased and programmed (in blocks of 128 bytes).

The flash memory supports read and write protection, where protection is implemented by logical sector.

Each memory unit consists up to 12 logical sectors (4K, 12K, 16K, 32K, 60K, 64K).

The ‘erase’ operation can be used on a page (128 bytes) or a physical sector (32*128 bytes). The ’program’ operation is applied pnly per page (128 bytes).

Read protection always globally covers the whole flash memory range (i.e. all four flash units). The bit field IMB_FSR_PROT.RPRO indicates the status.

Write protection can be globally enabled when Read protection is applied, or it can be specified for each logical sector (4x12) in each flash memory area. Registers IMB_PROCON0...3 (bit S12U…S0U) indicate the status.

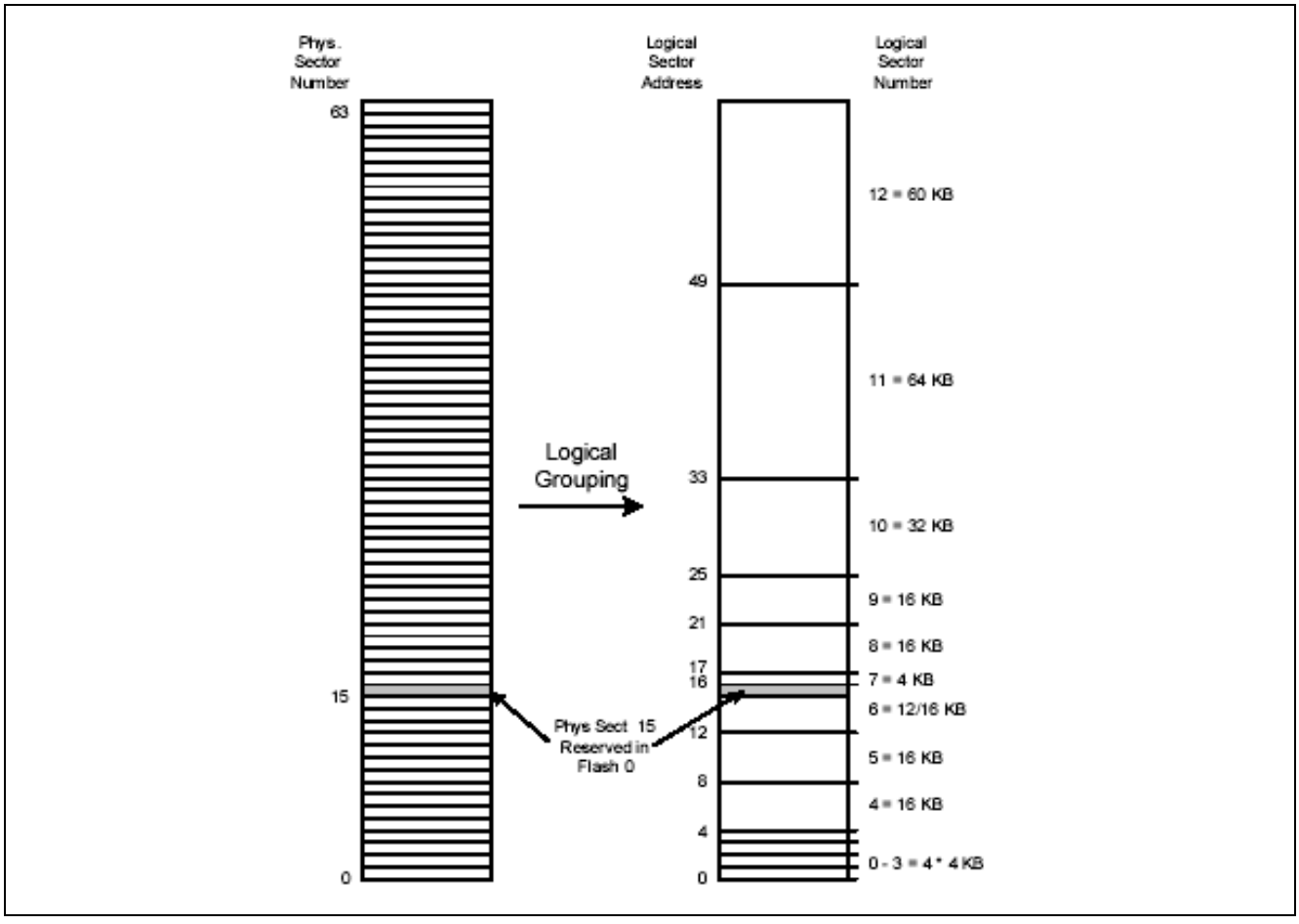

Figure 6 Logical Sectors in XC2000M/XE167FM (256K Area)

Note: The XC2000/XE167 devices all have different flash sizes and different allocations of logical sectors. For more details about flash memory size, location, protection and operating modes, please refer to the appropriate User's Manual or Data Sheet.

