

phyCORE-ST10HS/E with ST10F269

QuickStart Instructions

**Using ST Microelectronics' ST10 Flasher and the Keil μ Vision2
Software Evaluation Development Tool Chain**

Note: The PHYTEC Spectrum CD includes the electronic version of
the phyCORE-ST10HS/E English Hardware Manual

Edition: March 2003

A product of a PHYTEC Technology Holding company

In this manual are descriptions for copyrighted products that are not explicitly indicated as such. The absence of the trademark (™) and copyright (©) symbols does not imply that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this manual.

The information in this document has been carefully checked and is believed to be entirely reliable. However, PHYTEC Meßtechnik GmbH assumes no responsibility for any inaccuracies. PHYTEC Meßtechnik GmbH neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product. PHYTEC Meßtechnik GmbH reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages which might result.

Additionally, PHYTEC Meßtechnik GmbH offers no guarantee nor accepts any liability for damages arising from the improper usage or improper installation of the hardware or software. PHYTEC Meßtechnik GmbH further reserves the right to alter the layout and/or design of the hardware without prior notification and accepts no liability for doing so.

© Copyright 2003 PHYTEC Meßtechnik GmbH, D-55129 Mainz.

Rights - including those of translation, reprint, broadcast, photomechanical or similar reproduction and storage or processing in computer systems, in whole or in part - are reserved. No reproduction may occur without the express written consent from PHYTEC Meßtechnik GmbH.

	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: March 2003

1	Introduction to the Rapid Development Kit	5
1.2	Rapid Development Kit Documentation	5
1.3	Overview of this QuickStart Instruction.....	6
1.4	System Requirements	7
1.5	The PHYTEC phyCORE-ST10HS/E.....	8
1.6	The Keil μ Vision2 Software Development Tool Chain.....	11
2	Getting Started	15
2.1	Installing Rapid Development Kit Software	15
2.2	Installing ST Microelectronics ST10 Flasher.....	21
2.3	Interfacing the phyCORE-ST10HS/E to a Host-PC.....	22
2.4	Starting ST Microelectronics' ST10 Flasher.....	24
2.5	Downloading Example Code with ST10 Flasher	25
2.5.1	"Blinky"	27
2.5.2	"Hello"	30
3	Getting More Involved.....	35
3.1	Starting the μ Vision2 Tool Chain.....	35
3.2	Creating a New Project and Adding an Existing Source File	36
3.3	Modifying the Source Code.....	43
3.4	Saving the Modifications.....	44
3.5	Setting Options for Target	44
3.6	Building the Project	47
3.7	Downloading the Output File	48
3.8	"Hello2"	49
3.8.1	Creating a New Project.....	49
3.8.2	Modifying the Example Source.....	50
3.8.3	Setting Target Options.....	50
3.8.4	Building the New Project	50
3.8.5	Downloading the Output File	51
3.8.6	Starting the Terminal Emulation Program	52

4	Debugging	53
4.1	Creating a Debug Project and Preparing the Debugger	54
4.2	Preparing the Debugger.....	55
4.3	Preparing the Target Hardware to Communicate with the Debugger	56
4.4	Starting the Debugger.....	57
4.5	Using the Keil μ Vision2 Debug Features	59
4.5.1	Breakpoints.....	59
4.5.2	In Line Assembler	60
4.6	Single Stepping.....	61
4.6.1	Memory Window	62
4.6.2	Watch Window.....	62
4.7	Resetting Simulator and the phyCORE-ST10HSE	64
5	Advanced User Information.....	65
5.1	Start-pcST10HSE.a66	65
5.2	Linking and Locating	66
5.3	Debugging Using Monitor Kernel.....	68
5.4	Demo for the Optional Ethernet Controller CS8900A-CQ.....	70

Index of Figures

Figure 1:	phyCORE Development Board HD200 Overview.....	22
Figure 2:	Default Setting for the phyCORE Development Board HD200	23
Figure 3:	Power Connector	23

1 Introduction to the Rapid Development Kit

This QuickStart provides:

- general information on the PHYTEC phyCORE-ST10HS/E Single Board Computer (SBC),
- an overview of Keil's μ Vision2 software development tool chain evaluation version, and
- instructions on how to run example programs on the phyCORE-ST10HS/E, mounted on the PHYTEC phyCORE Development Board HD200, in conjunction with Keil software tools

Please refer to the [phyCORE-ST10HS/E 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-ST10HS/E Hardware Manual.

1.2 Rapid Development Kit Documentation

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

- the PHYTEC [phyCORE-ST10HS/E 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-ST10HS/E in conjunction with the Keil μ Vision2 software development tool chain evaluation version

1.3 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-ST10HS/E in conjunction with Keil μ Vision2. 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 ST Microelectronics ST10 Flasher.
- 2) The „*Getting More Involved*“ 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-ST10HS/E using the Keil tools and ST10 Flasher.
- 3) The “*Debugging*” section provides a third example program - "Debug" - to demonstrate simple debug functions using the Keil μ Vision2 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.

1.4 System Requirements

Use of this "Rapid Development Kit" requires:

- the PHYTEC phyCORE-ST10HS/E
- the phyCORE Development Board HD200 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



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

1.5 The PHYTEC phyCORE-ST10HS/E

The phyCORE-ST10HS/E 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 ST Microelectronics ST10F269 controller as well as a Real-Time Clock which, like the SRAM, 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-ST10HS/E to be plugged like a "big chip" into target hardware.

The standard module runs at a 40 MHz internal clock speed (delivering 50 ns instruction cycle) and offers 512 kByte (up to 1 MByte) SRAM, 256 kByte on-chip Flash as well as up to 2 Mbyte on-board Flash for DATA and CODE storage.

The module communicates by means of an RS-232 transceiver and 2 CAN bus interfaces. An optional external UART with an RS-232 transceiver provides the second asynchronous serial interface. The optional CS8900A 10Base-T Ethernet controller enable implementation of the module in embedded Internet devices. The phyCORE-ST10HS/E operates within a standard industrial range of 0 to +70°C and requires only a 220 mA power source.

ST Microelectronics' ST10 Flasher enables easy in-system programming of the on-chip Flash memory.

phyCORE-ST10HS/E (ST10F269) Technical Highlights

- SBC in subminiature dimensions (60 x 53 mm) achieved through modern SMD technology
- populated with an ST Microelectronics ST10F269 controller featuring two Full 2.0B on-chip CAN and operating in 16-bit, non-multiplexed bus mode at 40 MHz CPU speed (50 ns/instruction cycle)
- 512 kByte (up to 1 MByte) external SRAM (up to 1 MByte)¹
- 256 kByte on-chip Flash memory supporting on-board download of user code from a host-PC in conjunction with ST Microelectronics' ST10 Flasher¹
- no dedicated programming voltage required
- battery-buffered Real-Time Clock
- one serial interface via RS-232
- external UART as a second asynchronous serial interface
- optional Ethernet controller
- 16-channel A/D converter with 10-bit resolution
- two 2.0B Full-CAN interfaces
- requires a single power supply of 5 VDC/ <220 mA

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

The phyCORE Development Board HD200, 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-ST10HS/E, which plugs into the receptacle contact strips mounted on the Development Board HD200.

phyCORE Development Board HD200 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 3.3 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 HD 200 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)

1.6 The Keil μ Vision2 Software Development Tool Chain

Keil μ Vision2 fully supports the entire ST Microelectronics C166 microcontroller family. This includes a C compiler, macroassembler, Linker/Locator and the Simulator and Target Monitor within the μ Vision2 IDE. Specific chips supported are the ST10F168, ST10F269 and more. Future derivatives are easily accommodated due to the flexible Keil C compiler design.

μ Vision2 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 internal/external Flash on the PHYTEC phyCORE-ST10HS/E target board.

μ Vision2 consists of the following executables:

- **C Compiler** c166. exe
- **Assembler** a166. exe
- **Linker** l166. exe
- **Converter** oh166. exe
- **μ Vision2** 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 μ Vision2 is in *C:\Keil\Uv2*. Access to these programs from Windows is accomplished with μ Vision2. The entire tool set can be run from μ Vision2 or directly from DOS with batch files. The evaluation version is limited in code size to 8 kByte. 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 ST Microelectronics ST10 Family

The C166 ANSI compiler and A166 assembler are designed specifically for the ST Microelectronics ST10F168, ST10F269, 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

μ Vision2 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 μ Vision2 enable the following debugging functions:

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

μ Vision2 has a performance analysis feature to ensure your code runs efficiently. In addition, μ Vision2 has a disassembler/assembler that allows the modification of user code without recompiling. The evaluation version of μ Vision2 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 C166/ST10 and 8051 C compilers interface with the RTOS and CAN libraries. Keil supports various CAN microcontrollers including the ST10F168 and ST10F269 among others. Future CAN products based on these 8051 or C16x/ST10 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 ST10 Microelectronics' ST10 Flasher download utility
- interfacing the phyCORE-ST10HS/E, mounted on the Development Board, to a host-PC
- downloading example user code in hexfile format from a host-PC to the on-chip Flash memory using the ST10 Flasher

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.

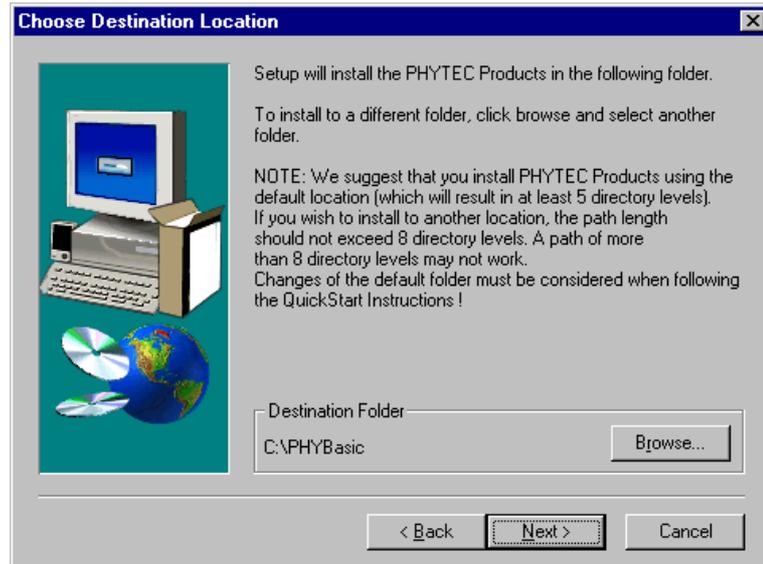
The following window appears:



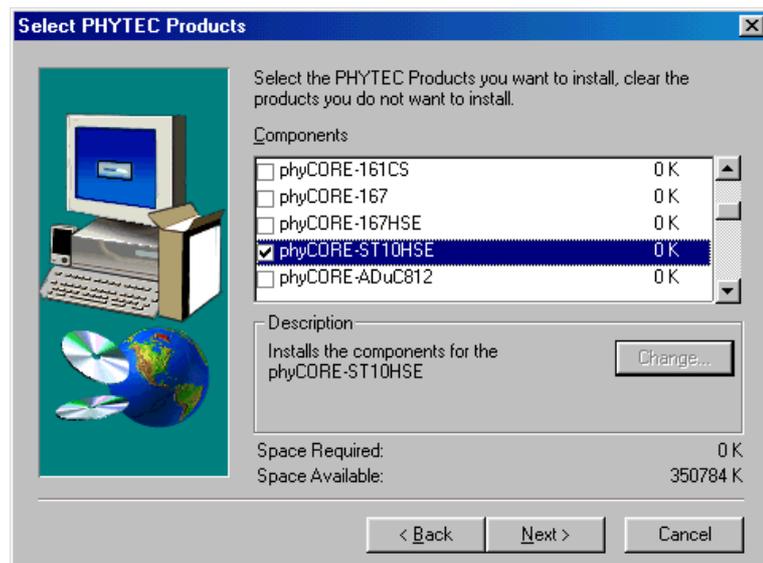
- Choose *Install Basic Product Files* Button.
- After accepting the *Welcome* window and license agreement select the destination location for installation of Rapid Development Kit software and documentation.

