phyCORE-XC161 with XC161CJ

QuickStart Instructions

Using PHYTEC FlashTools 3 and the Keil µVision2 Software Evaluation Development Tool Chain

Note: The PHYTEC Spectrum CD includes the electronic version of the English phyCORE-XC161 Hardware Manual

Edition: July 2003

A product of a PHYTEC Technology Holding company

	EUROPE	NORTH AMERICA
Address:	PHYTEC Technologie Holding AG Robert-Koch-Str. 39 D-55129 Mainz GERMANY	PHYTEC America LLC 203 Parfitt Way SW, Suite G100 Bainbridge Island, WA 98110 USA
Ordering Information:	+49 (800) 0749832 order@phytec.de	1 (800) 278-9913 sales@phytec.com
Technical Support:	+49 (6131) 9221-31 support@phytec.de	1 (800) 278-9913 support@phytec.com
Fax:	+49 (6131) 9221-33	1 (206) 780-9135
Web Site:	http://www.phytec.de	http://www.phytec.com

1st Edition: July 2003

Figures

1 Introduction to the Rapid Development Kit

This QuickStart provides:

- general information on the PHYTEC phyCORE-XC161 Single Board Computer (SBC),
- an overview of Keil's µVision2 software development tool chain evaluation version, and
- instructions on how to run example programs on the phyCORE-XC161, mounted on the PHYTEC phyCORE Development Board HD200 2.5V, in conjunction with Keil software tools

Please refer to the phyCORE-XC161 Hardware Manual for specific information on such board-level features as jumper configuration, memory mapping and pin layout. Selecting the links on the electronic version of this document links to the applicable section of the phyCORE-XC161 Hardware Manual.

1.1 Rapid Development Kit Documentation

This "Rapid Development Kit" (RDK) includes the following electronic documentation on the enclosed "PHYTEC Spectrum CD-ROM":

- the PHYTEC phyCORE-XC161 Hardware Manual
- controller User's Manuals and Data Sheets
- this QuickStart Instruction with general "Rapid Development Kit" description, software installation hints and three example programs enabling quick out-of-the box start-up of the phyCORE-XC161 in conjunction with the Keil µVision2 software development tool chain evaluation version

1.2 Overview of this QuickStart Instruction

This QuickStart Instruction gives a general "Rapid Development Kit" description, as well as software installation hints and three example programs enabling quick out-of-the box start-up of the phyCORE-XC161 in conjunction with Keil µVision2. It is structured as follows:

1) The *"Getting Started"* section uses two example programs: *Blinky* and *Hello* to demonstrate the download of user code to the Flash device using PHYTEC FlashTools 3.
2) The *"Getting More Invloved"* section provides step-by-step instructions on how to modify both examples, create and build new projects and generate and download output files to the phyCORE-XC161 using the Keil tools and FlashTools 3.
3) The *"Debugging"* section provides a third example program - "Debug" - to demonstrate simple debug functions using the Keil µVision2 debug environment.

In addition to dedicated data for this Rapid Development Kit, the PHYTEC Spectrum CD-ROM contains supplemental information on embedded microcontroller design and development.

1.3 System Requirements

Use of this "Rapid Development Kit" requires:

- the PHYTEC phyCORE-XC161
- the phyCORE Development Board HD200 2.5V with the included DB-9 serial cable and AC adapter supplying 5 VDC /min. 500 mA
- the PHYTEC Spectrum CD
- an IBM-compatible host-PC (486 or higher running at least Windows95/NT)

For more information and example updates, please refer to the following sources:

PHYTEC

http://www.phytec.com - or - http://www.phytec.de
support@phytec.com - or - support@phytec.de

KEIL SOFTWARE

http://www.keil.com
support@keil.com

1.4 The PHYTEC phyCORE-XC161

The phyCORE-XC161 represents an affordable yet highly functional Single Board Computer (SBC) solution in sub-miniature dimensions (60 x 53 mm). It is intended for use in memory-intensive applications running within a CAN bus system. The standard board is populated with an Infineon XC161CJ controller as well as a Real-Time Clock which can be buffered by an external battery.

All applicable data/address lines and signals extend from the underlying logic devices to two high-density Molex SMT pin header connectors (pin width is 0.635 mm/25 mil) lining the circuit board edges. This enables the phyCORE-XC161 to be plugged like a "big chip" into target hardware.

The standard module runs at a 40 MHz internal clock speed (delivering 25 ns instruction cycle) and offers 512 kByte (up to 1.5 MByte) SRAM and 256 kByte (up to 2 MByte) Flash on-board for DATA and CODE storage.

The module communicates by means of 2 RS-232 transceiver and 2 CAN bus interfaces. The optional CS8900A 10Base-T Ethernet controller enables implementation of the module in embedded Internet devices. The phyCORE-XC161 operates within a standard industrial range of 0 to +70°C and requires only a 220 mA power source.

PHYTEC FlashTools 3 enables easy on-board download of user programs.

phyCORE-XC161 (XC161CJ) Technical Highlights

- SBC in subminiature dimensions (60 x 53 mm) achieved through modern SMD technology
- populated with an Infineon XC161CJ controller featuring two Full 2.0B on-chip CAN and operating in 16-bit, non-multiplexed bus mode at 40 MHz CPU speed (25 ns/instruction cycle)
- 512 kByte (up to 1.5 MByte) external SRAM (up to 1 MByte can be buffered with an optional lithium battery)[1]
- 256 kByte (up to 2 MByte) external Flash memory supporting on-board download of user code from a host-PC in conjunction with PHYTEC FlashTools 3 [1]
- no dedicated programming voltage required through use of 5 V Flash-devices
- battery-buffered Real-Time Clock
- two serial interface via RS-232
- optional Ethernet controller
- 16-channel A/D converter with 10-bit resolution
- two 2.0B Full-CAN interfaces
- requires a dual power supply of 5 VDC and 2.5 VDC/ <220 mA

[1] : Please contact PHYTEC for more information about additional module configurations.

The phyCORE Development Board HD200 2.5V, in EURO-card dimensions (160 x 100 mm) is fully equipped with all mechanical and electrical components necessary for the speedy and secure insertion and subsequent programming of most PHYTEC phyCORE high-density series Single Board Computers. Simple jumper configuration readies the Development Board's connection to the phyCORE-XC161, which plugs into the receptacle contact strips mounted on the Development Board HD200 2.5V.

phyCORE Development Board HD200 2.5V Technical Highlights

- Reset signal controlled by push button or RS-232 control line CTS0
- Boot signal controlled by push button or RS-232 control line DSR0
- low voltage socket for supply with regulated input voltage 5 VDC
- additional supply voltage 2.5 VDC
- two DB-9 sockets (P1A, P1B) configurable as RS-232 interfaces
- two additional DB-9 plugs (P2A, P2B) configurable as CAN interfaces
- simple jumper configuration allowing use of the phyCORE Development Board HD200 2.5V with various PHYTEC phyCORE high-density SBC's
- RJ45 Ethernet transformer module
- one control LED D3 for quick testing of user software
- 2 x 160-pin Molex connector (X2) enabling easy connectivity to expansion boards (e.g. PHYTEC GPIO Expansion Board)

1.5 The Keil µVision2 Software Development Tool Chain

Keil µVision2 fully supports the entire Infineon C166 microcontroller family. This includes a C compiler, macroassembler, Linker/Locator and the Simulator and Target Monitor within the µVision2 IDE. Specific chips supported are the 161, 163, 164-CI, 165, 166, 167Cx and XC16x. Future derivatives are easily accommodated due to the flexible Keil C compiler design.

µVision2 supports all in-circuit emulators that adhere to the Infineon OMF166 debugging specification. The Keil OH166 Object-to-Hex converter converts an absolute object file into an Intel hexfile that is suitable for programming into an EPROM device or downloading into external Flash on the PHYTEC phyCORE-XC161 target board.

µVision2 consists of the following executables:

- **C Compiler** c166.exe
- **Assembler** a166.exe
- **Linker** l166.exe
- **Converter** oh166.exe
- **µVision2** Uv2.exe (a Windows-based application)

Once installed, the default destination location for the DOS based files is the ***C:\Keil\C166\bin*** directory while µVision2 is in ***C:\Keil\Uv2***. Access to these programs from Windows is accomplished with µVision2. The entire tool set can be run from µVision2 or directly from DOS with batch files. The evaluation version is limited to 4 kByte of object code. Other than these restrictions, all features operate normally.

µVision2 IDE

µVision2 is a Windows-based Graphical User Interface for the C compiler and assembler. All compiler, assembler and linker options are set with simple mouse clicks. µVision runs under Windows 95/98/Me, NT, 2000 and XP. This Integrated Development Environment (IDE) has been expressly designed with the user in mind and includes a fully functional editor.

All IDE commands and functions are accessible via intuitive pull-down menus with prompted selections. An extensive Help utility is included. External executables can be run from within µVision2, including emulator software.

C166 C Compiler for the Entire Infineon 166/167 Family

The C166 ANSI compiler and A166 assembler are designed specifically for the Infineon 161, 163, C164CI, 165,166, 167, 167Cx, XC16x and future derivatives. The C166 compiler easily integrates into the Keil RTOS and interfaces and passes debug information to the µVision2 Simulator and all in-circuit emulators. Extensions provide access to on-chip peripherals.

The Keil C166 compiler provides the fastest and smallest code using industry benchmarks.

A166 Macroassembler

The Professional Kit (PK) macroassembler is included with the PK Compiler package or is available separately. It is DOS-based or can be run from µVision2 and includes all utilities needed to complete your project.

Debug Environment

µVision2 contains a software simulator supporting debugging either via software on a host-PC or in target hardware. When operated in conjunction with the Keil Monitor resident in target hardware µVision2 enable the following debugging functions:

- run/halt,
- set breakpoints,
- examine/change memory,
- view the stack,
- view/set peripheral information
- apply virtual external signals.

µVision2 has a performance analysis feature to ensure your code runs efficiently. In addition, µVision2 has a disassembler/assembler that allows the modification of user code without recompiling. The evaluation version of µVision2 is restricted to a 8 kByte in manipulable code. Other than this restriction the evaluation tool chain (EK) functions exactly as does the full (PK) version. The evaluation version does not have a starting address restriction and produces useful object code. This allows you to fully evaluate the features and power of Keil products on the PHYTEC target board. The PK full version has no restrictions and is fully ANSI compliant.

FR166 Full-Function RTOS for the Infineon C166 Family

The FR166 is a multi-tasking real-time operating system for the Infineon 166 family. You can manage multiple tasks on a single CPU making your programs much easier to develop. The RTX166 Full includes CAN libraries. The RTX166 Tiny is a subset of the RTX166 Full and is included with all C166 C Compiler Kits. The EK version of the tool chain does not include an RTOS.

CAN (Controller Area Network) Library

The RTX166 Full RTOS supports CAN controllers with the included libraries. The CAN libraries are sold with the RTOS and support 11- and 29-bit message identifiers. Keil 166 and 8051 C compilers interface with the RTOS and CAN libraries. Keil supports all CAN microcontrollers based on the Infineon C505C, C515C, C164-CI, C167Cx and XC16x. Future CAN products based on these 8051 or C16x families are easily supported due to the flexible Keil Compiler design.

2 Getting Started

What you will learn with this Getting Started example:

- installing Rapid Development Kit software
- starting PHYTEC's FlashTools 3 download utility
- interfacing the phyCORE-XC161, mounted on the Development Board, to a host-PC
- downloading example user code in hexfile format from a host-PC to the external Flash memory using FlashTools 3

2.1 Installing Rapid Development Kit Software

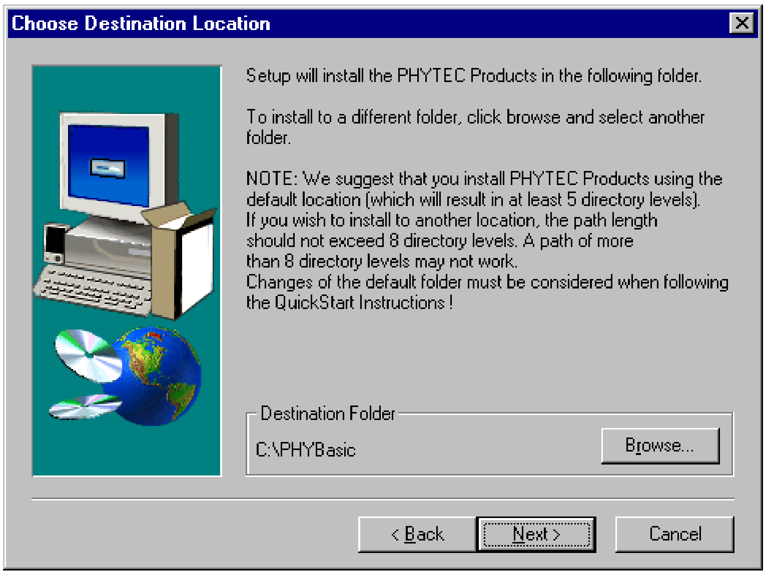
When you insert the PHYTEC Spectrum CD into the CD-ROM drive of your host-PC, the PHYTEC Spectrum CD should automatically launch a setup program that installs the software required for the Rapid Development Kit as specified by the user. Otherwise the setup program ***start.exe*** can be manually executed from the root directory of the PHYTEC Spectrum CD.

The following window appears:

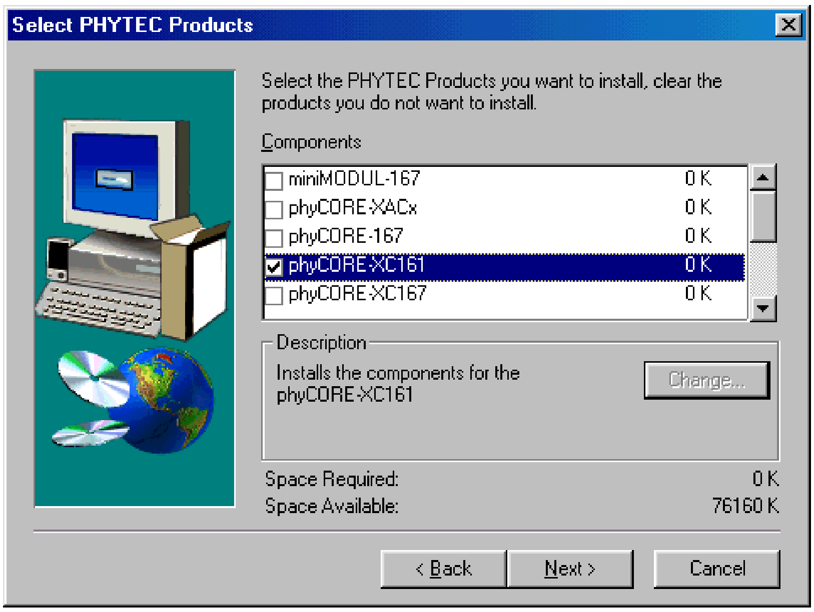
- Choose *Install Basic Product Files* Button.
- After accepting the *Welcome* window and license agreement select the destination location for installation of Rapid Development Kit software and documentation.

The default destination location is ***C:\PHYBasic***. All path and file statements within this QuickStart Instruction are based on the assumption that you accept the default install paths and drives. If you decide to individually choose different paths and/or drives you must consider this for all further file and path statements.

We recommend that you accept the default destination location.



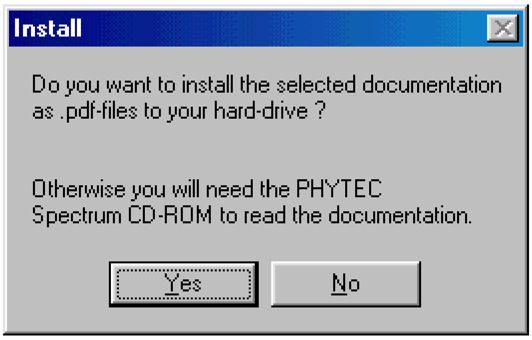
- In the next window select your Rapid Development Kit of choice from the list of available products. By using the *Change* button, advanced users can select in detail which options should be installed for a specific product.



All Kit-specific content will be installed to a Kit-specific subdirectory of the Rapid Development Kit root directory that you have specified at the beginning of the installation process.

All software and tools for this phyCORE-XC161 RDK will be installed to the ***\PHYBasic*** directory on your hard-drive.

- In the next dialog you must choose whether to copy the selected documentation as ****.pdf*** files to your hard drive or to install a link to the file on the Spectrum CD.

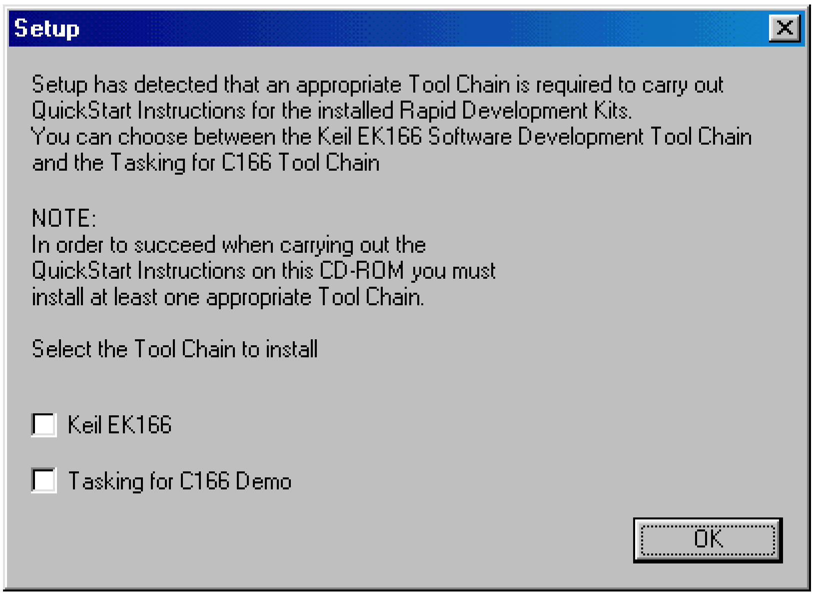


If you decide **not** to copy the documentation to your hard-drive you will need the PHYTEC Spectrum CD-ROM each time you want to access these documents. The installed links will refer to your CD-ROM drive in this case.

If you decide to copy the electronic documentation to your hard-drive, the documentation for this phyCORE-XC161 Kit will also be installed to the Kit-specific subdirectory.

- Setup will now add program icons to the program folder, named *PHYTEC*.

- In the next window, choose the Keil EK166 Software Development Tool Chain.



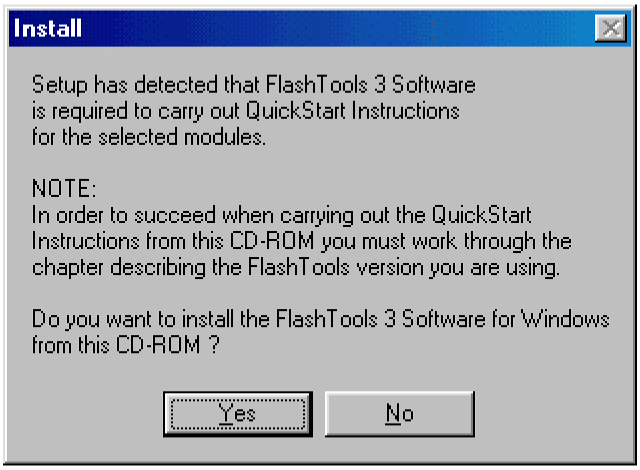
The applicable Keil tool chain must be installed to ensure successful completion of this QuickStart Instruction. Failure to install the proper software could lead to possible version conflicts, resulting in functional problems.

We recommend that you install the Keil EK166 resp. µVision2 from the Spectrum CD-ROM even if other versions of µVision2 are already installed on your system. These QuickStart Instructions and the demo software included on the CD-ROM have been specifically tailored for use with one another.

- After accepting the *Welcome* window and license agreement select the destination location for installation of the Development Tool Chain.