4.2 Flash Driver

To perform a 'program' operation on the flash memory, it is necessary to write some command sequences. The following table gives the possible operations and their command definitions as used in this application note:

Table 3 Command Sequence Definitions (Programming and Erasing)

Cycle	1	2	3	Flag BUSY
Erase Page	MOV xxAA$_H$, xx80$_H$	MOV xx54$_H$, xxAA$_H$	MOV PA, xx03$_H$	Yes
Enter Page	MOV xxAA$_H$, xx50$_H$	MOV PA, xxAA$_H$		Yes
Load Page Word	MOV xxF2$_H$, WD			-
Program Page	MOV xxAA$_H$, xxA0$_H$	MOV xx5A$_H$, xxAA$_H$		Yes
Erase Sector	MOV xxAA$_H$, xx80$_H$	MOV xx54$_H$, xxAA$_H$	MOV SA, xx33$_H$	Yes

Notes

1. *PA=Page Address; WD=Write Data; SA=Sector Address*
2. *All of the command sequences given above, except for Load Page Word, set the BUSY flag for the duration of their operation. Software must check the BUSY bit after the command instructions.*

The main part of SW **CAN_Loader** implements flash routines which provide the flash functionality listed in the table above (**Table 3**). However the user can still add other features as desired, such as the 'enable read protection' operation, or 'disable read protection' for example.

5 Communication Protocol

After downloading the **CAN_Loader** code to the XC2000/XE166 device PSRAM, the CAN Bootstrap Loader terminates and the XC2000/XE166 device will start executing the code in PSRAM ($0E0,0000_H$).

The acknowledge CAN_BSL_EXECUTION (see **Figure 4**) from XC2000/XE166 to the host, signals communication is established. The XC2000/XE166 then waits for commands from the PC. These command are forwarded using the CAN messages (process commands) of the host device.

5.1 Process Commands

5.1.1 Sector Erasing Operation

The Sector Erasing Operation process command.

Table 4 Process Command: Sector Erasing Operation

byte 7	byte 6	byte 5	byte 4	byte 3	byte 2	byte 1	byte 0
						Sector number	**CMD (0x03)**

- ASCII for the terminal program: for example, '**e012**', erase physical sector 12
- Sector number (8 bits): number of the flash sector to be erased.
- Command code: **CMD=0x03**

The XC2000/XE166 device sends either 0x00 or 0x01 to the host, to indicate either acknowledge or error.

5.1.2 Flash Page Programming Operation

The Flash Page Programming Operation process command.

Table 5 Process Command: Flash Page Programming Operation

byte 7	byte 6	byte 5	byte 4	byte 3	byte 2	byte 1	byte 0
				Page Address			**CMD (0x01)**

- ASCII for the terminal program: only used internally in the process command 'download an application example code' (see **Chapter 5.1.5**).
- Page address (24 bits): address of the flash page to be programmed.
- Command code: **CMD=0x01**

After receipt of this process command, including its 24bit page address, the XC2000/XE166 device initializes the flash address and enters a loop state, waiting to receive 128 bytes of data from the host (8 CAN data frames within 8 bytes). This page will then be programmed after a successful erase operation.

Table 6 Data Frame Format

byte 7	byte 6	byte 5	byte 4	byte 3	byte 2	byte 1	byte 0
data	**data**	**data**	**data**	**data**	**data**	**data**	**data**

The XC2000/XE166 device sends either 0x00 or 0x01 to the host as either acknowledgement or to indicate an error.

5.1.3 Application Reset for the Standard Start

The Application Reset for the Standard Start process command.

Table 7 Process Command: Application Reset for the Standard Start

byte 7	byte 6	byte 5	byte 4	byte 3	byte 2	byte 1	byte 0
							CMD (0x20)

- ASCII for the terminal program: '**r**'
- Command code: **CMD=0x20**

This command allows the user to directly start the downloaded software stored at $0C0,0000_H$.

The acknowledge MAIN_EXECUTION (see **Figure 4**) from the XC2000/XE166 device to the host, signals that communication is established.

5.1.4 Application Reset for the CAN BSL Mode

The Application Reset for the CAN BSL Mode process command.

Table 8 Process Command: Application Reset for the CAN BSL Mode

byte 7	byte 6	byte 5	byte 4	byte 3	byte 2	byte 1	byte 0
							CMD (0x21)

- ASCII for the terminal program: '**c**'
- Command code: **CMD=0x21**

This command allows the user to start the CAN Bootstrap Loader again, using an application reset.

The acknowledge CAN_BSL_EXECUTION (see **Figure 4**) from the XC2000/XE166 device to the host, signals that communication is established.

