Developer's Kit XC167CI

for XC167CI Microcontroller, Art-Nr.: 9020

P.O: Box 1103
Kueferstrasse 8
Tel. +49 (7667) 908-0
sales@fsforth.de

- D-79200 Breisach, Germany
- D-79206 Breisach, Germany
- Fax +49 (7667) 908-200
- http://www.fsforth.de

FS FORTH-SYSTEME GmbH
Postfach 1103, D-79200 Breisach a. Rh., Germany

Release of Document: January 29, 2004
Filename: XC167CI_UMa.doc
Author: Joachim Jaeger

Table of Contents

1. Introduction

The product line of FS FORTH-SYSTEME got a brand new member of user-friendly, easy-to-use and powerful evaluation kit which is offered as ”Developer’s Kit XC167CI” (9020). The development kit consists of a base board (223) and a module (349), which is equipped with external SRAM and external FLASH memory.

The development kit is shipped with a XC167CI controller which provides a 128 kByte on-chip FLASH memory with 64bit read accesses, a 2 kByte on-chip Program SRAM memory to store code and data, a 4 kByte on-chip Data SRAM memory to store general data, a 2 kByte on-chip Dual Port RAM memory to store user defined variables, system stack and general purpose register banks, a TwinCAN module according to CAN specification V2.0 part B and with many additional I/O features. The module supports up to 4MByte external Flash memory and up to 1MByte Fast-SRAM which allows to build a large number of sophisticated applications.

The XC167CI is a derivative component of the Infineon XC series with many features. Please refer to the XC167CI datasheet on the CD-ROM shipped with this package. Check the web page of Infineon (***www.infineon.com***) for updated versions of the device datasheet. The Developer’s Kit XC167CI promises a successful start-up development for beginners and experienced users. The kit comes with a Flash programming tool and a well prepared demo software which helps you to become an experienced specialist for XC167CI controller.

The engineers of FS FORTH-SYSTEME created these boards to fulfill a lot of current and future user requirements. Therefore the base board comes with one pre-assembled CAN transceiver. It can easily be used without the need of additional hardware. Furthermore the controller can run up to its maximum CPU-clock of 40 MHz. For fast prototyping the boards include a small wire wrap field, a serial EEPROM, a set of LEDs, 8 DIP switches, and a Reset push button to make development easy and simple. Please refer to the following chapters to get more details about the XC167CI, the board’s capabilities and the tools which are coming with the Developer’s Kit XC167CI.

2. Features of Developer's Kit XC167CI

- High Performance 16-bit CPU with 5-Stage Pipeline
- 1 MByte high speed SRAM
- 2MByte external Flash
- External serial EEPROM (24LC64, I2C-Interface) for user purposes
- External Real Time Clock (72423)
- 5MHz crystal oscillator device
- 16pin header for OCDS via JTAG interface
- One CAN transceiver (PCA82C250T, ±15V common mode offset voltage) on the base board
- One CAN plug (DSUB-9)
- Two RS232 interfaces one routed to a RS232 plug (DSUB-9) and the other one to a 10pin header
- 128 pin connector providing all CPU signals for user purposes
- Additional wire-wrap field for individual hardwired applications
- Power supply 90..264VAC/5VDC enclosed. Input voltage to the Evaluation Board (X4) connector is 5VDC only
- RS232-Interface cable enclosed
- Flash programming utility for easy downloading user specific code to external Flash memory.
- Hex-to-binary converter tool provided on CD-ROM as freeware
- C compiler (demo version only) provided by TASKING and KEIL
- Demo version of the UDE from PLS

3. Getting started

As already mentioned above, this paragraph intents to explain how you can run the board without writing own software. You will learn how the controller's internal bootstrap mode can be used to initiate a simple download of a binary program file to the embedded on-chip and external Flash memories. If you would like to use the environments of KEIL or TASKING you should refer to chapter 10.

First we recommend to create a project path to copy the demonstration software FLASH166, from CD-ROM. This software includes a bootstrap and a monitor program. The bootstrap module is a tiny piece of FS FORTH-SYSTEME firmware which is downloaded to the internal 2 kByte SRAM (IRAM) memory of the controller. It makes sure that other programs, like the monitor can be loaded to any memory location. For more details please refer to the following chapters. By the way: the user may select either, the external FLASH, the external SRAM memory, or the internal Flash memory to download his individual application software.

To setup your board successfully, please take care of the following steps:

- The board is pre-configured by FS FORTH-SYSTEME. See Figure 1 for jumper location. Refer to chapter 6 "System Start-up Configuration" for detailed information about these settings.

- Make sure that the BSL-Mode jumper J2 is closed. This will force the CPU to bootstrap mode right after reset or power on. The Bootstrap mode is always entered by the CPU whenever pin P0L.4 has been grounded prior to a reset.

- Connect the board's serial interface (X2) to COM1 or COM2 of your Personal Computer. Caution: CTS and RTS signals are not supervised by the board's serial interface and the bootstrap loader software.

- Copy the sample project, depending on your board, from the attached CD-ROM into your project path:
  "\Keil\C166\Examples\Boards\FORTH_EVA167_XC167\RTC72423\RTC72423.BIN"

- Copy "FLASH166.EXE" and "FLASH166.OVL" onto your hard disk. Both files allow you to run the bootstrap loader and monitor in an opened DOS box.

- Make sure that the path of the ”FLASH166” directory is included in your path environment, if you choose another directory than for your project before.

- Run the batch file PROG.BAT in a DOS box to program the application into the external Flash memory.

- The tool UDE from PLS is used to program the internal Flash memory, it is also provided on CD-ROM.

- Now remove the BSL-Mode jumper at J2, and press the reset button S2. Run a terminal program and select the line parameters: 9600Baud, No parity, 8 bit data, 1stopbit.

If everything is working properly a small text will appear on your PC screen as shown in Figure 3. Also the LEDs at port 6.6 and port 6.7 will represent a toggling light.

Figure 1: Jumper location at base board

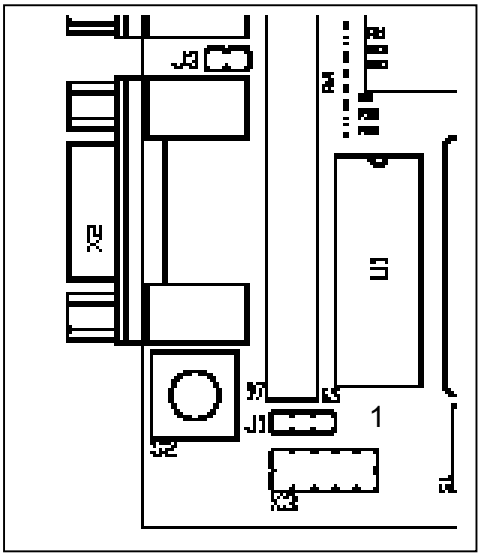
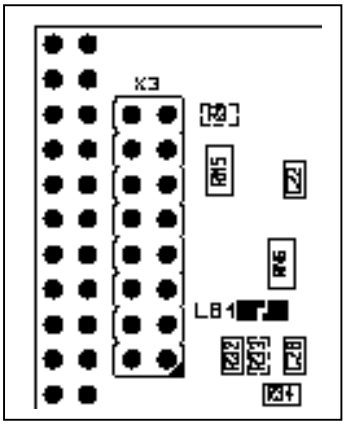
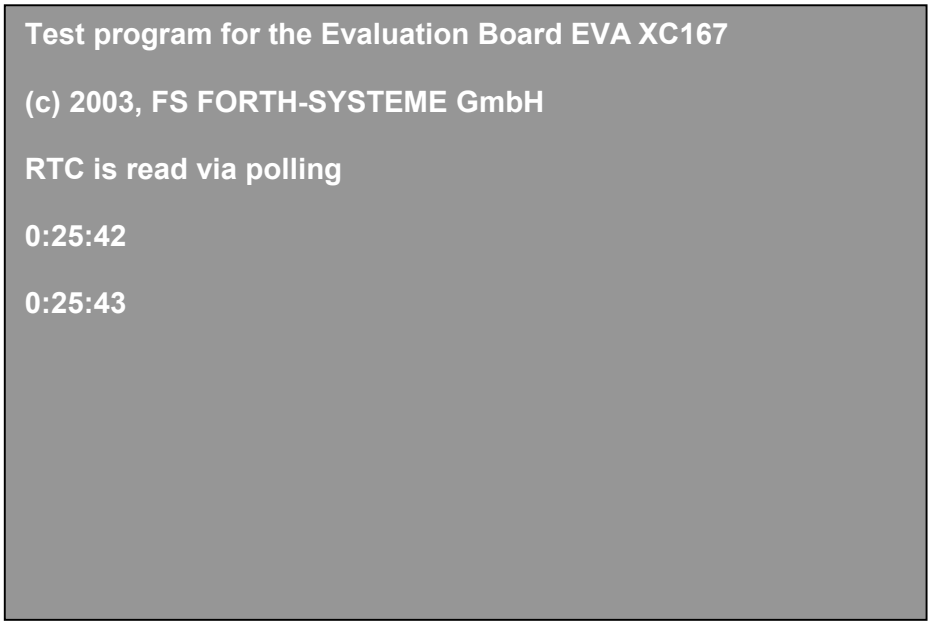


Figure 2: Jumper location at module



Single CHIP mode is disabled at the module by default (LB1 closed).

