phyCORE®-LPC3250

QuickStart Instructions

**Using IAR's JLINK and the IAR EWARM
Software Development Tool Chain**

Note: The PHYTEC Kit CD includes the electronic version of the English
phyCORE-LPC3250 Hardware Manual

Edition: September 2009

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 America LLC assumes no responsibility for any inaccuracies. PHYTEC America LLC neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product. PHYTEC America LLC reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages, which might result.

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

1 Introduction to the Rapid Development Kit

The phyCORE® System-on-Module (SoM) is designed to be plugged into a PHYTEC Carrier Board. The PHYTEC Carrier Board contains the I/O connectors as well as any other interface circuitry not provided on the phyCORE® SoM itself. The phyCORE® SoM, combined with the PHYTEC Carrier Board, provides a platform to jump start embedded designs and propel concept to prototype and finished product.

Each PHYTEC Rapid Development Kit contains a phyCORE SoM mounted on an applicable Carrier Board, cables, power supply, printed schematics, applicable evaluation software development tool CDs, and the PHYTEC Kit CD. The PHYTEC Kit CD provides this QuickStart guide, complete electronic documentation, and demos.

1.1 Rapid Development Kit Documentation

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

- the PHYTEC phyCORE-LPC3250 Hardware Manual
- LPC3250 controller User's Manuals and Data Sheets
- this QuickStart Instruction

1.2 Overview of this QuickStart Instruction

This QuickStart Instruction gives a general "Rapid Development Kit" description, as well as software installation hints and example programs enabling quick out-of-the box start-up of the phyCORE®-LPC3250 in conjunction with the IAR JLINK, IAR Embedded WorkBench® for ARM (EWARM) software tools, and Micrium tools. The QuickStart is structured as follows:

1. The *"Getting Started"* section describes how to install the IAR Embedded WorkBench®, interface the phyCORE®-LPC3250 target hardware to a host PC, and run the **Debug iRAM** project to demonstrate the download of user code to the internal RAM using the IAR Jlink and EWARM software tools.

2. The *"Debugging"* section introduces the main debugging features of the IAR EWARM and uses the **Debug xRAM** project to demonstrate the download of user code to the external SDRAM memory on the phyCORE®-LPC3250 and goes through a basic debugging exercise with use of a break point.

3. The *"Flash Programming"* section uses the **Debug xRAM boot from NAND** project to demonstrate the download of user code to the external NAND Flash memory on the phyCORE®-LPC3250

4. The *"Getting More Involved"* section provides instructions for creating a new project with the **GettingStarted** demo as a template.

5. The "*Micrium uC/OS-II Demo"* provides instructions for downloading and running **Micrium uC/OS-II**.

2 System Description

2.1 Hardware Description

The following PHYTEC hardware components are included in the phyCORE®-LPC3250 IAR Rapid Development Kit (KPCM-040-IAR) and are necessary for completing the instructions in this QuickStart:

- the PHYTEC phyCORE®-LPC3250 (PCM-040)

- the phyCORE®-LPC3250 Carrier Board (PCM-967)

- Bare PCB Expansion Board (PCM-988)

- AC adapter supplying 5 VDC /3.2A, center positive

- Straight Ethernet cable

- Serial cable

- USB Standard A to mini-B cable

- SBC Module extraction tool

- Hard copy schematics

- IAR JLINK JTAG-USB adapter[1]

- PHYTEC Kit CD

- IAR EWARM evaluation CD

2.2 Host System Requirements

- Operating System: Windows 2000, Windows XP or Windows Vista

- CPU architecture: Any x86 32-bit or 64-bit (x64: AMD64 or Intel EM64T) processor

- 60 Mbytes Free Hard Disk Space

- 128 Mbytes of RAM

---

[1] : The IAR JLINK is included in the Rapid Development Kit version with the part number KPCM-040-IAR.

2.3 The PHYTEC phyCORE®-LPC3250

The phyCORE®-LPC3250 is an ARM-9 based, small form factor, OEMable module populated with the NXP LPC3250. State-of-the-art power management, Vector Floating Point Unit (VFP), and rich peripherals such as USB OTG, Ethernet, and integrated LCD controller make this module the ideal candidate for embedded applications requiring high performance and low power consumption. The on-board MMU supports major operating systems, including Linux and Windows Embedded CE. Other chip-level features include 7 UARTs, SPI, I^{2}C, a Real-Time Clock with a separate power domain, and NAND Flash and DDR memory controllers. These features make the device particularly suitable for automotive and industrial control applications as well as medical systems.

Please refer to the phyCORE®-LPC3250 Hardware Manual for specific information on board-level features, jumper configuration, memory mapping, pin layout, and Carrier Board features.

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

North America	http://www.phytec.com	support@phytec.com
Europe	http://www.phytec.de	support@phytec.de

2.4 IAR EWARM Software Development Tool Chain

IAR Systems provides a complete set of development tools for ARM, ranging from evaluation environments to highly optimizing C/C++ compilers and executable UML tools. The IAR Embedded WorkBench® for ARM (EWARM) is a suite of tools for building and debugging embedded applications. It supports ARM7, ARM9, ARM9E, ARM10, ARM11 and XScale devices.

For more information on the IAR EWARM tools visit their website at: http://www.iar.com/

2.5 Micrium µC/OS-II Real-Time Kernel

The Micrium µC/OS-II Real-Time Kernel is a highly portable, ROMable, very scalable, preemptive real-time, multitasking kernel (RTOS) for microprocessors and microcontrollers. µC/OS-II can manage up to 255 tasks and provides the following services:

- Semaphores

- Mutual Exclusion Semaphores (to reduce priority inversions)

- Event Flags

- Message Mailboxes

- Message Queues

- Task Management (Create, Delete, Change Priority, Suspend/Resume etc.)

- Fixed Sized Memory Block management

- Time Management

- Timer Management

µC/OS-II runs on a large number of processor architectures.