5.1.5 Download an Application Example Code

The Download an Application Example Code process command.

Table 9 Process Command: Download an Application Example Code

byte 7	byte 6	byte 5	byte 4	byte 3	byte 2	byte 1	byte 0
							CMD (0x51)

- ASCII for the terminal program: '**a**'
- Command code: **CMD=0x51**

This command allows the user to download application example code from the host flash ($0C0,0000_H$ to $0C0,3FFF_H$, 4KBytes), to the internal flash of the XC2000/XE166 device.

Code will be received and programmed 'page-by-page' using the command sequence 'flash page programming operation command' (see **Chapter 5.1.2**).

6 Using this Application Note

To use this application note will require one **UConnect-CAN XE164** to act as host and one **XE167FM EasyKit** to act as slave. All hardware should be connected as shown in **Figure 1**.

- Insert the USB connector of the UConnect stick to the PC USB port.
- Connect the target connector CAN pins of the UConnect stick with the CAN pins on the Easy board (**Figure 3** and **Figure 5**).

The two boards have to been configured first:

- Set the slave board to active CAN BSL mode:
  - DIP switch S102 in **Figure 5** P10.0-P10.1-P10.2-P10.3-/TRST)=OFF-ON-OFF-xx-ON(1-0-1-x-1)
  - Make a Power-on Reset

Notes

1. *Because of a hardware bug (**BSL_CAN_X.001** - see **Chapter 8.1**), the CAN BSL can not be directly started on the **XE167FM EasyKit**. The quartz crystal **NX8045GB** on the board needs to be changed with an 8MHz ceramic resonator, OR -*
2. *Alternatively, if Memtool is available on the PC and your board can be started in the ASC BSL mode, the board does not have to be changed. Instead Memtool can be used to program the instructions of SW workaround into the flash of the slave board first. Then start it in the Standard Start Mode via a Power-on Reset. This is the software workaround for the hardware bug **BSL_CAN_X.001**. A hex file is provided in **Chapter 8.2**.*

After all of the steps above have been successfully completed, the tools can be started:

- The HyperTerminal program (19200Baud)
- Hitex debug tool **HiTOP53-UConnect-XE164**

The three pieces of code (two located in the flash and one located in the PSRAM) should first be programmed in the XE164 of the UConnect kit.

Because the **HiTOP5** places its own flash driver in the PSRAM at address $0E0,0000_H$, the PSRAM code **CAN_Loader** must be downloaded in the final stages to avoid being overwritten by the **HiTOP5** software (see **Figure 7**).