Figure 3: Output of the example RTC72423



4. Software Tools

The Development's Kit XC167CI always includes the bootstrap loader and monitor program (FLASH166) from FS FORTH-SYSTEME, as well as the UDE from PLS and the compiler from Keil and Tasking. That's everything you need to run your individual software on a XC167CI controller. This software is a part of the Development's Kit XC167CI.

FLASH166 package allows you

- to download your program to an external Flash or SRAM
- to request a memory dump from the target
- to test serial interface, CPU-I/Os and a lot of more things

UDE allows you

- to download your program to the internal Flash
- to debug your application

If you would like to use a sophisticated programming tool which provides you the advantages and performance of "C", we would appreciate to make you an interesting offer at any time. FS FORTH-SYSTEME is distributor for KEIL and TASKING. Both companies are manufacturer of well known "C/C++" compilers and debugging tools. For the very first beginning you may use a free demo of the KEIL and TASKING "C" compilers which allow you to create individual programs with a restricted size of program code.

A detailed and well documented description of writing software with TASKING and KEIL environments is attached to this manual. If you want to run the XC167CI as a standalone controller you may use the UDE from PLS. It allows a fast download of your program into the internal memory section of the CPU.

5. Module Memory Configuration

The XC167CI module supports a number of memory architectures according to the capability of the XC167CI controller.

The XC167CI-module is provided with the following components:

Table 1: Memory on the module

	Memory	Type	Chip-Select	Size
1	FLASH, Boot	AM29F800T-70	CS0#	512k * 16
2	FLASH	AM29F800T-70	CS3#	512k * 16
3	SRAM, High Speed	2 * D434008ALE-15	CS1#	2 * 512k * 8

6. System Start-up Configuration

Although most of the programmable features of the XC167CI are either selected during the initialization phase (software) or repeatedly done during the program execution (software), there are some configurations that MUST be selected earlier, because they are used for the very first access of the CPU (hardware). For example, the CPU has to know how to deal with the external bus interface since the external memory may be organized as an 8 bit multiplexed/non multiplexed or a 16 bit multiplexed/non multiplexed bus. These selections are made during reset at the pins of PORT0, which are read at the end of the internal reset sequence. During reset, CPU-internal pull-up devices at the PORT0 lines are active, so a high level will be read, if the respective pin is left open. To get a low level, the pin must be pulled down by an external resistor. These resistors are already provided on the module.

The following paragraphs explain the settings of these start-up configuration pins. The configuration is selected on the module by resistors connected to GND. If no resistor is fitted, the internal pull-up is active.

6.1. Adapt Mode

Pin P0L.1 selects the ADAPT Mode when low during reset. In this mode, the XC167CI goes into a passive, floating state, which is similar to the state during reset. This mode allows, to switch the XC167CI virtually off, so that an emulator may control the circuitry, since the pins are floating. This mode is used for special emulator purposes only.

Default: Adapt Mode off

6.2. Special Mode

PIN P0L.2 ... P0L.5
With these pins it is possible to select between user code, standard Bootstrap Loader and alternate Bootstrap Loader:

Table 2: Bootstrap Loader Mode Selection

P0L5...P0L.2	Bootstrap Loader Mode
0111	Alternate start
1001	Alternate bootstrap loader mode
1011	Bootstrap loader mode
1100	Use configuration out of on-chip program memory (Flash/ROM)
1111	Standard start (default)

All other combinations are reserved.

Refer to the user manual for more details.

6.3. Bus-Access-Types

With the pins P0L.6 and P0L.7 the bus type can be selected for /CS0.

Table 3: Boot Bus Mode Selection

P0L.7	P0L.6	External Data Bus Width	External Address Bus Mode
0	0	8-bit Data	Demultiplexed Addresses
0	1	8-bit Data	Multiplexed Addresses
1	0	16-bit Data	Demultiplexed Addresses
1	1	16-bit Data	Multiplexed Addresses

In multiplexed bus modes PORT0 drives both, the 16-bit intra-segment address and the 8 (16) bit output data, while PORT1 remains in high impedance state. In demultiplexed bus modes PORT1 drives the 16-bit intra-segment address, while PORT0 (P0H, P0L) drives the 16-bit Data output. If a 8-bit data width has been selected, only P0L is driving the data output.

Default: Data-Bus-Width = 16-bit; Bus Mode = Multiplexed

6.4. Write Configuration

Pin P0H.0 selects the initial operation of the control pins /WR and /BHE. When set high (P0H.0=1), this pin selects the standard mode, i.e. /WR and /BHE. When set low (P0H.0=0) the alternate function is selected: /WR = /WRL and /BHE = /WRH

Default: Pins WR and BHE operate as WRL / WRH

6.5. Chip-Select-Lines

The signals choose the number of chip select lines, which are active after reset. Unused chip select lines may be used as general purpose I/O.

Table 4: Active Chip Select

P0H.2	P0H.1	active chip select	Note
0	0	/CS0.../CS2	
0	1	/CS0.../CS1	
1	0	none	
1	1	/CS0.../CS4	Default

Default: All /CS signals used

6.6. Segment Address Lines

During reset these control lines define the number of active segment address lines. This allows to select pins of port 4 to work as general I/O lines as well as to become a part of the controller address logic. Depending on the selection of SALSELx, the required address space is chosen right after the system start-up and the program may address all locations without prior programming. Even if not all segment address lines are enabled on port 4, the XC167CI uses it's complete 24-bit addressing mechanism. This allows to generate all CS signals over the entire address space even if the external bus width is less in size.

CAUTION!
The selected number of segment address lines ***cannot*** be changed by software after reset.

Table 5: Segment Address Lines

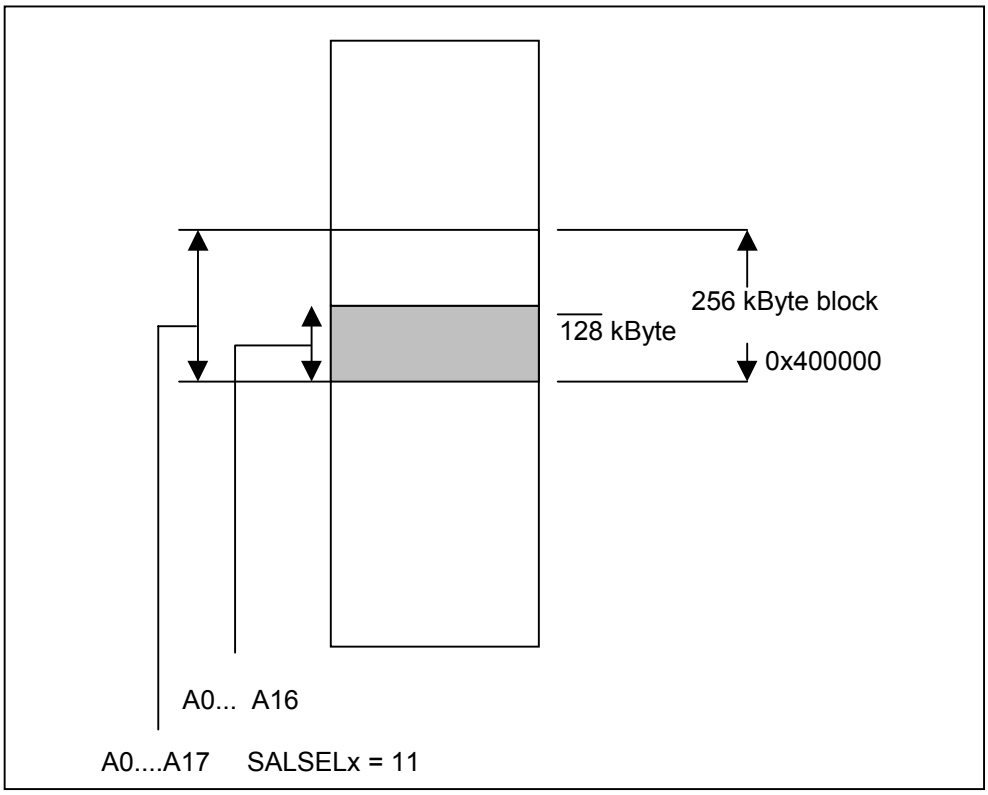
P0H.4	P0H.3	Segment Address Lines	Directly addressable space
0	0	A19.....A16	1 Mbyte; Default
0	1	None	64 kByte
1	0	A23.....A16	16 Mbyte, TwinCAN disabled
1	1	A17.....A16	256 kByte

Default: 1MByte address space

Example:

Assume that your application requires the 128 kByte SRAM block at the absolute address 0x400000. Furthermore the /CS1 signal should control both, reading from and writing to the SRAM. The controller must at least supervise the address lines A0 to A16 in the direct addressing mode.