µC/OS-II Kernel Awareness Plug-In displays µC/OS-II's internal data structures in a convenient series of Windows integrated with the C SPY Debugger within the IAR Embedded WorkBench[®]. This provides information about each of the active tasks in the target application, about each semaphore, mutex, mailbox, queue and event flag group along with a list of all the tasks waiting on these kernel objects, and more. This functionality supports testing and debugging applications.
For more information on the Micrium tools visit their website at: www.micrium.com

Micriµm
Empowering Embedded Systems

3 Getting Started

What you will learn with this Getting Started example:

- installing Rapid Development Kit software
- interfacing the phyCORE®-LPC3250, mounted on the Carrier Board, to a host-PC using the IAR JLINK
- downloading example user code to the LPC3250 internal SRAM memory.

3.1 Installing Rapid Development Kit Software

3.1.1 Installing Documentation from the PHYTEC Kit CD

The PHYTEC Kit CD installs documentation, datasheets, and demo files for the phyCORE-LPC3250. When you insert the PHYTEC Kit CD into the CD-ROM drive of your host-PC, the CD should automatically launch a setup program. Otherwise the setup program (setup.exe) can be manually executed from the root of the CD.

1. Insert the PHYTEC Kit CD.

2. On the *Welcome to the phyCORE-LPC3250 Rapid Development Kit Setup Wizard* window, select **Next** to continue.

3. Keep *C:\PHYTEC* as the installation path and select **Next** to continue.

4. On the *Ready to Install* window, select **Install**.

5. After the installation process is complete, select **Finish**.

The PHYTEC Kit CD installation path is *C:\PHYTEC\phyCORE-LPC3250* and the following five subfolders are located within the **phyCORE-LPC3250** directory:

Demos – This folder contains third party demos.

Documentation – This folder contains datasheets, hardware manual, QuickStarts, and other product related documentation.

Linux – This folder contains Linux BSP and tools.

Stage1 Loader – This folder contains the Stage1 Bootloader for the LPC3250.

WinCE – This folder contains a binary Eboot Bootloader, Windows Embedded CE images, demo application for Visual Studio, and SDK.

3.1.2 Installing IAR Embedded Workbench for ARM

When you insert the IAR EWARM CD into the CD-ROM drive of your host-PC, the CD should automatically launch a setup program that installs the required software. Otherwise the setup program (.exe) can be manually executed from the root folder of the CD.

Alternately, the IAR Embedded WorkBench® for ARM Evaluation tools can be installed from: https://www.iar.com

1. Select **Install IAR Embedded WorkBench**® from the IAR Embedded Workbench® setup screen:

2. Click to **Get a license**.

3. The IAR Product Registration page will open within a new Internet Browser window. Fill out your contact information and your registration.

4. You will receive a confirmation email from IAR. Follow the email instructions and confirm your registration to retrieve your License Number and License Key.

5. Select **Install IAR Embedded WorkBench**®

6. Follow the installation wizard for Installation of the IAR Embedded WorkBench®.

You have now successfully installed the IAR Embedded WorkBench® Kickstart edition.

Note: The Kickstart edition is limited to 32kB in output code generation. For unlimited evaluation you must install the 30-day evaluation edition.

3.2 Interfacing the phyCORE®-LPC3250 to a Host-PC

Connecting the phyCORE®-LPC3250, mounted on the phyCORE® Carrier Board, to your computer is simple:

1. Ensure proper jumper settings on the phyCORE® Carrier Board as indicated in phyCORE®-LPC3250 Hardware Manual.

2. Connect the 2.54 mm pitch ribbon cable from the IAR JLINK JTAG adapter to the JTAG connector X12 on the phyCORE® Carrier Board.

3. Connect the USB end of the JLINK JTAG adapter to the USB port of your host-PC using the included USB cable.

4. Connect the included serial cable between the bottom DB9 socket at P1 and an available RS-232 interface on your host-PC.

5. Connect the included 5V power adapter to the power socket X10 on the phyCORE® Carrier Board.

6. The red power LED D20, located next to the power socket at X10, should light. This indicates that proper voltage is supplied to the phyCORE® module and Carrier Board.

The phyCORE® module and Carrier Board should now be properly connected to a host PC via the IAR JLINK. You are now ready to use the IAR Development Tools to establish communication between the host-PC and target hardware.

3.3 Downloading Example Code with IAR EWARM

The IAR Embedded WorkBench® IDE should have been installed as described in section *3.1.2.*

1. Start the tool chain by selecting the IAR Embedded Workbench IDE from within the programs group: ***Start / IAR Systems / IAR Embedded Workbench for ARM Kickstart / IAR Embedded Workbench***

After you start the WorkBench®, the window shown below appears. From this window you can create projects, edit files, configure tools, assemble, link and start the debugger.
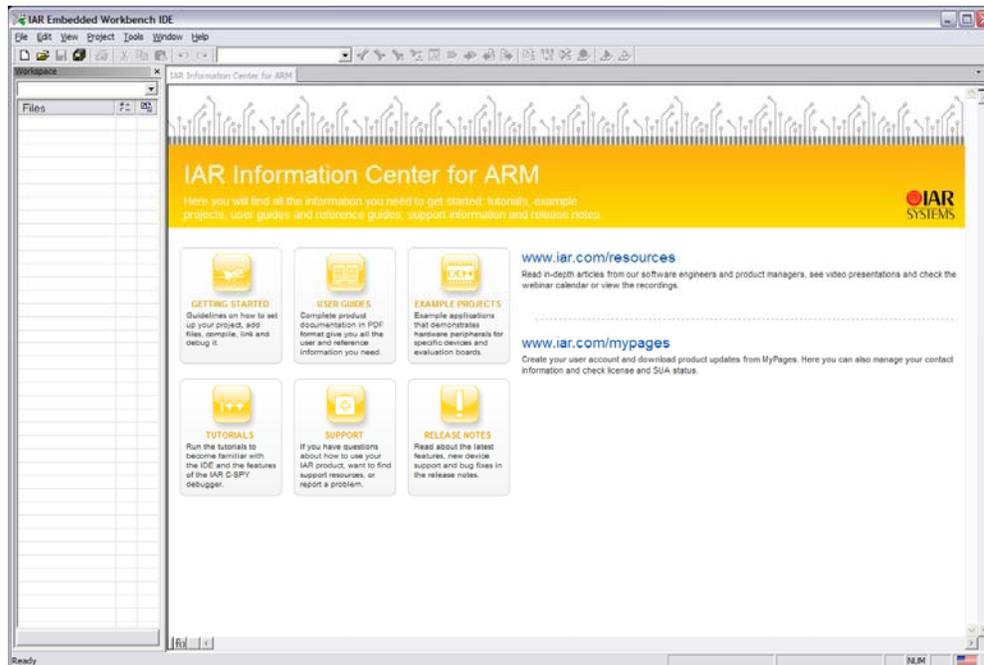
2. From the IDE menu select ***File / Open / Workspace***