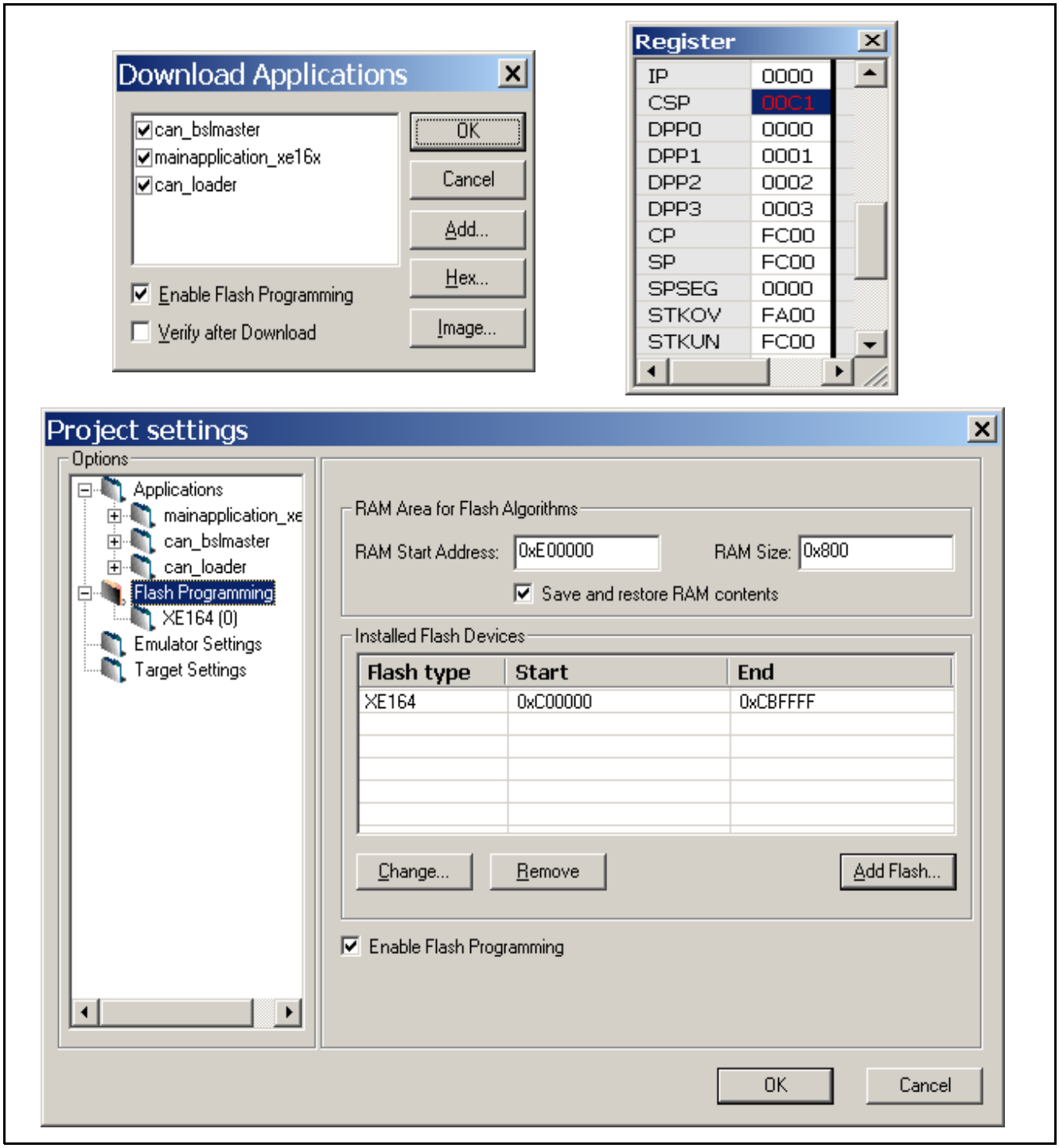

Figure 7 HiTOP Setup

Because the Transmitter Program **CAN_BSLMaster** is located at address $0C1,0000_H$ it is necessary to set the PC pointer to the $C1_H$ segment and then click the '**run**' button in **HiTOP5**.

After execution of the **CAN_BSLMaster** Transmitter Program, the **CAN_Loader** starts to be loaded to the slave board via the CAN bus. On the slave board the received data is stored in PSRAM, starting at address $0E0,0000_H$.

After receipt of the last CAN message object, the CAN BSL terminates and the code loaded to PSRAM is started.

At this point the communication between host and slave is established. ASCII can be keyed in; for example, '**a**' can be keyed in via the HyperTerminal to give the command to download the code **MainApplication_XE16x** and program it in the internal flash of the slave board.