Figure 4: Example



Define FCONCS1 Register:

ENCS1 = 1, RDYEN1 = 0; RDYMOD1 = 0 BTYP1 = 0x03;

Define TCONCS1 Register:

PHA1 = PHB1 = 0, PHC1 = 1, PHD1 = 0, PHE1 = 2, RDPHF1 = WRPHF1 = 0;

Define ADDRSEL1 Register

RGSZ (Range-Size-Selection) = 0101 (128 kByte block)
RGSAD (Range-Start-Address) = 0100000

6.7. Range Start Address

Depending on the chosen block size (see previous example) the range start address ”RRRRRRRxxxxx” specifies the upper bits (A23....A12) of the respective address area. An ”R” stands for a bit which has to be set within the RGSAD register if this address line is not supervised by the controller bus logic. All bits expressed as an ”x” are exclusively maintained by the bus address signals.

Referring to the example, the register configuration ”RRRRRRRxxxxx” lets A12 to A16 being controlled by the address bus controller. The address lines from A17 to A23 are configured within the RGSAD bits which will be taken by the CPU to compute the internal, absolute 24-bit address.

All memory areas which are not handled by ADRSEL(x) (x = 1...4) are maintained by ADRSEL0.

Now specify ADRSEL1 register:

0x400000 = 0100.0000.0000.0000.0000.0000
*

RGSAD = 0100.0000.0000.0000 **

ADRSEL1 = 0100.0000.0000.0101

* controlled by bus address lines A0..A16
**cursive bits are discarded since they are already controlled by address bus

6.8. XC167CI Memory Layout

The following table shows some address selections which are only suggestions by FS FORTH-SYSTEME. The memory areas may be changed to fulfill different user requirements. Use BUSCONx, ADRSELx registers and the configuration registers as already described in a previous chapter to setup your individual memory layout.

Table 6: Typical Memory Layout

Address Range	Chip-Select	Bus Width	Function
0x3F.FFFF 0x30.0000	/CS4	16 bit MUX	Not connected yet
0x1F.FFFF 0x10.0000	/CS3	16 bit MUX	Flash, up to 1 MByte
0x20.1FFF 0x20.1000	/CS2	8 bit MUX	RTC
0x4F.FFFF 0x40.0000	/CS1	16 bit MUX	SRAM, up to 1 MByte
0x0F.FFFF 0x00.0000	/CS0	16 bit MUX	Boot-Flash, up to 1 MByte

6.9. Clock Generation

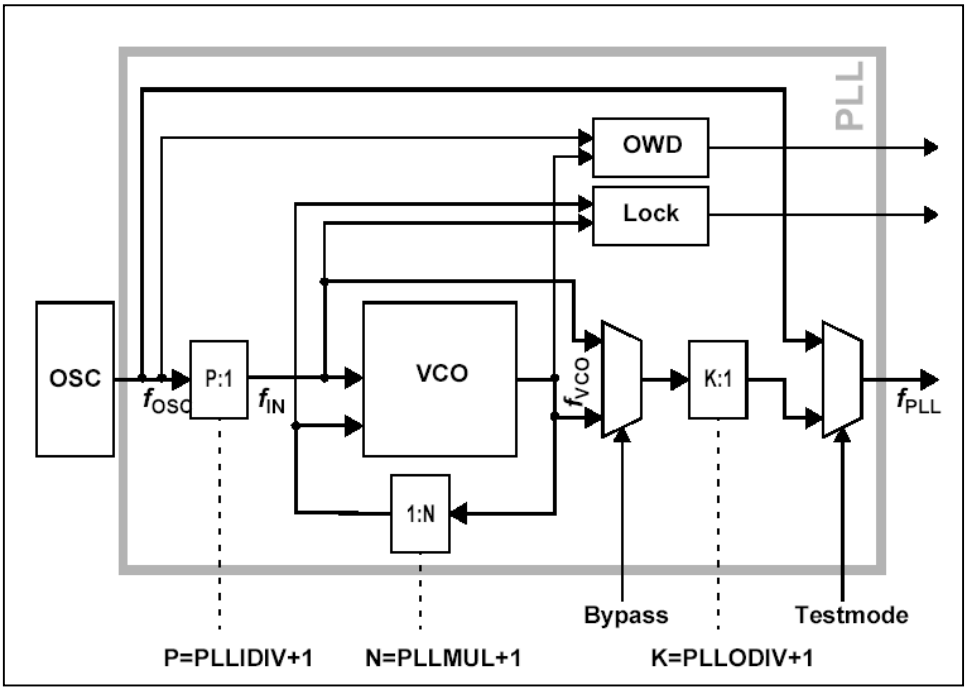
The external clock input range refers to a CPU clock range of 1...40 MHz. Besides, the PLL usage is limited to a 4-50MHz input frequency range. All configurations need a crystal to generate the CPU clock through the internal oscillator amplifier, except the Direct Drive mode (oscillator amplifier disabled, so no crystal or resonator can be used). Vice versa, the clock can be forced through an external clock source only in Direct Drive mode.

Table 7: Selection of CPU clock

P0H.7	P0H.6	P0H.5	CPU-Clock $f_{PLL} = f_{OSC} * F$	External Clock Input f_{OSC} Range in MHz (1)
0	0	0	*f*OSC * 0.5	4 - 50
0	0	1	*f*OSC * 2.5	12 - 16
0	1	0	*f*OSC * 2.5	8 - 12
0	1	1	*f*OSC * 1	4 - 40
1	0	0	*f*OSC * 5	4 - 6
1	0	1	*f*OSC * 2	12.5 -18.7
1	1	0	*f*OSC * 4.5	5.6 - 8.3
1	1	1	*f*OSC * 3	8.3 - 12.5

(1) The Developer's Kit XC161CI comes with an external 5 MHz quartz. The CLKCFG pins are set by default to a factor of 5 (CPU internal = 25 MHz !).

Figure 5: Clock generation



6.10. Single-Chip-Mode

In single chip mode the program is fetched from the on-chip Flash memory. Therefore the embedded Flash memory must contain a valid program to execute correctly. A utility for this can be found on the CD shipped with your Developer's Kit XC167CI.

Single chip mode is entered when pin EA# is set high during reset. The module is configured to boot from external Flash by default. To use single chip mode, the tin-solder has to be removed from the pads of LB1, before restarting the CPU. If the tin-solder is removed, a pull-up resistor configures the CPU to run in internal bus mode. Refer to Figure 2.

7. Module Specification

7.1. Power Supply

The module is powered with +5V only. Internal core voltage of +2.5V is generated with an on board fixed 2.5V LDO MAX8877 (U17). Maximal current of the core at 40MHz is 119mA (15 + 2.6X40), typical value is less than 100mA. Current limit of U17 is 150mA, junction temperature is about 35°C higher than ambient temperature, thermal shutdown is at 150°C junction temperature typical. A working temperature of +85°C of the module will lead to a junction temperature of about 120°C for U17. A copper plane under U17 may reduce this temperature rise.

7.2. Clock

Input clock at XTAL1 is 5MHz, generated by a SMD oszillator G1. This frequency will be transformed to a CPU clock of 40MHz with the internal PLL. The board starts with 25MHz after power up due to configuration settings at D13..15. Software can set the PLL multipliers and dividers to other values (PLLCON register). PLLCON is write protected after execution of EINIT. Successful tested settings are 40MHz, 53.3MHz and 80MHz CPU clock, measured at CLKOUT (P3.15).

7.3. Bus Timing XC167CI

Access times are programmable for 5 phases at the XC167CI: Phase A, Phase B, Phase C, Phase D, Phase E, Phase F.