3. Browse to and open the ***Gettingstarted.eww*** Workspace file located here:

    C:\Program Files\IAR Systems\Embedded Workbench 5.4 Kickstart\ARM\examples\ NXP\LPC3250\Phytec\Gettingstarted

The **GettingStarted** project shows basic use of CP15, I/O, timer, UART and the interrupt controllers. It starts by blinking LED_D15 and LED_D18 on the phyCORE® Carrier Board. Button BTN1 changes the blinking speed while button BTN2 changes the pattern. The button action is displayed over UART5 routed to the bottom DB-9 connector at P1 using 115200 baud, 8-bit, 1-stop and no parity/flow control.

The **GettingStarted** example contains 8 different configurations:

- **Debug iRAM**: configured to run in internal (on-chip) RAM.

- **Debug xRAM**: configured to run in external SDRAM.

- **Release iRAM boot from NAND512(64M)**: configured to run in internal RAM and boot from external 64 MB NAND.

- **Release iRAM boot from NAND256(32M)**: configured to run in internal RAM and boot from external 32 MB NAND.

- **Release xRAM boot from NAND512(64M)**: configured to run in external RAM and boot from external 64 MB NAND.

- **Release xRAM boot from NAND256(32M)**: configured to run in external RAM and boot from external 32 MB NAND.

- **Release iRAM boot level 2 from NAND256(32M)**: configured to use a second level boot loader to load the program into internal RAM (iRAM). This configuration can be used if the example exceeds the size of one NAND block.

- **Release iRAM boot level 2 from NAND512(64M)**: configured to use a second level boot loader to load the program into internal RAM (iRAM). This configuration can be used if the example exceeds the size of one NAND block.

4. From the Embedded Workspace Configuration drop-down menu select the **Debug iRAM** configuration as shown below:
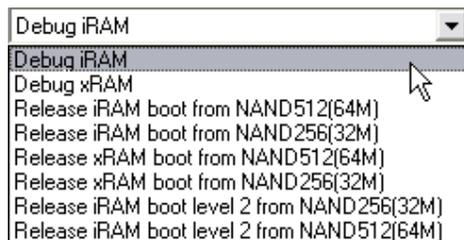
5. Start a HyperTerminal session with the following COM settings:

> **115200 bps**
> **8 bit**
> **No parity**
> **1 bit stop**
> **No flow control**

6. Start the debugger and download the code by selecting the **Download and Debug** button from the IDE toolbar .

If the data transfer was successful, a screen similar to the **Disassembly** window below will appear.. The **Debug Toolbar** is also displayed. In the lower part of the debug screen you will see the **Debug Log** window.

You may need to open, resize and /or move some windows to make your screen look similar to the following screen capture. You can open inactive windows by choosing the desired window from the **View** pull-down menu.
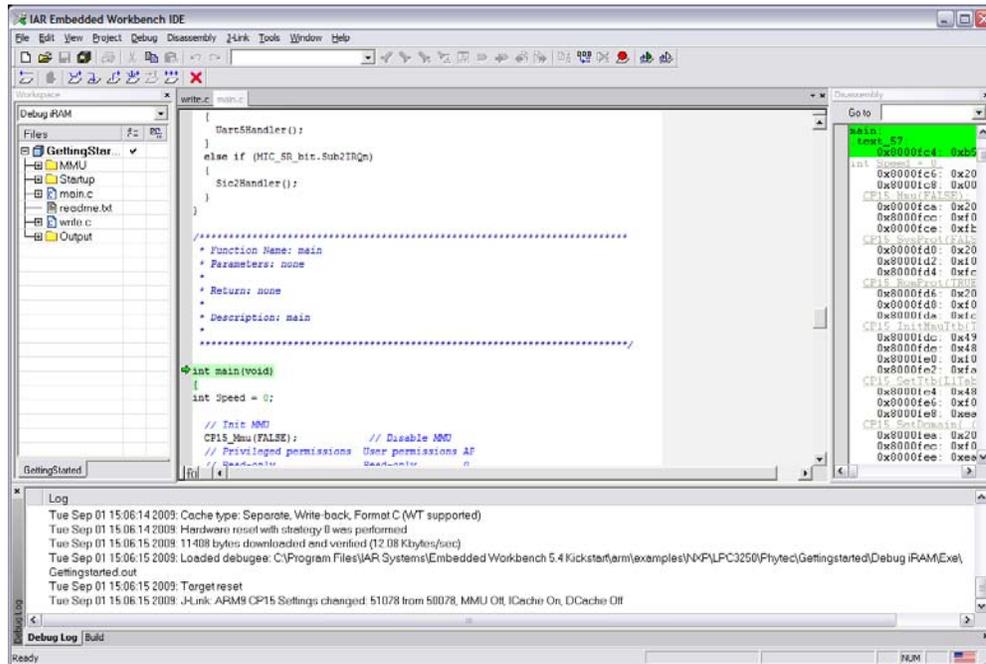
7. Execute the code by selecting the **Go** button from the debug toolbar .