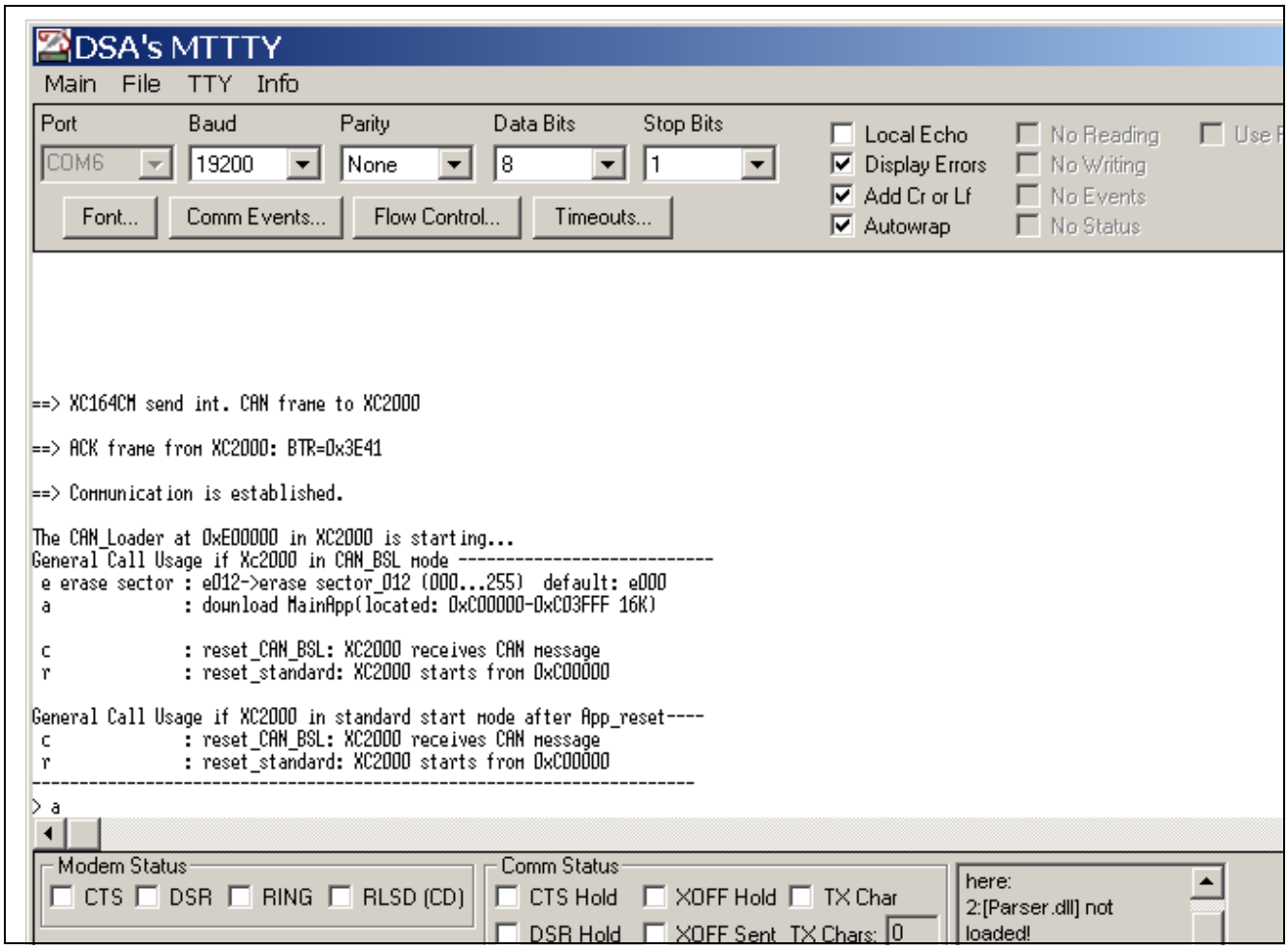

Figure 8 Terminal Report

To start the **MainApplication_XE16x** code after flash programming on the slave, enter the key '**r**'. The LED of Pin 10.2 will blink to indicate that the code in flash is running.

*Note: The implemented ASCII key codes are shown in **Figure 4**.*

7 Running this Application Note without HiTOP5

Alternatively, you can use two XE16x easy kits and Infineon freeware **Minimon** to run SW provided by this application note. **Minimon** is a monitor program that is started via ASC bootstrap loader. The main application of **Minimon** are hex editing, downloading and starting of user application SW, flash programing and SFR register view for example.

Minimon also supports script files which are nothing more then text files with a list of shell/script commands. These commands are executed sequentially. Here we will use script file '**AppNote16164_CAN_BSL.scm**' (**Chapter 8.5**) to start this application SW.

Two XE167FM EasyKit should be connected according **Figure 1**.

- Connect CAN pins (CANH and CANL) of host board with the CAN pins (CANH and CANL) of slave board (**Figure 5**).
- Need a USB cable and connect it with UART interface of host easy board and PC.

The two boards have to been configured first:

- Set the slave board to active CAN BSL mode:
  - DIP switch S102 in **Figure 5** P10.0-P10.1-P10.2-P10.3-/TRST)=OFF-ON-OFF-xx-ON(1-0-1-x-1)
  - Make a Power-on Reset
- Set the host board to active ASC BSL mode:
  - DIP switch S102 in **Figure 5** P10.0-P10.1-P10.2-P10.3-/TRST)=ON-OFF-OFF-xx-ON(0-1-1-x-1)
  - Make a Power-on Reset

After all of the steps above have been successfully completed, the tool **Minimon** can be started.

- Select 'XC2000' in the controller type under menu 'Target/Configuration'.
- Select the activated virtual COM port and configure '19200baud' under menu 'Settings/Communication'.
- Select ASCII text mode under menu 'Settings/Terminal Settings'.
- Assign the script file 'AppNote16164_CAN_BSL.scm' see **Chapter 8.5**.

Then you click the '**script**' button to start it.