- Phase A and Phase B are the time until addresses become valid. They need 1..2 cycles minimum with the same chip select; 0..3 cycles are inserted at a chip select change (address window change).
- Phase C is a command delay phase 0..3 cycles
- Phase D is the write data setup or MUX tristate phase with 0..1 cycles
- Phase E is the access time with 1..32 cycles
- Phase F is the address/data hold time phase with 0..3 cycles

The cycle time is 25ns @40MHz

XC167CI has TOH = 14ns min for output delay addresses (muxed and non muxed)

toutd = 6ns for output valid delay

tsetup = 24ns for data input setup time

Figure 6: Multiplexed Bus Read

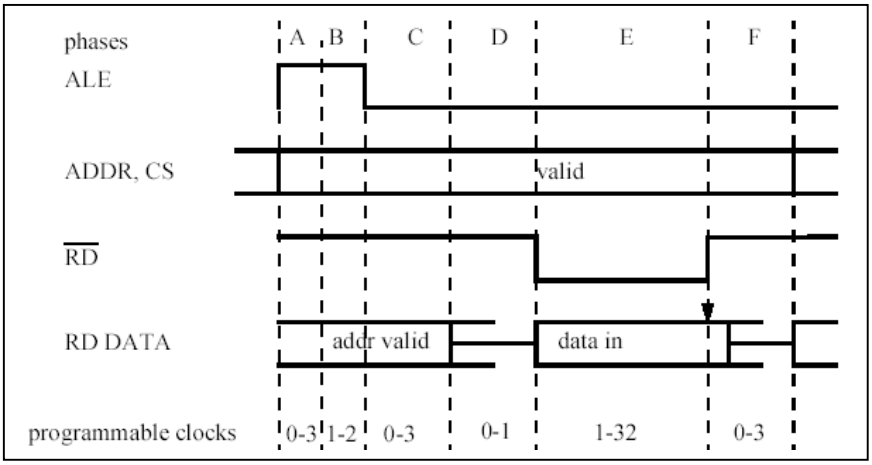
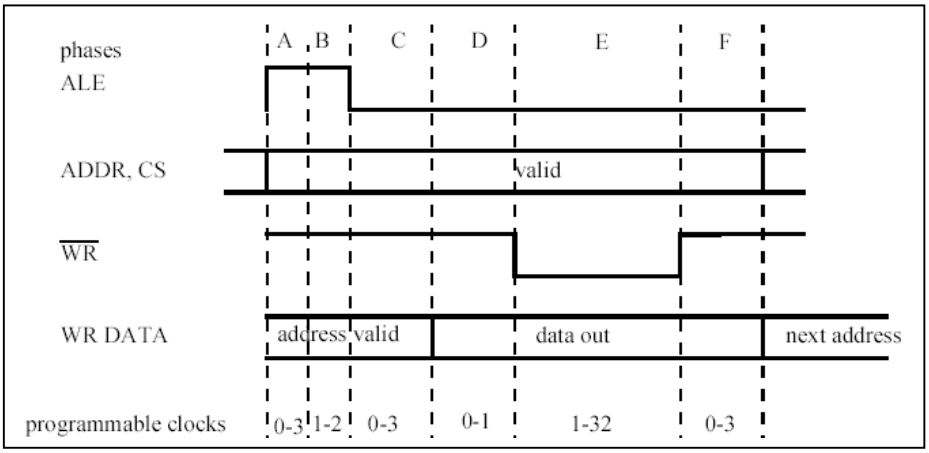


Figure 7: Multiplexed Bus Write



7.4. Flash Banks

Depending on the state of EA# at power up, the module will either start from the internal flash or the external flash connected to CS0#. This external flash U8 can have up to 2Mbytes (1MX16). 2Mbytes will need P4.4 as A20 disabling the TwinCAN unit, connected to P4.4..P4.7.

A 2nd external flash bank is implemented with U12 controlled by CS3#. Maximum size is 2Mbyte too with the restrictions mentioned above.

A 70ns device example AMD 29F800/AM29F160 has tRC=70ns, tACE=70ns, tDOE=30ns, tDF=20ns

fastest read cycle is toutd + tDOE + tsetup = 6 + 30 + 24 = 60ns

setting with Phase A = 0, Phase B = 1, Phase C = 1, Phase D = 0, Phase E = 2, Phase F = 0 is safe (4 cycles) for non multiplexed bus.

setting with Phase A = 0, Phase B = 1, Phase C = 1, Phase D = 0, Phase E = 2, Phase F = 0 is safe (4 cycles) for multiplexed bus.

7.5. SRAM

Up to 1Mbyte SRAM can be populated on the module. Either fast or slow SRAMs can be used (15..20ns devices U6,7 or 70nsec devices U20,21). CS1# is used to access the SRAM.

Calculating access times for 15ns example NEC µPD434008ALE: tRC=15ns, tACE=15ns, tDOE=7ns, tDF=7ns

Fastest read/write cycle is toutd + tDOE + tsetup = 6 + 7 + 24 = 37ns

setting with Phase A = 0, Phase B = 1, Phase C = 1, Phase D = 0, Phase E = 3, Phase F = 0 is ok (5 cycles) for multiplexed bus.

Calculating access time for 55ns example Samsung K6T1008C2E: tRC=55ns, tACE=55ns, tDOE=25ns, tDF=20ns

Fastest read/write cycle is toutd + tDOE + tsetup = 6 + 25 + 24 = 55ns

setting with Phase A = 0, Phase B = 1, Phase C = 1, Phase D = 0, Phase E = 2, Phase F = 0 is safe (4 cycles)

Slow SRAM can be used as battery buffered nonvolatile RAM. Battery buffering is not recommended for fast devices (standby current >10mA per device and no data retention at voltages < 4.5V guaranteed; U6,7 connected to +5V).

7.6. RTC

An external RTC (U16) is populated on the module. It is accessed by CS2#. The internal RTC of the CPU can be used, if no battery buffering of time is necessary. Internal RTC needs 5V and approximately 120..150µA (CPU in sleep mode).

The values of the RTC can be read either by polling or by interrupt requests. At the bottom side of the module there are the pads of LB2. If the LB2 is open (default) the values of the RTC have to be polled. When closing the pads with tin-solder, the interrupt output of the RTC is connected to port pin 2.8 of the XC167CI.

7.7. Battery Buffering Control, RESET# Generation

Battery supervisor BQ2201 (U14) switches from +5V power to battery at power fail. It provides battery buffered voltage for slow SRAMs (U20,21), if populated, and for RTC U16.

RESET generation is done by a supervisor MAX705 (U15). U15 supervises +5V, and +2.5V with its power fail path, and generates the system reset. The signal is connected to RSTIN# of the CPU. External signal RESET# (Reset button on base board) is connected to the input MR# of U15.

7.8. 2nd UART

XC167CI has a 2nd serial UART channel on chip (P3.0/TXD1 and P3.1/RXD1, P3.10/TXD0 and P3.11/RXD0). No hand shake signals are provided for both channels, but two handshake lines of the second RS232 interface (X3) are connected to the module header at the base board. These pins are routed at the module to port P4.0 (A16) and port P4.1 (A17) via a buffer with a fix direction. Therefore the handshake signals should not be used at X3 connector.

7.9. TwinCAN Unit

Two CAN nodes are provided by the XC167CI at P4.4..P4.7.

Node A is connected optional either to a driver U18 on board generating CANH at pin 83 and CANL at pin 115 or the direct port signals P4.6/CAN1_TXD to pin 83 and P4.5/CAN1_RXD to pin 115 at the base board connector.

Node B is connected with P4.4/CAN2_RXD at pin 84 and P4.7/CAN2_TXD at pin 114 at the base board connector.

Both nodes can be mapped by software to other port pins of the CPU:

- CAN1_TXD: P7.7 (pin 22) or P9.3 (pin 16)
- CAN1_RXD: P7.6 (pin 54) or P9.2 (pin 48)
- CAN2_TXD: P7.5 (pin 21) or P9.1 (pin 15)
- CAN2_RXD: P7.4 (pin 53) or P9.0 (pin 47)

7.10. External Bus

External signals ED0..7, EA0..18 (16..18 optional), RD#, WR#, CS4# and ERSTOUT# are buffered with U1..4. It allows external connection of 8-bit devices. Data direction is switched with CS4# and RD#.

7.11. Serial EEPROM

A serial non volatile EEPROM (U9) with IIC interface is populated on the module. Range is 128bytes..8Kbytes. Control is done with P2.10 (SCLK, X2 D31) and P2.11 (SDAT, X2 C30). Device address is 0xA0/1. Connection of external IIC devices with addresses <> 0xA0..1 is possible. Only 5V devices are allowed without additional circuitry.

8. Base Board Specification

8.1. Power Supply

The base board expects a stabilized 5 VDC (±0.25V) voltage at the X4 power connector (see Figure 8). An external power mains unit is also part of the Developer's Kit XC167CI.