Successful execution of the program will alternately flash LED1 and LED2 (D15 and D18) on the phyCORE® Carrier Board.

8. Switch to the HyperTerminal window.

9. Press the BTN1 button on the phyCORE® Carrier Board and notice the blinking speed has changed and **BTN1 button is pressed** is displayed in the HyperTerminal window.

10. Press the BTN2 button on the phyCORE® Carrier Board and notice the blinking pattern has changed and **BTN2 button is pressed** is displayed in the HyperTerminal window.
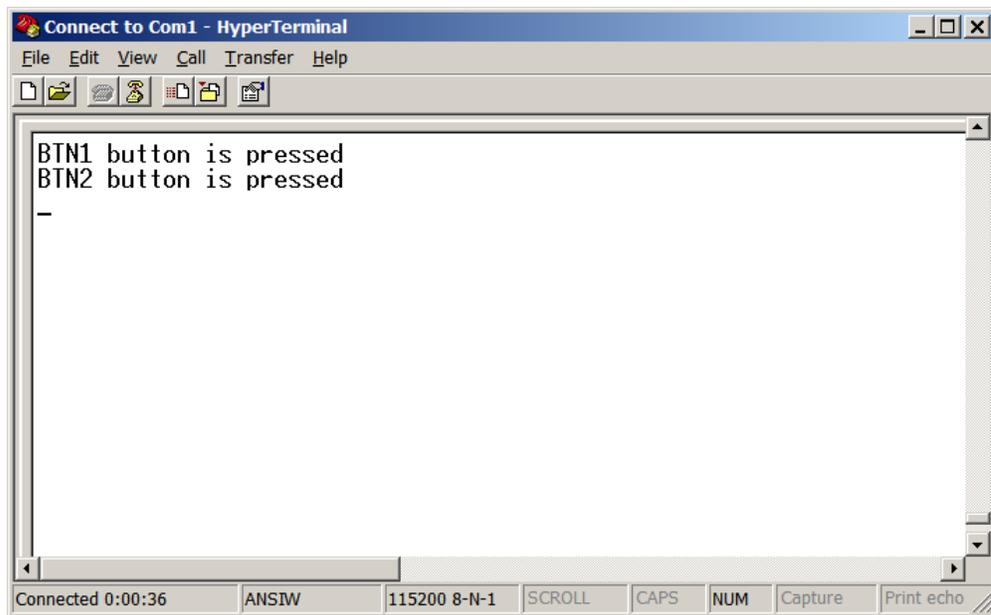
You have successfully downloaded to the on-chip SRAM, executed and monitored user interaction on the phyCORE-LPC3250 hardware via the HyperTerminal window.

4 Debugging

What you will learn with this example:

- how to familiarize yourself with simple debug functions provided by the IAR EWARM debug environment
- how to download example user code to the phyCORE-LPC3250 external SDRAM memory and go through a basic debugging exercise with use of a break point

This Debugging section provides a basic introduction to the debug functions included in the IAR Embedded WorkBench®. For a more detailed description of the debugging features, please refer to the appropriate documentation provided by IAR.

4.1 IAR EWARM Debug Features

The Debugger toolbar gives quick access to the following debug commands from left to right: **Reset, Break, Step Over, Step Into, Step Out, Next Statement, Run to Cursor, Go, and Stop Debugging**
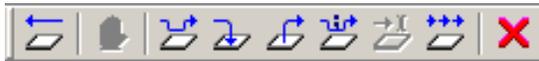
Reset	Resets the target processor.
Break	Stops the application execution.
Step Over	Executes the next statement, function call, or instruction, without entering C or C++ functions or assembler subroutines.
Set Into	Executes the next statement or instruction, entering C or C++ functions or assembler subroutines.
Set Out	Executes from the current statement up to the statement after the call to the current function.
Next Statement	Executes directly to the next statement without stopping at individual function calls.
Run to Cursor	Executes from the current statement or instruction up to a selected statement or instruction.
Go	Executes from the current statement or instruction until a breakpoint or program exit is reached.
Stop Debugging	Stops the debugging session and returns you to the project manager.

For more complete debug menu command descriptions see the ARM IAR Embedded WorkBench® Help by selecting from the IDE menu **Help / Index**.

4.2 Breakpoint Example

The **GettingStarted** demo is configured for button BTN1 to change the blinking speed while button BTN2 will change the pattern. BTN1 is physically connected to GPI_3 and BTN2 is physically connected to GPI_2. The LPC3250 interrupt controller is configured to respond to GPI_3 and GPI_2. The interrupt controller will not respond to BTN1 or BTN2 if the interrupt controller is configured improperly. In the following example we will change the interrupt controller to respond to GPI_4 rather than GPI_3 and with the use of a break-point demonstrate that the interrupt handler will not execute when BTN1 is toggled.

1. From the Embedded Workspace Configuration drop-down menu select the **Debug xRAM** configuration as shown below:

```
Debug xRAM                                      ▼
Debug iRAM
Debug xRAM                                    ⤡
Release iRAM boot from NAND512(64M)
Release iRAM boot from NAND256(32M)
Release xRAM boot from NAND512(64M)
Release xRAM boot from NAND256(32M)
Release iRAM boot level 2 from NAND256(32M)
Release iRAM boot level 2 from NAND512(64M)
```
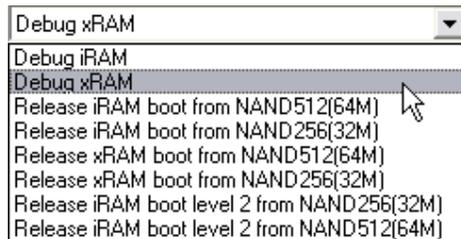
2. Start the debugger and download the code by selecting the **Download and Debug** button from the IDE toolbar .