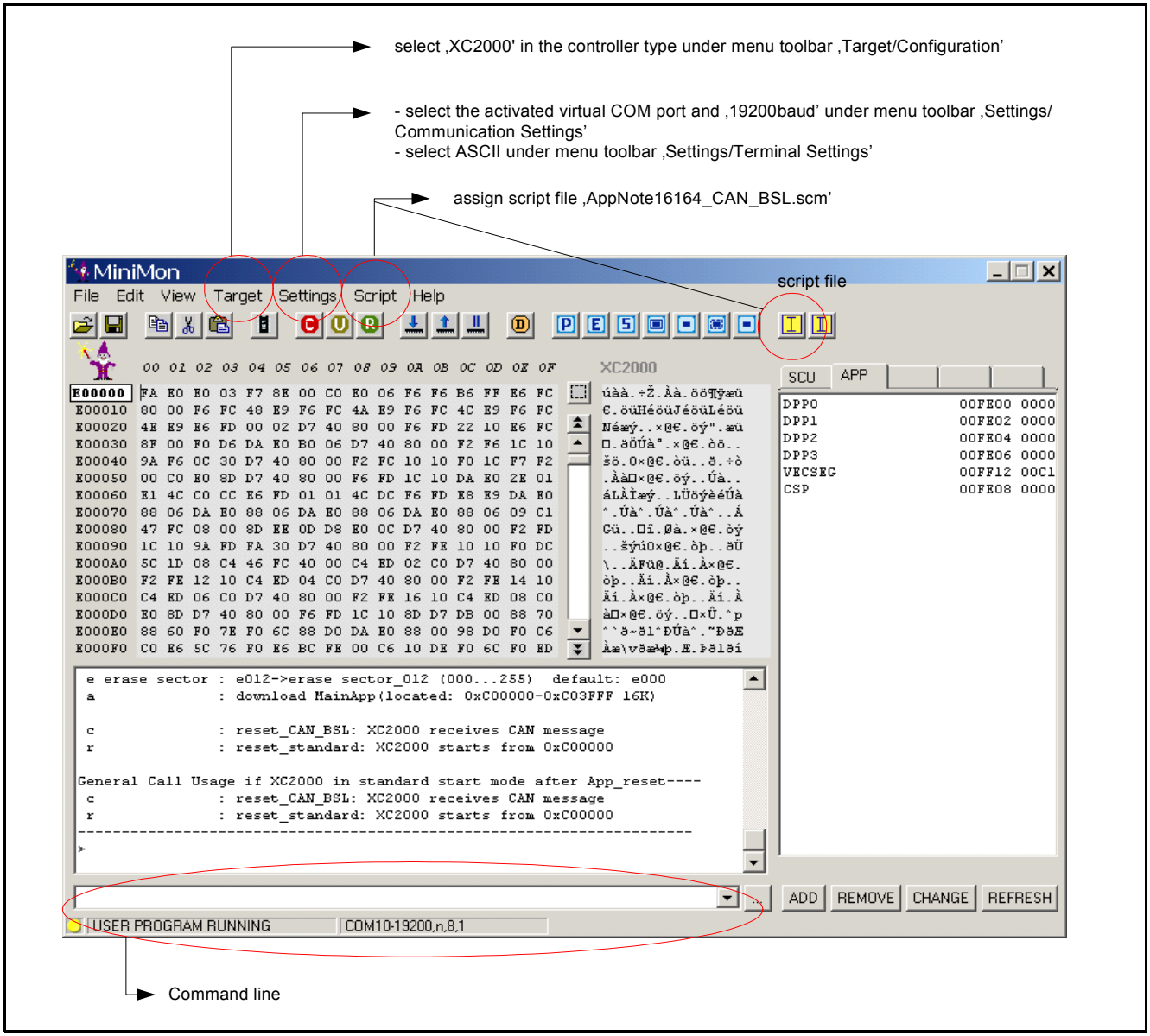

Figure 9 Minimon

8 Appendix

This Appendix is divided into the following sections:

- Hardware Bugs
- Hex Code for the software workaround of bug '**BSL_CAN_X.001**'
- XC2000/XE166 Easy Board V1.1
- XC2000/XE166 Easy Board V2.0 and V3.0

8.1 Hardware Bugs

There are two hardware bugs that the user needs to be aware of: **BSL_CAN_X.001** and **RESET_X.002**.

BSL_CAN_X.001: Quartz Crystal Settling Time after PORST too Long for CAN Bootstrap Loader

The startup configuration of the CAN bootstrap loader when called immediately after $\overline{\text{PORST}}$ limits the settling time of the external oscillation to 0.5 ms. For typical quartz crystal this settling time is too short. The CAN bootstrap loader generates a time-out and goes into Startup Error State.

Workaround:

• For low performance CAN application a ceramic resonator with settling time less then 0.5 ms can be used.

• An alternative is the Internal Start from on-chip flash memory as startup mode after $\overline{\text{PORST}}$. Then switch the system clock to external source and trigger a software reset with CAN bootstrap loader mode selected. Now the device starts with a CAN bootstrap loader without limitation of the oscillator settling time.

Notes

Normally there are two types of external clock circuits used in board design. The effort for the determination of a ceramic resonator oscillator is much more extensive than a quartz crystal oscillator. The following gives the typical start-up time for a ceramic resonator and a quartz crystal:

- The rise time of ceramic resonator: 0.01...0.5 msec
- The rise time of quartz crystal: 1...10 msec

The oscillator rise time (start-up time) is defined from 50% to 90% maximum voltage of the stable oscillation. For example, **Murata** 8MHz **CSTCE8M00G55A** rise time < 0.25 msec.

Before using the built-in CAN BSL, ensure to check the oscillator's rise time in the associated Data Sheet.