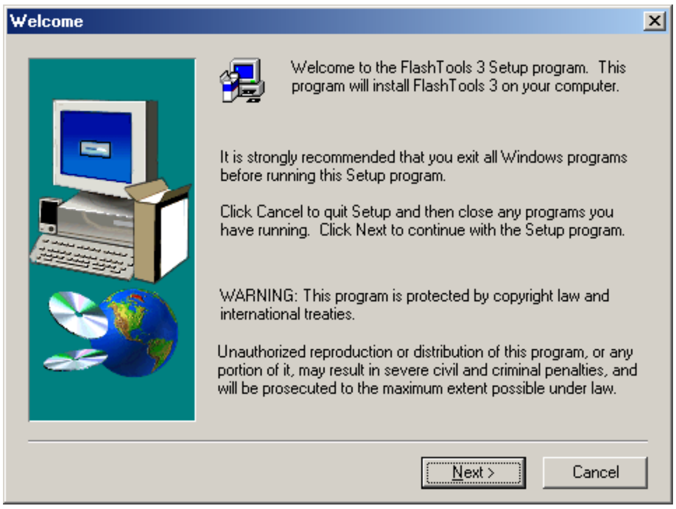
The applicable Keil µVision2 evaluation development tool chain will be installed to your hard-drive. Additional software, such as Adobe Acrobat Reader, will also be offered for installation.

In the following windows you can decide to install FlashTools 3 software and the Acrobat Reader.

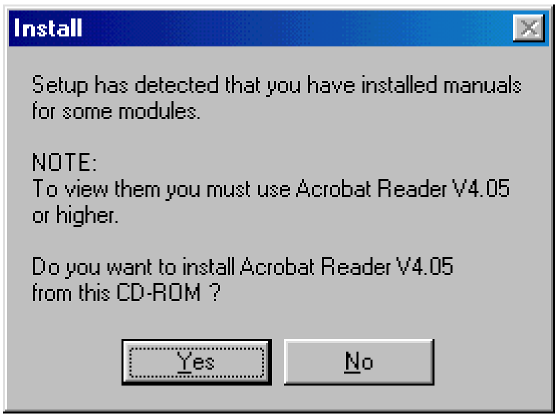


The applicable FlashTools 3 software must be installed to ensure successful completion of this QuickStart Instruction. Failure to install the proper software could lead to possible version conflicts, resulting in functional problems.

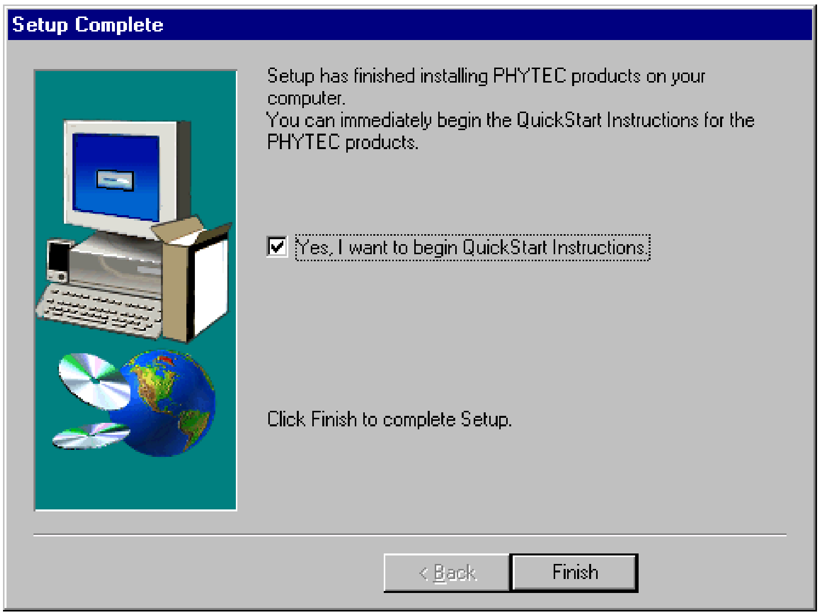
- Click on *Yes* and complete the FlashTools 3 Setup program.



- If a version of Acrobat Reader is already installed on your system you can skip the next step by clicking on *No*.



- Decide if you want to begin the QuickStart Instruction immediately by selecting the appropriate checkbox and click on *Finish* to complete the installation.



2.2 Interfacing the phyCORE-XC161 to a Host-PC

Connecting the phyCORE-XC161, mounted on the phyCORE Development Board HD200 2.5V, to your computer is simple.

- As shown in the figure below, if the phyCORE module is not already pre-installed, mount it connector side down onto the Development Board's receptacle footprint (X6).

Ensure that pin 1 of the phyCORE-connector (denoted by the hash stencil mark on the PCB) matches pin 1 of the receptacle on the phyCORE Development Board HD200 2.5V.

Ensure that there is a solid connection between the Molex connector on the module and the Development Board receptacle.

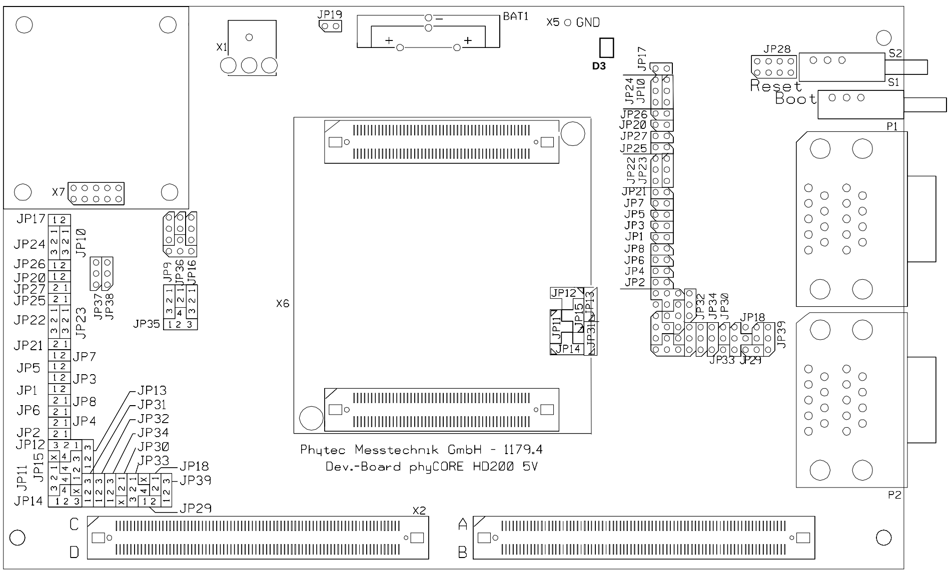


Figure 1: phyCORE Development Board HD200 2.5V Overview

- Configure the jumpers on the Development Board as indicated in *Figure 2*. This correctly routes the RS-232 signals to the DB-9 socket (P1A = bottom) and connects the Development Board's peripheral devices to the phyCORE module.

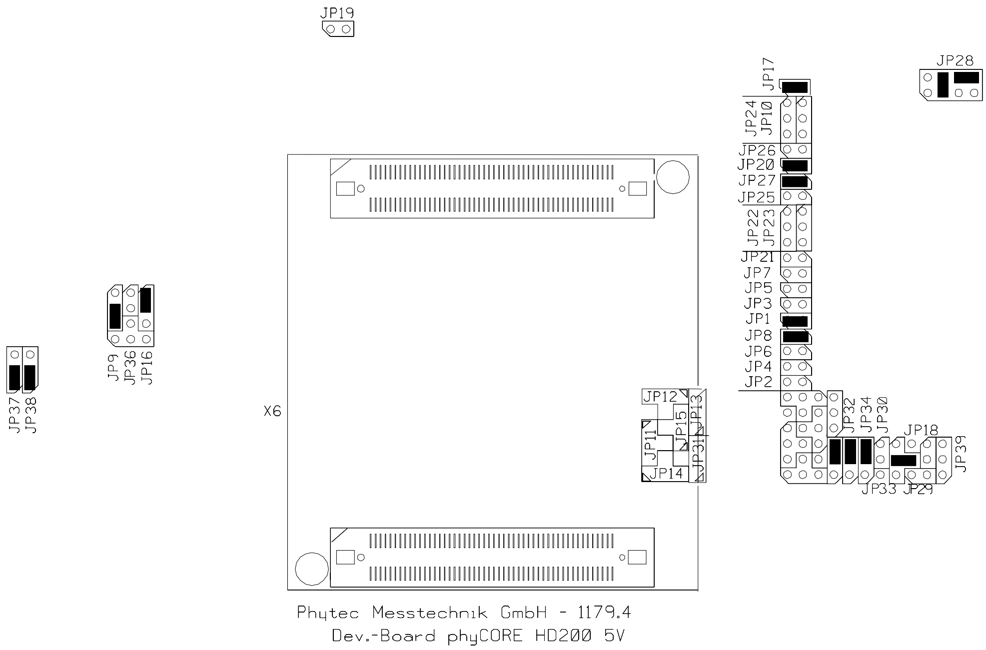


Figure 2: Default Setting for the phyCORE Development Board HD200 2.5V

- Connect the RS-232 interface of your computer to the DB-9 RS-232 interface on the phyCORE Development Board HD200 2.5V (P1A = bottom) using the included serial cable.
- Using the included power adapter, connect the power socket on the board (X1) to a power supply (*refer to Figure 3 for the correct polarity*).

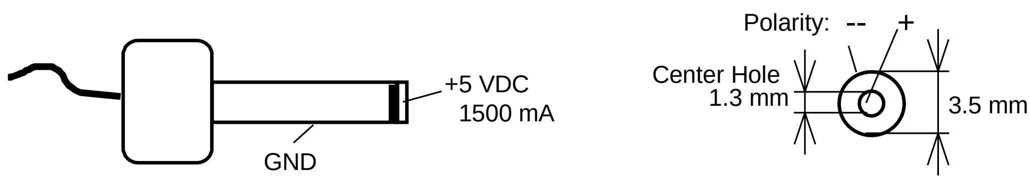


Figure 3: Power Connector

- Simultaneously press the Reset (S2) and Boot (S1) buttons on the phyCORE Development Board HD200 2.5V, first releasing the Reset and then, two or three seconds later, release the Boot button.

This sequence of pressing and releasing the Reset (S2) and Boot (S1) buttons renders the phyCORE-XC161 into the Bootstrap mode. Use of FlashTools 3 always requires the phyCORE-XC161 to be in Bootstrap mode. *See section 5.1, "FlashTools 3" for more details.*

The phyCORE module should now be properly connected via the phyCORE Development Board HD200 2.5V to a host-PC and power supply. After executing a Reset and rendering the board in Flash programming mode, you are now ready to program the phyCORE-XC161. This phyCORE module/phyCORE Development Board HD200 2.5V combination is also referred to as "target hardware".

2.3 Starting PHYTEC FlashTools 3

FlashTools 3 for Windows is a utility program that allows download of user code in Intel ***.hex** or ***.h86** file format from a host-PC to a PHYTEC SBC via an RS-232 connection.

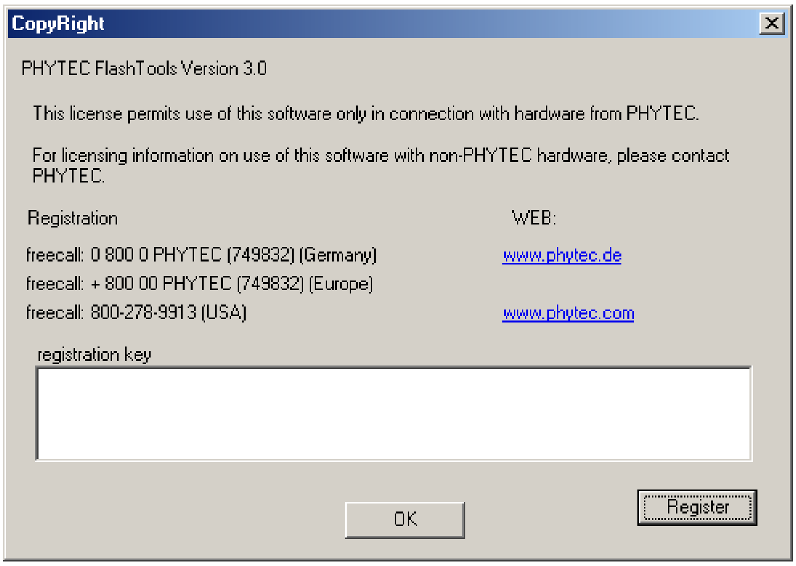
Proper connection of a PHYTEC SBC to a host-PC enables the software portion of FlashTools 3 to recognize and communicate to the Bootstrap loader on the phyCORE-XC161.

- You can start FlashTools 3 by selecting it from the *Programs* menu using the Windows *Start* button.

It is recommended that you drag the FlashTools 3 icon onto the desktop of your PC. This enables easy start of FlashTools 3 by double-clicking on the icon.

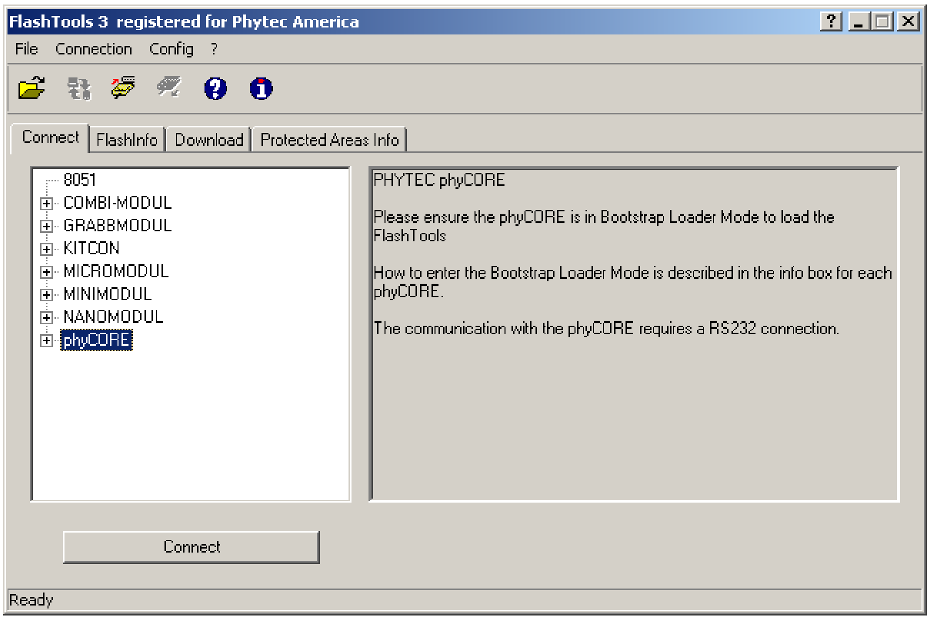
2.4 Downloading Example Code with FlashTools 3

- Start FlashTools 3 for Windows by double-clicking on the FlashTools icon or by selecting *FlashTools 3* from within the ***Programs|Phytec*** program group.
- The FlashTools 3 registration window will now appear. Please contact PHYTEC to obtain your registration key.

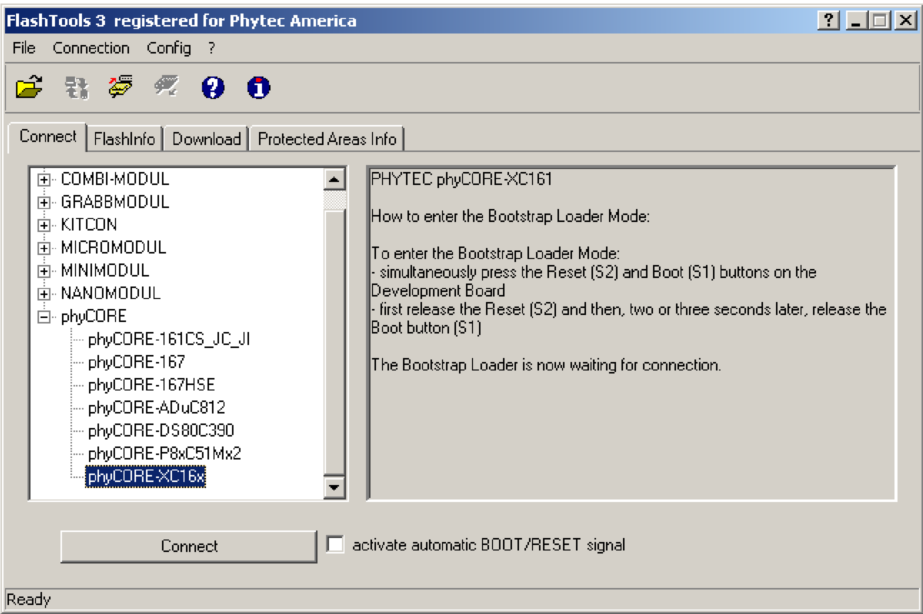


- Copy your registration key into the applicable field and click on the *Register* button.

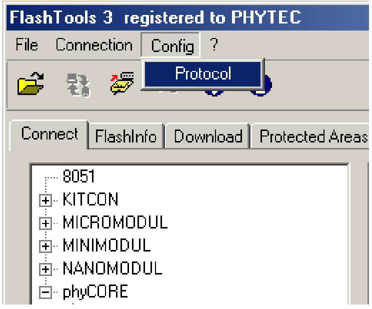
- The FlashTools 3 start window with the *Connect* tab will now appear.



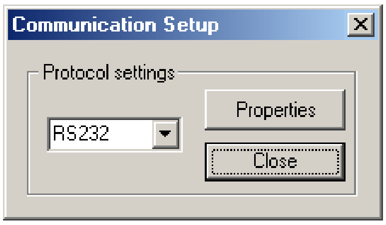
- Select the phyCORE-XC16x in the target hardware list in the *Connect* window. Click on the + sign in front of the phyCORE string to expand the view and to see all available modules.



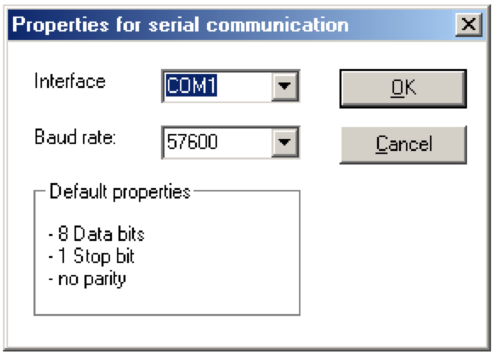
- Click on *Config* and select *Protocol* in the pull-down menu to advance to the *Communication Setup* window.



- Communication properties between the target hardware and the host-PC are configured in the *Communication Setup* window.



- The RS-232 protocol is configured as default. Click on the *Properties* button to advance to the properties for the serial communication window.

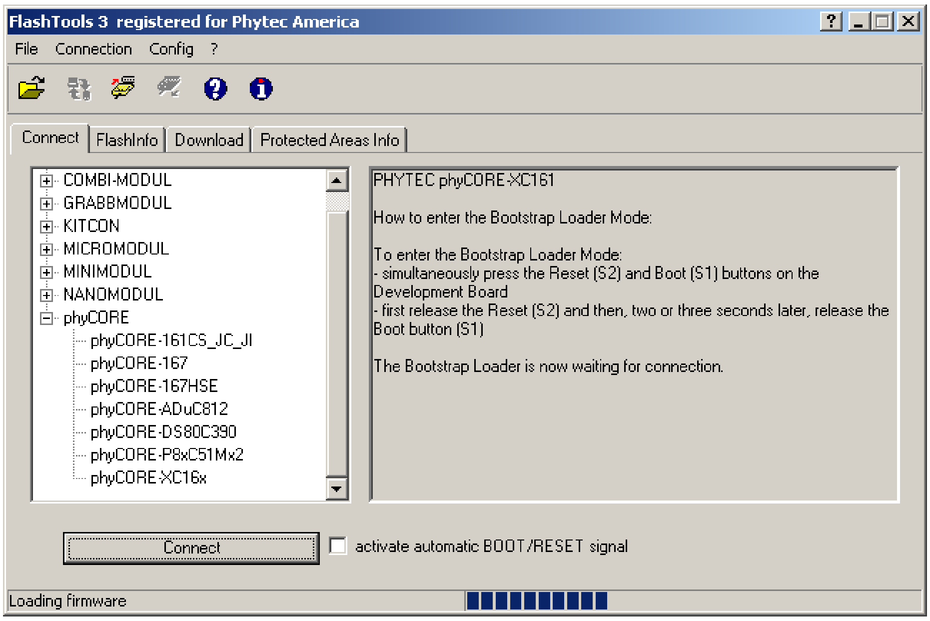


- Choose the correct serial port for your host-PC and a 57,600 baud rate.
- Click on *OK* to save these settings.
- Click on the *Close* button in the *Communication Setup* window to exit the communication setup.