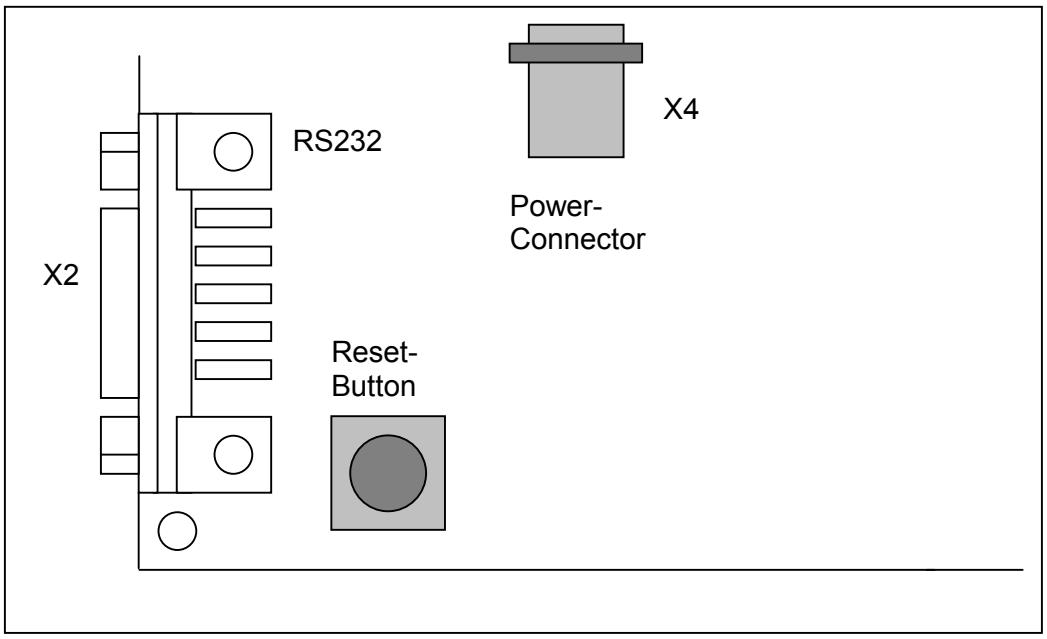


Figure 8: Power Supply

CAUTION:
DON'T USE ANY OTHER POWER SUPPLY THAN THAT ONE COMING WITH THE DEVELOPER'S KIT XC167CI. OTHERWISE YOU TOLERATE THE DAMAGE OF THE ENTIRE BORD !

IN THIS CASE WE CANNOT GRANT ANY WARRANTY!

8.2. Switch S1

The signals have pull-up resistors provided. If the switch is closed, the signal is tied to GND.

Table 8: User Switch S1

PIN	Port Pin at XC167CI
1	P9.0
2	P9.1
3	P9.2
4	P9.3
5	P9.4
6	P9.5
7	P1L.7
8	P1H.0

8.3. LED V2

All user LEDs are connected to a high level (+5V) through series resistors (2K2). A low level at the corresponding port pin of the CPU switches these LEDs **ON.**

Table 9: User LED V2

LED	Port Pin at XC167CI
1	P1L.0
2	P1L.2
3	P1L.4
4	P1H.4
5	P1H.5
6	P6.5
7	P6.6
8	P6.7
9	ERSTOUT#
10	GND

8.4. Push button S2

Push button S2 is the Reset key, which is connected to the Reset Generator device on the module.

8.5. Jumper J1

With this jumper setting it can be choosen, if the RTC, or the slow SRAMs on the module are supplied by the battery.

PIN	A	B	Description	Pre-configured
1	▌		Pin BATT of X7 connected to battery	Yes
2	▌	▌		
3		▌	Pin BATT of X7 connected to GND	No

Table 10: Jumper settings for J1 ▌ = *connected pins*

8.6. Jumper J2

This jumper enables the Bootstrap-Mode.

PIN	A	B	Description	Pre-configured
1	▌		A: Bootstrap Mode enabled	Yes
2	▌		B: Bootstrap Mode disabled	No

Table 11: Jumper settings for J2 ▌ = *connected pins*

8.7. Jumper J3

This jumper switches on the CAN bus termination of 120R.

PIN	A	B	Description	Pre-configured
1	▌		A: CAN bus termination on	Yes
2	▌		B: CAN bus termination off	No

Table 12: Jumper settings for J3 ▌ = *connected pins*

A CAN bus system has to be terminated at both ends of the network with two resistors. The resistors should have a value of nominal 124 Ω (typically 120 Ω). The terminating resistor can be activated through jumper J3.

Caution: Never use more than two termination resistors for a network. This might result into any unexpected bus transmission errors or may force the transceiver circuits to drop down their operation.

The connector X1 provides one D-Sub male connector for the CAN-interface.

8.8. RS232 Interface

The RS232 is interfaced to the board through D-Sub female connector X2. It allows the download of the monitor and bootstrap loader as well as the user application file. If the monitor is not in use, this interface can be taken as a standard RS232 for individual purposes. The RS232 hardware does not support any handshake signals, like RTS, CTS or DTR.

A second RS232-interface is provided at the 10pin header X3. This interface also doesn’t support handshake signals.

9. Connectors

9.1. Module connector X1

Pin	Row A	Description	Row B	Description
1	P1L.1/A1/COUT60	Direct port pin, 1.)	P1L.0/A0/CC60	Direct port pin, 1.)
2	P1L.3/A3/COUT61	Direct port pin, 1.)	P1L.2/A2/CC61	Direct port pin, 1.)
3	P1L.5/A5/COUT62	Direct port pin, 1.)	P1L.4/A4/CC62	Direct port pin, 1.)
4	P1L.7/A7/CTRAP#/CC22IO	Direct port pin, 1.)	P1L.6/A6/COUT63	Direct port pin, 1.)
5	P1H.7/A15/CC27IO	Direct port pin, 1.)	NMI#	Direct pin, Pull-Up
6	P1H.5/A13/CC25IO	Direct port pin, 1.)	P1H.6/A14/CC26IO	Direct port pin, 1.)
7	P1H.3/A11/SCLK1/EX0IN_A	Direct port pin, 1.)	P1H.4/A12/CC24IO	Direct port pin, 1.)
8	P1H.1/A9/CC6POS1#/MRST1	Direct port pin, 1.)	P1H.2/A10/CC6POS2#/MTSR1	Direct port pin, 1.)
9	CC60 (P1L.0)	Direct Port Pin, 1.)	P1H.0/A8/CC6POS0#/EX0IN_B/CC23IO	Direct port pin, 1.)
10	CC25IO (P1H.5)	Direct Port Pin, 1.)	CC62 (P1L.4)	Direct Port Pin, 1.)
11	CC7IO (P6.7)	Direct Port Pin	CC6IO (P6.6)	Direct Port Pin
12	CC24IO (P1H.4)	Direct Port Pin, 1.)	CC5IO (P6.5)	Direct Port Pin
13	P6.5	Direct Port Pin	CC61 (P1L.2)	Direct Port Pin, 1.)
14	P6.7	Direct Port Pin	P6.6	Direct Port Pin
15	P9.1	Direct Port Pin	P9.0	Direct Port Pin
16	P9.3	Direct Port Pin	P9.2	Direct Port Pin
17	P9.5	Direct Port Pin	P9.4	Direct Port Pin
18	CC23IO (P1H.0)	Direct Port Pin, 1.)	CC22IO (P1L.7)	Direct Port Pin, 1.)
19	COUT61 (P1L.3)	Direct Port Pin, 1.)	COUT60 (P1L.1)	Direct Port Pin, 1.)
20	COUT63 (P1L.6)	Direct Port Pin, 1.)	COUT62 (P1L.5)	Direct Port Pin, 1.)
21	P7.5	Direct Port Pin	P7.4	Direct Port Pin
22	P7.7	Direct Port Pin	P7.6	Direct Port Pin
23	P5.1	Analog Input	P5.0	Analog Input
24	P5.3	Analog Input	P5.2	Analog Input
25	P5.5	Analog Input	P5.4	Analog Input
26	P5.7	Analog Input	P5.6	Analog Input
27	P5.9	Analog Input	P5.8	Analog Input
28	P5.14	Analog Input	P5.15	Analog Input
29	P5.12	Analog Input	P5.13	Analog Input
30	P5.10	Analog Input	P5.11	Analog Input
31	VAREF	Analog Ref. Voltage	VAGND	Analog Ground
32	GND	-	GND	-