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.



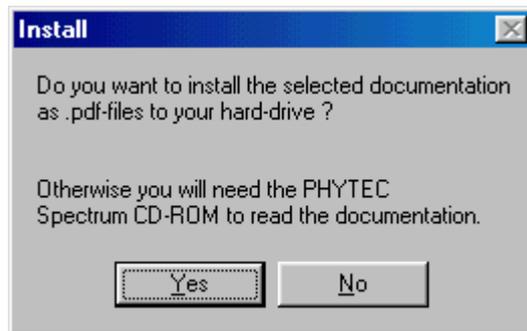
- In the next window select your Rapid Development Kit of choice from the list of available products. By using the *Change* button, advanced users can select in detail which options should be installed for a specific product.



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-ST10HS/E RDK will be installed to the |**PHYBasic** directory on your hard-drive.

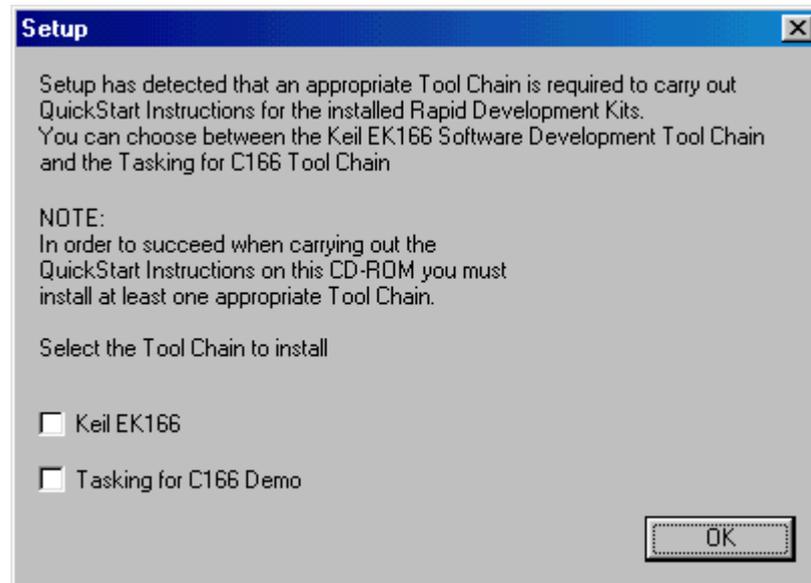
- In the next dialog you must choose whether to copy the selected documentation as *.pdf files to your hard drive or to install a link to the file on the Spectrum CD.



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-ST10HSE 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.



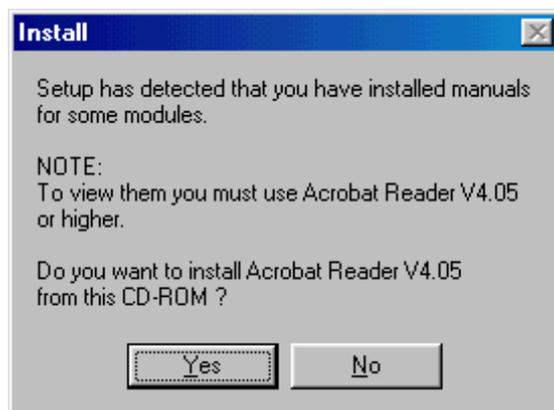
The applicable Keil tool chain must be installed to ensure successful completion of this QuickStart Instruction. Failure to install the proper software could lead to possible version conflicts, resulting in functional problems.

We recommend that you install the *Keil EK166* resp. μ Vision2 from the Spectrum CD-ROM even if other versions of μ Vision2 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.

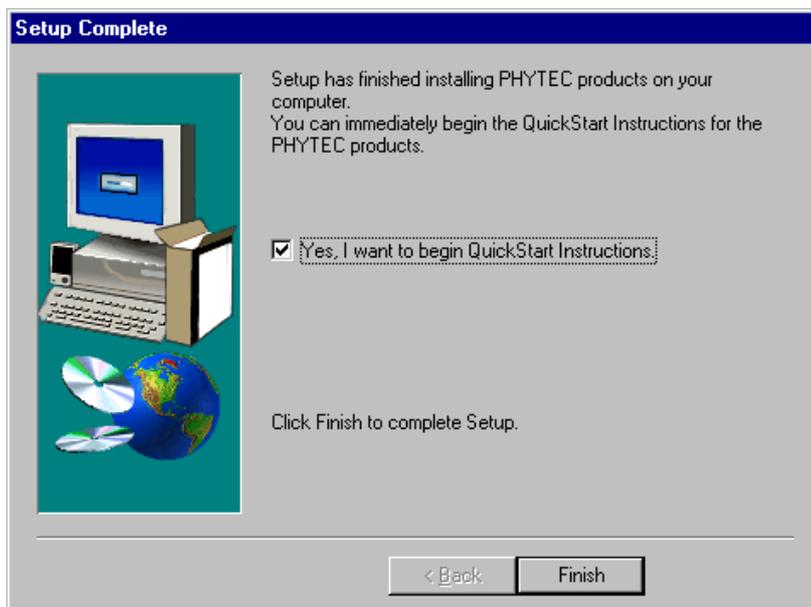
- After accepting the Welcome window and license agreement select the destination location for installation of the Development Tool Chain.

The applicable Keil μ Vision2 evaluation development tool chain will be installed to your hard-drive. Additional software, such as Adobe Acrobat Reader, will also be offered for installation.

- 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.

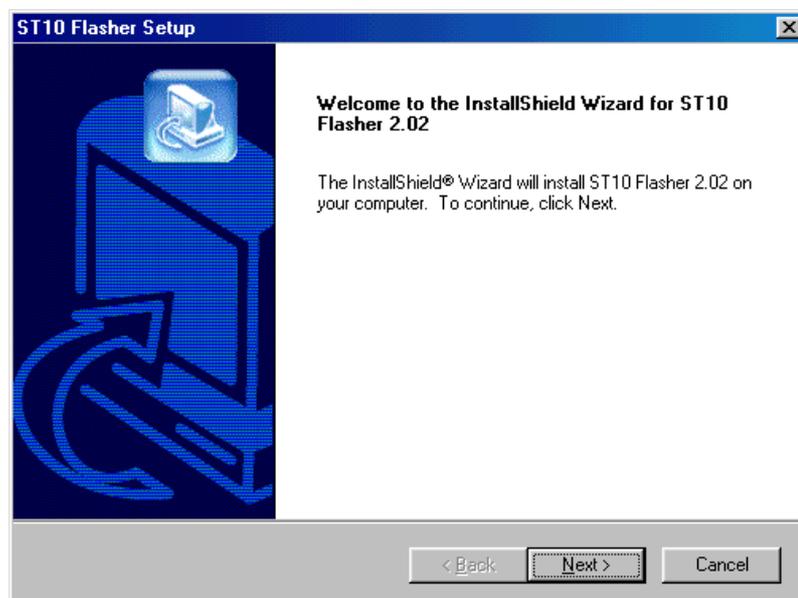


2.2 Installing ST Microelectronics ST10 Flasher

The applicable ST10 Flasher software must be installed to ensure successful completion of this QuickStart Instruction.

Install ST Microelectronics ST10 Flasher by executing *Setup.exe* from within the *\Software\ST-Flasher\Disk1* directory of your PHYTEC Spectrum CD and follow the installation instructions.

- In the ST10 Flasher Setup window, click on *Next* and complete the ST10 Flasher setup program.



2.3 Interfacing the phyCORE-ST10HS/E to a Host-PC

Connecting the phyCORE-ST10HS/E, mounted on the phyCORE Development Board HD200, 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.

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

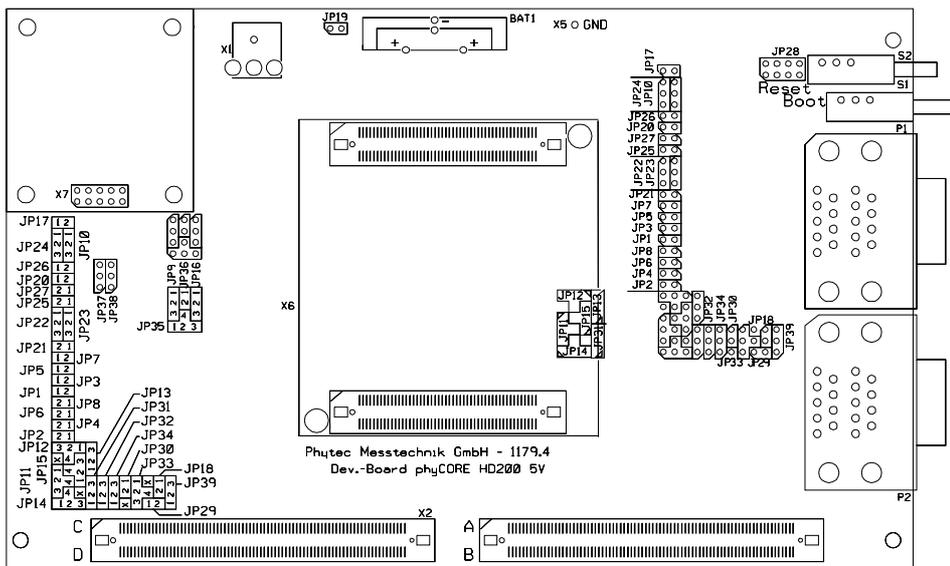


Figure 1: phyCORE Development Board HD200 Overview

- 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.

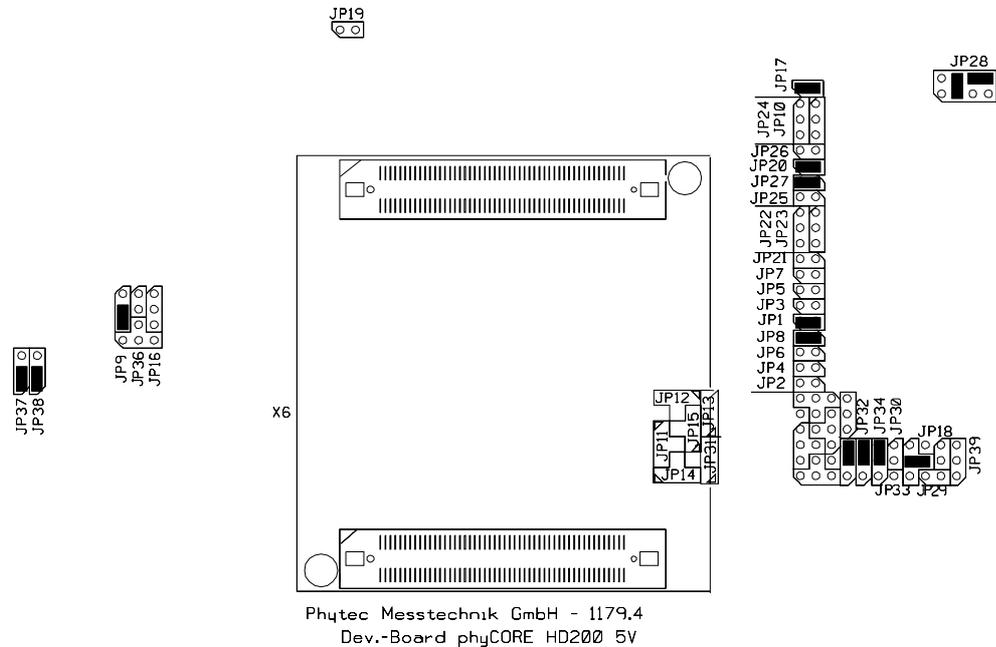


Figure 2: Default Setting for the phyCORE Development Board HD200

- Connect the RS-232 interface of your computer to the DB-9 RS-232 interface on the phyCORE Development Board HD200 (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*).