3. Execute the code by selecting the **Go** button from the debug toolbar .

Successful execution of the program will alternately flash LED1 and LED2 (D15 and D18) on the phyCORE® Carrier Board.

4. Press the BTN1 button on the phyCORE® Carrier Board and notice the blinking speed will change.

5. Stop the code execution by selecting the **Break** button from the debug toolbar .

6. From the Project Workspace open the *main.c* file.

7.   From the IDE menu select **Tools / Options**.

8.   From the IDE Options window select **Editor** and check **Show line numbers.**

9.   You will now see the line numbers in the *main.c* file.

10. Locate the Gpi03_handler function starting on line 126:

```
125 ******************************************************************/
126 static void Gpi03_handler (void)
127 {
128   ButtonDly = BUTTON_DLY;
129   printf("BTN1 button is pressed\r\n");
130   SIC2_RSR_bit.GPI_03 = 1;              // clear interrupt flag (for edge only
131 }
132
```

11. Place the cursor anywhere in the Gpi03_handler function and place a break point by selecting the **Toggle Breakpoint** button in the IDE toolbar 🔴.

```
125 ******************************************************************/
126 static void Gpi03_handler (void)
127 {
128   ButtonDly = BUTTON_DLY;
129   printf("BTN1 button is pressed\r\n");
130   SIC2_RSR_bit.GPI_03 = 1;              // clear interrupt flag (for edge only
131 }
132
```

12. Execute the code by selecting the **Go** button from the debug toolbar.

Successful execution of the program will alternately flash LED1 and LED2 (D15 and D18) on the phyCORE® Carrier Board.

13. Press the BTN1 button on the phyCORE® Carrier Board and notice the execution of Gpio3_handler.

```
125 ******************************************************************/
126 static void Gpi03_handler (void)
127 {
128   ButtonDly = BUTTON_DLY;
129   printf("BTN1 button is pressed\r\n");
130   SIC2_RSR_bit.GPI_03 = 1;              // clear interrupt flag (for edge only
131 }
132
```

14. Stop the debugger by selecting the **Stop Debugging** button from the debug toolbar ❌.

15. Locate the interrupt controller configuration code for GPI_3 beginning on line 282:

```
280    // Init button (BTN1 - GPI3)
281    // Init GPI_03
282    SIC2_APR_bit.GPI_03     = 1;        // Interrupt is generated on a high level s
283    SIC2_ATR_bit.GPI_03     = 1;        // Interrupt is edge sensitive
284    SIC2_ITR_bit.GPI_03     = 0;        // The interrupt is routed to the IRQ outpu
285    SIC2_RSR_bit.GPI_03     = 1;        // clear interrupt flag (for edge only)
286    SIC2_ER_bit.GPI_03      = 1;        // Enable GPI_03 interrupt
287
```

16. Change **GPI_03** to **GPI_04** on line 282-286.

```
280    // Init button (BTN1 - GPI3)
281    // Init GPI_03
282    SIC2_APR_bit.GPI_04     = 1;        // Interrupt is generated on a high level s
283    SIC2_ATR_bit.GPI_04     = 1;        // Interrupt is edge sensitive
284    SIC2_ITR_bit.GPI_04     = 0;        // The interrupt is routed to the IRQ outpu
285    SIC2_RSR_bit.GPI_04     = 1;        // clear interrupt flag (for edge only)
286    SIC2_ER_bit.GPI_04      = 1;        // Enable GPI_03 interrupt
287
```

17. Start the debugger and download the code by selecting the **Download and Debug** button from the IDE toolbar .

18. Execute the code by selecting the **Go** button from the debug toolbar .

19. Again locate the Gpi03_handler function. The breakpoint set in step 11 should still be in place:

```
125   **********************************************************************/
126 static void Gpi03_handler (void)
127 {
128     ButtonDly = BUTTON_DLY;
129     printf("BTN1 button is pressed\r\n");
130     SIC2_RSR_bit.GPI_03 = 1;                    // clear interrupt flag (for edge only]
131 }
132
```

20. Press the BTN1 button on the phyCORE® Carrier Board and notice the Gpi03_handler does not execute and the blinking speed does not change.

Because BTN1 is physically connected to GPI_3, the interrupt controller will not respond to BTN1 if the interrupt controller is configured improperly to GPI_4.

16

21. Return to the interrupt controller configuration code for GPI_3 beginning on line 282 and change **GPI_04** back to **GPI_03** and save changes:

```
280   // Init button (BTN1 - GPI3)
281   // Init GPI_03
282   SIC2_APR_bit.GPI_03    = 1;      // Interrupt is generated on a high level s
283   SIC2_ATR_bit.GPI_03    = 1;      // Interrupt is edge sensitive
284   SIC2_ITR_bit.GPI_03    = 0;      // The interrupt is routed to the IRQ outpu
285   SIC2_RSR_bit.GPI_03    = 1;      // clear interrupt flag (for edge only)
286   SIC2_ER_bit.GPI_03     = 1;      // Enable GPI_03 interrupt
287
```

You have now successfully download example user code in the phyCORE-LPC3250 external SDRAM memory and gone through a basic debugging exercise with use of a break point.

5 Flash Programming

What you will learn with this example:

- how to use the IAR secondary bootloader for the LPC3250.
- downloading example user code to the external Flash memory

5.1 Secondary Bootloader

IAR provides a secondary bootloader for the NAND Flash. To better understand the reasons for the need of a secondary bootloader on the LPC3250 a concise explanation of the boot process is presented below.

After a reset the LPC3250 executes the on-chip bootstrap software located in the on-chip boot ROM. This software is responsible for reading code out of NAND Flash and loading it in internal SRAM and executing it. Because the SRAM is limited to 64kB on the LPC3250, it is not possible to execute code which exceeds 64kB in size out of internal RAM. Further, despite the presence of 64kB iRAM in the LPC3250, the bootstrap software will only copy 15.5kB from the NAND Flash into iRAM for execution.