1.) Usable as port lines only when multiplexed address/data, otherwise address

9.2. Module connector X2

Pin	Row C	Description	Row D	Description
1	VCC	+5V	VCC	+5V
2	RESET#	Reset Input	BATT	Standby Power SRAM/RTC
3	EA13	Demuxed Addr. buffered	EA15	Demuxed Addr. buffered
4	EA8	Demuxed Addr. buffered	EA14	Demuxed Addr. buffered
5	EA7	Demuxed Addr. buffered	EA9	Demuxed Addr. buffered
6	EA6	Demuxed Addr. buffered	EA10	Demuxed Addr. buffered
7	ERSTOUT#	Buffered RSTOUT#	ECS4#	External Chip Select
8	EA5	Demuxed Addr. buffered	EA12	Demuxed Addr. buffered
9	EA4	Demuxed Addr. buffered	ERD#	Buffered RD#
10	EA3	Demuxed Addr. buffered	EA11	Demuxed Addr. buffered
11	EA2	Demuxed Addr. buffered	EA1	Demuxed Addr. buffered
12	EALE	Buffered ALE	EWR#/WRL#	Buffered P3.13/WR#
13	EA0	Demuxed Addr. buffered	ED7	Buffered Data
14	ED0	Buffered Data	ED6	Buffered Data
15	ED1	Buffered Data	ED5	Buffered Data
16	ED2	Buffered Data	ED4	Buffered Data
17	BOOTSTR#	Bootstrap Enable Input	ED3	Buffered Data
18	TXD1/EA18	optional	P4.7	Direct Port Pin
19	CANH/CAN_TXD	optional	CANL/CAN_RXD	optional
20	P4.4/A20/CAN2RXD/EX5IN_B	optional	RXD1/EA16	optional
21	n.c./EA17	optional	P3.15	Direct Port Pin
22	P3.13/SCLK	Direct Port Pin	P3.11/RXD0	Direct Port Pin
23	P3.10/TXD0	Direct Port Pin	P3.9/MTSR	Direct Port Pin
24	P3.8/MRST	Direct Port Pin	P3.7	Direct Port Pin
25	P3.6	Direct Port Pin	P2.13	Direct Port Pin
26	P2.14	Direct Port Pin	P2.15	Direct Port Pin
27	P3.0	Direct Port Pin	P3.1	Direct Port Pin
28	P3.2	Direct Port Pin	P3.3	Direct Port Pin
29	P3.4	Direct Port Pin	P3.5	Direct Port Pin
30	P2.11	Direct Port Pin	-	n.c.
31	P2.8	Direct Port Pin	P2.10	Direct Port Pin
32	P2.12	Direct Port Pin	P2.9	Direct Port Pin

9.3. JTAG Connector X3

X3: JTAG/Emulator Connector on the module, 2 X 8 pin RM2.54

PIN	Signal	Description
1	TMS	JTAG
2	+5V	Power
3	TDO	JTAG
4	GND	
5	CPU_CLK	Connected to GND
6	GND	
7	TDI	JTAG
8	RSTIN#	System RESET
9	TRST#	JTAG
10	BRKOUT#	Emulator
11	TCK	JTAG
12	GND	
13	BRKIN#	Emulator
14	NC	reserved for OCDS_E#
15	NC	RCAP1, free for user
16	NC	RCAP2, free for user

9.4. X1, CAN Connector

Sub-D 9 pin male connector on base board

PIN	Signal
1	NC
2	CAN_L, differential Low signal
3	GND, optional
4	NC
5	NC
6	GND, optional
7	CAN_H, differential High signal
8	NC
9	NC

Pins 2 and 7 may be connected via a terminating resistor (120 Ω) via jumper J3.

9.5. X2, RS232 Interface

Sub-D 9 pin female connector on base board

PIN	Signal
1	NC
2	TXD0
3	RXD0
4	NC
5	GND
6	DTR0, static 10V, no handshake
7	NC
8	NC
9	NC

9.6. X3, RS232 Interface

Header connector 10 pin, on base board

PIN	Signal
1	NC
2	DTR1, static 10V, no handshake
3	TXD1
4	CTS1, don’t use this signal
5	RXD1
6	RTS1, don’t use this signal
7	NC
8	NC
9	GND
10	NC

10. CD-ROM

Within the Developer's Kit XC167CI you will find everything you need for a fast start-up development. Two C compilers (demo versions) coming from KEIL and TASKING allow you to write small sample programs or to run the examples provided on the CD-ROM. The use of the compiler is reduced in memory size. You will find datasheets of the devices on the CD-ROM as well as a utility to program and erase the embedded Flash memory.

10.1. Content of CD-ROM

On your CD, you can find four main directories. The content is described below. Please refer to the TXT files in the sub-folders for detailed information.

DC166	Tasking demo, board configuration for the monitor and examples.
Keil	Keil demo, monitor, examples.
Manual	Manual of the Developer's Kit XC167CI and datasheets of the XC167CI.
Tools	This directory contains tools for programming the external/internal Flash of the XC167CI and the UDE demo debugger from PLS.

10.2. DEMO for KEIL

The example RTC72423 can be built, using a KEIL C compiler (full or demo version). Within the workspace you can choose between a Debug-version, a version running from the internal Flash and a version, running from the external Flash as well as an OCDS version.

Debug:	Shows how to make a project for a debug session.
On-Chip Flash:	Shows how to make a project for the internal Flash-memory of the XC167CI.
External Flash:	Shows how to make a project for the external Flash-memory.
OCDS:	Shows how to debug with OCDS.

10.3. DEMO for TASKING

The example RTC72423 can be built, using a TASKING C compiler (full or demo version). Within the workspace you can choose between a Debug-version, a version running from the internal Flash and a version, running from the external Flash as well as an OCDS version.

Debug:	Shows how to make a project for a debug session.
On-Chip_Flash:	Shows how to make a project for the internal Flash-memory of the XC167CI.
ExtFlash:	Shows how to make a project for the external Flash-memory.
OCDS:	Shows how to debug with OCDS.

10.4. TOOLS\FLASH166

In this folder you will find the FLASH166 tool which has already been mentioned in this manual. Furthermore this directory keeps the technical documentation of the Flash tool, where you can find all needed information for debugging and downloading binaries into the Flash circuit. FLASH166 supports a lot of different Flash memory types.

Each sample folder for the release version includes a certain batch execution file (prog.bat) which makes it easy for you to create a binary file. FCONV is used to generate a binary output file, based on a hex file typically generated by all C-compilers. Since FLASH166 expects a binary formatted file you should use FCONV and FLASH166 to program the XC167CI controller. You must not pay attention to these subjects if you are using any other programming tool.

We are permanently upgrading the FLASH166 tool. So we actually support a large number of different standard FLASH memories. If you are planning to use a Flash type which is not supported by FLASH166 yet, you should not hesitate to contact us.

Furthermore, if you are planning to use FLASh166 for your own application later on, please contact us by phone (+49 7667 908 122), or email (sales@fsforth.de) We will make you an interesting offer for the tool and additional licences.

10.5. Board configuration

For each demonstration program refer to the associated text file, like readme.txt or abstract.txt. Herein you will find all technical details about the board configuration, CPU speed and other parameters. Also have a look at the PDF file, provided by Infineon, which represents the datasheet of the XC167CI.

10.6. Error conditions FLASH166

Batch file does not run:	Make sure that you have adapted the batch file according to your individual path structure. You may be forced to edit the batch file slightly.
Wrong Flash number:	Please note that FLASH166 expects a Flash device at the address 0x00.0000 (/CS0). If you have changed the chip selection lines this error message occurs. The switch ”/CSx” changes the chip select line.
Bootstrap mode timeout.	Make sure that you have connected a serial (RS232) cable to COM1 (COM2) and X2 of your board. Then press the RESET push button again. Check if the BSL jumper J2 is in bootstrap mode. If you are using a serial interface different than COM1 you must revise the ”prog.bat” file according to: ”flash166 /P <file.bin> /COM2”.

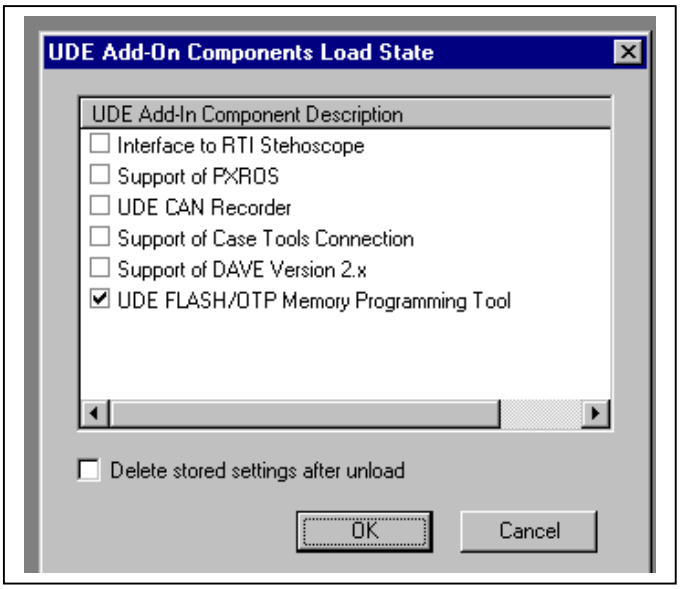
10.7. PLS UDE

This Debugger allows also programming the internal Flash memory of the XC167CI controller. The tool is provided within the tool directory on CD-ROM.