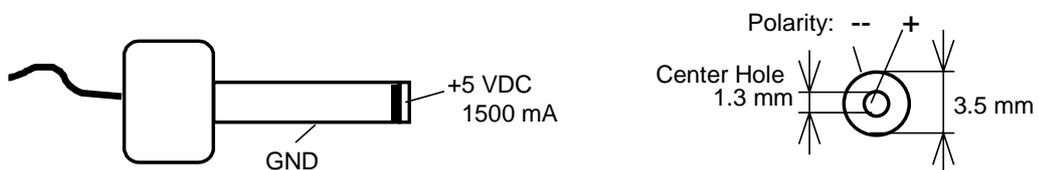


Figure 3: Power Connector

- Simultaneously press the Reset (S2) and Boot (S1) buttons on the phyCORE Development Board HD200, 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-ST10HS/E into the Bootstrap mode. Use of ST10 Flasher always requires the phyCORE-ST10HS/E to be in Bootstrap mode.

The phyCORE module should now be properly connected via the phyCORE Development Board HD200 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-ST10HS/E. This phyCORE module/phyCORE Development Board HD200 combination is also referred to as "target hardware".

2.4 Starting ST Microelectronics' ST10 Flasher

ST10 Flasher for Windows is a utility program that allows download of user code in Intel **.hex* file format from a host-PC to the controller's on-chip Flash on a PHYTEC SBC via an RS-232 connection.

Proper connection of a PHYTEC SBC to a host-PC enables the software portion of ST10 Flasher to recognize and communicate to the Bootstrap loader on the phyCORE-ST10HS/E.

Note:

Always ensure that the phyCORE-ST10HS/E is in Bootstrap mode before starting ST10 Flasher (*see section 2.2*).

- You can start ST10 Flasher by selecting it from the *Programs* menu using the Windows *Start* button.
- ST10 Flasher tries to establish connection to the target hardware.

It is recommended that you drag the ST10 Flasher icon onto the desktop of your PC. This enables easy start of ST10 Flasher by double-clicking on the icon.

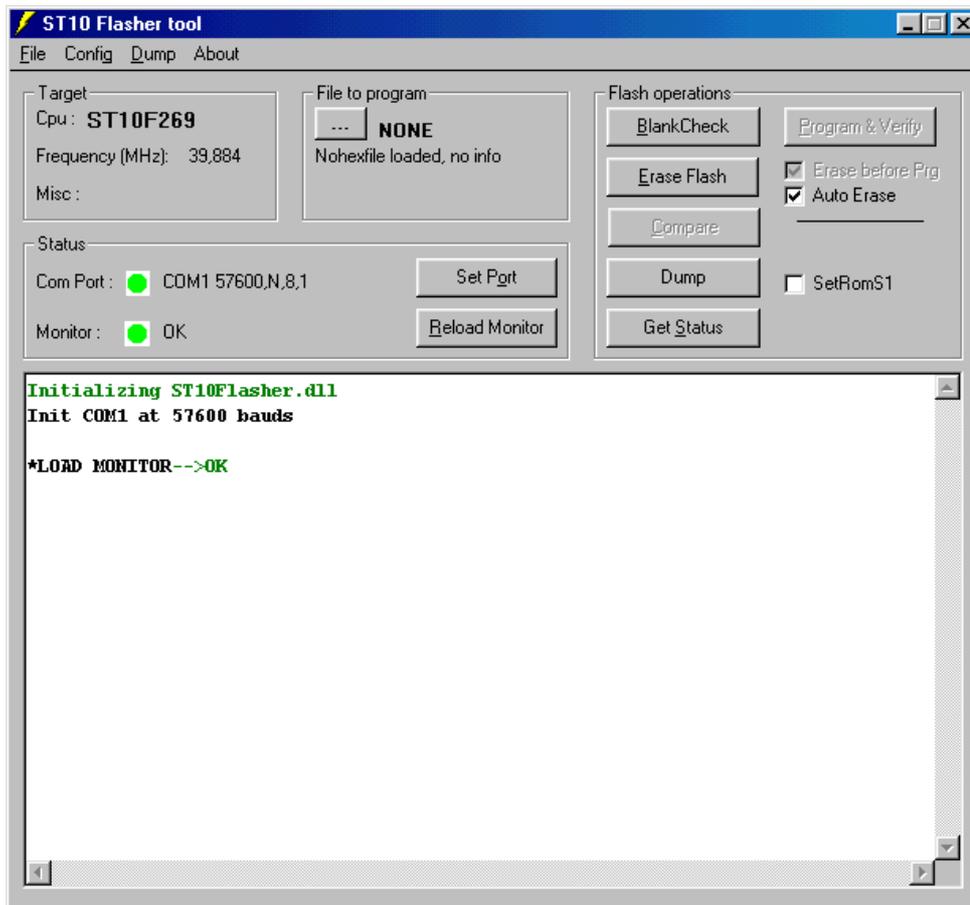
The microcontroller tries to automatically adjust to the baud rate used by ST10 Flasher (default 57,600 baud). *See the ST10 Flasher manual for details on how to change the default setting.* However, it may occur that the selected baud rate can not be attained. This results in a connection error. In 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.*

2.5 Downloading Example Code with ST10 Flasher

- 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.
- Start ST10 Flasher for Windows by double-clicking on the ST10 Flasher icon or by selecting ST10 Flasher from within the *Programs* program group.
- ST10 Flasher tries to establish connection to the target hardware.

The default settings for the ST10 Flasher configure COM1 with 57,600 baud as communication parameters. If you want to change these default settings follow the instructions given in the ST10 Flasher User's Manual located in the installation folder *|Program Files|STMicoelectronics|ST10-Flasher-2.02|Doc.*

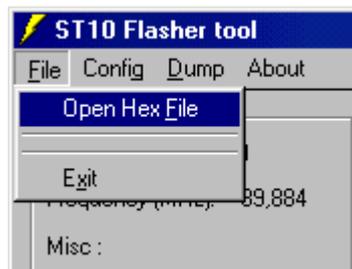
- The ST10 Flasher startup window will now appear.



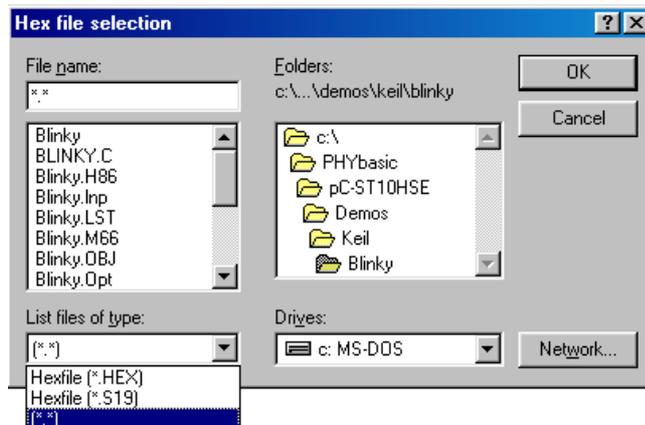
2.5.1 "Blinky"

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

- Choose *Open Hex File* from the **File** menu.

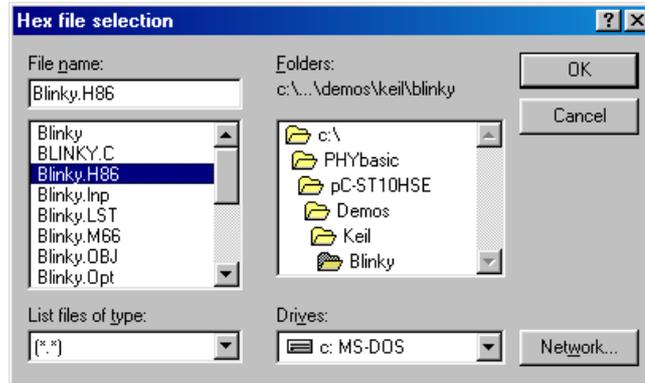


- Select (*.*) in the *List file of type:* pull-down menu.

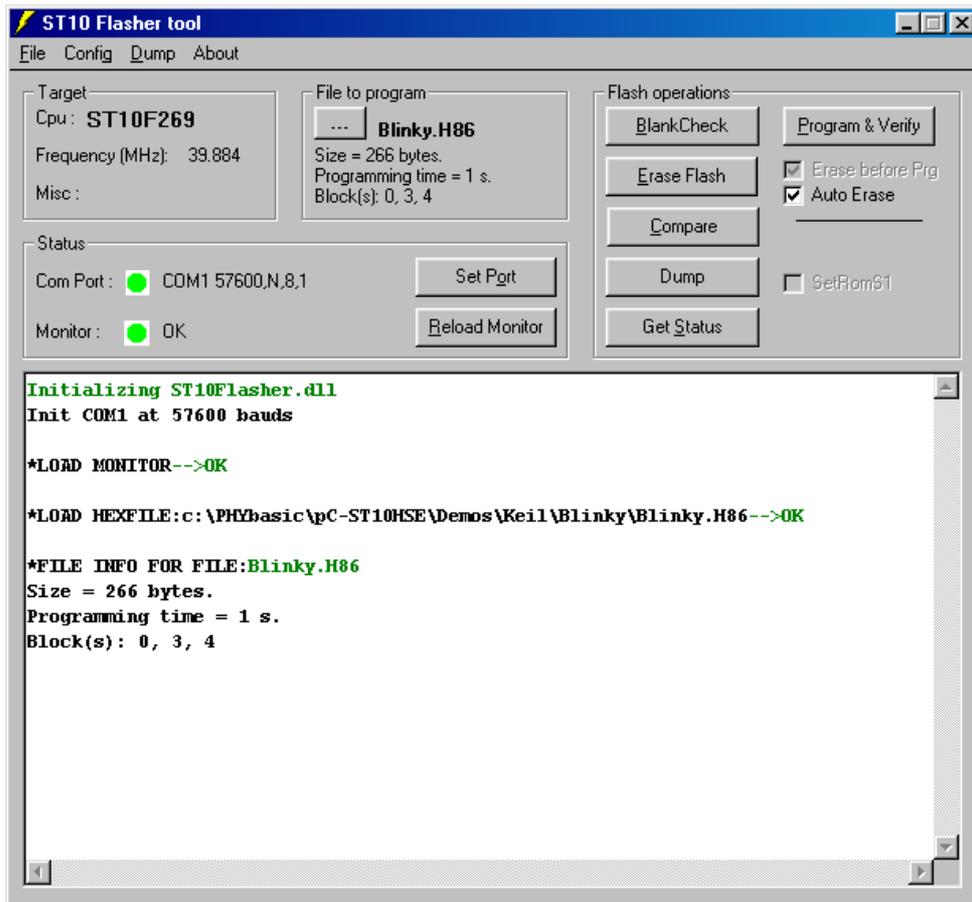


The hexfile has already been installed to your hard drive during the installation procedure.