To get around this limitation code must be executed from external SDRAM which is much greater in size than internal SRAM. To execute code out of SDRAM three basic steps must be followed:

> (1) the SDRAM interface must be initialized, and
> (2) the code must be copied from NAND Flash into SDRAM, and
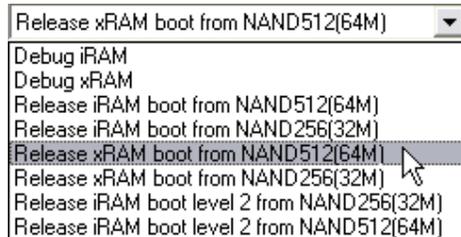> (3) execution must be transferred to SDRAM.

The LPC3250 bootstrap software is not capable of initialization SDRAM, copying code into it from NAND Flash, and executing it. For this reason a secondary bootloader must be implemented on the LPC3250. The secondary bootloader is the first code placed in NAND Flash which the LPC3250 bootstrap software will load into SRAM and execute. This secondary bootloader will then initialized the SDRAM, copy application code from NAND Flash into SDRAM, and then transfer execution to SDRAM.

For a more detailed description of the boot process, *please refer to the phyCORE-LPC3250 Hardware Manual*.

Note: Bad block skipping is supported by the secondary bootloader provided with this Rapid Development Kit.

5.2 Loading Example to Flash

1. From the Embedded Workspace Configuration drop-down menu select the **Debug xRAM boot from NAND512(64M)** configuration as shown below:

```
Release xRAM boot from NAND512(64M)          ▼
Debug iRAM
Debug xRAM
Release iRAM boot from NAND512(64M)
Release iRAM boot from NAND256(32M)
Release xRAM boot from NAND512(64M)
Release xRAM boot from NAND256(32M)
Release iRAM boot level 2 from NAND256(32M)
Release iRAM boot level 2 from NAND512(64M)
```

2. This target will program both the secondary bootloader and the application code to the on-board NAND Flash.

3. Start a HyperTerminal session with the following COM settings:

> **115200 bps**
> **8 bit**
> **No parity**
> **1 bit stop**
> **No flow control**

4. Start the debugger and download the code by selecting the **Download and Debug** button from the IDE toolbar .

5. Execute the code by selecting the **Go** button from the debug toolbar .

6. Stop the debugger by selecting the **Stop Debugging** button from the debug toolbar .

7. Press the RESET (S1) button on the phyCORE® Carrier Board. The example code downloaded to the phyCORE external NAND Flash memory should execute and flash LED1 and LED2 (D15 and D18) on the phyCORE® Carrier Board.

8. Switch to the HyperTerminal window.

9. Press the BTN1 button on the phyCORE® Carrier Board and notice the blinking speed has changed and **BTN1 button is pressed** is displayed in the HyperTerminal window.

10. Press the BTN2 button on the phyCORE® Carrier Board and notice the blinking pattern has changed and **BTN2 button is pressed** is displayed in the HyperTerminal window.

You have now successfully download example user code in the phyCORE-LPC3250 external NAND Flash memory.

6 Getting More Involved

What you will learn with this example:

- how to create a new project

6.1 Create a new Project

For ease of creating a new project for the phyCORE-LPC3250 it is recommended that the given **GettingStarted** workspace is used as a template. The **GettingStarted** workspace already has project examples with the target, linker, compiler, debugger, and startup code correctly configured for the phyCORE-LPC3250 internal RAM, external SDRAM, and NAND Flash target. Here are some suggestions for getting you started:

- Create a new directory for your workspace and copy the contents of **GettingStarted** into the new directory.

- From the new **Workspace** you can manage components of your project and add or remove source files, add code libraries, etc…

Advanced users can modify the compiler, assembler, debugger and linker settings by selecting *Project / Options* in the IDE toolbar menu.

7 Micrium uC/OS-II Demo

What you will learn with this example:

- installing and downloading the **µC**/**OS-II** example demo
- enabling the **µC**/**OS-II** Kernel Awareness Plug-In in IAR EWARM
- using **µC**/**Probe** as a debugging and monitoring tool.

7.1 Installing the Example Files

Code for **µC**/**OS-II** is provided in source form along with IAR EWARM project files that allows you to run **µC**/**OS-II** on the PHYTEC phyCORE-LPC3250 Rapid Development Kit. To install the software, simply run the self-extracting executable.

The self-extracting executable is called   **Micrium-NXP-uCOS-II-LPC3250-Phytec.exe**.

You will be prompted to accept the simple terms of the licensing agreement. If you answer 'Yes', the software will be installed on your PC under the ***Micrium*** directory.

7.2 Using the IAR Micrium Example Project

To view the example project, start an instance of IAR Embedded WorkBench®, and open the project file ***LPC320-Phytec-OS-Probe-v5-2.ewp***, located in the directory:

> ***Micrium\\Software\\EvalBoards\\NXP\\LPC3250-Phytec\\IAR\\OS-Probe***

To do this, use the ***Add Existing Project...*** menu command under the ***Projec****t* menu as shown below.

Note: The example project, ***LPC320-Phytec-OS-Probe-v5-2.ewp****,* is intended for IAR EWARM v5.2x.