Note:
Always ensure that the phyCORE-XC161 is in Bootstrap mode before pressing the *Connect* button (*see section 2.2)*

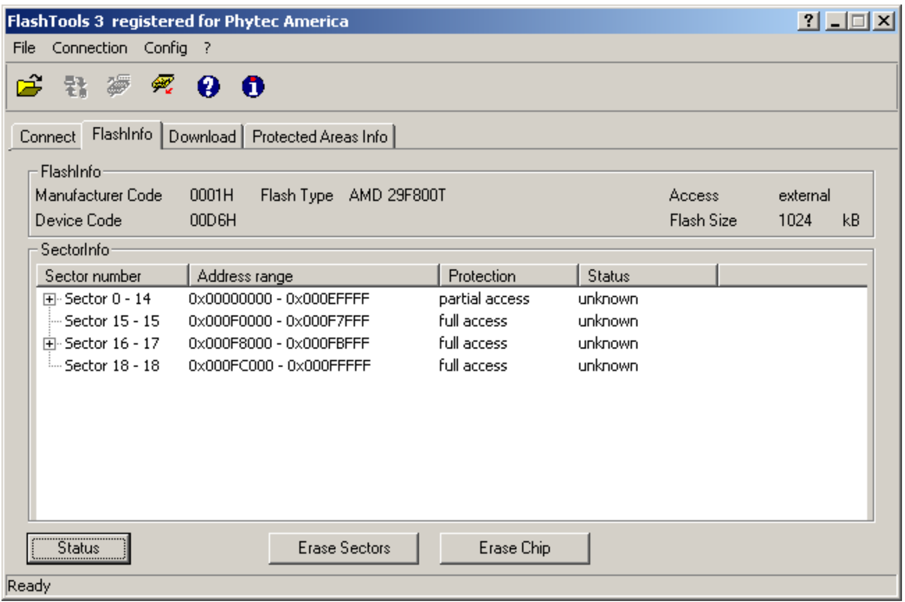
- Now click the *Connect* button to establish connection to the target hardware.

The microcontroller tries to automatically adjust to the baud rate selected within the baud rate pull-down menu. However, it may occur that the selected baud rate can not be attained. This results in a connection error. In this case, try other baud rates to establish a connection. Before attempting each connection, be sure to reset the target hardware and render it into Bootstrap mode as described in *section 2.2.*

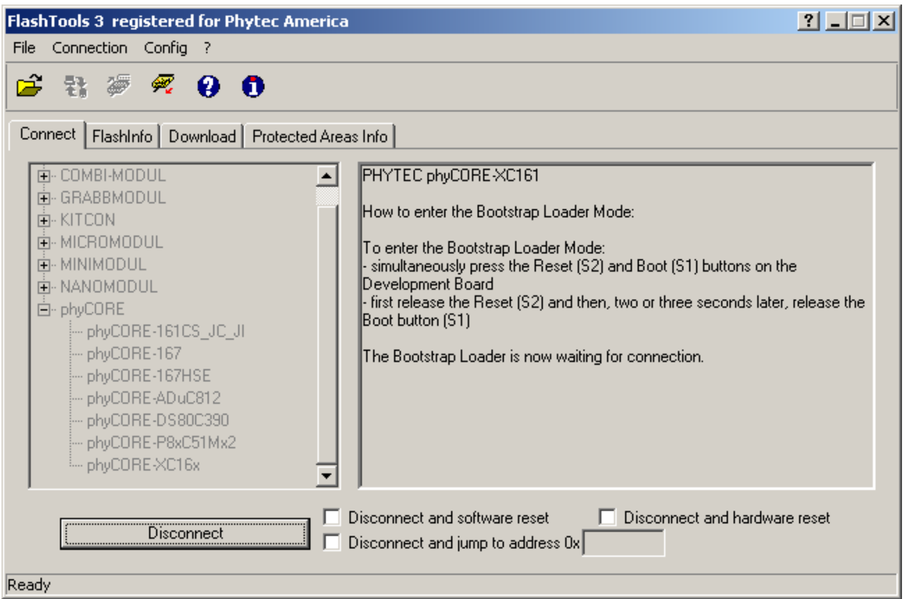


Returning to the FlashTools 3 tabsheet window, you will see tabs for the following:

FlashInfo[1] displays Flash-specific information and enables erasure and status check of Flash sectors specified by the user:



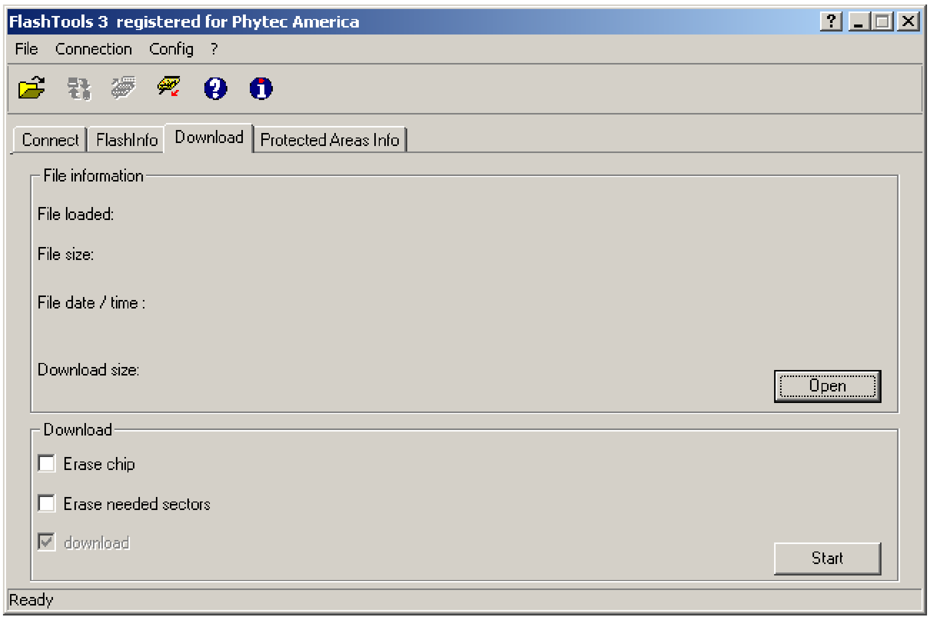
The *Connect* tab allows connection to and disconnection from the board. This is the same window that was used when you first entered FlashTools 3:



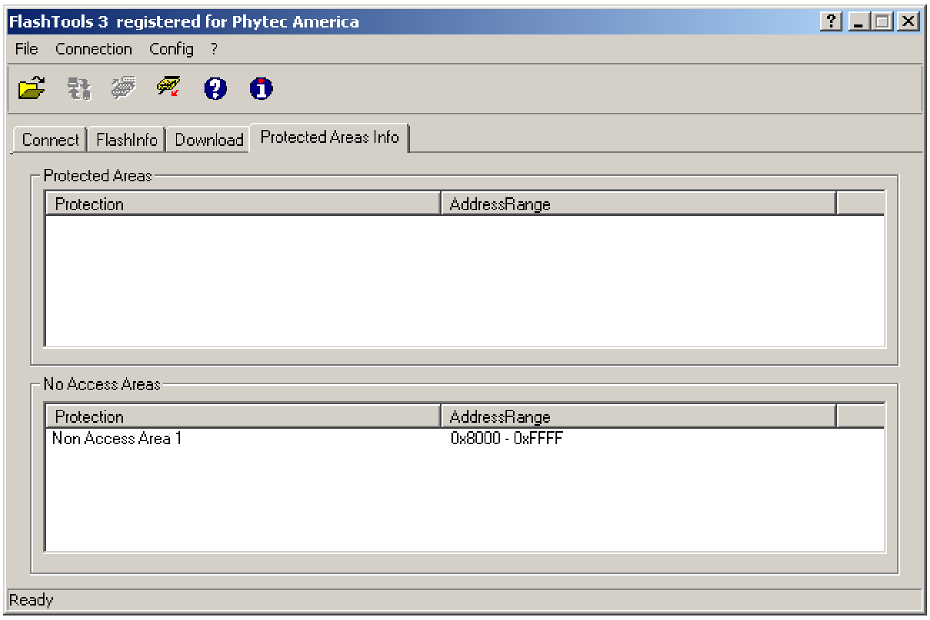
1 : The number of banks shown on the *FlashInfo* tabsheet varies depending on the size and type of the Flash mounted on the phyCORE-XC161.

Note:
To use the option ***Disconnect and hardware reset*** the handshake signals on the Development Board must be connected to the host-PC. This requires closing Jumpers JP22 and JP23 at position 2+3.

Download downloads specified hexfiles to the target hardware:



Protected Areas Info shows protected areas of Flash memory:



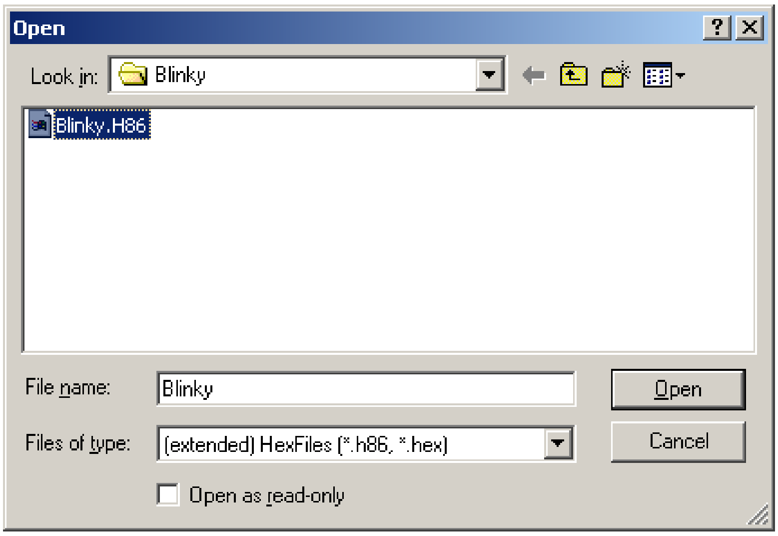
2.4.1 "Blinky"

The "Blinky" example downloads a program to the Flash that, when executed, manipulates the LED D3 on the phyCORE Development Board HD200 2.5V that is located above the jumper field (*refer to Figure 1*).

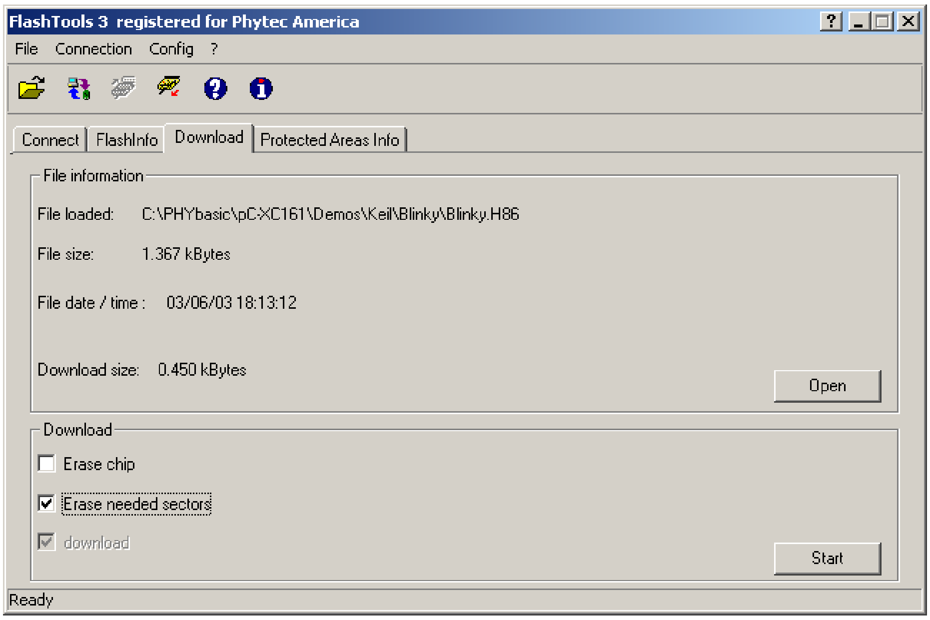
- Returning to the FlashTools 3 tabsheet, choose the *Download* tab and click on the *Open* button.

The hexfile has already been installed to your hard drive during the installation procedure.

- Browse to the correct drive and path for the phyCORE-XC161 Demo folder (default location ***C:\PHYBasic\pC-XC161\Demos\Keil\Blinky\Blinky.h86***) and click *Open.*



- The FlashTools 3 *Download* window will re-appear. Now select the *Erase needed sectors* checkbox in the lower left corner. FlashTools 3 will then erase the required Flash sectors first before downloading the specified hexfile.



- Click on the *Start* button in the lower right corner. You can watch the status of the Flash erasure and code download of ***Blinky.h86*** into external Flash memory in the lower right corner of the *Download* window.
- The individual steps of the Flash download procedure can be viewed in the status bar at the bottom of the FlashTools 3 window. Wait until the status check finishes before returning to work with the board. Once you see the *Ready* message in the lower left corner, the downloaded code can be executed.
- Returning to the *Connect* tab, click on the *Disconnect* button and exit FlashTools 3.
- Press the Reset button (S2) on the phyCORE Development Board HD200 2.5V to reset the target hardware and to start execution of the downloaded software.
- Successful execution of the program will flash the LED D3 with equal on and off duration.

2.4.2 "Hello"

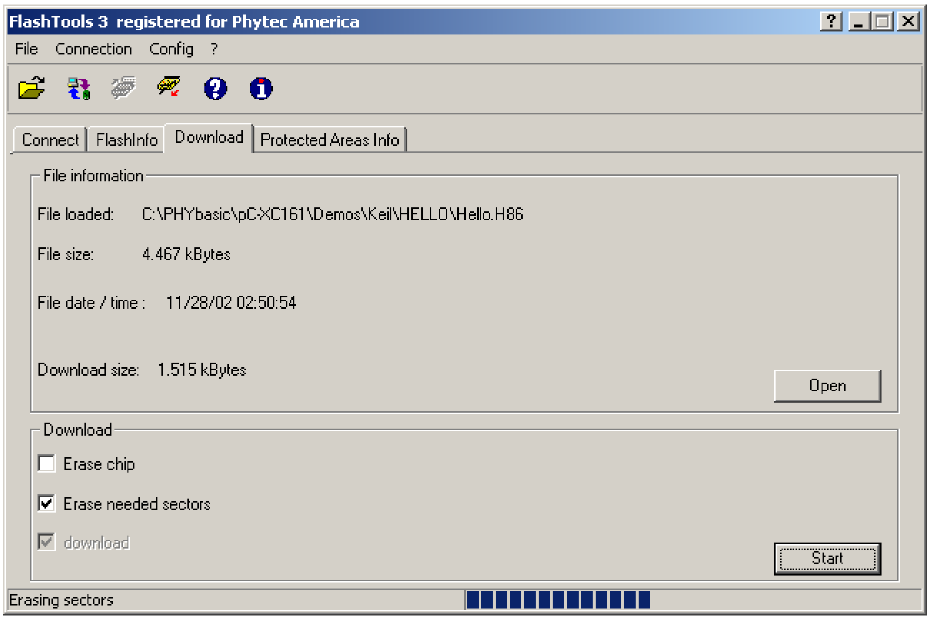
The "Hello" example downloads a program to the Flash that, when executed, performs an automatic baud rate detection and sends a character string from the target hardware back to the host-PC. The character string can be viewed with a terminal emulation program. This example program provides a review of the FlashTools 3 download procedure. For detailed commentary on each step, described below in concise form, *refer back to sections 2.3 through 2.4.1*.

- Ensure that the target hardware is properly connected to the host-PC and a power supply.
- Reset the target hardware and force it into Bootstrap mode by simultaneously pressing the Reset (S2) and Boot (S1) buttons on the phyCORE Development Board HD200 2.5V and then releasing first the Reset and, two or three seconds later, the Boot button.
- Start FlashTools 3.
- At the *Connect* tab of the FlashTools 3 tabsheet, select the phyCORE-XC161 and click the *Connect* button in the lower left corner to establish connection to the target hardware.
- Returning to the FlashTools 3 tabsheet, choose the *Download* tab and click on the *Open* button.

The demo hexfile has already been installed to your hard drive during the installation procedure.

- Browse to the correct drive and path for the phyCORE-XC161 Demo folder (default location ***C:\PHYBasic\pC-XC161\Demos\Keil\Hello\Hello.h86***) and click *Open.*
- The FlashTools 3 *Download* window will re-appear. Now select the *Erase needed sectors* checkbox in the lower left corner. FlashTools 3 will then erase the required Flash sectors first before downloading the specified hexfile.

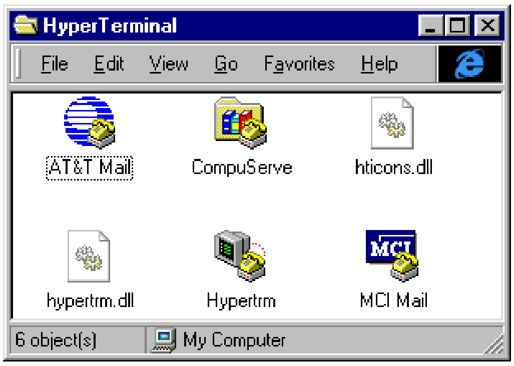
- Click on the *Start* button. You can watch the status of the Flash erasure and code download of ***Hello.h86*** into the external Flash memory in the lower right corner of the *Download* window.



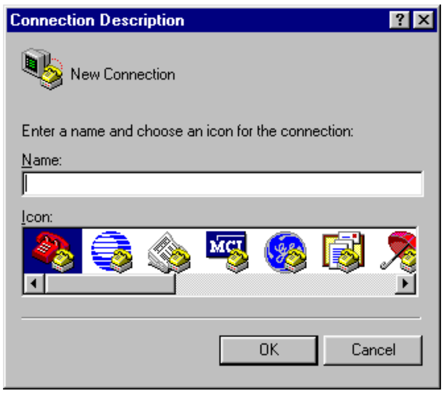
- The individual steps of the Flash download procedure can be viewed at the bottom of the FlashTools 3 window. Wait until the status check finishes before returning to work with the board. Once you see the *Ready* message in the lower left corner, the downloaded code can be executed.
- Returning to the *Connect* tab, click on the *Disconnect* button and exit FlashTools 3.

Monitoring the execution of the Hello demo requires use of a terminal program, such as the HyperTerminal program included within Windows.

- Start the HyperTerminal program within the *Programs|Accessories* bar.
- The HyperTerminal main window will now appear[1]:
- Double-click on the HyperTerminal icon "*Hypertrm*" to create a new HyperTerminal session.



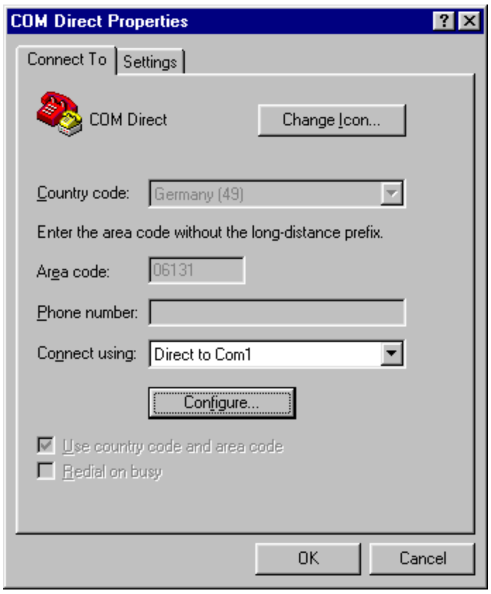
- The Connection Description window will now appear. Enter "COM Direct" in the *Name* text field.