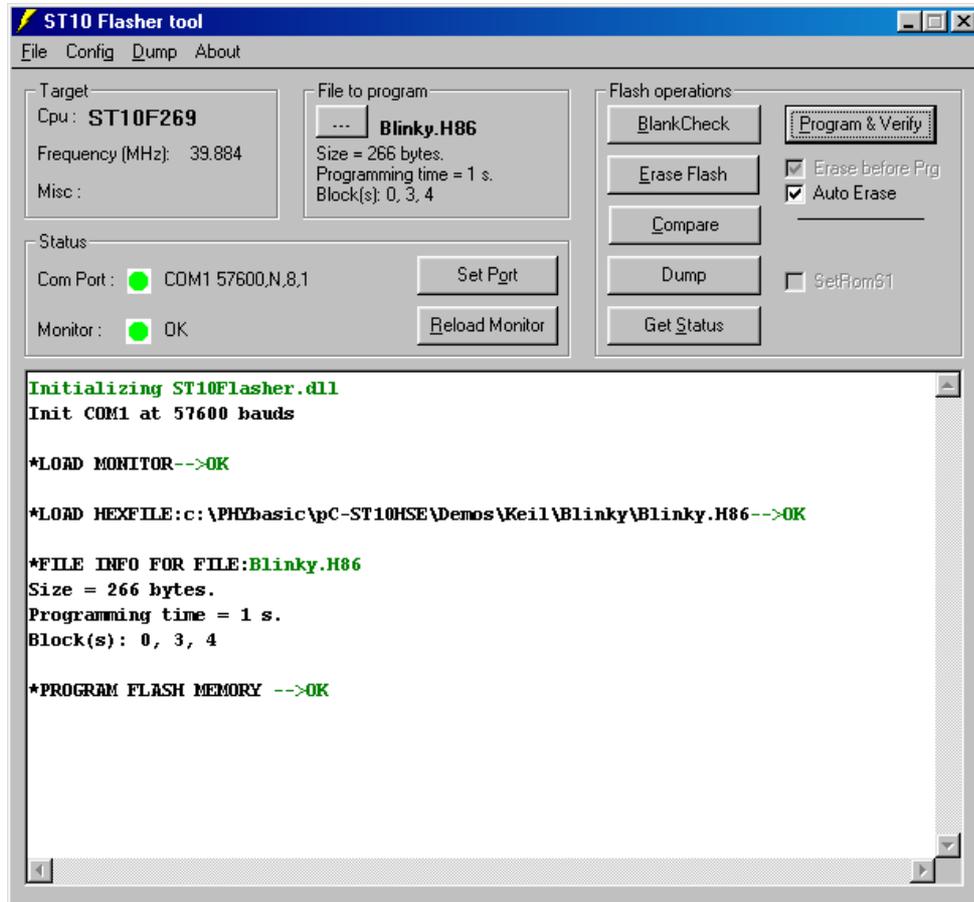
- Browse to the correct drive and path for the phyCORE-ST10HS/E Demo folder (default location **C:\PHYBasic\pC-ST10HSE\Demos\Keil\Blinky\Blinky.h86**), select the file **Blinky.h86** and click **OK**.



- The ST10 Flasher main window will re-appear.



- Click on the *Program & Verify* button in the upper right corner.
- Wait until programming is done. Once you see the *OK* message in the Message Window, the downloaded code can be executed.



- Exit ST10 Flasher.
- Press the Reset button (S2) on the phyCORE Development Board HD200 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.

2.5.2 "Hello"

The "Hello" example downloads a program to the on-chip Flash that, when executed, 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 ST10 Flasher download procedure. For detailed commentary on each step, described below in concise form, *refer back to sections 2.2 through .*

- 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.
- Start ST10 Flasher.
- Choose *Open Hex File* from the **File** menu.

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

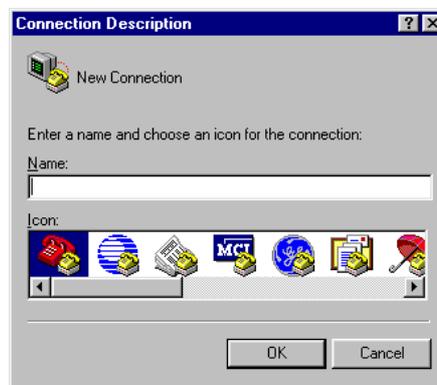
- Browse to the correct drive and path for the phyCORE-ST10HS/E Demo folder (default location **C:\PHYBasic\pC-ST10HSE\Demos\Keil\Hello\Hello.h86**) and click *OK*.
- The ST10 Flasher main window will re-appear.
- Click on the *Program & Verify* button in the upper right corner.
- Wait until the Flash programming is done. Once you see the *OK* message in the *Message* window, the downloaded code can be executed.
- Exit ST10 Flasher.

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 appear²:
- 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.



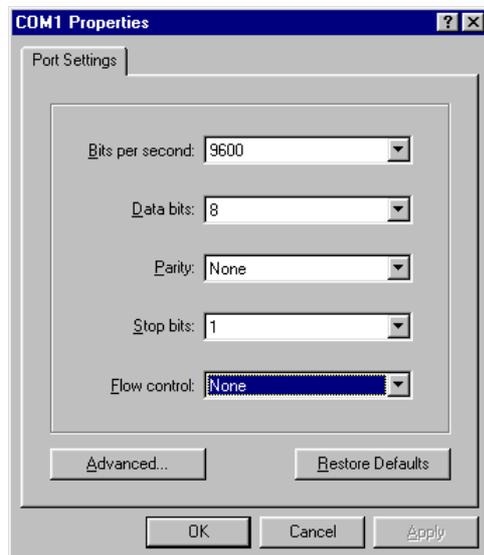
- Next click on *OK*. This creates a new HyperTerminal session named “COM Direct” and advances you to the next HyperTerminal window.

²: 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).

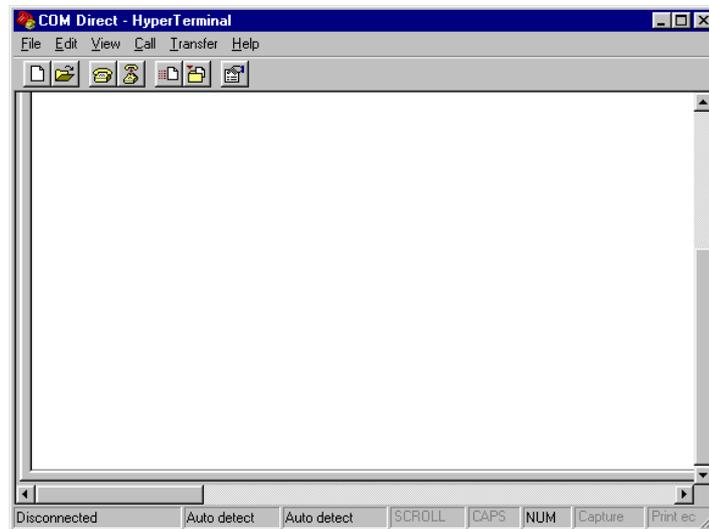


- Click the *Configure* button in the *COM Direct Properties* window to advance to the next window (*COM1/COM2 Properties*).



- 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 (at S2) will execute the *Hello.hex* 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 demo application within the file *hello.h86* initializes the serial port of your phyCORE-ST10HS/E 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 10 MHz populates the phyCORE-ST10HS/E. Using the internal PLL (Phase Locked Loop) device results in an internal 40 MHz CPU frequency. If your phyCORE-ST10HS/E is equipped with a different speed oscillator, the demo application might transmit using another baud rate.

You have now successfully downloaded and executed two pre-existing example programs in Intel **.h86* file format.

3 Getting More Involved

What you will learn with this example:

- how to start the μ Vision2 tool chain
- how to configure the μ Vision2 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 μ Vision2 Tool Chain

The μ Vision2 evaluation software development tool chain should have been installed during the install procedure, as described in *section 2.1*.

You can also manually install μ Vision2 by executing *ek166v424A.exe* from within the `\Software\Keil\muVision2.30` directory of your PHYTEC Spectrum CD.

Caution:

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 μ Vision2* from within the programs group.

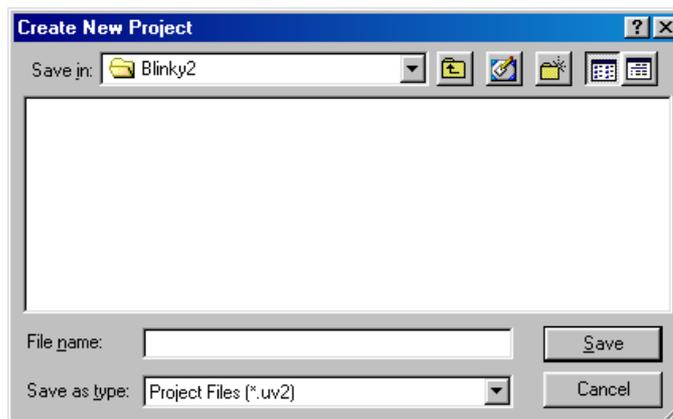
After you start μ Vision2, the window shown below appears. From this window you can create projects, edit files, configure tools, assemble, link and start the debugger. Other 3rd party tools such as emulators can also be started from here.



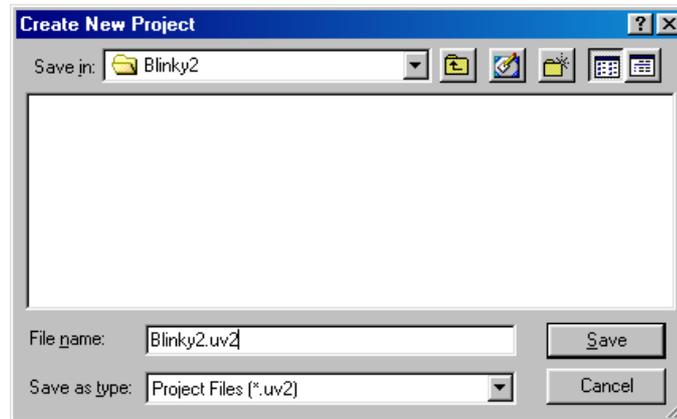
3.2 Creating a New Project and Adding an Existing Source File

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-ST10HSE\Demos\Keil\Blinky2*).