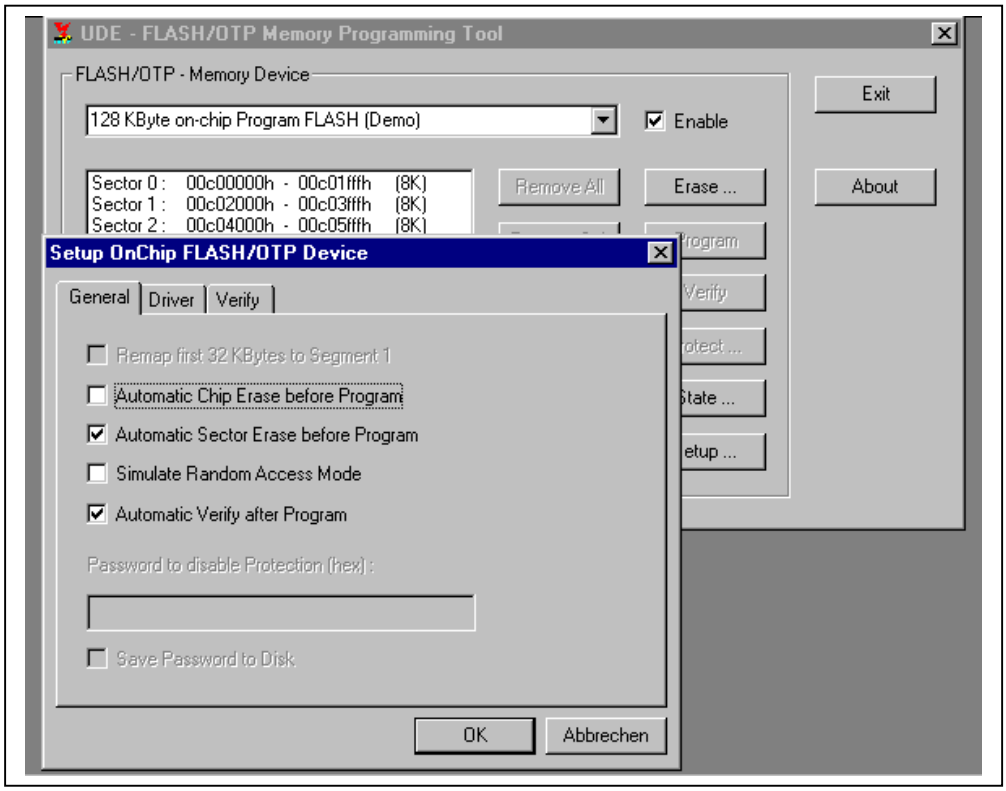
Run “SETUP.EXE” to install the tool.

Before running UDE, establish the connection between your PC and the Base board, by using the enclosed OCDS adapter and switch on the power supply.

Now you can run UDE, open the corresponding workspace of your board and open the Add-In Components menu from the Config-menu, where you have to select UDE FLASH/OTP Memory Programming Tool.



Save the workspace and open the Flash Programming tool, to enable the internal Flash support. Setup the programming tool:



Then you can select the Hex-file to download into the internal Flash via the “Load” button:

“\dc166\Examples\FORTH_EVA167_XC167\RTC72423\on-chip_flash.hex”

After the programming was successful, switch off the power supply and remove the OCDS adapter. Then switch on the power supply.

Attention: Before the CPU can be used in Single Chip mode, the tin-solder at LB1 on the module has to be removed.

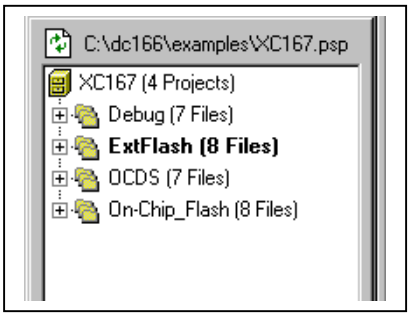
11. Building the sample project with the TASKING C-Compiler

First of all you have to install the Tasking C166 Compiler:
Copy the file DC166-801.zip to a free directory on your harddisk and unzip this file. Then run the setup.exe to install the Tasking Compiler. We recommend to use the default directories and to copy the files from the \TOOLS\FLASH166-directory of the CD-ROM into a directory on your Hard disk, which is known by your system.

Now copy the files from the structure \DC166\... from the CD-ROM to the corresponding directories on your harddisk, remove the “Read-only”-attribute from all of these files.

Then run the Tasking Compiler and select from the menu:
Project -> Project Space -> Open
From the next menu select the file “XC167.PSP” from the directory \DC166\EXAMPLES\.

Now you will see the project space from the XC167-sample project:



Go to the menu Build and select the item “Scan all dependencies”

If you have the license for the full version of EDE, you can remove the macro _TASKING_DEMO from the preprocessing project options of the C-compiler options.

11.1. Project for the debugger

To build the project for the debugger, choose the project “Debug” as current project and “Rebuild” the project. The results can be found in the project-directory.

Connect the delivered cable for the RS232-interface between your PC and the connector X2 on the Base board. Plug on jumper JP2 and switch on the power supply of the board.

Now you can start the CrossView debugger by choosing the item “Debug” at the menu “Project”. After the correct download of the project you will see the environment of the debugger. Here you can start the application by choosing the item “Run” from the menu “Run”.

Now you will see the output of the target in the Terminal window.

If you will get an error message from the CrossView debugger, belonging to a directory, you have to quit the debugger and enter the correct drive letter into the appearing window:

11.2. Project for the external FLASH

To build the project, running from the external FLASH, select the project “ExtFlash” as current project and “Rebuild” the project. The results can be found in the project-directory.

Connect the delivered cable for the RS232-interface between your PC and the connector X2 on the Base board. Plug on jumper JP2 and switch on the power supply of the board.

Start the batch-file “PROG.BAT”, which converts the hex-file of the project into a binary file, which will be downloaded into the external FLASH, by the FLASH166 program.

After the correct download, you have to remove jumper JP2, press the Reset push button S2 and start a terminal program on your PC with the line parameters 9600Baud, no parity, 8 bit data and 1 stop bit.

At the terminal program you will receive the output from the Base board.

11.3. Project for OCDS

To debug the application via OCDS perform the following steps:

- Build and download the project for the external Flash, as described above
- Select the project “OCDS” as current project and “Rebuild” the project
- Switch off the power supply
- Connect the OCDS adapter to JTAG connector X3 on the module
- Connect the OCDS adapter to LPT1 of your PC
- Remove jumper JP2
- Switch on the power supply
- Enter the Crossview Debugger

11.4. Project for the internal Flash

To build the project, running from the internal Flash, select the project “On-Chip_Flash” as current project and “Rebuild” the project.

Attention: Before the CPU can be used in Single Chip mode, the tin-solder at LB1 on the module has to be removed.

Use UDE from PLS to download your application into the internal Flash of XC167CI.

12. Building the sample project with the KEIL C-Compiler

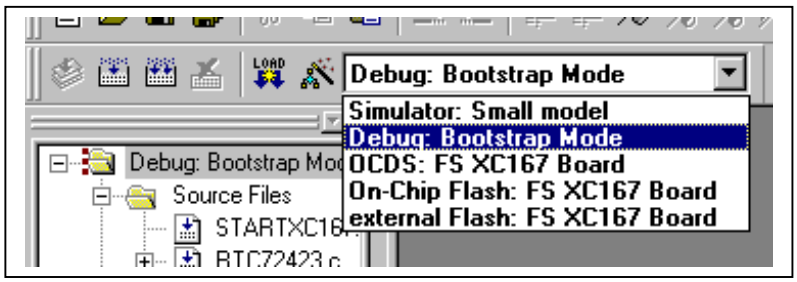
First of all you have to install the Keil C166 Compiler:
Copy the file ek166v427.exe to a free directory on your harddisk and run it to install the Keil Compiler. We recommend to use the default directories and to copy the files from the \TOOLS\FLASH166-directory of the CD-ROM into a directory on your Hard disk, which is known by your system.

Now copy the files from the structure \Keil\... from the CD-ROM to the corresponding directories on your harddisk, remove the “Read-only”-attribute from all of these files.

Before starting to work with the debugger, you have to choose the correct monitor for the debug session:

Open the project “RTC72423.UV2” from the directory:
“\Keil\C166\EXAMPLES\BOARDS\FORTH_EVA167_XC167\RTC72423\”

Select Debug as target.