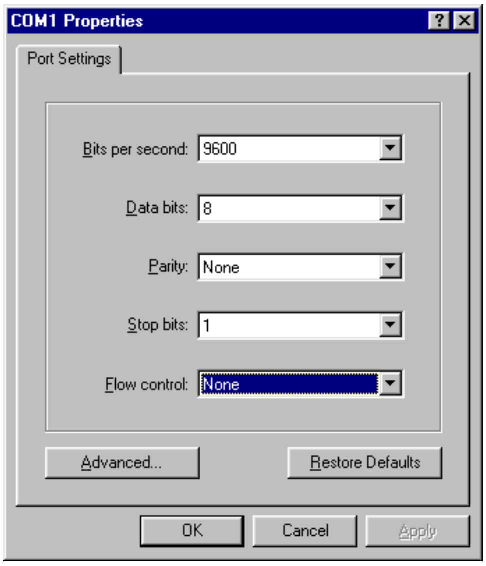
- Next click on *OK*. This creates a new HyperTerminal session named "COM Direct" and advances you to the next HyperTerminal window.

[1]: The HyperTerminal Window has a different appearance for different versions of Windows.

- The *COM Direct Properties* window will now appear. Specify *Direct to COM1/COM2* under the *Connect Using* pull-down menu (be sure to indicate the correct COM setting for your system).

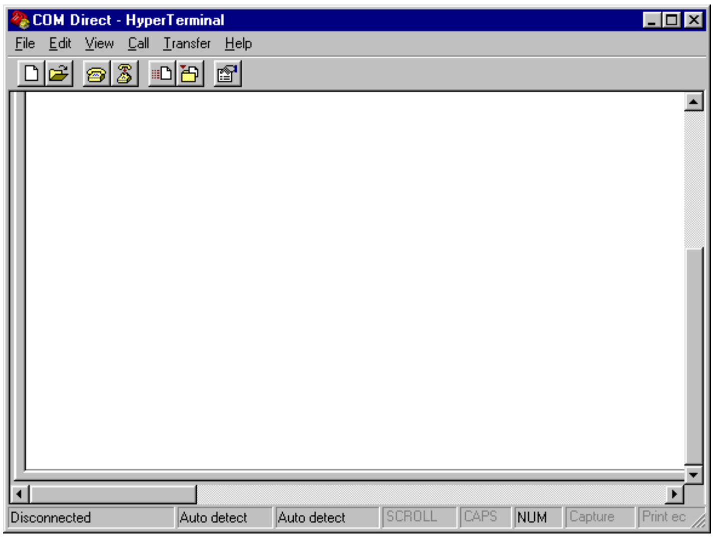


- Click the *Configure* button in the *COM Direct Properties* window to advance to the next window (*COM1/COM2 Properties*).



- Then set the following COM parameters: Bits per second = *9,600*; Data bits = *8*; Parity = *None*; Stop Bits = *1*; Flow Control = *None*.

- Selecting *OK* advances you to the *COM Direct–HyperTerminal* monitoring window. Notice the connection status report in the lower left corner of the window.



- Resetting the phyCORE Development Board HD200 2.5V (at S2) will execute the ***Hello.h86*** file loaded into the Flash.
- Successful execution will send the character string "*Hello World*" from the target hardware to the HyperTerminal window.
- Click the disconnect icon  in HyperTerminal toolbar and exit HyperTerminal.
- If no output appears in the HyperTerminal window check the power supply, the COM parameters and the RS-232 connection.

The code within the demo application ***Hello.h86*** initializes the serial port of your phyCORE-XC161 to 9600 baud. The initialization values are based on the assumption that the microcontroller runs at a 40 MHz internal clock frequency. Please note that an oscillator with a frequency of 16 MHz populates the phyCORE-XC161. Using the internal PLL (Phase Locked Loop) device results in an internal 40 MHz CPU frequency. If your phyCORE-XC161 is equipped with a different speed oscillator, the demo application might transmit using another baud rate. This may lead to incoherent characters appearing in the HyperTerminal window following execution of code.

3 Getting More Invloved

What you will learn with this example:

- how to start the µVision2 tool chain
- how to configure the µVision2 IDE (Integrated Development Environment)
- how to modify the source code from our examples, create a new project and build and download an output ***.h86** file to the target hardware

3.1 Starting the µVision2 Tool Chain

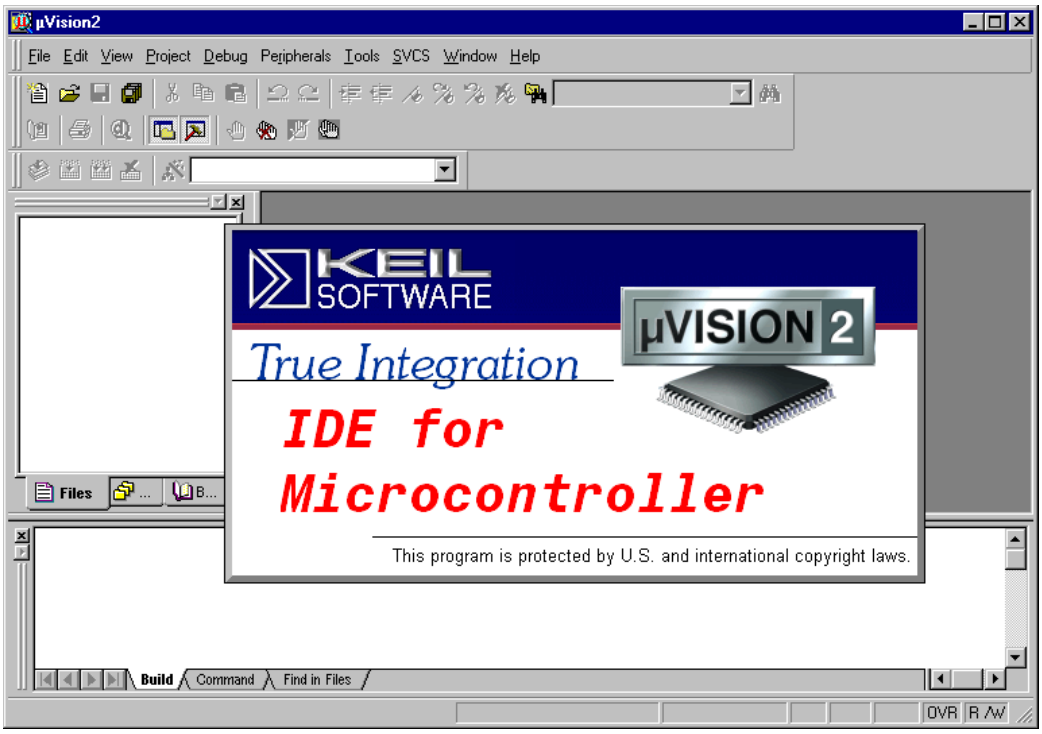
The µVision2 evaluation software development tool chain should have been installed during the install procedure, as described in *section 2.1*.

You can also manually install µVision2 by executing ***ek166v427.exe*** from within the ***\Software\Keil\µVision2.30*** directory of your PHYTEC Spectrum CD.

Note:
It is necessary to use the Keil tool chain provided on the accompanying Spectrum CD in order to complete this QuickStart Instruction successfully. Use of a different version could lead to possible version conflicts, resulting in functional problems.

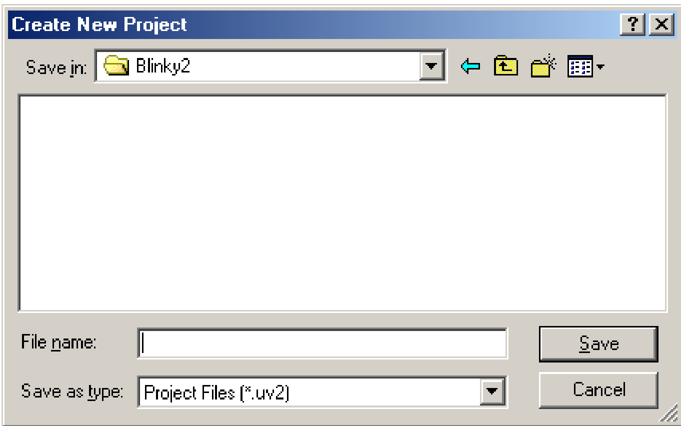
Start the tool chain by selecting *Keil µVision2* from within the programs group.

After you start µVision2, the window shown below appears. From this window you can create projects, edit files, configure tools, assemble, link and start the debugger. Other 3rd party tools such as emulators can also be started from here.

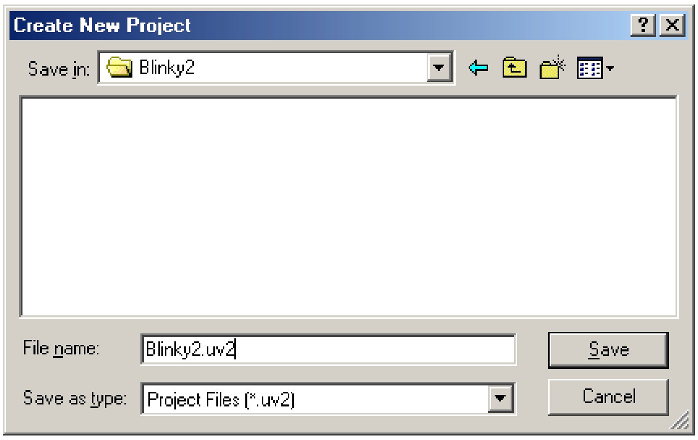


3.2 Creating a New Project and Adding an Existing Source File

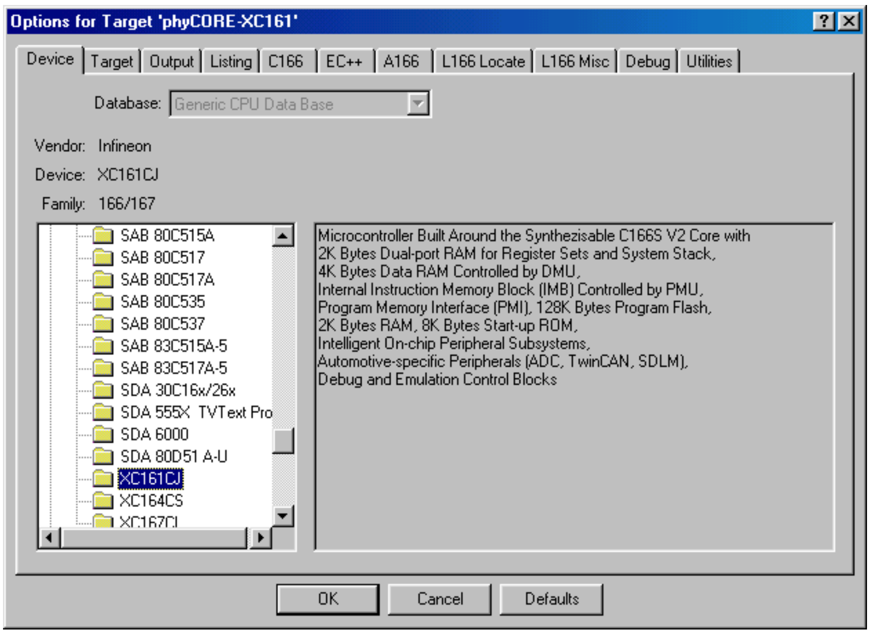
- To create a new project file select from the µVision2 menu ***Project|New Project…***. This opens a standard Windows dialog that asks you for the new project file name.
- Change to the project directory created by the installation procedure (default location ***C:\PHYBasic\pC-XC161\Demos\Keil\Blinky2***).



- In the text field '*File name*', enter the file name of the project you are creating. For this tutorial, enter the name ***Blinky2.uv2*** and click on *Save*.



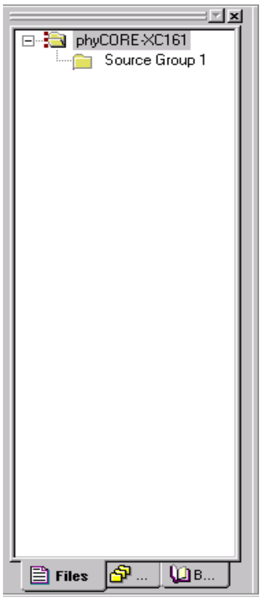
- The ***Select Device for Target 'Target1'*** will automatically appear. Double click on *Infineon* as manufacturer for the CPU within the next window which opens automatically[1]. The phyCORE-XC161 is equipped with a *XC161CJ CPU*. Choose on of this controller type from the list as shown below. This selection sets necessary tool options for the XC161CJ device and simplifies in this way the tool configuration.



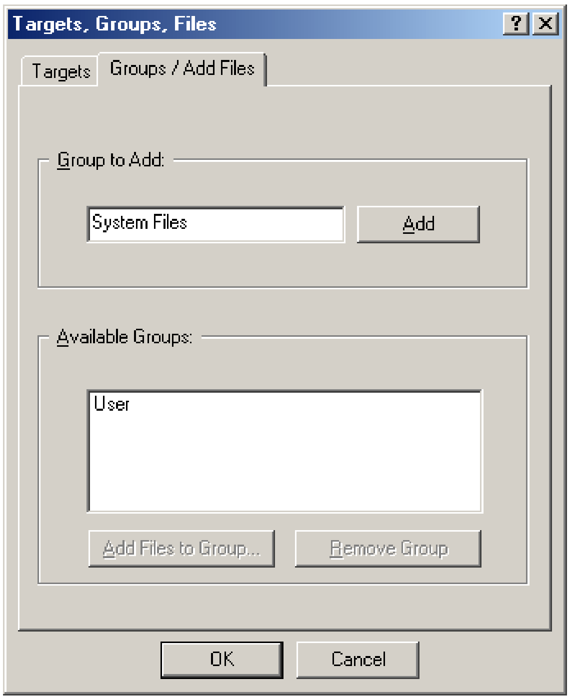
[1] : The same window opens by choosing *Select Device for Target* from the *Project* menu.

- Click on *OK* to save this setting.
- The following pop-up window will appear. Click *No* to continue. The applicable startup code will be added later. Do not use the startup code offered here!

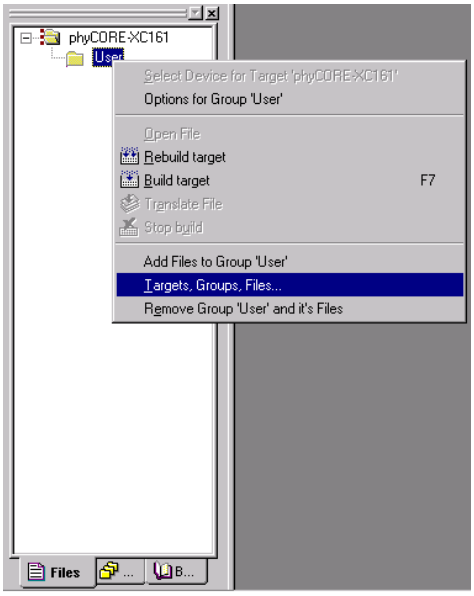
- Now click on *Target1* within the **Project Window - Files** tab. *Target1* is now highlighted. Click on *Target1* again to enable the edit mode. Change the default name of the target to *phyCORE-XC161*.



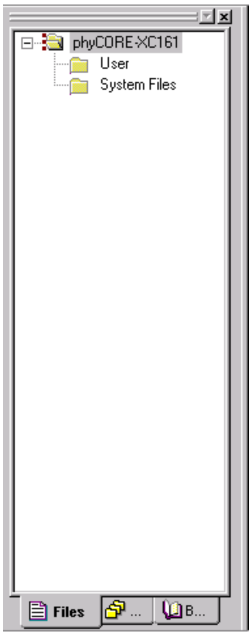
- Select the file group *Source Group 1* in the **Project Window – Files** tab and click on it to change the name into *User*.
- Right-click in the **Project Window – Files** to open a new window. Choose the option ***Targets, Groups, Files...***.



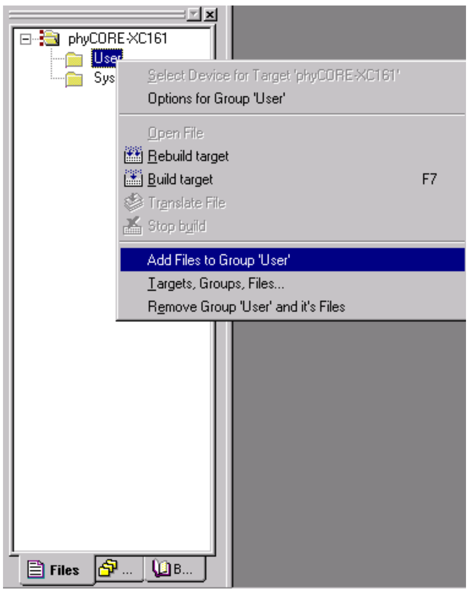
- Select the **Groups / Add Files** tabsheet and type the new group name *System Files* in the ***Group to Add*:** section.



- Click on *Add* and then on *OK*.
- Your project file structure should now look like this:

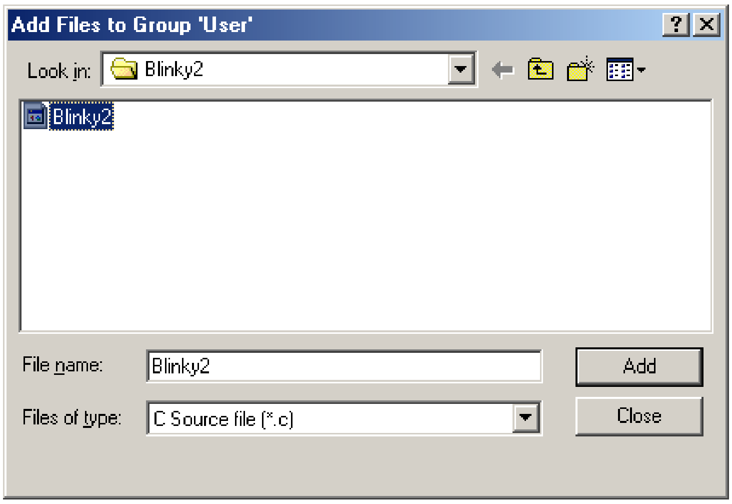


- You are now ready to add source files to the project. Right-click on the *User* group to open a local menu. The option *Add Files to Group 'User'* opens the standard files dialog.

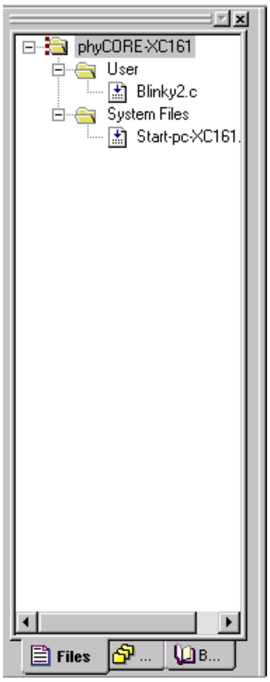


- Select the file ***Blinky2.c***.

- Click on the *Add* button to add the ***Blinky2.c*** file to your current project window.