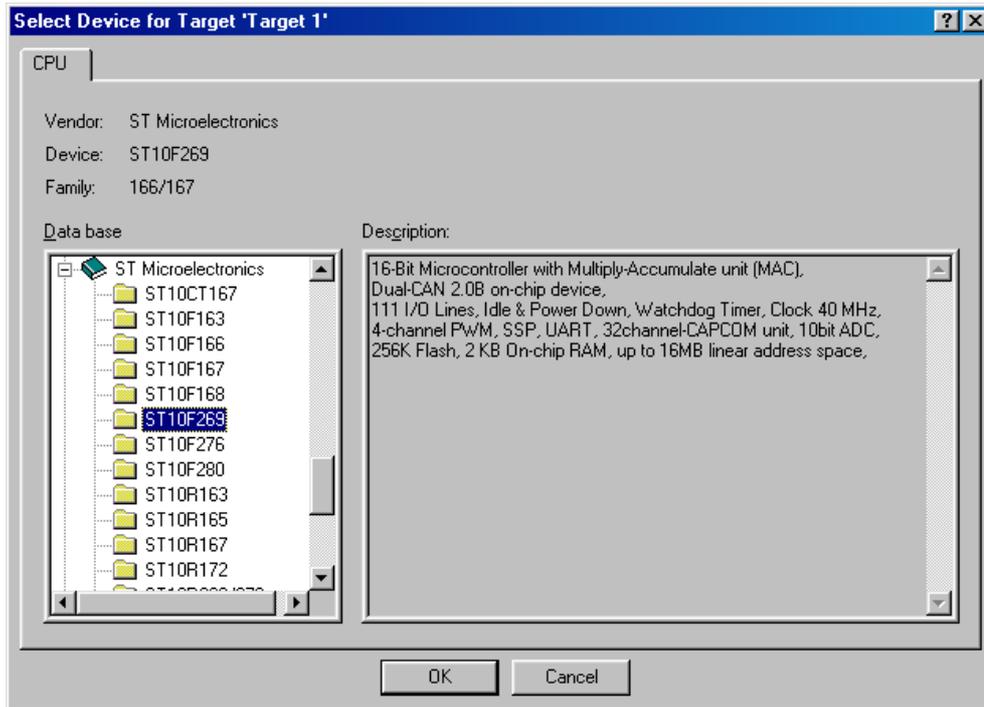


- In the text field '*File name*', enter the file name of the project you are creating. For this tutorial, enter the name ***Blinky2.uv2*** and click on *Save*.



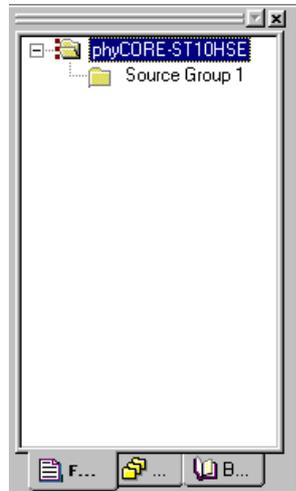
- The ***Select Device for Target 'Target1'*** will automatically appear. Double click on *ST Microelectronics* as manufacturer for the CPU within the next window which opens automatically³. The phyCORE-ST10HS/E is equipped with a *ST10F269 CPU*. Choose this controller type from the list as shown below. This selection sets necessary tool options for the ST10F269 device and simplifies in this way the tool configuration.

³: The same window opens by choosing *Select Device for Target* from the *Project* menu.

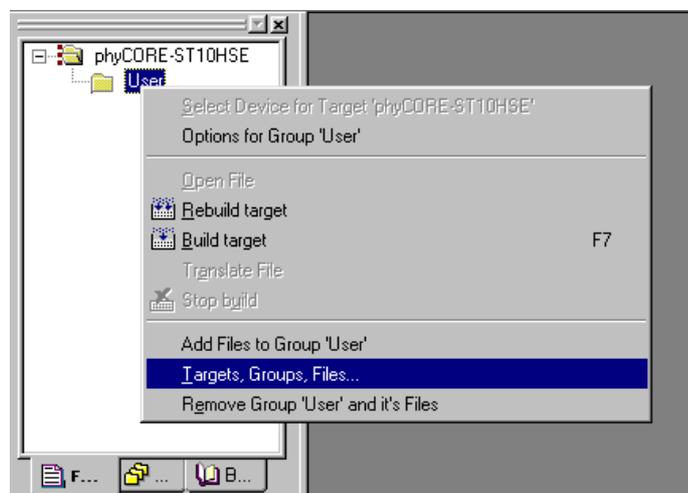


- Click on *OK* to save this setting.

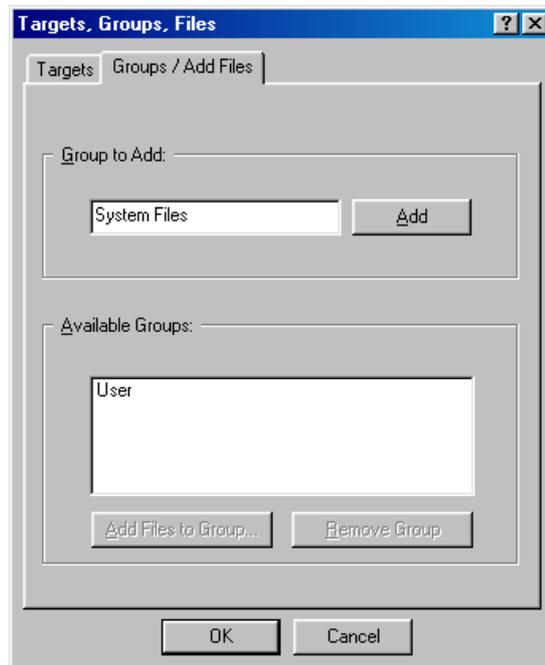
- Now click on *Target1* within the **Project Window - Files** tab. *Target1* is now highlighted. Click on *Target1* again to enable the edit mode. Change the default name of the target to *phyCORE-ST10HSE*.



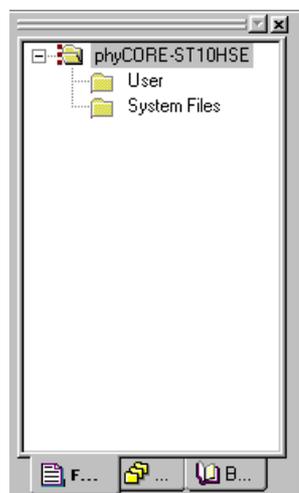
- Select the file group *Source Group 1* in the **Project Window - Files** tab and click on it to change the name into *User*.
- Right-click in the **Project Window - Files** to open a new window. Choose the option *Targets, Groups, Files...*



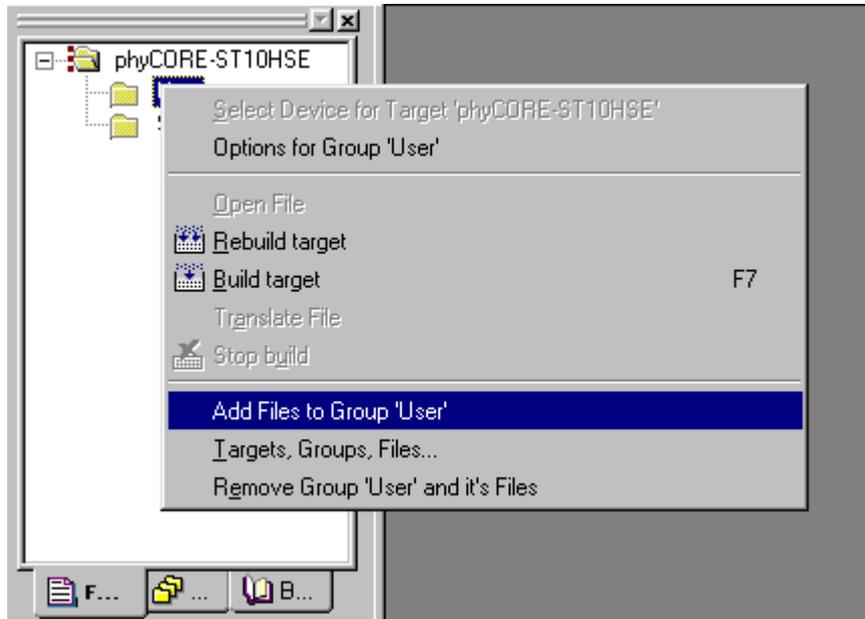
- Select the **Groups / Add Files** tabsheet and type the new group name *System Files* in the **Group to Add:** section.



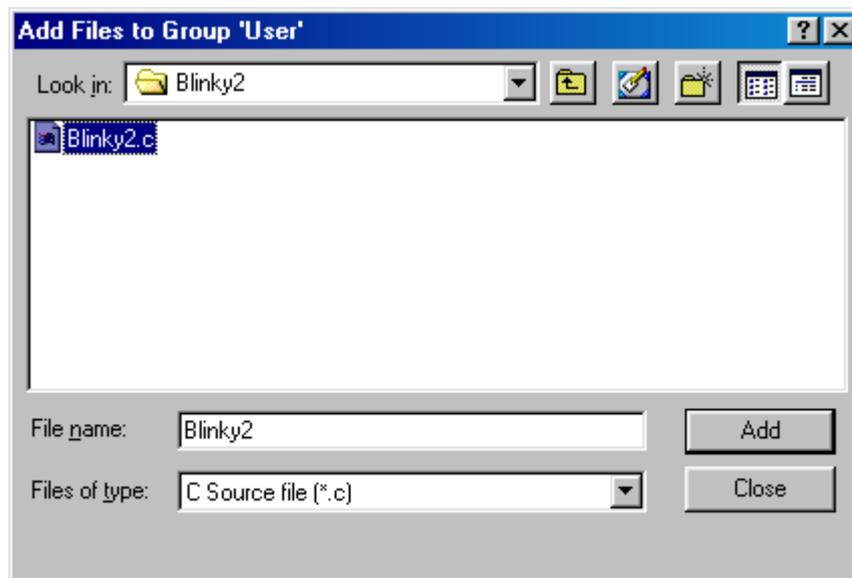
- Click on *Add* and then on *OK*.
- Your project file structure should now look like this:



- Now it's time to add some source code to our project. To do so, click with the right mouse key on the *User* group to open a local menu. The option *Add Files to Group 'User'* opens the standard files dialog.

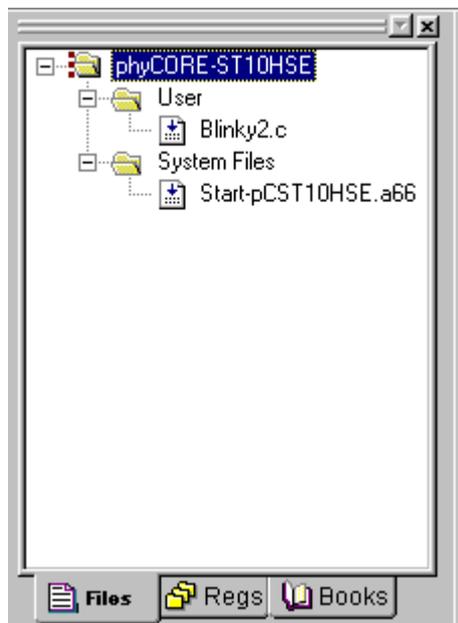


- Select the file *Blinky2.c*.