The project includes two configurations, which can be selected from the drop-down menu above the workspace explorer:

- iRAM: This project option is configured to load the code into the processor's internal RAM (iRAM) and generate an ELF-file used by **µC**/**Probe.**

- XRAM: This project option is configured to load the code into the board's external SDRAM (XRAM) and generate an ELF-file used by **µC**/**Probe.**

Published by

PHYTEC
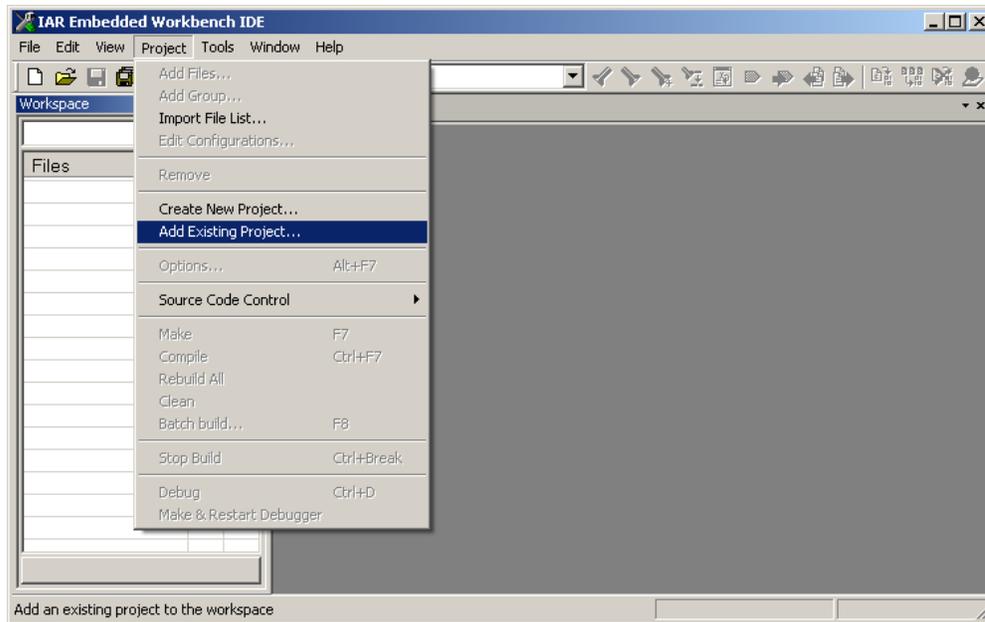
To build the code, select the **Rebuild All** menu item from the **Project** menu. To load the code through the JTAG debugger onto the connected evaluation board, select the **Debug** menu item from the **Project** menu.

7.3 Enabling the uC/OS-II Kernel Awareness

When running the IAR C-Spy debugger, the **µC/OS-II** Kernel Awareness Plug-In can be used to provide useful information about the status of **µC/OS-II** objects and tasks. If the **µC/OS-II** Kernel Awareness Plug-In is currently enabled, then a *µC/OS-II* menu should be displayed while debugging. Otherwise, the plug-in can be enabled.

- Stop the debugger (if it is currently active) and select the *Options* menu item from the *Project* menu.

- Select the **Debugger** entry in the list box and then select the *Plugins* tab pane.

- Find the **µC/OS-II** entry in the list and select the check box beside the entry, as shown below.

When the code is reloaded onto the phyCORE-LPC3250, the *µC/OS-II* menu should appear. Options are included to display lists of kernel objects such as semaphores, queues, and mailboxes, including for each entry the state of the object. Additionally, a list of the current tasks may be displayed, including pertinent information for each task such as used stack space, task status, and task priority, in addition to showing the actively executing task.

Published by

PHYTEC

Ordering No. L-718e_2
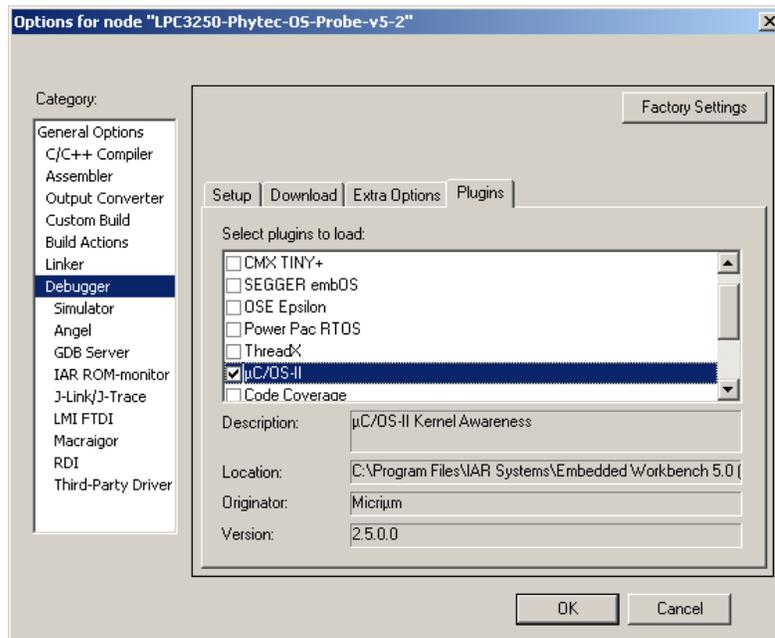
# 7.4 Running the Example Application

The example projects include a basic demonstration of the **µC/OS-II** and **µC/Probe**. After you load the sample project to the phyCORE-LPC3250, the LEDs will start to blink.

Text will be output to the lower RS-232 port (at 115200 baud) giving the state of the system as shown in the HyperTerminal screen capture below. The push buttons can be used to change the output to a new item.