RESET_X.002: Startup Mode Selection is not Valid in SCU_STSTAT.HWCFG

Reading from SCU_STSTAT.HWCFG-bits field returns all zeros instead of the information which startup mode has been entered after the last reset.

Workaround

Read the initial value from VECSEG register to evaluate where from the user code is started:

• VECSEG[7:0]=00_H -start from an off-chip memory, external startup mode

• VECSEG[7:0]=$C0_H$ -start from on-chip flash, internal startup mode

• VECSEG[7:0]=$E0_H$ -start from on-chip PSRAM, bootstrap loader mode (UART, CAN or SSC)

Notes:

This is does not affect in **M-, N-, 5-, H-series** devices (see ***Table 1***).

8.2 Hex Code

Software Instructions:

```
SCU_SWRSTCON = 0x0D00;       //SWCFG =0000,1101b (CAN BSL start)
SCU_RSTCON0 = 0xC000;        //.SW=application reset
SCU_SWRSTCON|=0x0003;        //generate reset
```

ASM and Hex file (see attached file 'CAN_BSL_SW.hex'):

```
MOV     R8,#0D00h     ; E6F8000D
MOV     0F0AEh,R8     ; F6F8AEF0
MOV     R9,#0F0AEh    ; E6F9AEF0
MOV     R8,#0C000h    ; E6F800C0
MOV     0F0B8h,R8     ; F6F8B8F0
MOV     R6,[R9]       ; A869
OR      R6,#03h       ; 7863
MOV     [R9],R6       ; B869
```

Note: All SCU registers are protected by the register security mechanism after the EINIT instruction has been executed.

8.3 XC2000/XE166 Easy Board V1.1

The XC2000/XE166 Easy Board, V1.1.

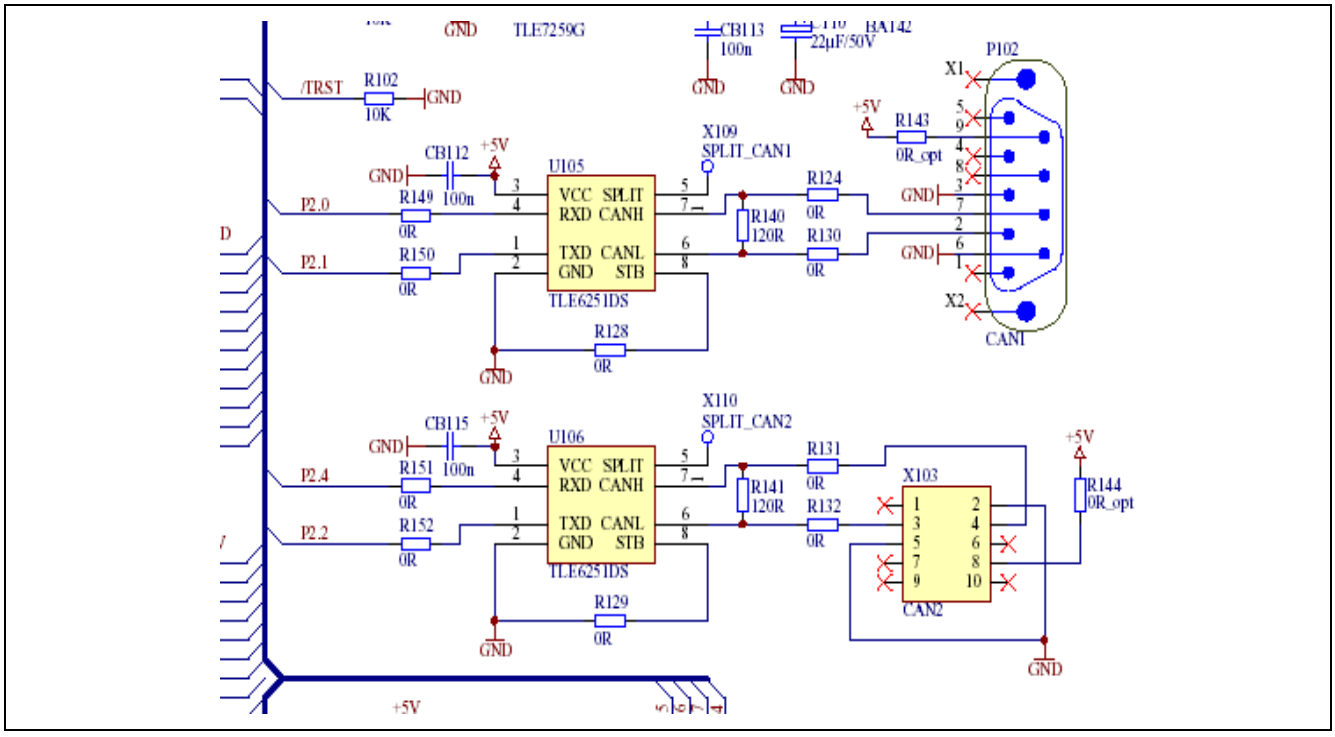

Figure 10 XC2000/XE166 Easy Board V1.1

8.4 XC2000/XE166 Easy Board V2.0 and V3.0

The XC2000/XE166 Easy Board, V2.0 and V3.0.