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

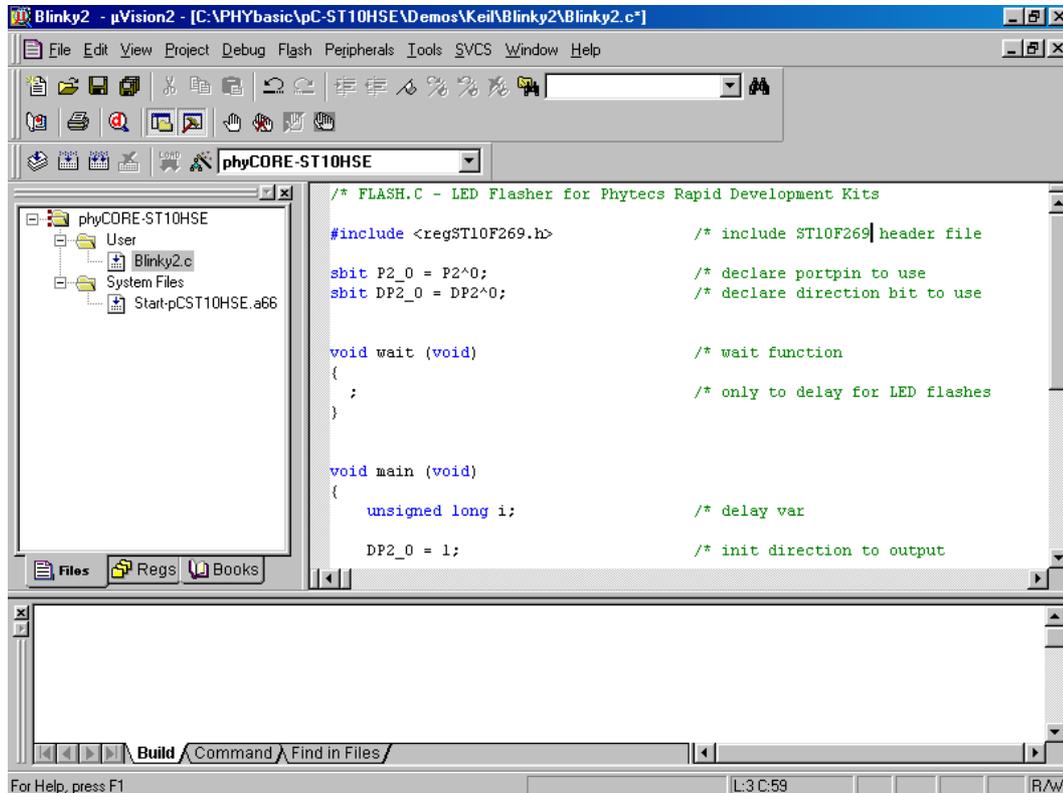
- Now right-click on group *System Files* and add the file *Start-pcST10HSE.a66*. You have to change the file type to "*Asm Source file (*.a, *.src)*" in the *File of types* pull-down menu to see this file.
- Your project window should now look like this:



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-pcST10HSE.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 (150,000) counts to the indicated values:

```

while (1) {                      /* loop forever          */
    P2_0 = 0;                     /* output to LED port      */
    for (i=0; i<225000; i++) { /* delay for 225000 counts */
        wait ();                 /* call wait function      */
    }

    P2_0 = 1;                     /* output to LED port      */
    for (i=0; i<75000; i++) { /* delay for 75000 counts */
        wait ();                 /* call wait function      */
    }
}

```

3.4 Saving the Modifications

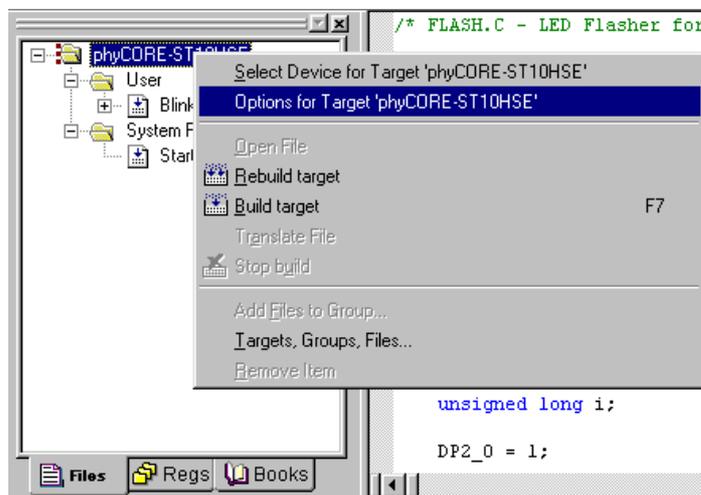
Save the modified file by choosing *File/Save* or by clicking the floppy disk icon  .

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.

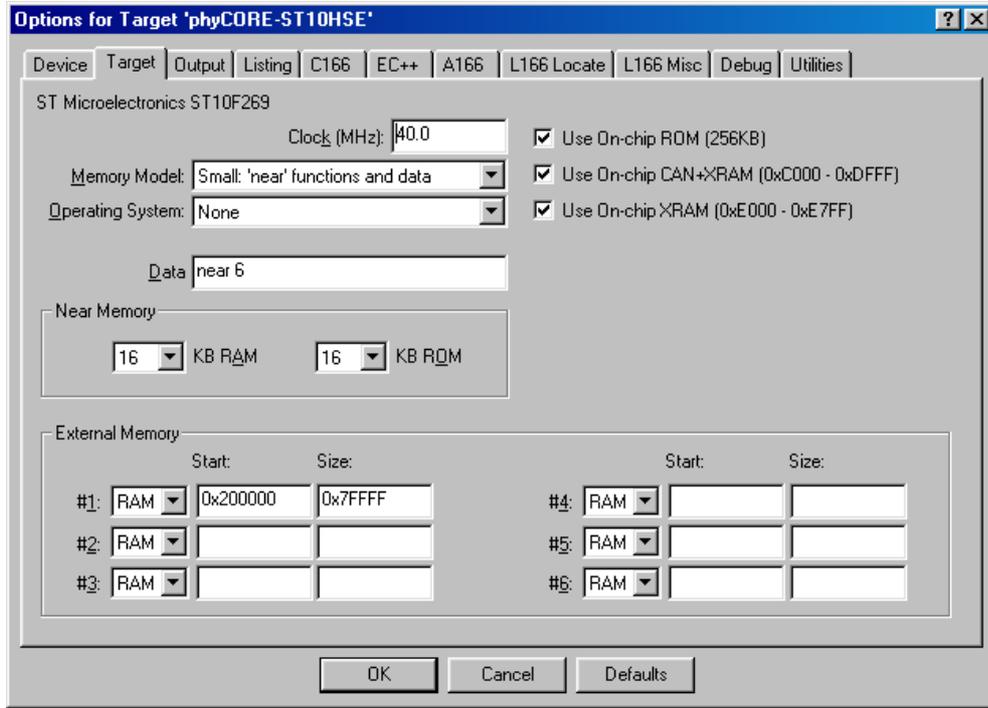
Enter the changes as indicated below and leave all other options set to their default values. μ Vision2 allows you to set various options with mouse clicks and these are all saved in your project file.

- Click with the right mouse key in the *Project* window to open a local menu. Choose the option *Options for Target 'phyCORE-ST10HSE'*.



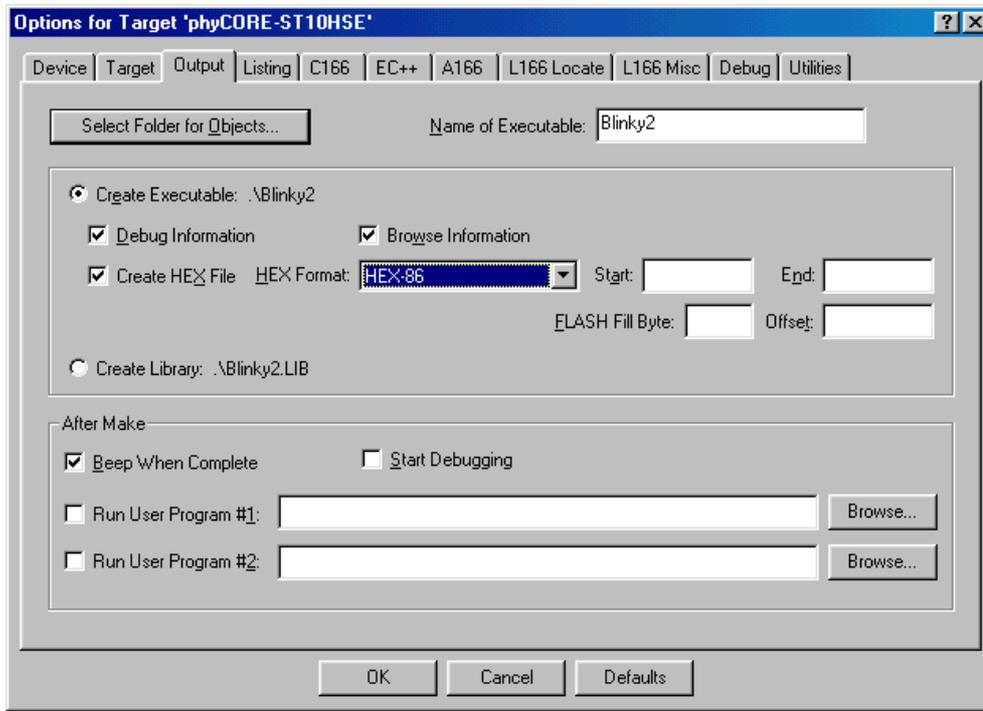
To configure the Target:

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



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.

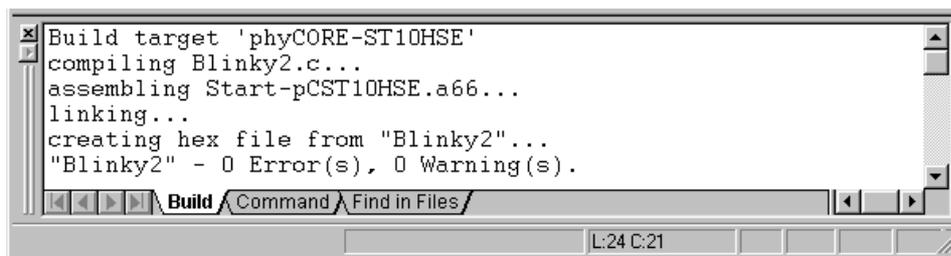


- No other configurations are necessary for this example.
- Click *OK* to save these settings

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 μ Vision2 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*).



```
Build target 'phyCORE-ST10HSE'  
compiling Blinky2.c...  
assembling Start-pCST10HSE.a66...  
linking...  
creating hex file from "Blinky2"...  
"Blinky2" - 0 Error(s), 0 Warning(s).
```

- 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 and then releasing first the Reset and, two or three seconds later, the Boot button.
- Start ST10 Flasher.
- Choose *Open Hex File* from the *File* menu.
- Browse to the correct drive and path for the phyCORE-ST10HS/E Demo folder (default location **C:\PHYBasic\pC-ST10HSE\Demos\Keil\Blinky2\Blinky2.h86**) and click *OK*.
- The ST10 Flasher main window will re-appear.
- Click on the *Program & Verify* button in the upper right corner.
- Wait until the Flash programming is done. Once you see the *OK* message in the *Message* window, the downloaded code can be executed.
- Exit ST10 Flasher.
- 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 "Hello2"

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 1.3.

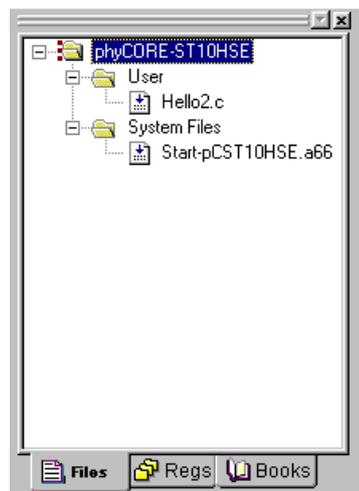
3.8.1 Creating a New Project

Open the *Project* menu and create a new project called *Hello2.uv2* within the existing project folder

C:\PHYBasic\pC-ST10HSE\Demos\Keil\Hello2

(default location) on your hard-drive. Select the *ST Microelectronics ST10F269* in the CPU vendor database list.

- Add *Hello2.c* and *Start-pcST10HSE.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-pcST10HSE.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 1.6*.

- Type the settings for the *External Memory* as shown below. Make sure that the option “*Use On Chip Rom (256KB)*” is selected.

RAM: Start: 0x200000 Size: 0x7FFFF

- 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-ST10HS/E 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.