To communicate with the board through RS-232, connect a serial cable between the phyCORE Rapid Development Kit hardware and your host-PC and open a HyperTerminal window (often located in the start menu, on the **Communications** submenu of the **Accessories**). Configure the RS-232 interface with the following settings:

- Bits per Second: 115200

- Data bits 8

- Parity: None

- Stops bits: 1

- Flow Control: None

The upper RS-232 port (at 115200 baud) is used for **µC/Probe**, which allows you to view (in real-time) the value of static / global variables in the target system. A 30-day trial version of this program can be downloaded from the Micriµm website.
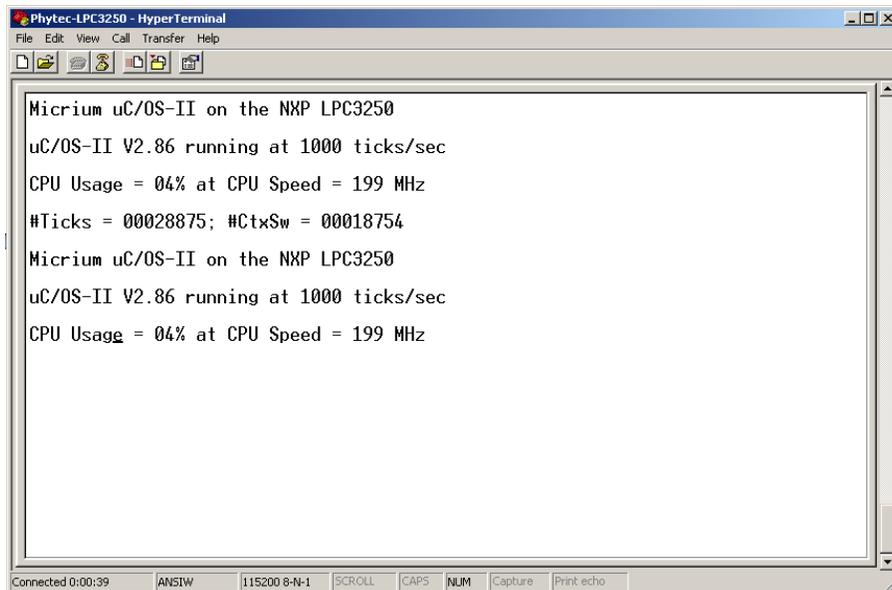
Published by

PHYTEC

7.5 Using the uC/Probe Example Workspace

µC/Probe is a Windows program which retrieves the values of global variables from a connected embedded target and displays the values in an engineer-friendly format. To accomplish this, an ELF file, created by the user's compiler and containing the names and addresses of all the global symbols on the target, is monitored by **µC/Probe**. The user places components (such as gauges, labels, and charts) into a Data Screen in a **µC/Probe** workspace and assigns each one of these a variable from the Symbol Browser, which lists all symbols from the ELF file. The symbols associated with components placed on an open Data Screen will be updated after the user presses the start button (assuming the user's PC is connected to the target).

To use **µC/Probe** with the example project (or your application), do the following:

- **Download and Install µC/Probe.** A trial version of **µC/Probe** can be downloaded from the Micriµm website at

   http://www.micrium.com/products/probe/probe.html

- **Open µC/Probe**. After downloading and installing this program, open the example **µC/Probe** workspace for **µC/OS-II** *LPC3250-Phytec-OS-Probe-Workspace.wsp*, which should be located in your project directory at:

   \Micrium\Software\EvalBoards\NXP\LPC3250-Phytec\IAR\OS-Probe

- **Connect Target to PC**. You should connect a RS-232 cable between your target (upper RS-232 port) and your computer.

- **Load Your ELF File**. The example projects described in this QuickStart are already configured to output an ELF file. (If you are using your own project, please refer to Appendix A of the **µC/Probe** user manual for directions for generating an ELF file with your compiler.) This file should be in

   /<Project Directory>/<Configuration Name>/exe/

   where *<Project Directory>* is the directory in which the IAR EWARM project is located (extension *.ewp) and *<Configuration Name>* is the name of the configuration in that project which was built to generate the ELF file and which will be loaded onto the target. The ELF file will be named
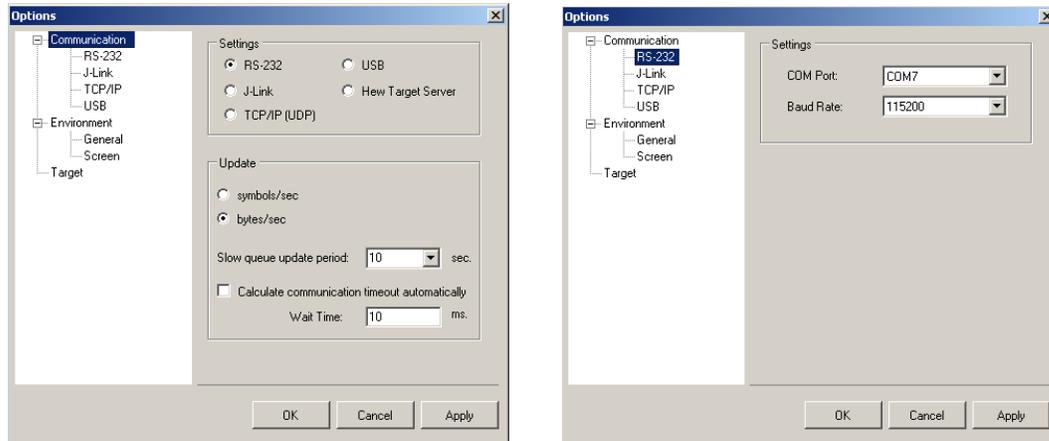
   <Project Name>.out

   To load this ELF file, right-click on the symbol browser and choose "Add Symbols".

- **Configure the RS-232 Options**. In **µC/Probe**, choose the *Options* menu item on the *Tools* menu. A dialog box as shown below, left, should appear. Choose the *RS-232* radio button.

Published by   **PHYTEC**

- Next, select the *RS-232* item in the options tree, and choose the appropriate COM port and baud rate.

1. **Start Running**. You should now be ready to run **µC/Probe**. Just press the run button (▶) to see the variables in the open data screens update. Figure 7-3 displays the phyCORE-LPC3250 IAR Rapid Development Kit data screen where the pushbuttons state and the serial output are being monitored. In addition to this data screen there are **µC/OS-II** data screens where detailed information about each task is displayed.

Published by

PHYTEC

You have now successfully installed and downloaded the **µC/OS-II** example demo, enabled the **µC/OS-II** Kernel Awareness Plug-In in IAR EWARM and used **µC/Probe** as a debugging and monitoring tool

Published by