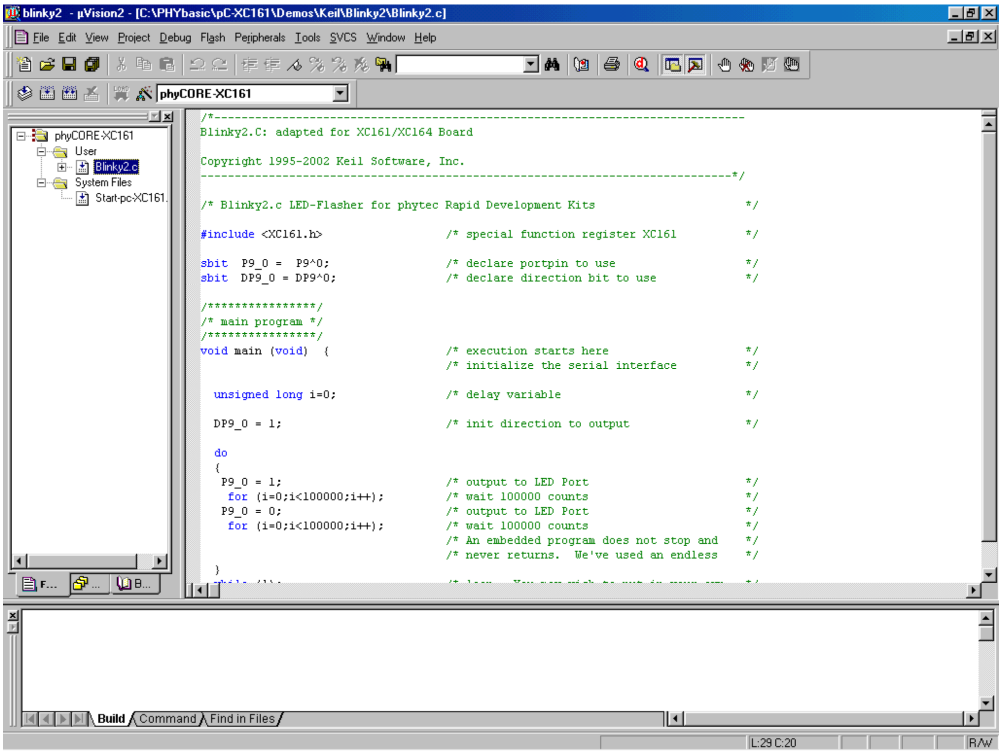
- Close the window.
- Now right-click on the *System Files* group and add the ***Start-pc-XC161.a66*** file. You have to change the file type to "***Asm Source file (*.a, *.src)***" in the *File of types* pull-down menu to see this file.
- Your project window should now look like this:



At this point you have created a project called ***Blinky2.uv2*** and added an existing C source file called ***Blinky2.c*** and an existing Assembler file called ***Start-pc-XC161.a66***. The next step is to modify the C source before building your project. This includes compiling, linking, locating and creating the hexfile.

3.3 Modifying the Source Code

- Double click on ***Blinky2.c*** to open it in the source code editor.



- Locate the following code section. Modify the section shown below (the values shown in bold and italic) from the original (100,000) counts to the indicated values:

```
do {                            /* loop forever           */

    P9_0 = 0;                   /* output to LED port     */
    for (i=0; i<225000; i++);   /* delay for 225000 counts */

    P9_0 = 1;                   /* output to LED port     */
    for (i=0; i<75000; i++)     /* delay for 75000 counts */
  }
while(1);
```

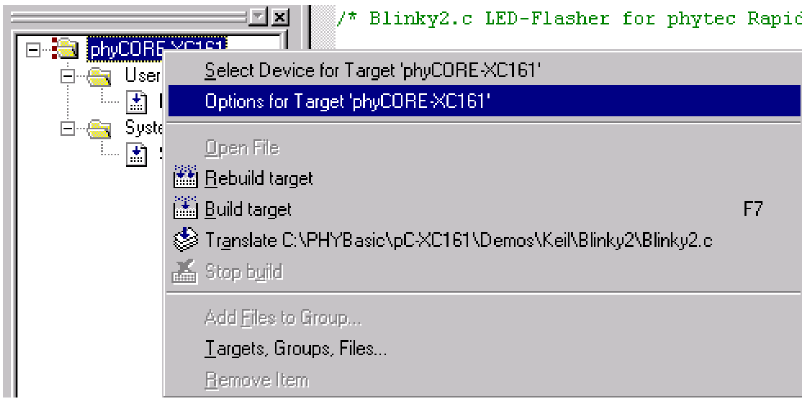

3.4 Saving the Modifications

- Save the modified file by choosing *File|Save* or by clicking the floppy disk icon  .

3.5 Setting Options for Target

Keil includes a Make utility that can control compiling and linking source files in several programming languages. Before using the Make utility, macroassembler, C compiler or linker you must configure the corresponding options. Most of the options are set when specifying the target device for the project. Only the external memory and output options must be set.

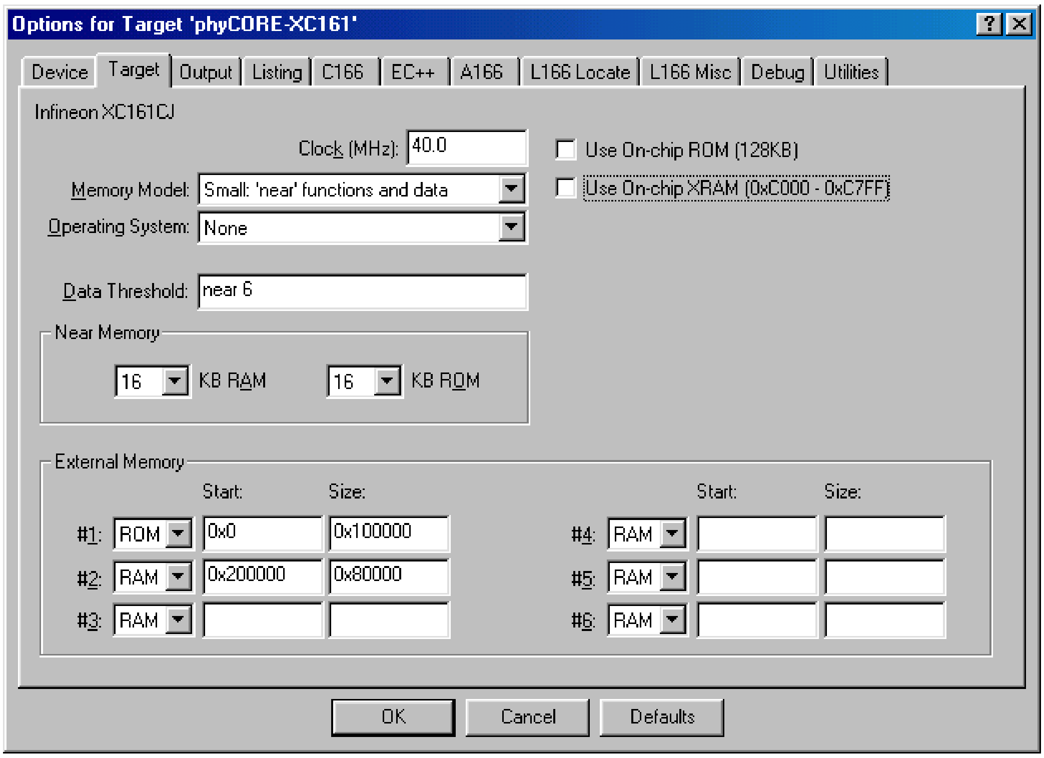
Enter the changes as indicated below and leave all other options set to their default values. µVision2 allows you to set various options with mouse clicks and these are all saved in your project file.



- Click with the right mouse key in the *"Project"* window to open a local menu. Choose the option *Options for Target ‘phyCORE-XC161’*.

To configure the Target:

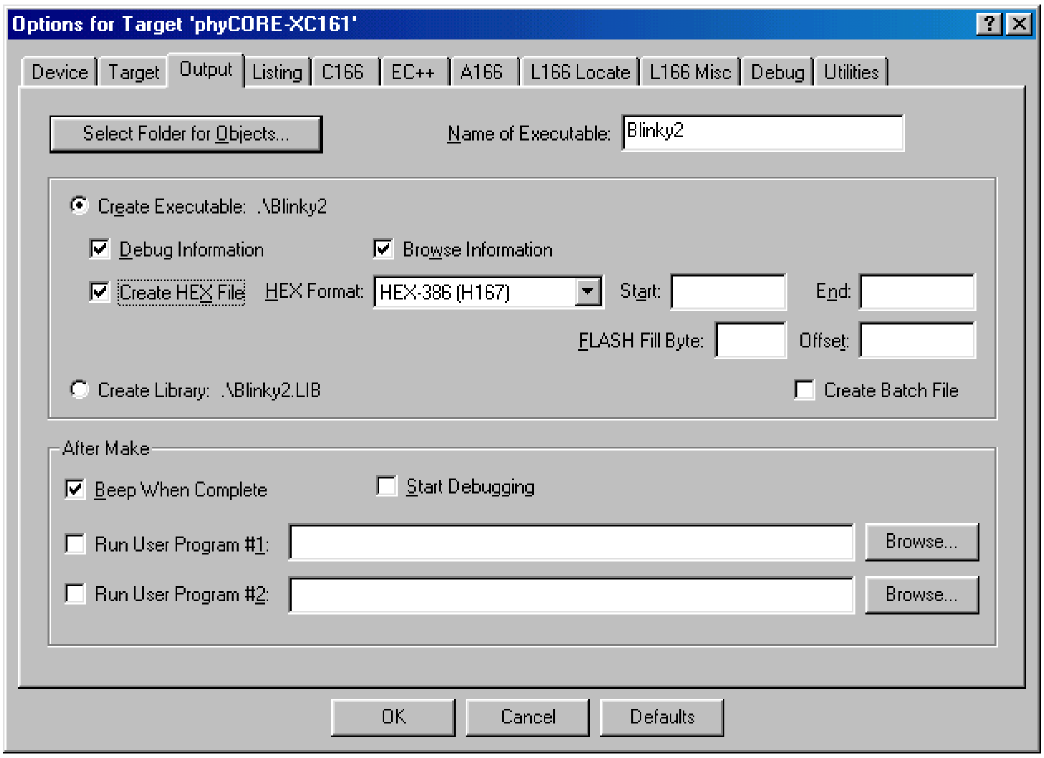
- Type the settings for the *External Memory* as shown below. Make sure that *#1* is set to *ROM*.[1]



[1] : Memory configuration based on standard version of the phyCORE-XC161 as included in the Rapid Development Kit featuring 1 MByte of Flash and 512 kByte of fast SRAM. Modify these values if you are using a different memory configuration.

To configure the Output options:

- Select the **Output** tab and activate the *Create HEX-File* option. With this option a downloadable Intel ****.h86*** file will be created.

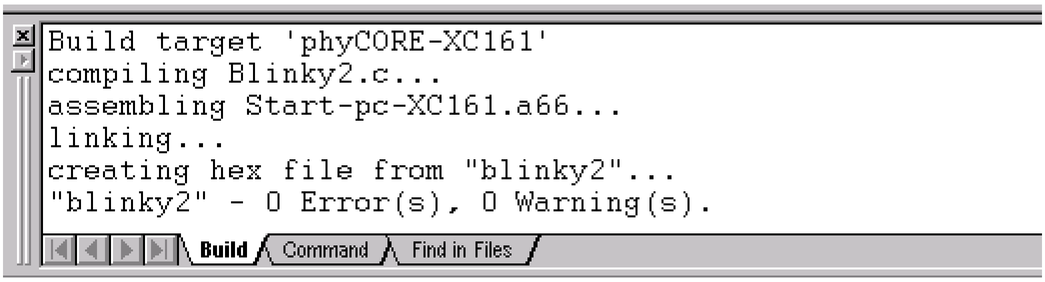


- No other configurations are necessary for this example.
- Click *OK* to save these settings

3.6 Building the Project

You are now ready to run the compiler and linker using the Make utility.

- Click on the *Build Target* icon  from the µVision2 tool bar or press <F7>.
- If the program specified (***Blinky2.c***) contains any errors, they will be shown in an error dialog box on the screen.
- If there are no errors, the code is assembled and linked and the executable code is ready to be downloaded to the board. This is shown in the Output Window, which indicates *"Blinky2" – 0 Errors*, *0 Warnings*. The code to be downloaded to the board will be the name of the project with ***.h86*** as filename extension (in this case ***Blinky2.h86***).



- If a list of errors appears, use the editor to correct the error(s) in the source code and save the file and repeat this section.

3.7 Downloading the Output File

- Ensure that the target hardware is properly connected to the host-PC and a power supply.
- Reset the target hardware and force it into Bootstrap mode by simultaneously pressing the Reset (S2) and Boot (S1) buttons on the phyCORE Development Board HD200 2.5V and then releasing first the Reset and, two or three seconds later, the Boot button.
- Start FlashTools 3.
- At the *Connect* tab of the FlashTools 3 tabsheet, select the phyCORE-XC161 and click the *Connect* button in the lower left corner to establish connection to the target hardware.
- Returning to the FlashTools 3 tabsheet, choose the *Download* tab and click on the *Open* button.
- Choose the *Download* tab, and click on the *Open* button.
- Browse to the correct drive and path for the phyCORE-XC161 Demo folder (default location
  C:\PHYBasic\pC-XC161\Demos\Keil\Blinky2\Blinky2.h86)
  and click *Open.*
- The FlashTools 3 *Download* window will re-appear. Now select the *Erase needed sectors* checkbox in the lower left corner. FlashTools 3 will then erase the required Flash sectors first before downloading the specified hexfile.
- Click on the *Start* button. You can watch the status of the Flash erasure and code download of ***Blinky2.h86*** into the external Flash memory in the lower right corner of the *Download* window.
- The individual steps of the Flash download procedure can be viewed at the bottom of the FlashTools 3 window. Wait until the status check finishes before returning to work with the board. Once you see the *Ready* message in the lower left corner, the downloaded code can be executed.
- Returning to the *Connect* tab, click on the *Disconnect* button and exit FlashTools 3.
- Press the Reset (S2) button on the Development Board.

If the modified hexfile properly executes, the LED should now flash in a different mode with different on and off durations.

You have now modified source code, recompiled the code, created a modified downloadable hexfile, and successfully executed this modified code.

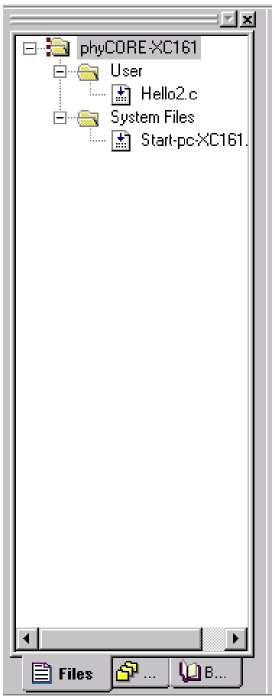
3.8 "Hello"

A return to the "Hello" program allows a review of how to modify source code, create and build a new project, and download the resulting output file from the host-PC to the target hardware. For detailed commentary on each step, described below in concise form, *refer back to the “Blinky2” example starting at section 3.2.*

3.8.1 Creating a New Project

- Open the ***Project*** menu and create a new project called ***Hello2.uv2*** within the existing project folder ***C:\PHYBasic\pC-XC161\Demos\Keil\Hello2*** (default location) on your hard-drive. Select the *Infineon XC161CJ* in the CPU vendor database list.
- Add ***Hello2.c*** and ***Start-pc-XC161.a66*** from within the project directory to the project ***Hello2.uv2***.

- Your project should now look like this:



- Save the project.

At this point you have created a project called ***Hello2.uv2*** consisting of a C source file called ***Hello2.c*** and an assembler file called ***Start-pc-XC161.a66***.

3.8.2 Modifying the Example Source

- Double click the file ***Hello2.c*** from within the project window.
- Use the editor to modify the *printf* command:

```
printf ("\x1AHello World\n")
```

to

```
printf ("\x1APHYTEC... Stick It In!\n")
```

- Save the modified file under the same name ***Hello2.c***.

3.8.3 Setting Target Options

- Open the ***Project|Options for Target...*** menu and change the default settings to the correct values as shown in *section 3.5*.
- Type the settings for the *External Memory* as shown below. Make sure that *#1* is set to *ROM*.

ROM:	Start: 0x000000	Size: 0x100000[1]
RAM:	Start: 0x200000	Size: 0x80000[1]

- Modify the default options for the output file by selecting the *Create HEX File* checkbox in the ***Project|Options for Target....|Output*** tab. This will automatically create a hexfile for download to the phyCORE-XC161 after compiling.

3.8.4 Building the New Project

- Build the project
- If any source file of the project contains any errors, they will be shown in an error dialog box on the screen. Use the editor to correct the error(s) in the source code and save the file and repeat this section.
- If there are no errors, the code is assembled and linked and the executable code is ready to be downloaded to the board.

[1] : Memory configuration based on standard version of the phyCORE-XC161 as included in the Rapid Development Kit featuring 1 MByte of Flash and 512 kByte of fast SRAM. Modify these values if you are using a different memory configuration.

3.8.5 Downloading the Output File

- Ensure that the target hardware is properly connected to the host-PC and a power supply.
- Reset the target hardware and force it into Bootstrap mode by simultaneously pressing the Reset (S2) and Boot (S1) buttons on the phyCORE Development Board HD200 2.5V and then releasing first the Reset and, two or three seconds later, the Boot button.
- Start FlashTools 3.
- At the *Connect* tab of the FlashTools 3 tabsheet, select the phyCORE-XC161 and click the *Connect* button in the lower left corner to establish connection to the target hardware.
- Returning to the FlashTools 3 tabsheet, choose the *Download* tab and click on the *Open* button.
- Choose the *Download* tab, and click on the *Open* button.
- Browse to the correct drive and path for the phyCORE-XC161 Demo folder (default location
  C:\PHYBasic\pC-XC161\Demos\Keil\Hello2\Hello2.h86)
  and click *Open.*
- The FlashTools 3 *Download* window will re-appear. Now select the *Erase needed sectors* checkbox in the lower left corner. FlashTools 3 will then erase the required Flash sectors first before downloading the specified hexfile.
- Click on the *Start* button. You can watch the status of the Flash erasure and code download of ***Hello2.h86*** into the external Flash memory in the lower right corner of the *Download* window.
- The individual steps of the Flash download procedure can be viewed at the bottom of the FlashTools 3 window. Wait until the status check finishes before returning to work with the board. Once you see the *Ready* message in the lower left corner, the downloaded code can be executed.
- Returning to the *Connect* tab, click on the *Disconnect* button and exit FlashTools 3.

3.8.6 Starting the Terminal Emulation Program

- Start the HyperTerminal and connect to the target hardware using the following COM parameters: Bits per second = *9600*; Data bits = *8*; Parity = *None*; Stop Bits = *1*; Flow Control = *None*
- Resetting the Development Board (at S2) will execute the ***Hello2.h86*** file loaded into the Flash.
- Successful execution will send the modified character string ***"PHYTEC... Stick It In!"*** to the HyperTerminal window.
- Click the Disconnect icon .
- Close the HyperTerminal program

You have now modified source code, recompiled the code, created a modified download hexfile, and successfully executed this modified code.

4 Debugging

This Debugging section provides a basic introduction to the debug functions included in the Keil µVision2 evaluation tool chain. Using an existing example, the more important features are described. For a more detailed description of the debugging features, *please refer to the appropriate manuals provided by Keil.*

The µVision2 Debugger offers two operating modes that can be selected in the ***Project|Options for Target phyCORE-XC161*** dialog:

- The **Simulator** allows PC-based microcontroller simulation of most features of the 166/ST10 microcontroller family without actually having target hardware. You can test and debug your embedded application before the hardware is ready. µVision2 simulates a wide variety of peripherals, including the serial port, external I/O, and timers. The peripheral set is selected when you select a CPU from the device database for your target.

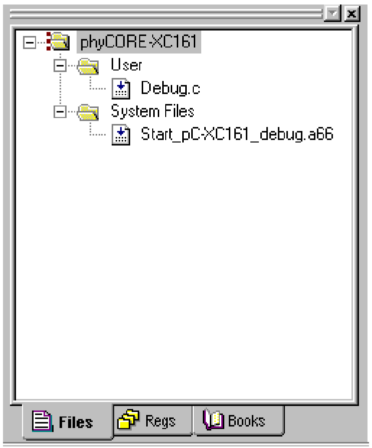
- Advance GDI drivers, like the **Keil Monitor 166** or **OCDS Driver for XC16x** interface, allow target-based debugging. With the Advanced GDI interface you may connect directly to the target hardware. Debugging on the target hardware also enables testing peripheral components of the application.