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 and then releasing first the Reset and, two or three seconds later, the Boot button.
- Start ST10 Flasher.
- Choose *Open Hex File* from the *File* menu.
- Browse to the correct drive and path for the phyCORE-ST10HS/E Demo folder (default location *C:\PHYBasic\pC-ST10HSE\Demos\Keil\Hello2\Hello2.h86*) and click *OK*.
- The ST10 Flasher main window will re-appear.
- Click on the *Program & Verify* button in the upper right corner.
- Wait until the Flash programming is done. Once you see the *OK* message in the *Message* window, the downloaded code can be executed.
- Exit ST10 Flasher.

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 internal 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 μ Vision2 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 μ Vision2 Debugger offers two operating modes that can be selected in the *Project/Options for Target phyCORE-ST10HSE* 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. μ Vision2 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** 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.

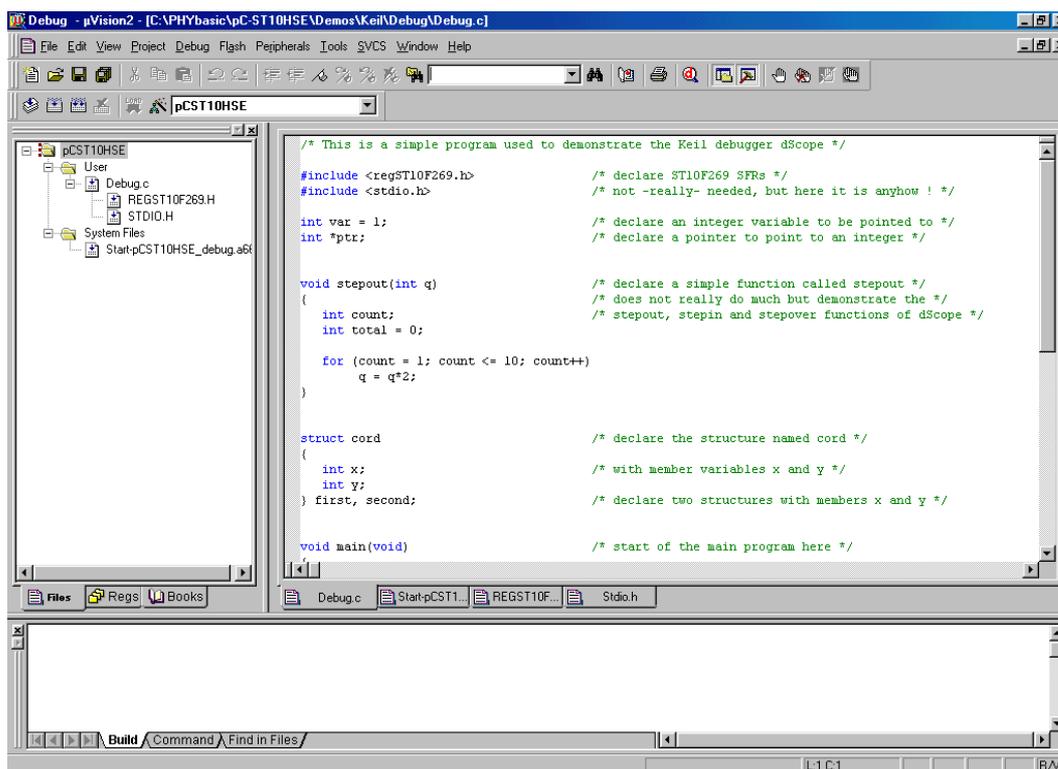
The following examples utilize the **Keil Monitor 166** environment.

4.1 Creating a Debug Project and Preparing the Debugger

Within the default location

C:\PHYBasic\pC-ST10HSE\Demos\Keil\Debug you will find an already build project called *Debug*.

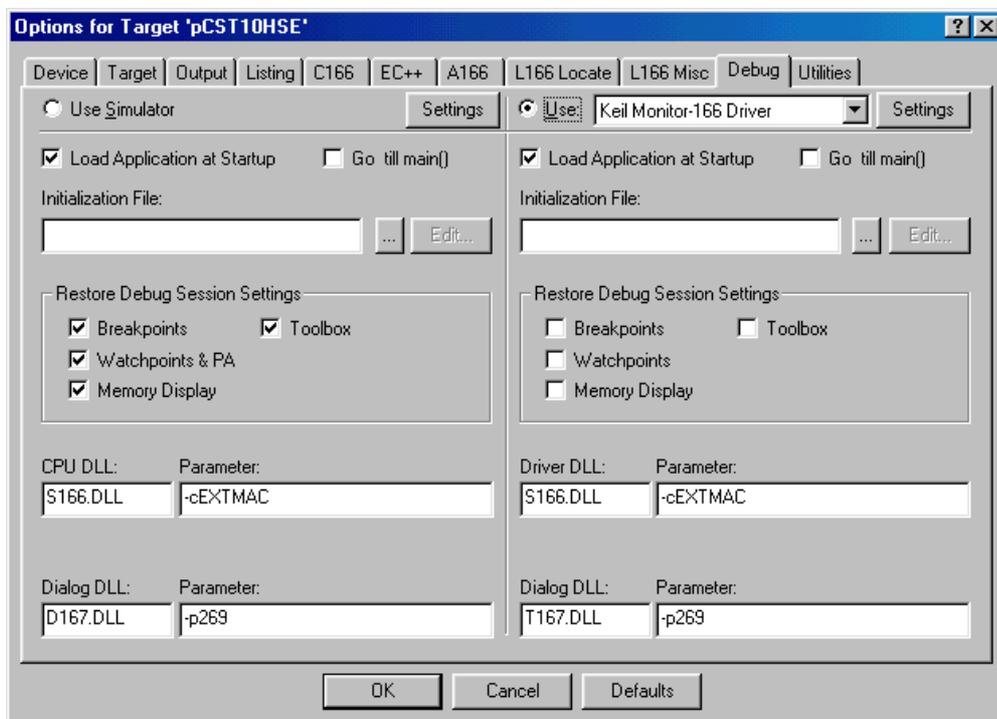
- Close all possible open projects and open the project *Debug.uv2* with the option *Project/Open Project*.



4.2 Preparing the Debugger

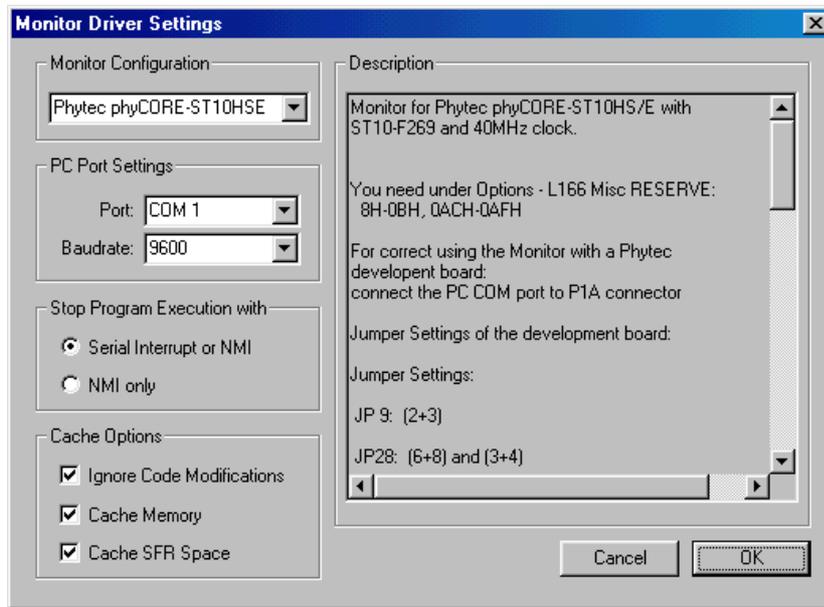
This example utilizes the Monitor interface, which automatically downloads a special Monitor kernel to the target hardware using the Bootstrap mode. Depending on the *Project/Options for Target 'pCST10HSE'/Debug* configuration, μ Vision2 will load the application program and run the startup code.

- Open the *Project/Options for Target 'pCST10HSE'* menu and select the *Debug* tabsheet.
- Enable the checkboxes *Use: Keil Monitor -166 Driver* and *Load Application at Startup*.



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

- Select *Phytec phyCORE-ST10HSE* 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 'pCST10HSE'* menu will appear.
- Click on the *OK* button again.

4.3 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 of the target system.

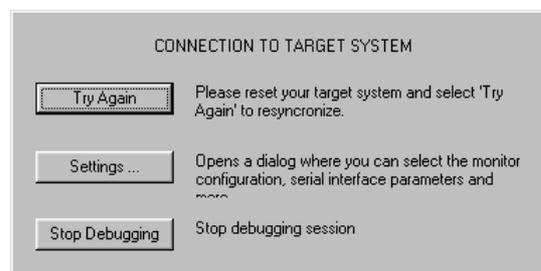
4.4 Starting the Debugger

To start the μ Vision2 debug environment, click on the debugger icon  on the μ Vision2 toolbar. A pop-up window will appear indicating that this is an evaluation version. Click on *OK*.



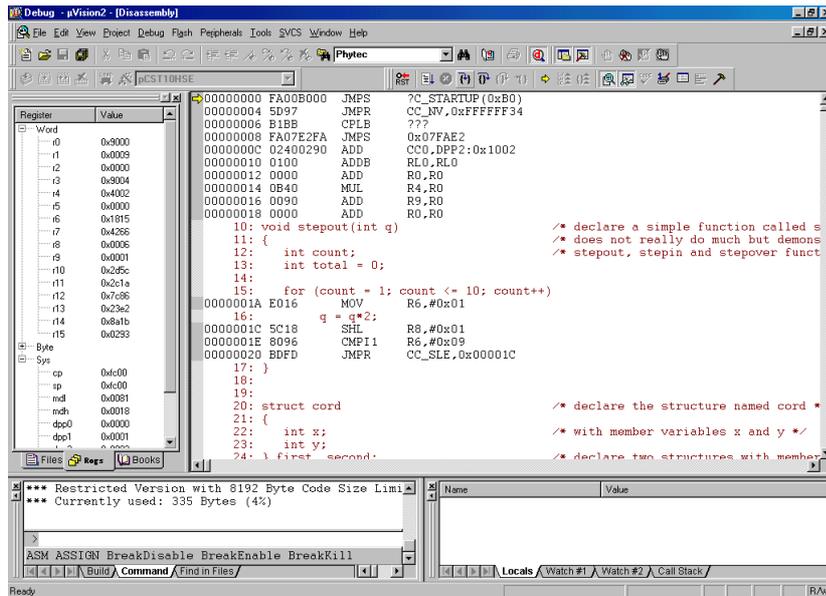
- You will see a blue status bar from left to right at the bottom of your screen indicating the download process of the debug program.

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 1.4*) and click on *Try Again*.

- If the data transfer was successful choose *View/Dissassembly Window* and you will get a screen similar to this shown below. The project window changed to the register value view and the debug toolbar is shown. The lower right screen shows the Watch & Call Stack Window. (If you can't see one of these enable it in the *View* menu).



- You will see the source code on your screen in the Disassembly Window. Note that you will not see your C source code because the program starts at 00000H – which is the reset vector - with some assembler instructions. There is an assembler jump to a section of initialization code which then branches to your C main program. In this example you will see the JMPS instruction at 00000H and some parts of the C code.
- Click once on *StepInto!*  . You will see the assembly code of the initialization routine. DISWDT (disable watchdog) is the first instruction. EINIT (E8H) signals the end of the initialization routine of the CPU.
- To view the Output window, if not already visible, select *View/Output window*.