12.1. Project for the debugger

To build the project for the debugger, select “Debug” as target and “Rebuild” the project. The results can be found in the directories:

\RTC72423\Debug\List and \ RTC72423\Debug\Obj

Connect the delivered cable for the RS232-interface between your PC and the connector X2 on the Base board. Plug on jumper JP2 and switch on the power supply of the board.

To start the debugger, Select Debug -> Start/Stop Debug Session. After the correct download of the project you will see the environment of the debugger. Here you can start the application by choosing the item Go from the menu Debug.

After you have started the application, you will see the output from the target in the serial #1-window.

12.2. Project for the external FLASH

To build the project, running from the external FLASH, select “External Flash” as target and “Rebuild” the project. The results can be found in the directories:

\ RTC72423\Release\List and \ RTC72423\Release\Obj

Connect the delivered cable for the RS232-interface between your PC and the connector X2 on the Base board. Plug on jumper JP2 and switch on the power supply of the board.

To make the project ready to download you can either open a DOS-box to start the batch-file “PROG.BAT” in your project-directory, or you can enter the call of the batch-file “PROG.BAT” at “Customize Tools Menu” from the menu Tools:

Don’t forget to select the option “Run independant” !

The batch-file “PROG.BAT” converts the hex-file of the project into a binary file, which will be downloaded into the external FLASH, by the FLASH166 program.

After the correct download, you have to remove the jumper JP2, press the Reset push button S2 and start a terminal program on your PC with the line parameters 9600Baud, no parity, 8 bit data and 1 stop bit.

At the terminal program you will receive the output from the Base board.

12.3. Project for OCDS

To debug the application via OCDS perform the following steps:

- Build and download the project for the external Flash, as described above
- Select the project “OCDS” as current project and “Rebuild” the project
- Switch off the power supply
- Connect the OCDS adapter to JTAG connector X3 on the module
- Connect the OCDS adapter to LPT1 of your PC
- Remove jumper JP2
- Switch on the power supply
- Enter the Debugger

12.4. Project for the internal Flash

To build the project, running from the internal Flash, select “On-Chip Flash” as target and “Rebuild” the project. The results can be found in the directories:

\RTC72423\Intern\List and \ RTC72423\Intern\Obj

Attention: Before the CPU can be used in Single Chip mode, the tin-solder at LB1 on the module has to be removed.

Use UDE from PLS to download your application into the internal Flash of XC167CI.

13. Know-how for handling XC167CI IDEs

There is no general way how to handle IDEs to get fast and good results but we can give some useful hints to avoid mistakes.

There are a few differences in handling projects for debugging the application and for making a release version for the ROM (Flash).

For a debug session, you need a RAM or ROM monitor. This is a small program which initializes registers and provides functions in order to make your hardware able to communicate with a debugger, usually via serial line. The most important point is, that the registers for RAM memory are set to a correct value by the monitor, because the remote debugger loads your application into RAM and has to modify it for setting breakpoints. The configuration for the Keil monitor can be found in the project MONITOR.UV2 and the configuration for the Tasking monitor can be found, when opening the file \ FORTH_EVA167_XC167\ fsf_debug_xc167ci.cfg

A RAM monitor is loaded via bootstrap loader, the ROM monitor has to be programmed into the ROM (Flash) and is automatically activated after RESET. This Evaluation Kit works mainly with RAM monitors because the download time is so short that a direct start from ROM yields no advantages.

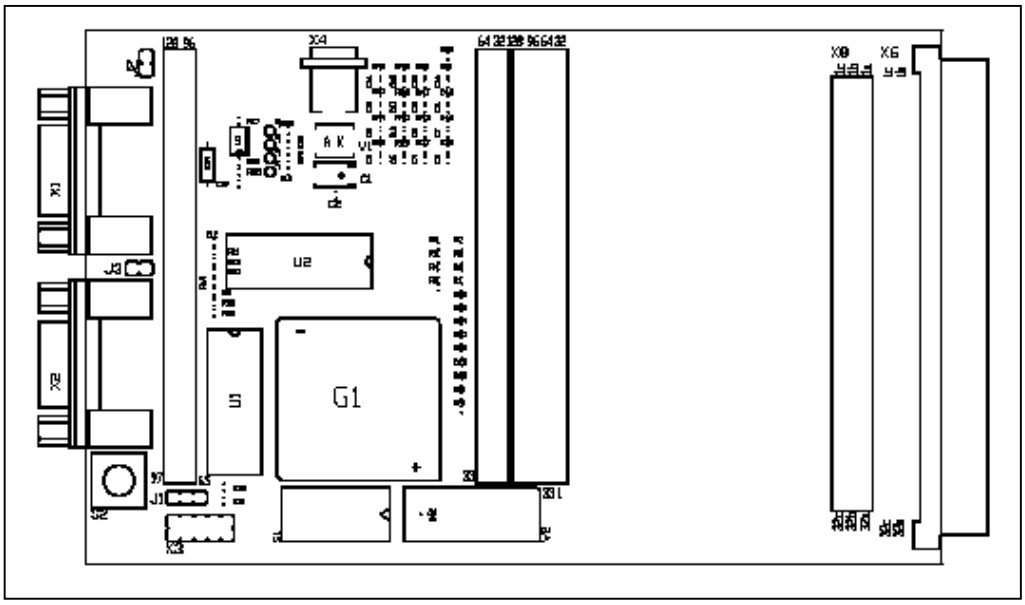
Please make sure that your application doesn't overwrite the monitor in the RAM!

For a release session, you have to link a "start-up" to your application, which is automatically started after RESET. The start-up is usually an assembler program, linked to your application (Keil example **RTC72423**: STARTXC167.A66), or is automatically generated via the project settings (Tasking example **XC167**). The memory settings for linker/locator has to match with the start-up settings! I.e., if you set the address selection register for RAM to 100 000h and locate your data to 10 000h your program won't run.

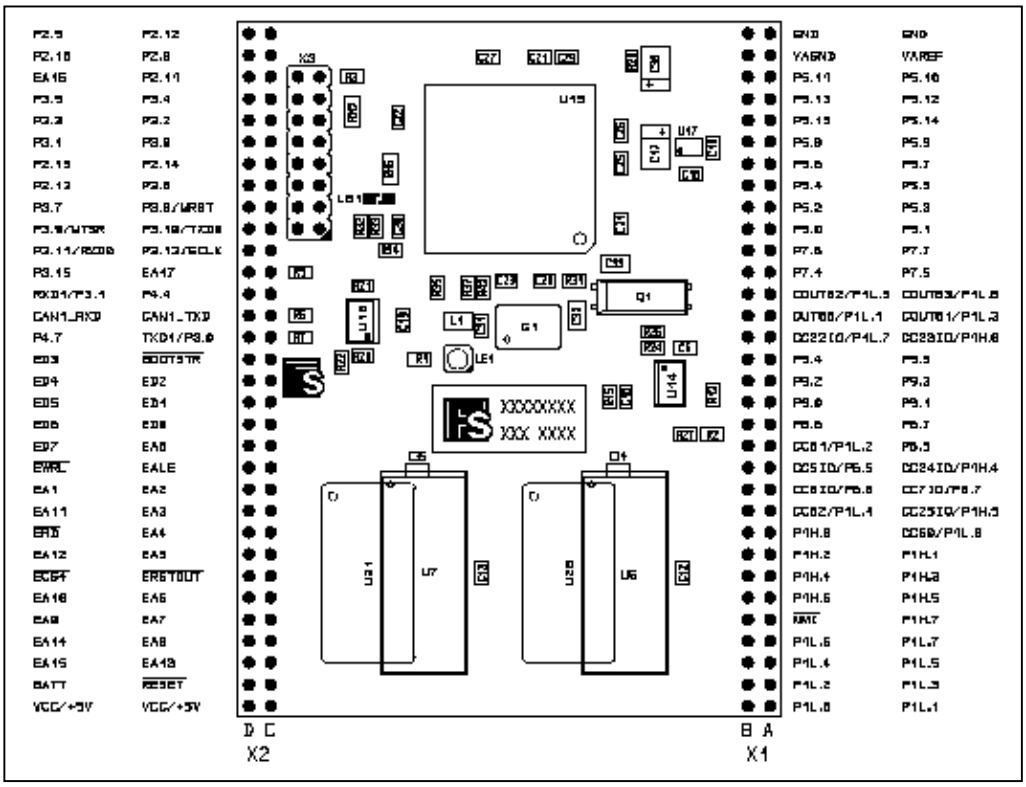
Generally, the start-up initializes your hardware. Please check the settings for memory access times for RAM and ROM (Flash) to avoid your hardware running slower than it could. **If you have any questions, please don't hesitate to contact the technical support of FS FORTH-SYSTEME!**

14. Appendix

14.1. Top View of the Base board



14.2. Top View of the Module



14.3. Bottom View of the Module