The following examples utilize the **Keil Monitor 166** environment.

4.1 Creating a Debug Project and Preparing the Debugger

4.1.1 Creating a New Project

- Start the Keil µVision2 environment and close all projects that might be open.
- Open the **Project** menu and create a new project called ***Debug.uv2*** within the existing project directory ***C:\PHYBasic\pC-XC161\Demos\Keil\Debug*** (default location) on your hard drive. Select the *Infineon XC161CJ* in the CPU vendor data base list.
- Rename the target of your project within the **Project Window – Files** tab into *phyCORE-XC161*.
- Rename the file group *Source Group 1* within the **Project Window – Files** tab into *User* and add an additional file group named *System Files.*
- Add ***Debug.c*** to the file group *User* and ***Start_pC-XC161_debug.a66*** to the file group *System Files* from within the project folder.
- Your project should now look like this:



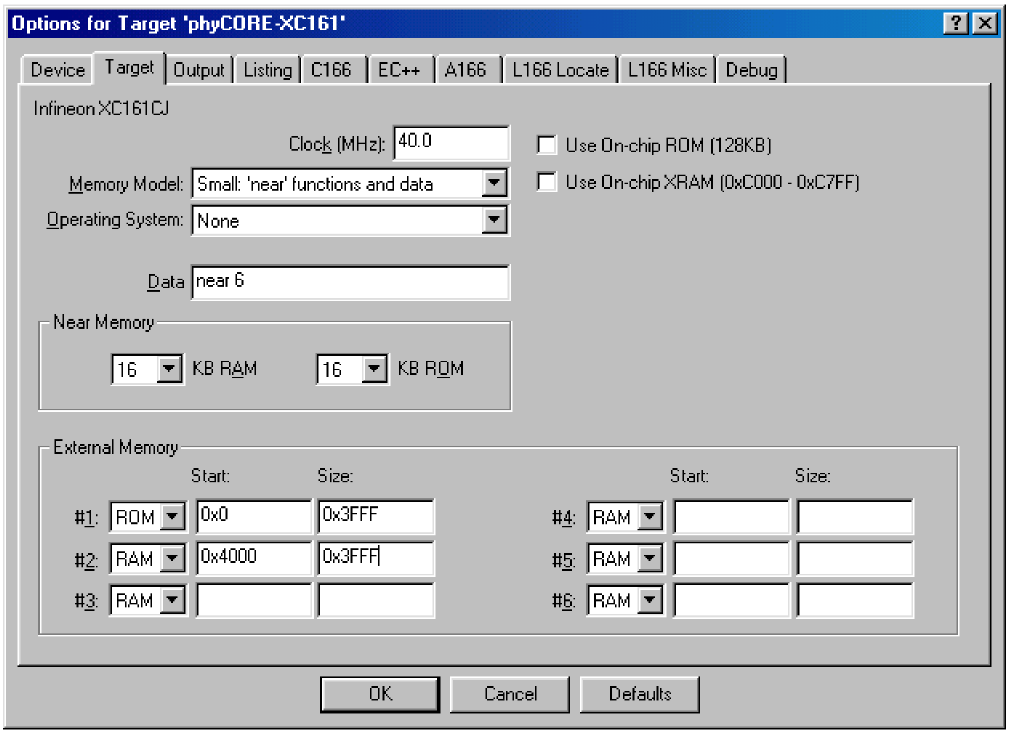
- Save the project.

At this point you have created a project called ***Debug.uv2,*** consisting of a C source file called ***Debug.c*** and the assembler file ***Start_pC-XC161_debug.a66***.

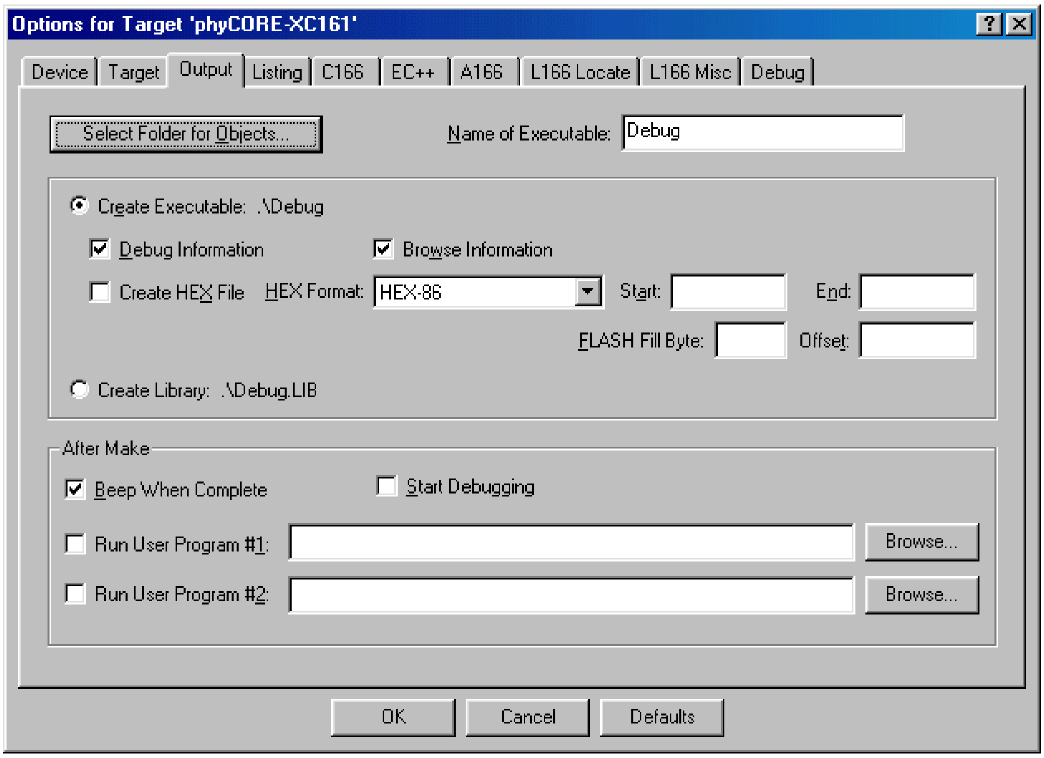
The ***Start_pC-XC161_debug.a66*** startup file differs from the standard startup file as included in the *Blinky(2)* and *Hello(2)* demo projects in terms of its BUSCON1 settings. The Keil monitor maps RAM to the controller's address space starting at address 00:0000h. In contrast, ROM/Flash would be assigned to this address range for all projects that are created for "out-of-Flash" execution. Furthermore, RAM is accessed via /CS1 and mapped to an address range starting at 20:0000h for "out-of-Flash" execution. These address ranges are configured within the ***Start-pC-XC161.a66*** startup file. BUSCON1 is disabled in the ***Start-pC-XC161_debug.a66*** startup file since RAM is already mapped to start address 00:0000h by the Keil monitor (also refer to *Abstract.txt* located in the ***Keil\C166\Monitor\Phytec pC-XC161*** folder for details).

4.1.2 Setting Options for Target

- Open the ***Project|Options for Target 'phyCORE-XC161'*** menu and change the default settings to the correct values as shown in the figure below. This includes settings for the clock frequency of your phyCORE-XC161, the memory model and the off-chip memory.

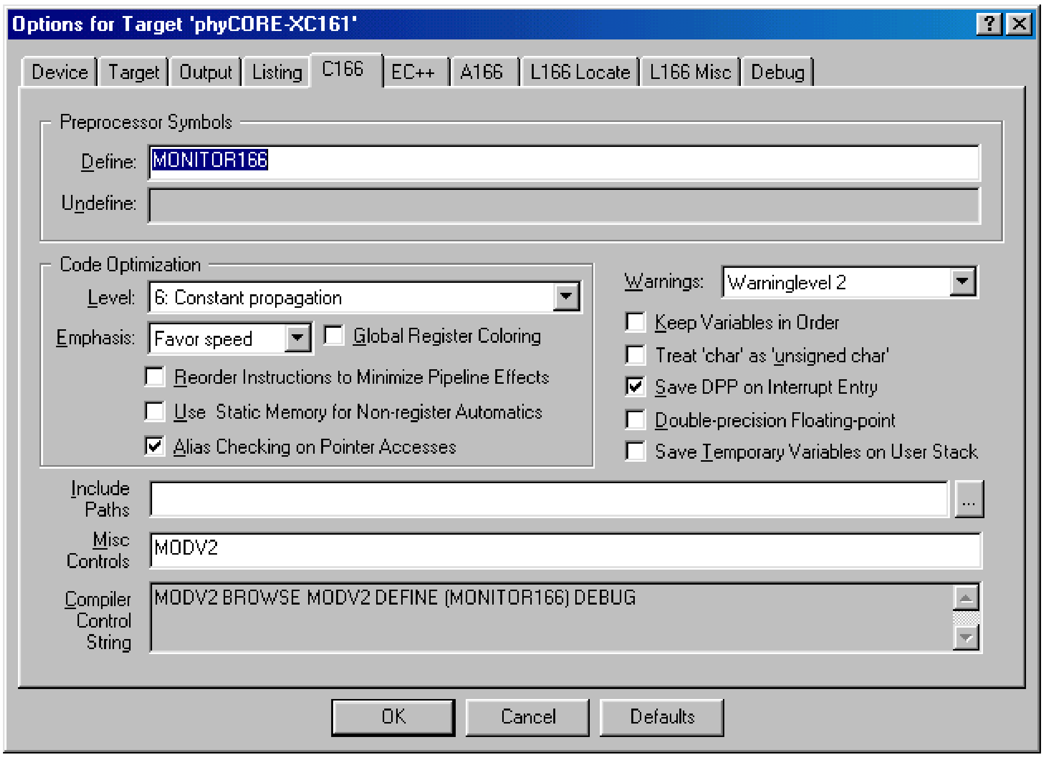


- Select the ***Output*** tabsheet and enable the *Browse Information* checkbox. This will include additional browser information into the object files that can be viewed with the ***Source Browser*** included in Keil µVision2.

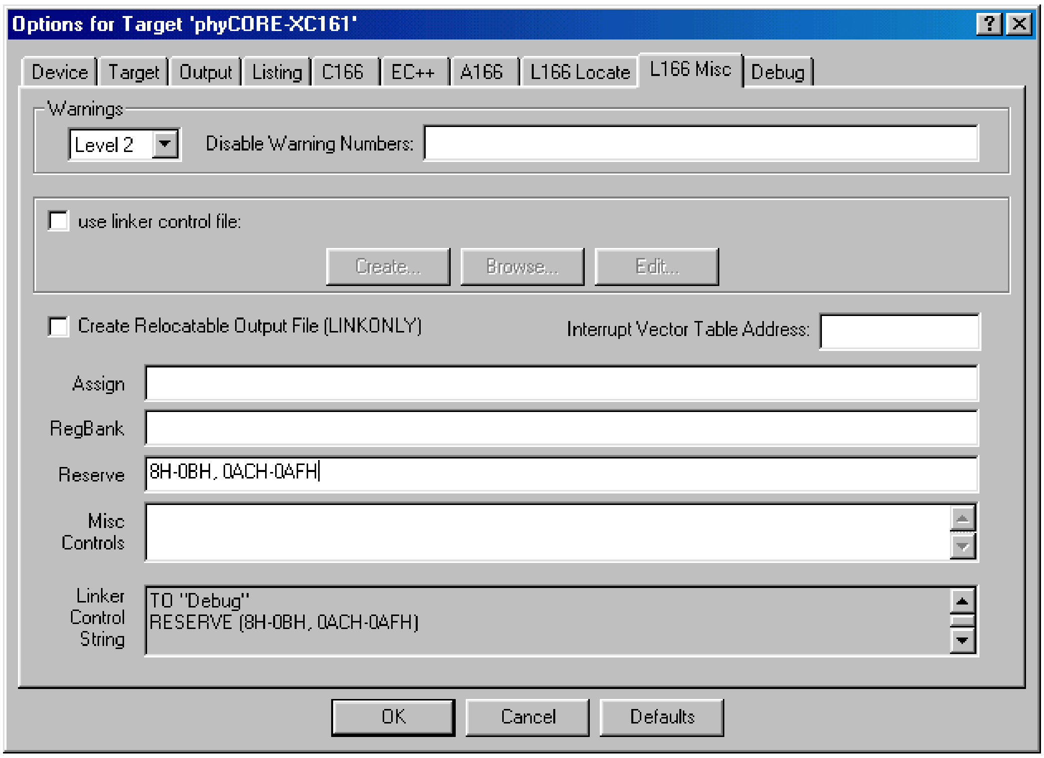


Compiling of the ***Debug.c*** program can be controlled with the two *define* statements **MONITOR166** and **INTERRUPT**. Setting the *define* **MONITOR166** reserves space for the serial interrupt and disables the configuration of the serial interface by the ***Debug.c*** program. This is necessary to avoid overwriting the configuration done by the monitor kernel. The *define* **INTERRUPT** disables all serial output of the ***Debug.c*** program. If debugging is controlled with the serial interrupt, (*see settings for the Keil Monitor-166 Driver in section 4.1.3*) any serial input and output of the user application conflicts with the communication of the monitor program. Such serial communication functions cannot be active in user code to ensure proper debugging.

- Select the ***C166*** tabsheet and add **MONITOR166** in the *Define:* input field. Adjust the settings for *Code Optimization* to the settings shown in the figure below.



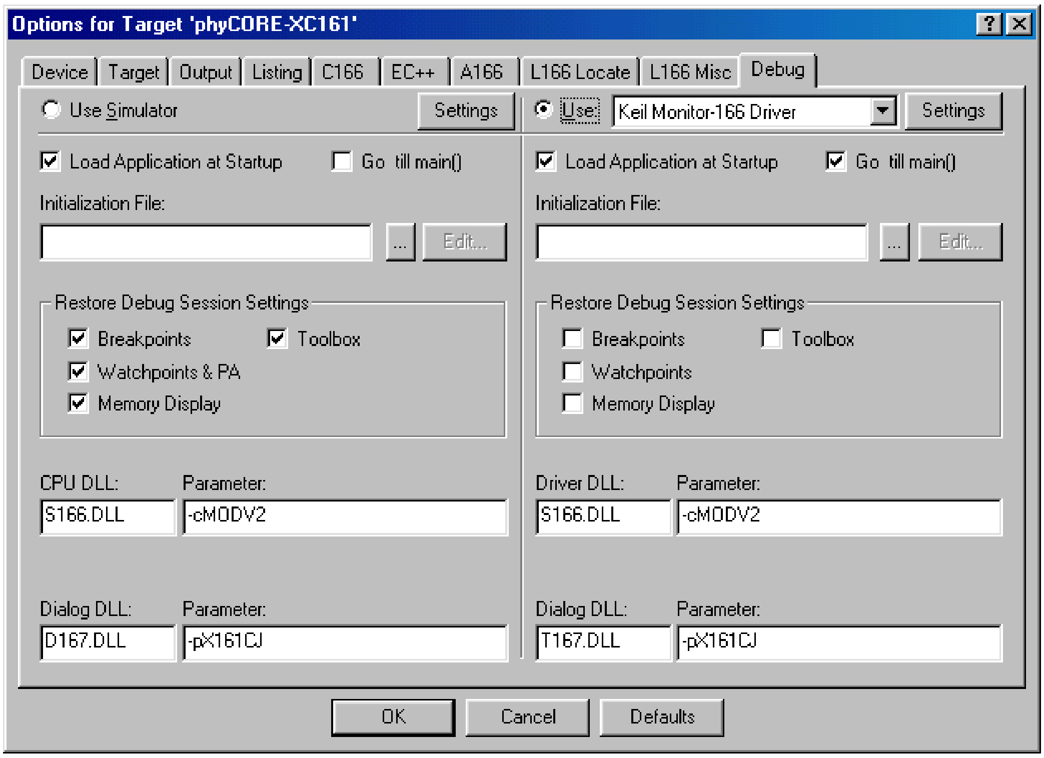
- Select the ***L166 Misc*** tabsheet and add reserved memory areas as shown in the figure below in the *Reserve* input field.



- Ensure that the configurations on the ***EC++***, ***A166*** and ***L166 Locate*** tabsheets are set to their default settings.
- Click the *OK* button to save the settings.
- Click on the **Rebuild all target files** button to compile and link your project.

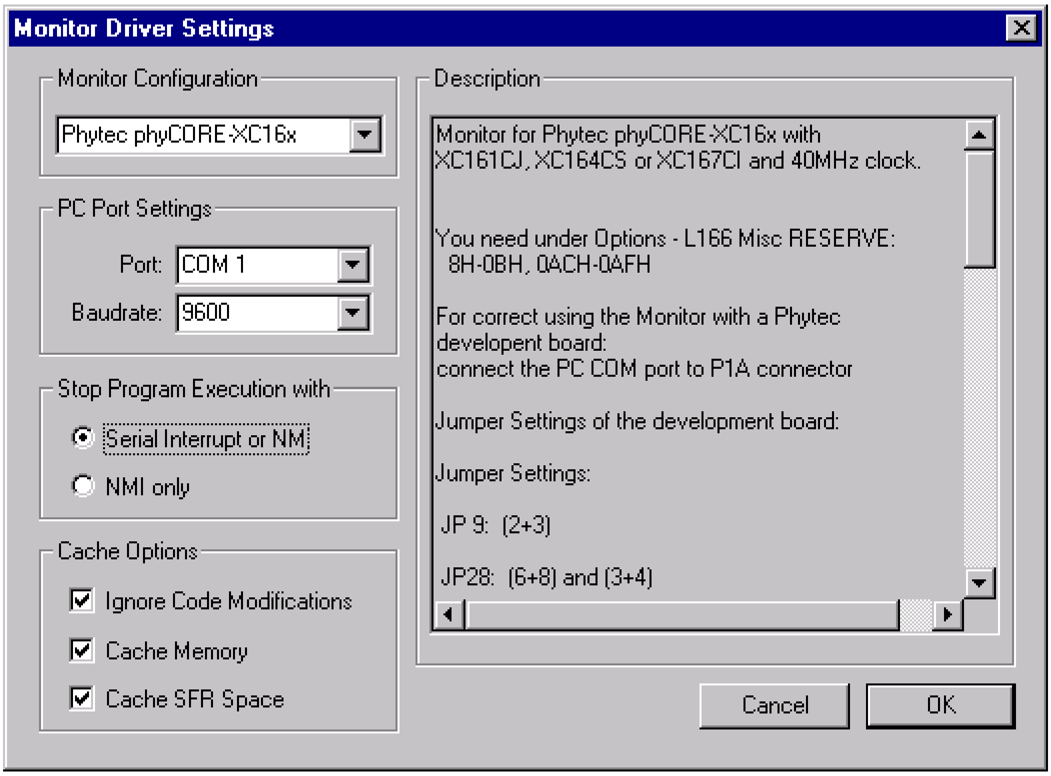
4.1.3 Preparing the Debugger

- Open the ***Project|Options for Target 'phyCORE-XC161'*** menu and select the ***Debug*** tabsheet.
- Enable the checkboxes *Keil Monitor–166 Driver*, *Load Application at Startup* and *Go till main()*.



- Click on the **Settings** button in the upper right-hand corner of the ***Debug*** tabsheet.

- Select *Phytec phyCORE-XC16x* from the *Monitor Configuration* pull-down menu. A short description for the selected module is shown in the *Description* section on the right.
- Select the correct COM *Port* and *Baudrate* in the *PC Port Settings*.



- Click *OK* to save these settings and exit the *Monitor Driver Settings* window.
- The ***Options for Target 'phyCORE-XC161'*** menu will reappear.
- Click on the *OK* button again.

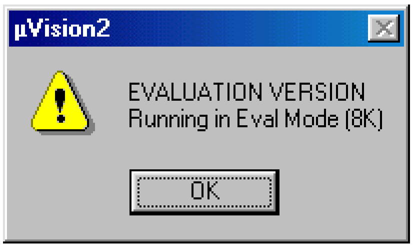
4.2 Preparing the Target Hardware to Communicate with the Debugger

- Ensure that the target hardware is properly connected to the host-PC and a power supply
- Reset the target hardware and force it into Bootstrap mode by simultaneously pressing the Reset (S2) and Boot (S1) buttons on the phyCORE Development Board HD200 and then releasing first the Reset and, two or three seconds later, the Boot button.

Since the required microcontroller portion to communicate with the Keil Monitor 166 will be automatically downloaded using the Bootstrap mode there is no further preparation required for the target system.

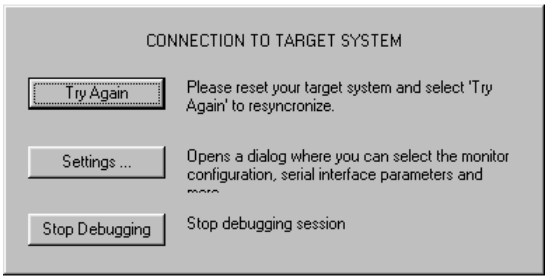
4.3 Starting the Debugger

- To start the µVision2 debug environment, click on the debugger icon  on the µVision2 toolbar. A pop-up window will appear indicating that this is an evaluation version. Click on *OK*.



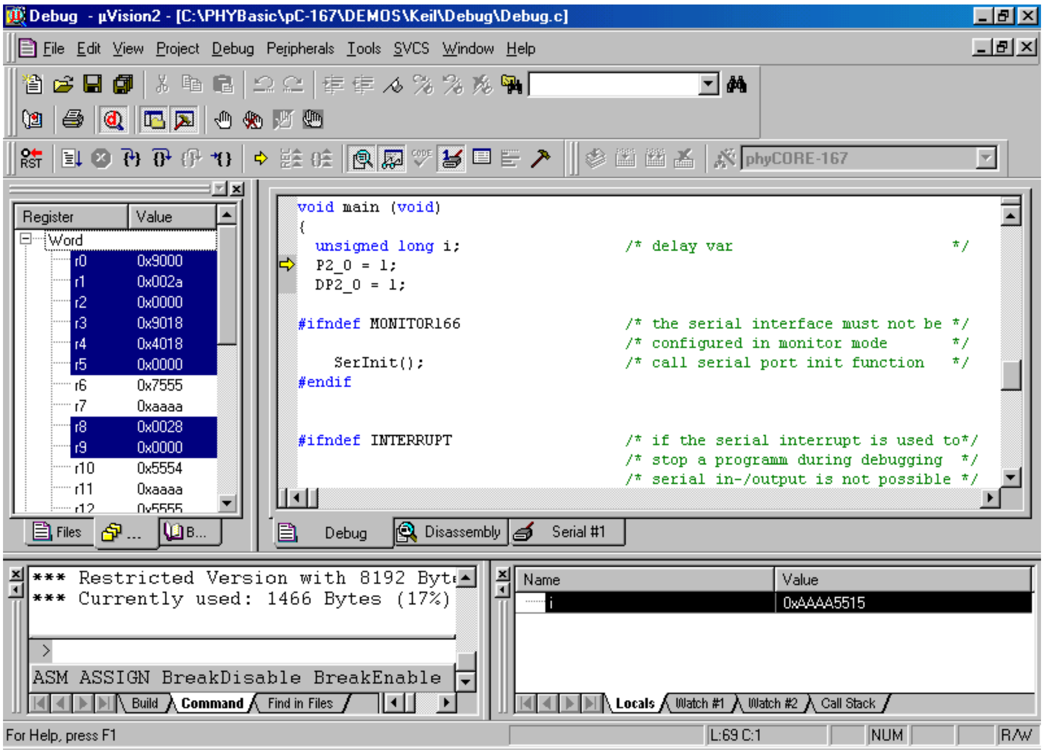
- You will see a blue status bar from left to right at the bottom of your screen indicating the download process of the debug program.

If a problem occurs during data transfer, the following window will appear:



Click on *Settings* and verify the COM port and the baud rate (9600 baud). Reset the target hardware, force it into Bootstrap mode (*refer to section 4.2*) and click on *Try Again*.

If the data transfer was successful, a screen similar to the one shown below will appear. The ***Project*** window changed to the ***Register*** page. The debug toolbar is also displayed. In the lower part of the debug screen you will see the ***Command*** and ***Watch/Stack*** window.

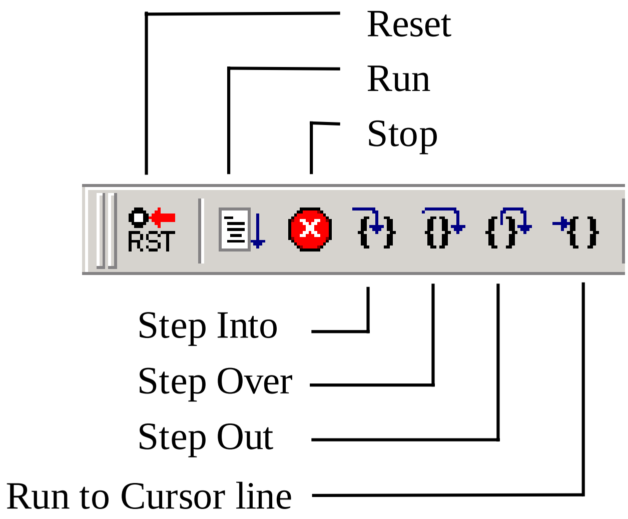


You may need to open, resize and /or move some windows to make your screen look similar to the screen capture. You can open inactive windows by choosing the desired window from the ***View*** pull-down menu. The following screen capture has *Workbook Mode* enabled to allow easy access to various overlapped windows with tabs.

The debugger will run to the *'main'* function and stop automatically. Notice the yellow arrow pointing to the first command in the *'main'* function. Also notice the program counter (**PC $**) within the **Project Window – Register** page showing the start address of the *'main'* function.

4.4 Keil µVision2 Debug Features

- The ***Debugger*** window toolbar gives access to the following debug commands: *Reset*, *Run*, *Stop*, *Step Into*, *Step Over*, *Step Out* and *Run to Cursor line*.



- The first button on the debugger toolbar is the *Reset*  button.

  The *Reset* command sets the program counter to 0.

- The button to the right of the *Reset* button starts the *Run* command.

  Clicking this button runs the program without active debug functions. To stop program execution at a desired point, a breakpoint can be placed before the *Run* button is pushed.

- The next button on the debugger toolbar is the *Stop* button.

  The *Stop* button interrupts and stops the running program at an undetermined location.

- The first button allowing exact control of the program execution is the *Step Into* button.

  The *Step Into* command performs the execution of the command line to which the *Current-Statement Arrow* points. This can be a C command line or a single assembler line, depending on the current display mode. If the command line is a function call, *Step Into* jumps to the C function or subroutine, enabling you to explore the code contained in the accessed subroutine.

- The *Step Over* button is next on the debugger toolbar.

  The *Step Over* command executes the command line, to which the *Current-Statement Arrow* points. This can be a C command line or a single assembler line, depending on the current display mode. If the command line is a function call, the function will be executed without single stepping into the function.

- The next button is the *Step Out* button..

  Step Out is used to exit a function you are currently in. *Step Out* is very useful if you find yourself in a function you are not interested in and need to return quickly to your intended function.

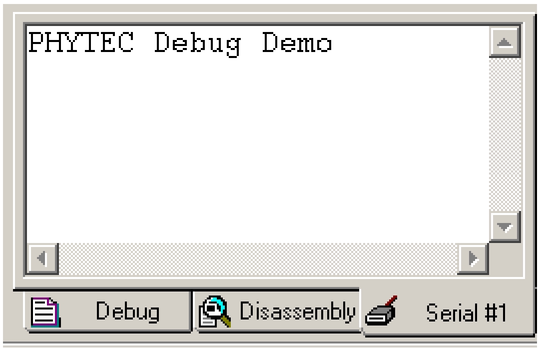
- The last button on the debugger toolbar performs the *Run to Cursor line* command.

  The *Run to Cursor line* command executes the program to the current cursor position within the code window. This allows use of the cursor line as a temporary breakpoint.

4.5 Using the Keil µVision2 Debug Features

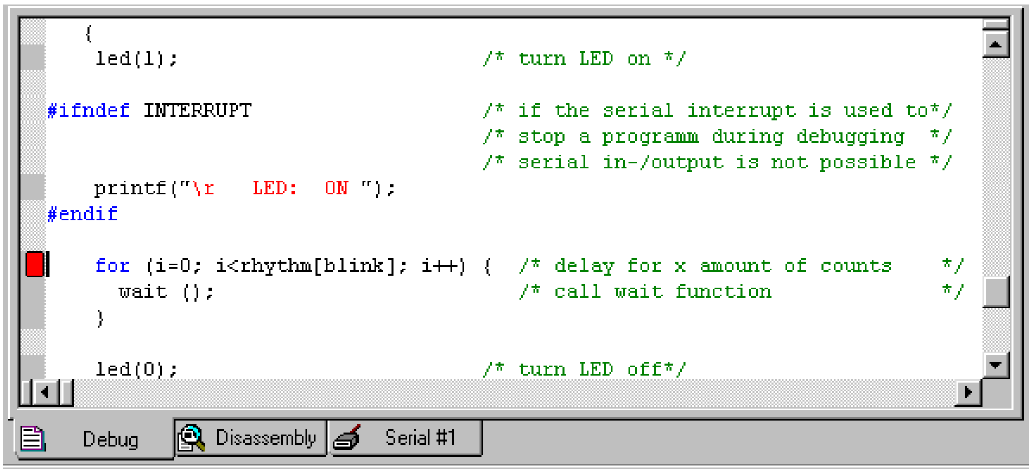
4.5.1 Serial Window

- Click in the source code line with the first **printf** command and choose ***Run to Cursor line*** from the debug toolbar. Your program will be executed until it reaches this line.
- Click on the *Step Into*  button. The **printf** command will be executed and the serial output will appear in the ***Serial #1*** window of the debugger.



4.5.2 Breakpoints

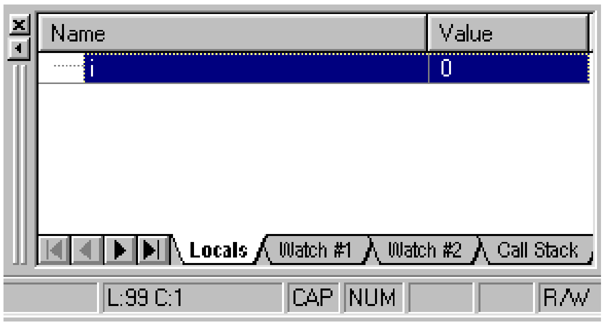
- Click in the line `for (i=0; i<rhythm[blink]; i++) {`.
- Click on *Insert/Remove Breakpoint* to set a breakpoint here. The red marker on the left-hand side of the selected line indicates the breakpoint. You can also set a breakpoint by double-clicking in the desired code line.



- Click on the *Run* icon and the program will run and stop at the breakpoint.
- Notice that the LED (D3) on the Development Board now illuminates. This is because the *led(1)* function call has been executed and the status of the LED is shown in the ***Serial*** window.
- Click again on *Insert/Remove Breakpoint* to remove the breakpoint.

4.5.3 Single Stepping and Watch Window

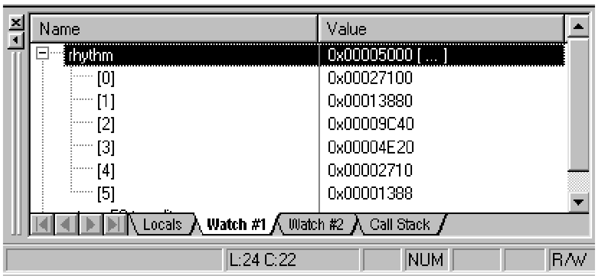
- Click on the ***Step Into*** icon to enter the *'for'* loop.
- The ***Watch*** window automatically shows the value of the local variable *i*. In order to change the number base from hexadecimal to decimal, right-click on the variable.



- Click ***Step Over*** several times and watch the value of *i* count up.
- As you can see in the source code, the for{} loop will end if *i* becomes equal to the first element of the constant field *rhythm[]* which has the value of 160,000. To leave the *wait* function, change the value of *i* by typing ***i=159996*** in the command line and pressing <Enter>. Now repeat clicking on ***Step Over*** until you leave the wait function.

- Click in the source code line *blink++* and choose ***Run to Cursor line*** from the debug toolbar. Your program will be executed until it reaches this line.
- Notice that the LED D3 on the Development Board is off now and the new status of the LED is shown in the ***Serial*** window.

- As a last example, the constant "**rhythm[]**" will be evaluated. Go to the source code line where the constant "**rhythm[]**" is declared. Right-click on "**rhythm[]**" and choose the **"ADD rhythm to watch window" -> #1** option. Select the "**Watch #1**" tabsheet at the bottom of the watch window. The constant is shown with its address and a small ⊞ sign in front which indicates that "**rhythm[]**" is an array with a group of array elements. Click the ⊞ sign to expand the view and to see all array elements of "**rhythm[]**".



4.6 Running, Stopping and Resetting

- To run your program without stopping at any time, delete all breakpoints by clicking on the  button.
- Click the ***Run***  button.

The LED now blinks and its current status is displayed in the ***Serial*** window.

You can use of the ***Stop***  button to stop program execution at any time.

4.7 Resetting the Debugger and the phyCORE-XC161

Debugger in Monitor mode:

The monitor kernel runs on the target hardware. A debugger reset sets the IPC to zero and performs other initialization routines if no user application was started. This type of reset is not as complete as a hardware-reset (pressing push button S2 on the Development Board). The best method to stop a running application is to click on the *Halt* button rather than using the *Reset CPU* in monitor mode. *Halt* tries to stop a running application when the *'Serial interrupt or NMI'* option is enabled. If this option is not enabled, a dialog box is displayed in which you can select further options. This has the advantage of detecting any 'infinite loop' in which your program might be stuck. With reset you start again at address 0.

Debugger in Simulator mode:

When you are using the simulator (i.e. no connection to target hardware), pressing the *Reset CPU* icon does not cause a running simulation to stop at the current point of execution. *Reset CPU* starts the application from the start address (0x0) again. Press the *Halt* button to halt a program under normal conditions.

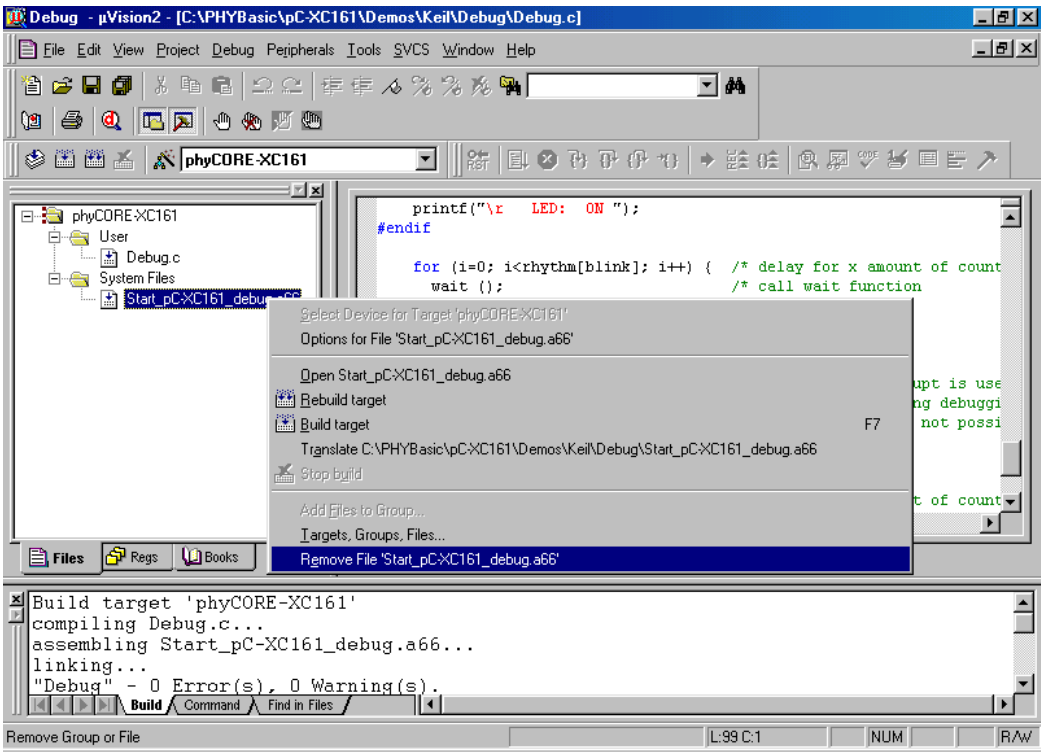
Hardware NMI:

The phyCORE Development Board HD200 does not provide a hardware NMI button. A NMI is the most reliable way to stop a running program. This is even more important when you are using the Bootstrap version of the monitor because it is difficult to re-synchronize the debugger and the kernel. For example, it is not possible for the debugger to re-synchronize automatically by pressing the hardware reset button (S1) if the monitor has been downloaded with the Bootstrap Loader. It must be restarted manually by restarting the debugger.

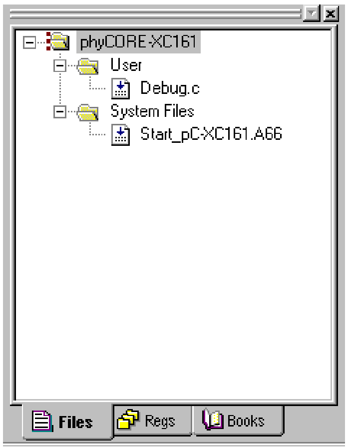
4.8 Changing Target Settings for the "Executable Version"

After successfully debugging the program, next change the project and the target settings in order to create an Intel hexfile. This can then be downloaded to and executed out of the Flash memory on the phyCORE-XC161.

- Exit the current debug session by selecting ***Debug\Start/Stop Debug Session***.
- Within the **Project Window – Files** tab remove the startup file that was used for the debug session as described in *section 4.1.1*. First highlight the ***Start_pC-XC161_debug.a66*** file. Then right-click in the **Project Window – Files** to open a new window. Choose the option ***Remove File 'Start_pC-XC161_debug.a66'***.

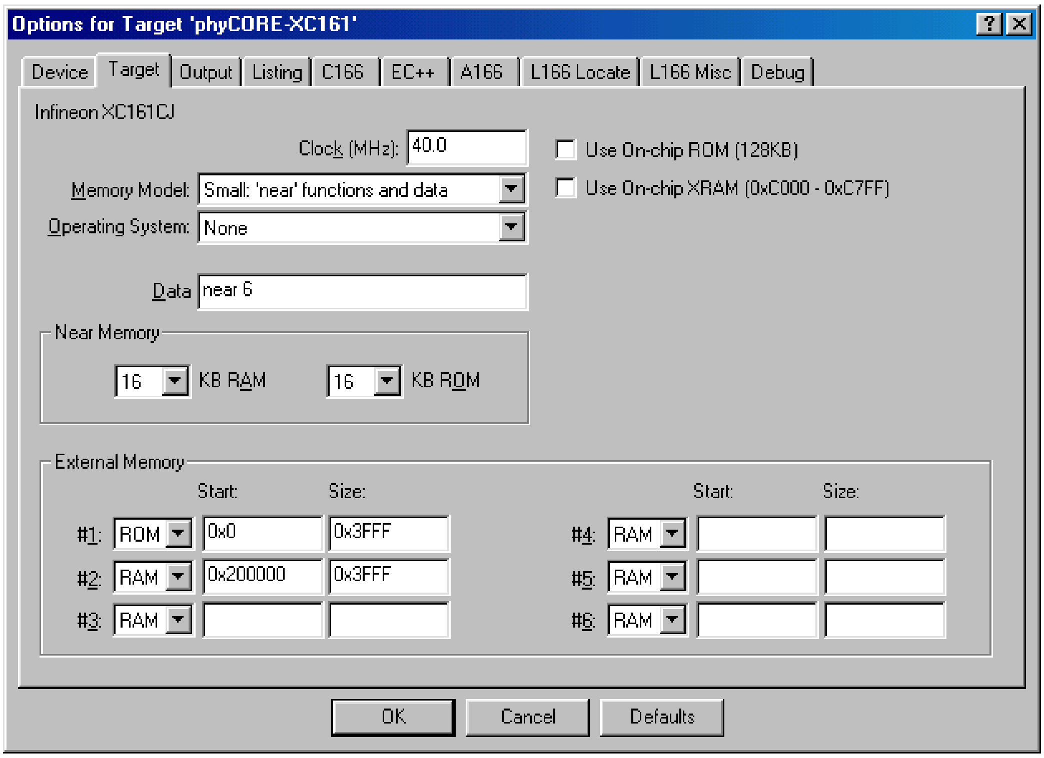


- Now add the ***Start_pC-XC161.a66*** to the file group *System Files*. This will add the applicable startup code to the debug demo when running out of Flash as opposed to debugging code in RAM as described in *sections 4.1 through 4.6. Refer back to section 4.1.1 for more details on the different startup files.*
- Your project should now look like this:

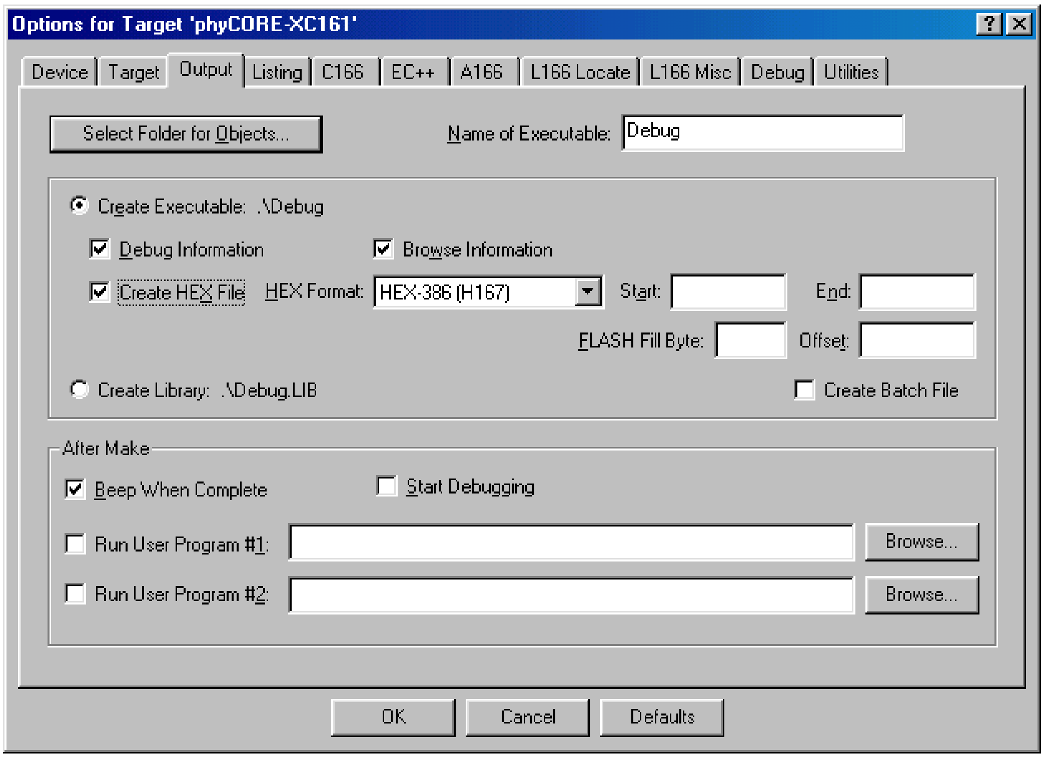


- Save the project.

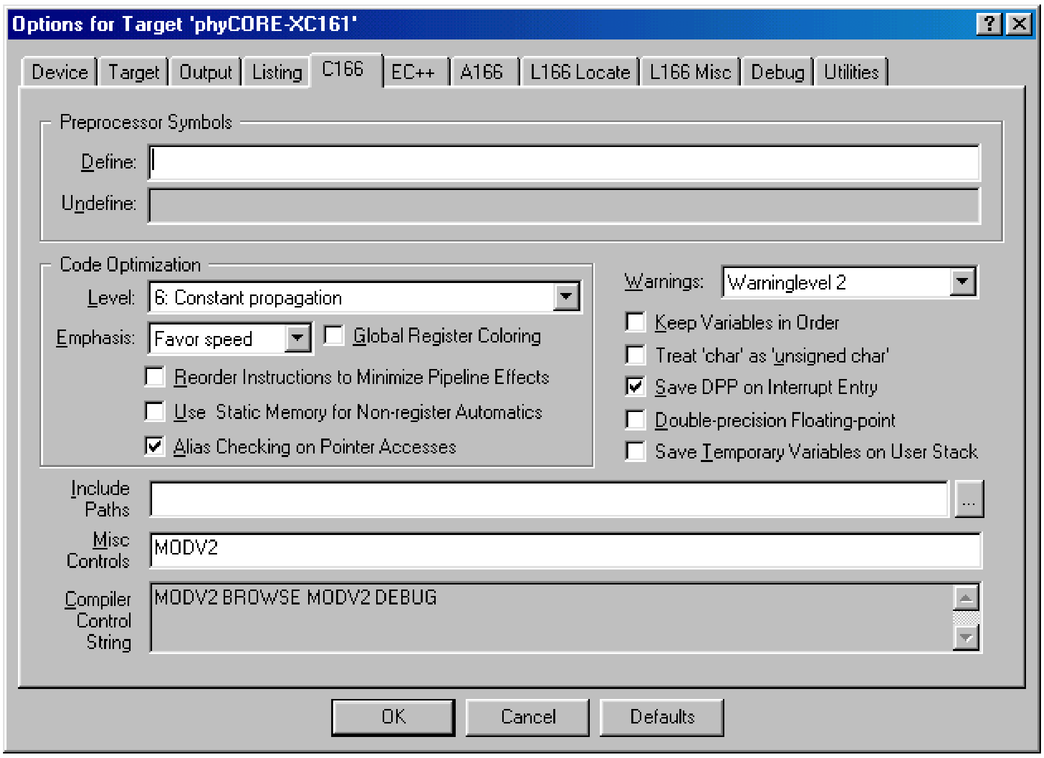
- Open the ***Project|Options for Target 'phyCORE-XC161'*** menu and change the settings for off-chip memory to the new values as shown in the figure below. The start address for RAM, accessed via Chip Select signals /CS1, is 0x200000 which is also configured in the BUSCON1 register *(see **start_pC-XC161.a66** file)*.



- Select the ***Output*** tabsheet and enable the checkbox *Create HEX File*.



- Go to the ***C166*** tabsheet and erase all defines from the *Define:* input field and adjust the optimization to your needs.



- Click on the *OK* button to save the settings.
- Click on the *Rebuild all target files* icon to compile and link your project.
- Download the created ***Debug.h86*** file (located in ***C:\PHYBasic\pC-XC161\Demos\Keil\Debug***) to the Flash memory. *For general download procedure information refer to sections 2.2 through 2.4.1.*
- Exit FlashTools3.
- Start the HyperTerminal program as described in *section 2.4.2*.
- Press the Reset button S1 on the Development Board to start the program.
- Now you can watch your final debug example execute.

5 Advanced User Information

This section provides advanced information for successful operation of the phyCORE-XC161 with the µVision2 software tool chain.

5.1 FlashTools 3

Flash is a highly functional means of storing non-volatile data. One of its advantages is the possibility of on-board programming. Flash programming tools for the phyCORE-XC161 are provided in the form of executable and binary files that run under Windows 9x/Me/NT/2000/XP. These tools make use of the Bootstrap mode to transfer executable code to the phyCORE-XC161 that, in turn, download user code into the Flash. Additionally, the re-programmable Flash device on the phyCORE-XC161 enables easy update of user code and the target application in which the phyCORE-XC161 has been implemented.

Currently the phyCORE-XC161 can be populated with four different sized Flash devices: a 29F200 with 256 kByte, a 29F400 with 512 Byte, a 29F800 with 1 MByte or a 29F160 with 2 MByte.

FlashTools 3 always uses the Bootstrap mode to transfer the required microcontroller firmware to the phyCORE-XC161. Hence, there is no restriction in using the Flash memory for storing user code.

Before Flash programming FlashTools 3 will adjust the internal Chip Select Unit and the bus configuration of the microcontroller during the Bootstrap download. This results in the following programming memory model:

- Flash bank using /CS0 addressable at 000000H-1FFFFFH (up to 2 MByte Flash)
- RAM bank 1 using /CS1 addressable at 200000H-27FFFFH (up to 512 KByte RAM)
- RAM bank 2 using /CS3 addressable at 300000H-3FFFFFH (up to 1 MByte RAM)

Using the Bootstrap mode to transfer the required microcontroller firmware to the phyCORE-XC161 ensures that FlashTools 3 maintains its independent Flash programming capability.

5.2 Start-pc-XC161.a66

The code within the assembly file ***start-pc-XC161.a66*** is responsible for the initial controller configuration and the startup initialization of your C project. This includes adjusting the properties of the external bus signals and Chip Select signals, setting of the system stack, initialization of variables and clearing of memory areas.

It is very important that this code will execute prior to the execution of user code. To ensure this, it is recommended that the startup code occupies the reset vector of the application, which is the location where the microcontroller starts execution after reset (0x0000). After performing the initialization steps your individual *main()* function is called by the startup code.

Since some of the settings are hardware-dependent, we recommend use of the prepared ***start-pc-XC161.a66*** from within the default location ***C:\PHYBasic\pC-XC161\Tools\Startup\Keil***. The properties of the external bus interface are already configured for the phyCORE-XC161. You may want to change the values for the Chip Select Unit.

To accommodate the startup code to the needs of your application copy it from the directory described above to your project directory. You can then edit, modify and compile it using the Keil macro-assembler.

Since the startup code will usually be implicitly taken into consideration from the default runtime libraries, you must now explicitly tell your linker to instead consider your individual startup object file. To do this we recommend adding your modified ***start-pc-XC161.a66*** file to your project. Be sure that it is always included in the Link/Lib process of your project (see options within the *Project* window of the Keil tool chain).

5.3 Linking and Locating

The Linker has to combine several re-locatable object modules contained in object files and/or libraries to generate a single absolute object.

In addition the Linker must locate several segments of code type, constants and data to fixed address locations within the address range of the microcontroller: This ensures the natural or explicitly declared properties of these segments.

Data segments must always be located in Random Access Memory (e.g. RAM), code and constant segments should be located in any kind of non-volatile memory (e.g. Flash). The C166 family has a von Neumann architecture which uses the same read signal to fetch data and also code or constants. To distinguish between non-volatile and modifiable memory, physically different memory devices must be addressable within different address ranges.

To enable easy accommodation of the linking process the Linker collects segments of equal type to classes.

The Keil tool chain distinguishes the following classes:

*x*CODE:	code for several addressing modes (*x* = N, I, F, H or X)
*x*DATA:	not initialized data for several addressing modes (*x* = N, I, F, H or X)
*x*DATA0:	pre-initialized data for several addressing modes (*x* = N, I, F, H or X)
*x*CONST:	constant for several addressing modes (*x* = N, I, F, H or X)

It is required that all *x*DATA and *x*DATA0 classes segments are located to any internal RAM of the XC161CJ or to any external RAM on the phyCORE-XC161.

All *x*CODE and *x*CONST classes must also be located to any internal non-volatile memory (e.g. Flash, OTPROM) of the XC161CJ or the external Flash memory of the phyCORE-XC161.

A near address data area (NDATA, NDATA0) must reside in one data page (16 kByte). A near address code area (NCODE, NCONST) must reside in one code segment (64 kByte).

To ensure proper execution of your application you must take the runtime memory model into account when linking and locating. This means that you must instruct the Linker where to assume external RAM for locating data classes and Flash for locating code and constant classes.

The recommended operating mode of the phyCORE-XC161 allows the use of the Chip Select Unit of the XC161CJ to define the physical memory layout. By modifying the file ***start-pc-XC161.a66*** as part of your project you can adapt the memory layout to your needs.

The external use of the Chip Select signals is predefined by the hardware in the following way:

- Flash Bank uses /CS0 (up to 2 MByte Flash)
- RAM Bank uses /CS1 (up to 512 kByte RAM)

The default configuration of the phyCORE-XC161 has 1 MByte Flash (/CS0) and 512 kByte fast SRAM (/CS1) (order code PCM-020-0020E).

Caution:
You will see multiple mirrors of a memory device that has a physical smaller address range than the associated address range of the Chip Select signal. For instance if you adjust Chip Select signal /CS1 to be active within an address range of 1 MByte and the actual memory size populating the phyCORE is just 512 kByte, you will get one mirrored image of your RAM.

We recommend that you generate a ***.m66*** map file for your project and inspect the memory map information within this file. Compare this information with the physical memory model resulting from your settings selected within the ***start-pc-XC161.a66*** file.

5.4 Debugging Using Monitor Kernel

Whenever you decide to use the µVision2 target debugger or Mon166 target Monitor to debug your application, some special precautions must be taken into consideration to ensure proper code execution of your application.

Your application and the Keil Monitor kernel contained in the files ***boot*** and ***monitor*** must share some memory locations within the target system. If you do not consider the physical Memory-Model already claimed by the kernel and the memory requirements of the kernel, you may get conflicts in memory use. This typically leads to variables containing not their assigned value, functions returning bad results and modified code.

We recommend the use of the ***start-pc-XC161_debug.a66*** within the default destination ***C:\PHYBasic\pC-XC161\Tools\Mon\Keil*** if you want to debug your application using the Monitor kernel. This file will adjust the external bus properties and the Chip Select Unit in exactly the same manner as did the Monitor kernel.

To obtain information about the memory requirements of the Monitor, the corresponding memory map file ***monitor.m66*** is made available together with the ***monitor*** executable file. This file contains a detailed memory map of the Monitor and is also located in the default destination mentioned above.

You must link your application to prevent any overlapping memory ranges. Since the Monitor also uses some special interrupts for communication with the host-PC at runtime, you should add a *Reserve:* statement for 0x008 – 0x011, 0x0AC – 0x0AF to the *L166Misc* tab of your *Options for Target* option to reserve at least these ranges.

You should always ensure that segments of your application do not reach or overlap the segments of the Monitor. The Monitor segments will usually be linked to the top of the memory, leaving you the most possible memory space for the application code.

The Monitor is linked under the assumption of maximum memory upgrading for Flash bank and RAM bank. Remember that you will have multiple additional mirrors of the physical devices actually mounted on the phyCORE-XC161 if their capacity is less than the maximum value of 1 MByte.

For instance if you have 512 kByte of RAM mounted on the phyCORE-XC161 you will have one additional mirror image of the RAM within the reserved 1 MByte range. Note that in this case all associated address ranges of 0x80000 – 0xFFFFF will actually address the same physical device address range of 0x00000 – 0x7FFFF. This means that exactly the same physical memory location can be addressed using two different internal addresses. This must be taken into consideration when verifying your memory mappings.

5.5 Demo for the Optional Ethernet Controller CS8900A-CQ

As an option, the Crystal LAN CS8900A-Q Ethernet Controller can populate the phyCORE-XC161 at position U16. The PHYTEC Spectrum-CD, included in the phyCORE-XC161 Rapid Development Kit, contains an example program that demonstrates the use of a tiny webserver. This example program can be found in the folder:

C:\PHYBasic\pC-XC161\Demos\Keil\easy-WEB)

This tiny web server was taken from the magazine *'Design & Elektronik'*, special issue *'Embedded Internet'*. It can be downloaded from the following web site: ***www.elektroniknet.de/extraheft***. This software has been modified to work with a PHYTEC phyCORE-XC161 board and the Keil C166 C Compiler.

All modifications in the source code are marked with *'// Keil:'* and *'//Phytec:´*.

The web page shows the values of two analog inputs (AN0 and AN1). This tiny webserver needs very few resources and therefore has some restrictions:

- only one active TCP session at a given time
- no support for fragmented IP datagrams
- no buffer for TCP datagrams received in wrong order
- only one web page, no GIF/JPG graphics possible

The IP address can be modified in the *tcpip.h* header file in order to match your existing LAN (see MYIP_x).

The easyWEB project is set up for two different targets:

- pC-XC161: Settings for PHYTEC phyCORE-XC161 module
- Simulation: Settings for simulation.

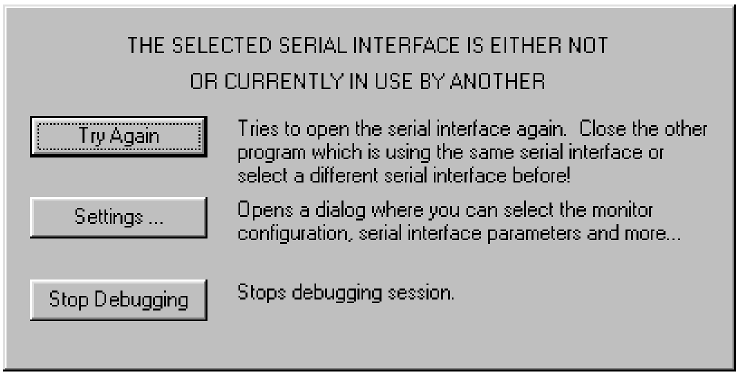
Note:
The Ethernet controller cannot be simulated.

A Appendices

A1 Troubleshooting

A1.1 µVision2 Debugger in Monitor Mode

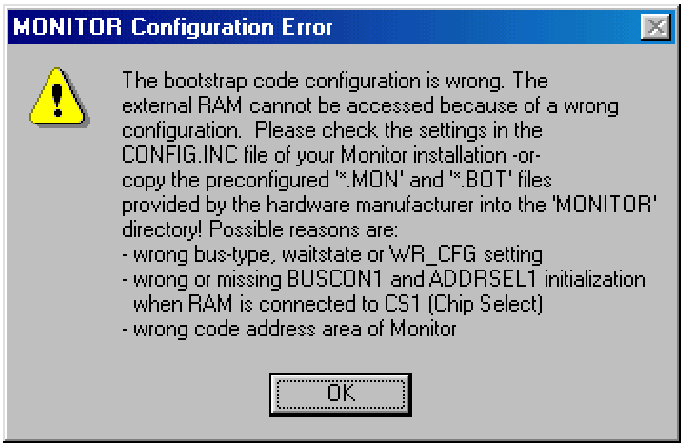
If µVision2 communication to the target system is not successful, an error window will appear as shown below. If this should occur, check if the correct serial port is selected or try other baud rates. The correct transmission rate is 9600 baud. Click on *Try Again* after correcting the communication parameters and forcing the target system into Bootstrap mode again.



The serial FIFO buffer in MS Windows95 can cause transmission problems. µVison2 debugger may have problems completing the communication initialization process. This can be intermittent. The FIFO can be disabled under *Settings/Control Panel|System|Device Manager|Port Settings|Advanced.* Make sure *Use FIFO buffers* is not activated in this menu.

A.2 Monitor Configuration Error

If the Monitor 166 kernel download is not successful, an error window "*MONITOR Configuration Error*" will appear as shown below. If this should occur, make sure that the correct target hardware is selected under *Options for Target|Debug|Settings|Monitor configuration*. This step is crucial to ensure proper communication between the target hardware and debug environment.



Document: phyCORE-XC161 QuickStart Instructions
Document number: L-646e_1, July 2003

How would you improve this manual?

Did you find any mistakes in this manual? page

Submitted by:
Customer number: ______________________

Name: ______________________

Company: ______________________

Address: ______________________

____

Return to:

PHYTEC Technologie Holding AG
Postfach 100403
D-55135 Mainz, Germany
Fax : +49 (6131) 9221-33

Published by

PHYTEC

Ordering No. L-646e_1
Printed in Germany