- At the Output window prompt,  type in `g,main` and press <Enter> (notice the comma!) to run the CPU to the start of the 'main' function. The program counter will advance to 'main'. Note you can press the up arrow key to get the history of keys typed in the Output Window.
- You can single step using the *StepInto* icon  . You are now ready to use the debugger window to step through code, set breakpoints, and issue the Go command to start program execution. You can examine special function registers, memory locations, and register values, etc.

4.5 Using the Keil μ Vision2 Debug Features

4.5.1 Breakpoints

Click on a memory location such as line number 42 `first.x++`. A colored bar appears marking this position.

- You can click on *Run to Cursor line*  to reach this point or you can double-click and set a breakpoint. Set a breakpoint here with the double-click. The red mark on the left-hand of the selected line indicates the breakpoint.
- Click on *RUN*  and the program will run and stop at the breakpoint.
- Double click on the *breakpoint* to remove it.

4.5.2 In Line Assembler

Open the Watch & Call Stack Windows with a click on this icon  .

- You could see the actual values of the variable *bigcount* in the Watch window.
- Double click in line 48 to set a breakpoint.
- Click on *RUN* a few times. This will continue the endless loop until program execution stops at the breakpoint. See how the value of *bigcount* changes with each RUN.
- Open the Inline Assembler window by selecting *Debug/Inline Assembly*.
- Move the Window with your mouse to your upper left hand so you can see your source code and the Watch Window.
- Type the address *0x64* in the *Current Assembly Address* field and press <Enter> on your keyboard. The assembly code at this address appears in the *Current Instruction* field. It is the addition of the variable *bigcount* with 2.
- Type the assembler instruction *ADD R10,#3* in the *Enter New Instruction* field and press <Enter>. You could see the change of your source code at address *0x64H*.
- Close the *Inline Assembler* window.
- Click on *RUN* a few times and see that the value of *bigcount* now is added with 3 at each run.
- Remove the breakpoint at line 48 with a double mouse click.

4.6 Single Stepping

The debugger uses “*Step Into*” to single step one instruction at a time. “*Step Into*” is also used to enter a function in the same fashion.

- “*Step Over*”  means to skip over a function that you are not interested in.
- “*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.
- If the debugger gets stuck, execute a Reset using the Reset icon located next to the RUN  icon. To isolate any problems from the target hardware board you could use the software simulator instead of the Monitor. You can also use the hardware Reset button on the board.
- With the cursor on line 42 click on *StepInto* until you enter the function **stepout()**. Note that you must click on *StepInto* 10 times to exit the *for* loop within the function **stepout()**.
- Repeat the process using *StepOver* and you will not enter the function although it will be executed. Use the command `$=\42` within the Output Window to reposition the program counter to line 42 prior to further code execution. This command takes a line number preceded by a backslash to set the program counter. The *StepOver* feature is useful to skip function calls you are not interested in debugging.

Note:

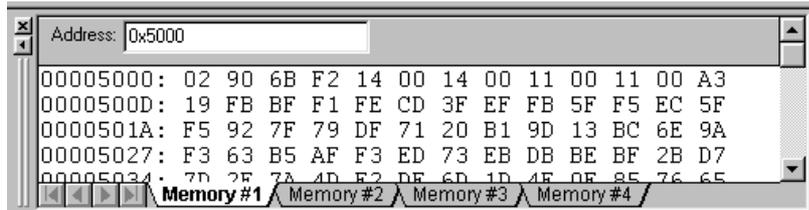
The *StepOut* button is grayed out and not available in Monitor mode. *StepOut* is available in the simulator and provides a quick escape from a function by executing the next *return* instruction.

- As you step through the program - the appropriate values in the *Regs* window will change color as their values are updated. Most of the variables have been allocated to fast internal CPU registers by the μ Vision2 compiler.

4.6.1 Memory Window

If the Memory Window is not already visible, activate it by selecting *View/Memory window* from the μ Vision2 tool bar.

- Type *0x5000* at the *Address:* field and press <Enter>. The addresses displayed in the Memory Window should change starting at this address. This is the memory area for the variables of the project.

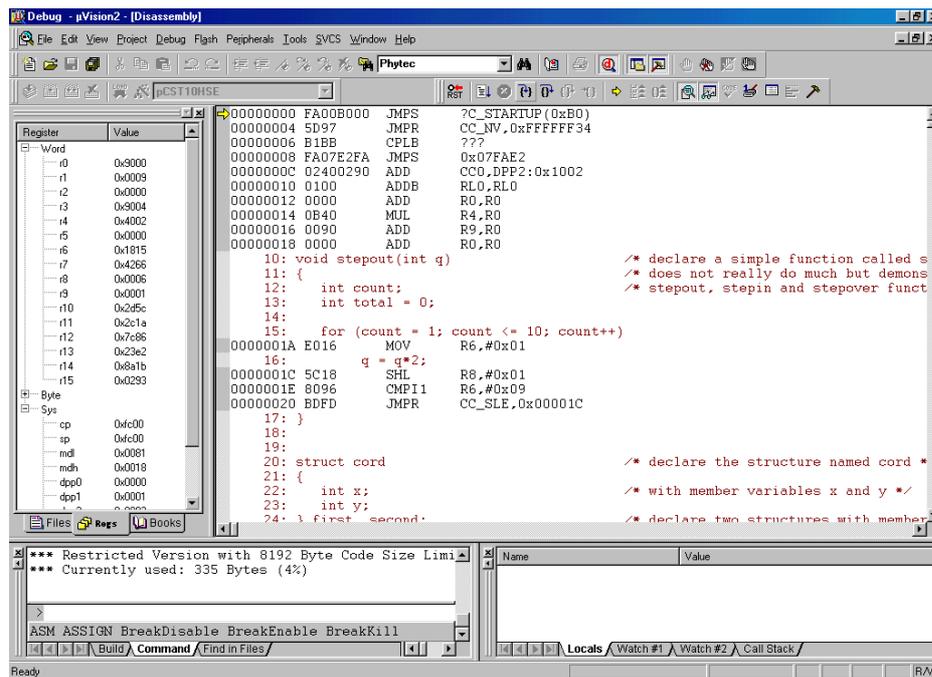


- Set a breakpoint at line 47 (count++). Note that as you press *Run*, the contents of the memory changes as the structure values are incremented at lines 42 through 45 (e.g. bigcount at 0x5000, first at 0x500A). Remove the breakpoint at line 47 with a double mouse click.
- Close the Memory Window

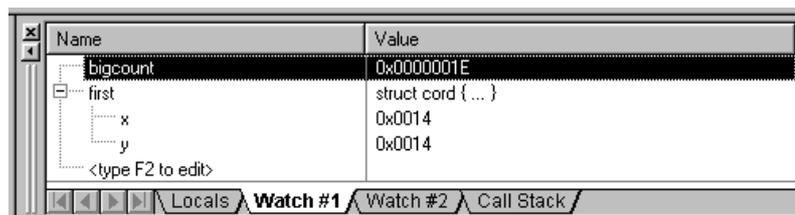
4.6.2 Watch Window

If not already visible view the Watch Window by selecting *View/Watch & Call Stack window* at the μ Vision2 menu.

- The Watch Window displays memory contents as specified by name. Structures can also be displayed.
- Click on the *Files tab* in the *Project window* and double click on the *Debug.c* filename to open it. Scroll down till you could see the yellow program counter arrow.



- Click on the *Watch #1* tab at the bottom of the *Watch Window*
- To add variable symbols just click with the right mouse button on the symbol name in your source code and choose *Add “...” to Watch Window*. Add the variables *bigcount* and *first* to *Watch Window #1*. Till *first* is a structure with the members *x* and *y* the *Watch Window* shows a + symbol left from *first*. Click on + to see the structure members *x* and *y*.



- Open the *Disassembly Window* with a click on this icon .
- Double click in line 42 of the C source code (the line-counter is displayed at the bottom of the µVision2 window) to set a breakpoint and choose *Run to Cursor line* from the *Debug Toolbar*.
- Now choose *Step into* a few times and look how the variable values are changing in the *Watch Window*.

4.7 Resetting Simulator and the phyCORE-ST10HSE

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 beginning address (0) again. Press the Halt  button to halt a program under normal conditions.

- Debugger in Monitor mode: The Monitor runs in the target hardware. A Debugger reset sets the IPC to zero and performs other initialization routines if no user application was started. It is not as complete as a hardware-reset (pressing S2 on the Development Board). The best method of stopping a running application is to press the *Halt* button rather than 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 the next step. This has the advantage of detecting any 'infinite loop' in which your program might be stuck. With Reset, you start again at address 0.
- The PHYTEC Development Board does not have 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 (S2) if the monitor has been downloaded with the Bootstrap Loader. It must be restarted manually by restarting the Debugger.

5 Advanced User Information

This section provides advanced information for successful operation of the phyCORE-ST10HS/E with the μ Vision2 software tool chain.

5.1 Start-pcST10HSE.a66

The code within the assembly file *start-pcST10HSE.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-ST10HSE.a66* from within the default location **C:\PHYBasic\pC-ST10HSE\Tools\Startup\Keil**. The properties of the external bus interface are already configured for the phyCORE-ST10HS/E. 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-pcST10HSE.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).

5.2 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:

- `xCODE`: code for several addressing modes
($x = N, I, F, H$ or X)
- `xDATA`: not initialized data for several addressing modes
($x = N, I, F, H$ or X)
- `xDATA0`: pre-initialized data for several addressing modes
($x = N, I, F, H$ or X)
- `xCONST`: constant for several addressing modes
($x = N, I, F, H$ or X)

It is required that all `xDATA` and `xDATA0` classes segments are located to any internal RAM of the ST10F269 or to any external RAM on the phyCORE-ST10HS/E.

All `xCODE` and `xCONST` classes must also be located to any internal non-volatile memory (e.g. Flash) of the ST10F269.

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 internal Flash for locating code and constant classes.

The recommended operating mode of the phyCORE-ST10HS/E allows the use of the Chip Select Unit of the ST10F269 to define the physical memory layout. By modifying the file *start-pcST10HSE.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) (not populated at standard version of the phyCORE-ST10HS/E)
- RAM Bank uses /CS1 (up to 1 MByte RAM)

The default configuration of the phyCORE-ST10HS/E has 256 kByte on-chip Flash and 512 kByte RAM (/CS1).

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-pcST10HSE.a66* file.

5.3 Debugging Using Monitor Kernel

Whenever you decide to use the μ Vision2 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-pcST10HSE_debug.a66* within the default destination *C:\PHYBasic\pCST10HSE\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-ST10HS/E 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-ST10HS/E 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.

5.4 Demo for the Optional Ethernet Controller CS8900A-CQ

As an option, the Crystal LAN CS8900A-Q Ethernet Controller can populate the phyCORE-ST10HS/E at position U23. The PHYTEC Spectrum-CD, included in the phyCORE-ST10HS/E 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-ST10HSE\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-ST10HS/E 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 less 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 module tcpip.h to fit into your existing LAN (see MYIP_x).

The easyWEB project is set up for two different targets:

- pC-ST10HSE: Settings for PHYTEC phyCORE-ST10HS/E module
- Simulation: Settings for Simulation.

Note:

The Ethernet controller cannot be simulated.

Document: phyCORE-ST10HSE QuickStart Instructions
Document number: L-633e_1, March 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

© PHYTEC Meßtechnik GmbH 2003

Ordering No. L-633e_1
Printed in Germany