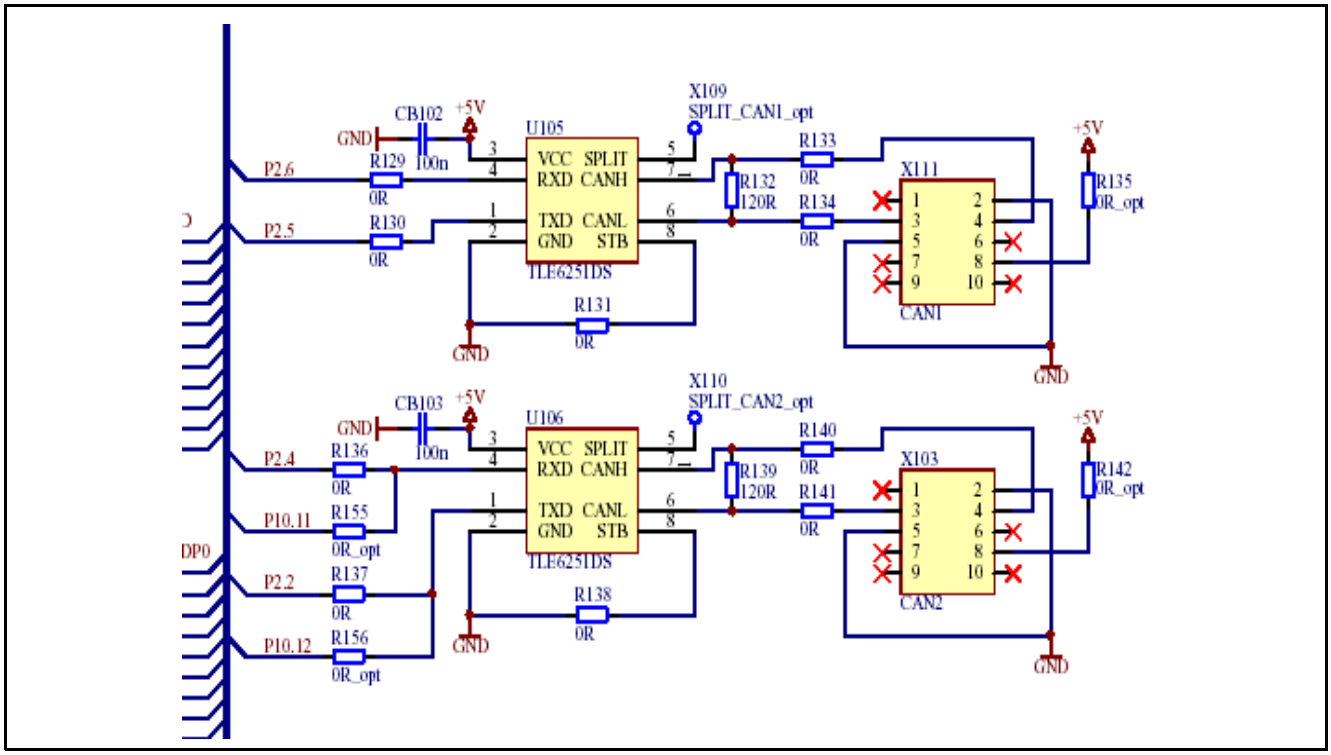

Figure 11 XC2000/XE166 Easy Board V2.0 and V3.0

8.5 Script File: AppNote16164_CAN_BSL.scm

see attathed file 'App16164_CAN_BSL.scm'

```
; Minimon Script file for testing AppNote16164_CAN_BSL SW
_connect                                                        ; connect terminal program

_MOVEMONITOR e07000                                             ; remove monitor PG

_load C:\AppNote16164\CAN_BSLMaster\CAN_BSLMaster.hex           ; load AppSW
_iprogram                                                       ; flash programming

_load C:\AppNote16164\MainApplication_XE16x\MainApplication_XE16x.hex   ; load AppSW
_iprogram                                                       ; flash programming

_load C:\AppNote16164\CAN_Loader\CAN_Loader.hex                 ; load AppSW
_download                                                       ; download in PSRAM

_MOV VECSEG, 00C1                                               ; rewrite register

_jmp C10000                                                     ; jump to C1,0000
```

Note: Before starting the script file, ensure to check the SW in your directory

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG