P&E Microcomputer Systems, Inc. PKGPPCNEXUS

Quick Start Guide
for the
PHYTEC phyCORE-MPC5554
Rapid Development Kit

Visit us on the web: www.pemicro.com

CONTENTS

1 Topics Included In This Guide

This document is a step-by-step guide to using PKGPPCNEXUS with the Phytec phyCORE-MPC5554 evaluation board. The guide covers creating a C language project, compiling the source code with the GNU compiler, programming code to internal Flash memory and internal SRAM, and debugging with the P&E ICDPPCNEXUS in-circuit debugger.

The "PKGPPCNEXUS User's Manual" includes detailed information about using WinIDE to configure the GNU compiler. The "Application Framework and Solutions Guide" summarizes the process of customizing the target application for your system.

2 PKGPPCNEXUS - Software Overview

The P&E PKGPPCNEXUS Windows development environment for PowerPC Nexus includes software to edit and compile C source code for the PowerPC Nexus family of processors that use the e200z6 core. PKGPPCNEXUS contains software for programming Flash and debugging with a PowerPC Nexus target system.

This version of PKGPPCNEXUS provides templates for building an application for the MPC5553 and MPC5554 processors. Although the templates are intended to be used with select evaluation boards, you may use PKGPPCNEXUS to create an application for any MPC5553 or MPC5554 board. The GNU compiler supports any PowerPC processor that implements the e200z6 core or e200z3 core operating in non-VLE mode.

PKGPPCNEXUS includes the following software:

WinIDE – Integrated Development Environment for the PowerPC Nexus architecture.

This is the primary editing environment for the software kit. It is the launch pad for other applications, such as the P&E in-circuit debugger, P&E in-circuit Flash programmer, and the GNU C compiler. Control the settings for the GCC compiler within the WinIDE environment.

ICDPPCNEXUS – In-Circuit Debugger for PowerPC Nexus series devices.

This debugger supports reading and writing memory and variables, loading code to RAM, controlling processor execution, and more. Use the debugger to step through both assembly language and C language source code. The debugger loads the Elf/Dwarf (version 2.0) debug file format, which is output by many compilers, such as the GNU compiler included in this package.

PowerPC EABI SPE GCC Compiler – Windows GNU C compiler for select PowerPC 55xx targets.

Without requiring a UNIX shell, this compiler executes under the Windows operating system. WinIDE controls the operation of the compiler, providing one-touch compilation of a C project. The GNU software produces both the Elf/Dwarf 2.0 and S19 file formats. P&E's in-circuit Flash programmer will program with the S19 file, and P&E's in-circuit debugger will use the Elf/Dwarf file for debugging.

The GNU toolset includes the following components:

GCC version 3.4.4
Binutils version 2.15.91
Newlib version 1.13.0

Both GCC and Binutils have been patched to support the e200z6 core and e200z3 core operating in non-VLE mode.

PROGPPCNEXUS – In-circuit Flash programmer for PowerPC Nexus.

Register File Viewer for PowerPC Nexus

This software simplifies viewing and editing PowerPC registers and register bits.

3 Installing the PKGPPCNEXUS Development Environment

You must have administrator privilege on the Windows 95/98/ME/NT/XP computer that will host the software. After launching the installer, follow the instructions to install all of the software in the default location, C:\pemicro\pkgppcnexus.

Take note of several subdirectories in the PKGPPCNEXUS installation directory:

\gnu
GNU/GCC software (P&E Build)

\gnu\bin
GNU/GCC executables (P&E Build)

3.1 Changing the Default Installation Location

The default installation location for PKGPPCNEXUS is C:\pemicro\pkgppcnexus. Using the installer software, you may install the software at any location.

We do not recommend that you manually change the location of PKGPPCNEXUS after installation. Do not copy-and-past the GNU compiler to a different location. However, you may place your source code and WinIDE project files at any location.

4 PKGPPCNEXUS – Quick Start Guide

The following sections walk you through select tasks using PKGPPCNEXUS. *For more information regarding each of the P&E software components, see the Windows help files for WinIDE, ICDPPCNEXUS, and PROGPPCNEXUS.*

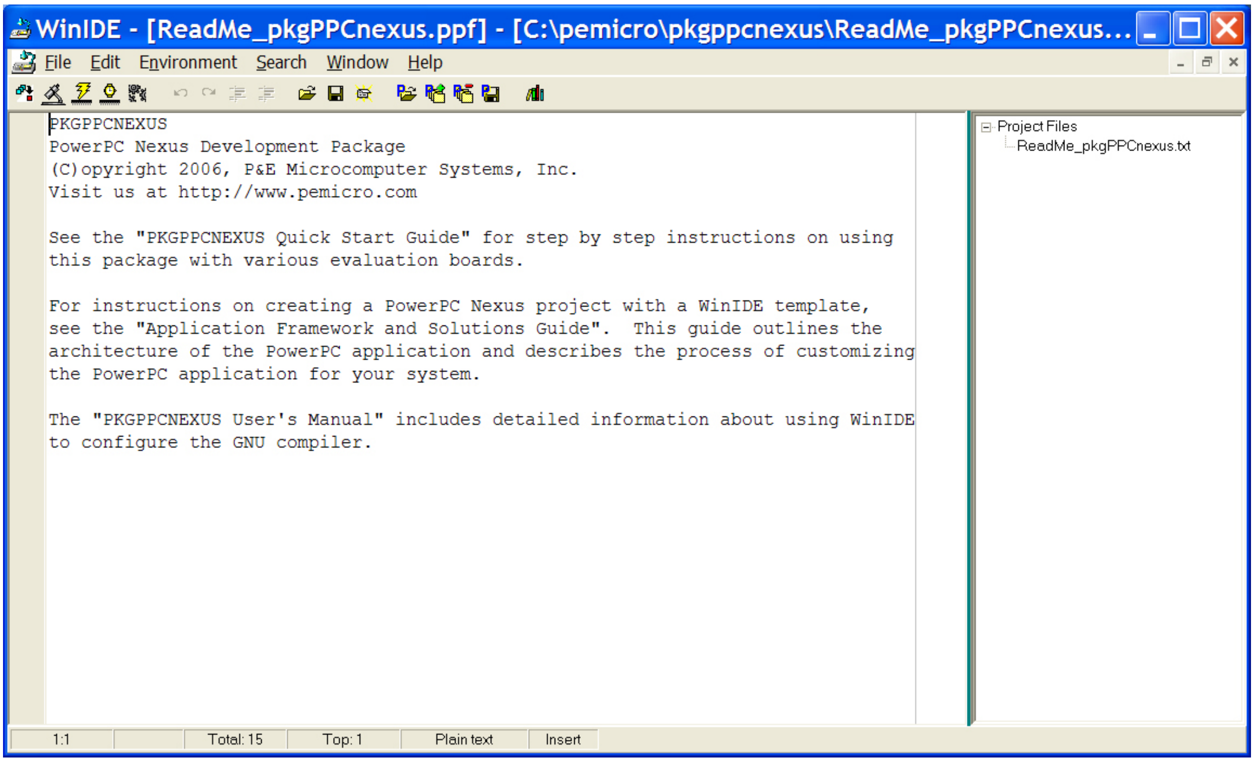
4.1 Configuring the Target Board

There are templates to create projects for both internal Flash and internal SRAM using the phyCORE-MPC5554 board. First, we will create a project that runs in internal Flash. Be sure that your target board is configured to boot from internal Flash memory. Later, we will create a project for RAM only.

4.2 Starting WinIDE

After installing PKGPPCNEXUS, launch the WinIDE development environment by using the Windows Start menu. Select the WinIDE icon in the program group for the P&E PowerPC Nexus kit. lternatively, double-click on the **WinIDE32.exe** icon in the installation directory, ***C:\pemicro\pkgppcnexus.***

When the WinIDE development environment launches for the first time, it will display a file containing a description of the documentation included with PKGPPCNEXUS.


WinIDE - [ReadMe_pkgPPCnexus.ppf] - [C:\pemicro\pkgppcnexus\ReadMe_pkgPPCnexus...
File Edit Environment Search Window Help
Project Files
ReadMe_pkgPPCnexus.txt
1:1
Total: 15
Top: 1
Plain text
Insert

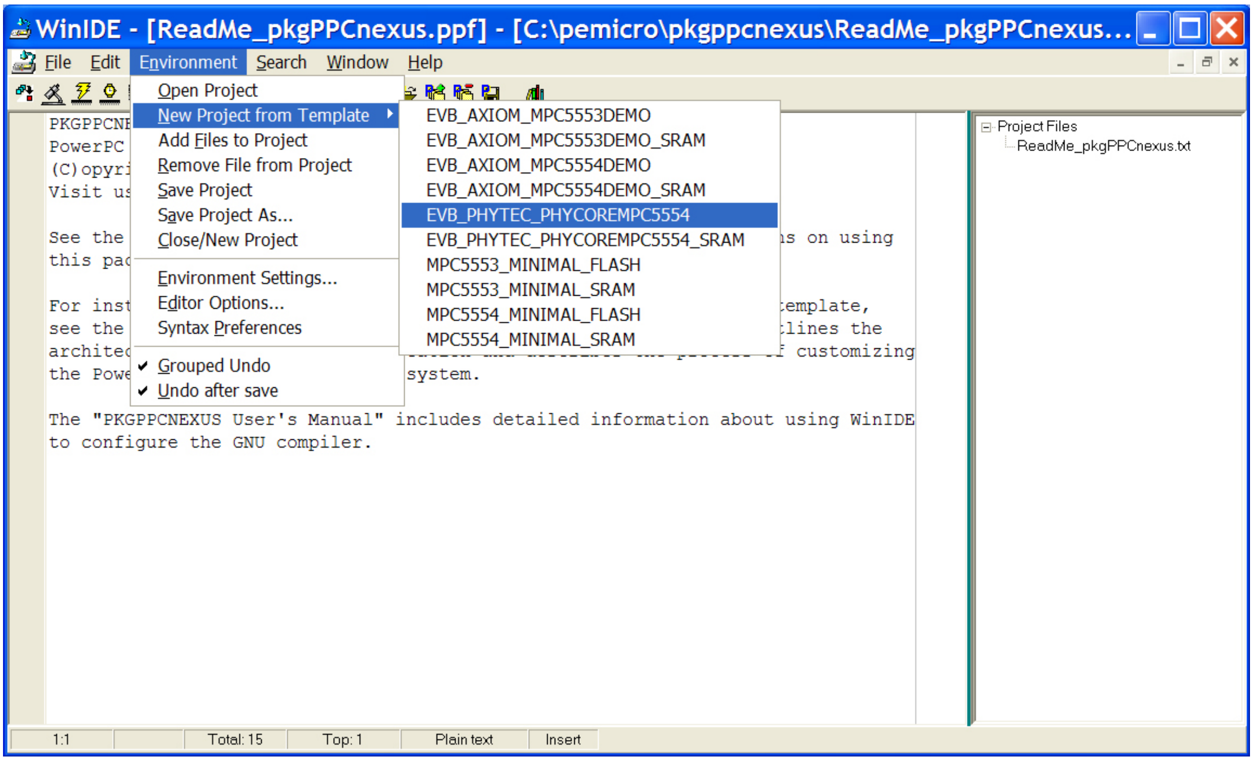
```
PKGPPCNEXUS
PowerPC Nexus Development Package
(C)opyright 2006, P&E Microcomputer Systems, Inc.
Visit us at http://www.pemicro.com

See the "PKGPPCNEXUS Quick Start Guide" for step by step instructions on using
this package with various evaluation boards.

For instructions on creating a PowerPC Nexus project with a WinIDE template,
see the "Application Framework and Solutions Guide".  This guide outlines the
architecture of the PowerPC application and describes the process of customizing
the PowerPC application for your system.

The "PKGPPCNEXUS User's Manual" includes detailed information about using WinIDE
to configure the GNU compiler.
```

A WinIDE project file contains the names of user files associated with a project as well as a wide range of compilation and editing settings. The WinIDE project name appears in the title bar of the main WinIDE window, shown as ReadMe_pkgPPCnexus.ppf, above.

4.3 Creating a New Project with a WinIDE Template

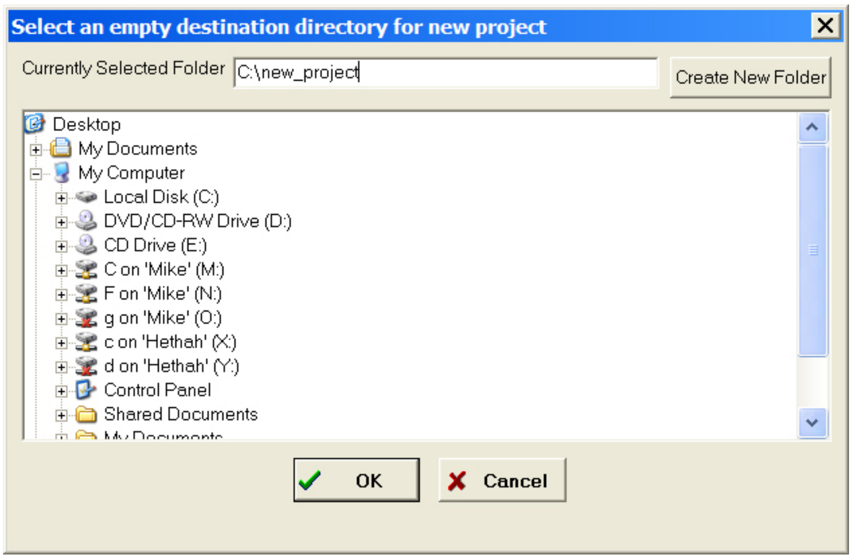
Use WinIDE's templates to create a C language project for your target system. Use the template to create a new WinIDE project and to automatically configure the GNU C compiler. After WinIDE copies default source files to your new project folder, it will tailor the compiler settings for your target system.

Begin by selecting the WinIDE Environment menu, New Project from Template. A list of processors or evaluation boards will appear. Select the template "***EVB_PHYTEC_PHYCOREMPC5554***". This will create a project that will reside in internal Flash and utilize internal SRAM and cache. We will discuss the template "EVB_PHYTEC_PHYCOREMPC5554_SRAM" later.



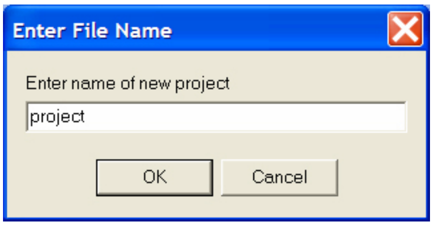
```
PKGPPCNE
PowerPC
(C)opyri
Visit us

See the                                                   ns on using
this pac

For inst                                                  template,
see the                                                   lines the
architec                                                  f customizing
the Powe                      system.

The "PKGPPCNEXUS User's Manual" includes detailed information about using WinIDE
to configure the GNU compiler.
```

A dialog will ask you to select an empty directory for the new project.

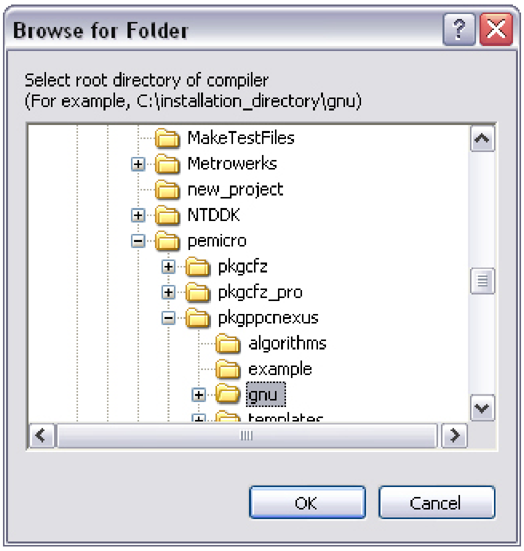


Instead of selecting an existing directory, type the path and name of a new folder in the edit box. Then, hit the Create New Folder button, then ***OK***. WinIDE will create the new, empty folder.

Next, type the name of your new project file.



WinIDE will ask you for the root directory of the GNU compiler. In most cases, WinIDE will automatically select the correct folder. The default compiler root directory is ***C:\pemicro\pkgppcnexus\gnu.***



WinIDE will ask if you want to save the initial project, ReadMe_pkgPPCnexus.ppf; select ***No*** or ***Yes***.

The default project files will now be located in the new project directory, ***C:\new_project***. WinIDE will open the new WinIDE project file, project.PPF.

4.4 Files Included in the Template Project

A PowerPC Nexus template creates a project whose source code is very similar to the Freescale code for the processor. Freescale publishes this processor initialization code on the Freescale website. Differences between the Freescale source code and the P&E code are clearly noted in the P&E source code.

The P&E template includes Freescale's header files and initialization code for the PowerPC processor, as well as P&E's linker script and WinIDE project file. Take note of several subdirectories and files within the new project directory:

Subdirectory \headers

This folder contain Freescale header files for the PowerPC 55xx.

Subdirectory \src

This includes the Freescale initialization source code for the PowerPC 55xx.

Subdirectory \PEGnu

The folder contains the P&E linker script, WinIDE project file, and compiler output files, such as the ELF/DWARF, s-record, and compiler error log files.

P&E GNU Linker Script (*.ld)

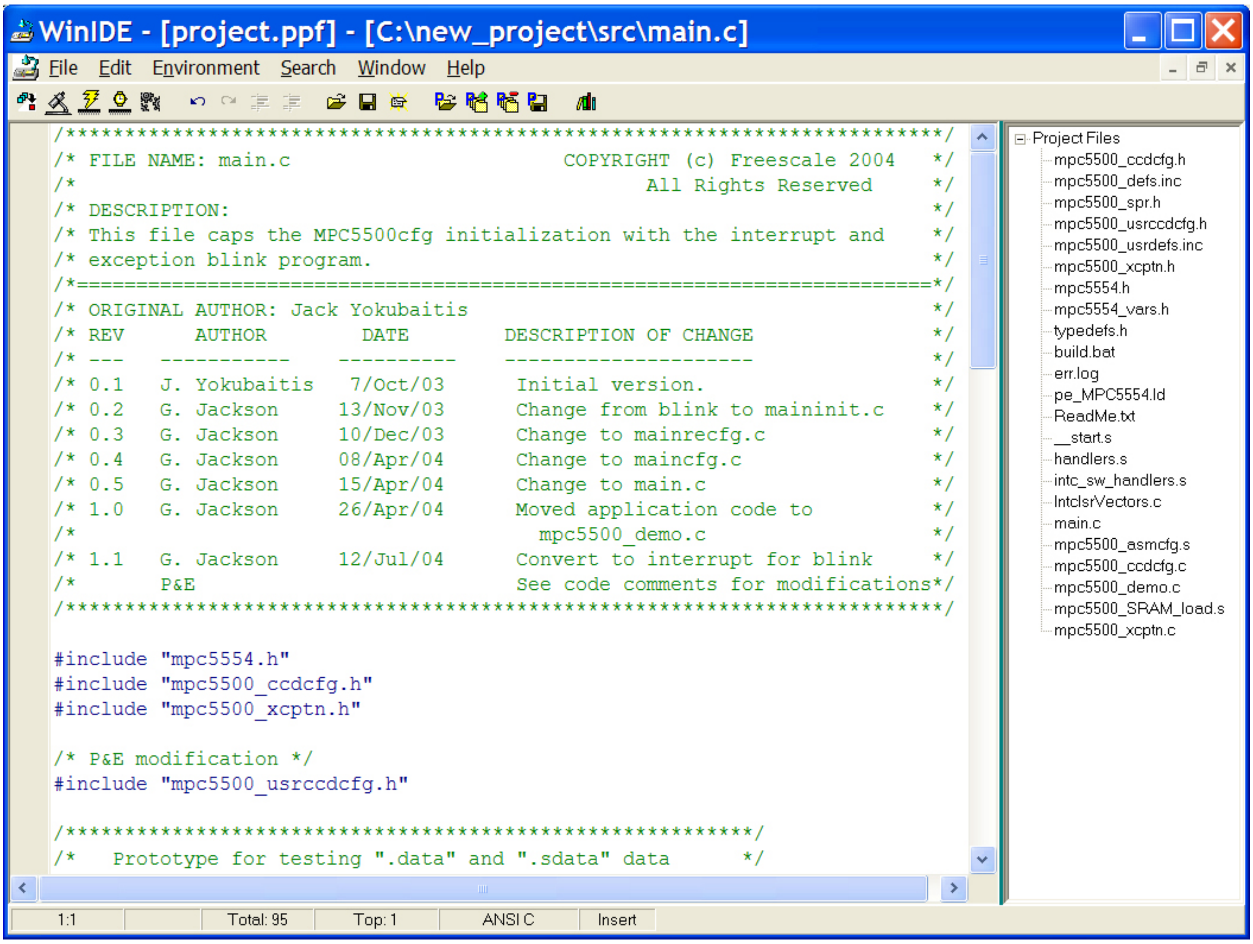
This file works in conjunction with the Freescale code to control the memory map of the PowerPC application. The top of the linker script contains user defined settings, such as stack location and other memory settings.

WinIDE Project File (*.ppf)

This file contains all of the WinIDE settings, including compiler settings. Do not alter this file manually in a text editor.

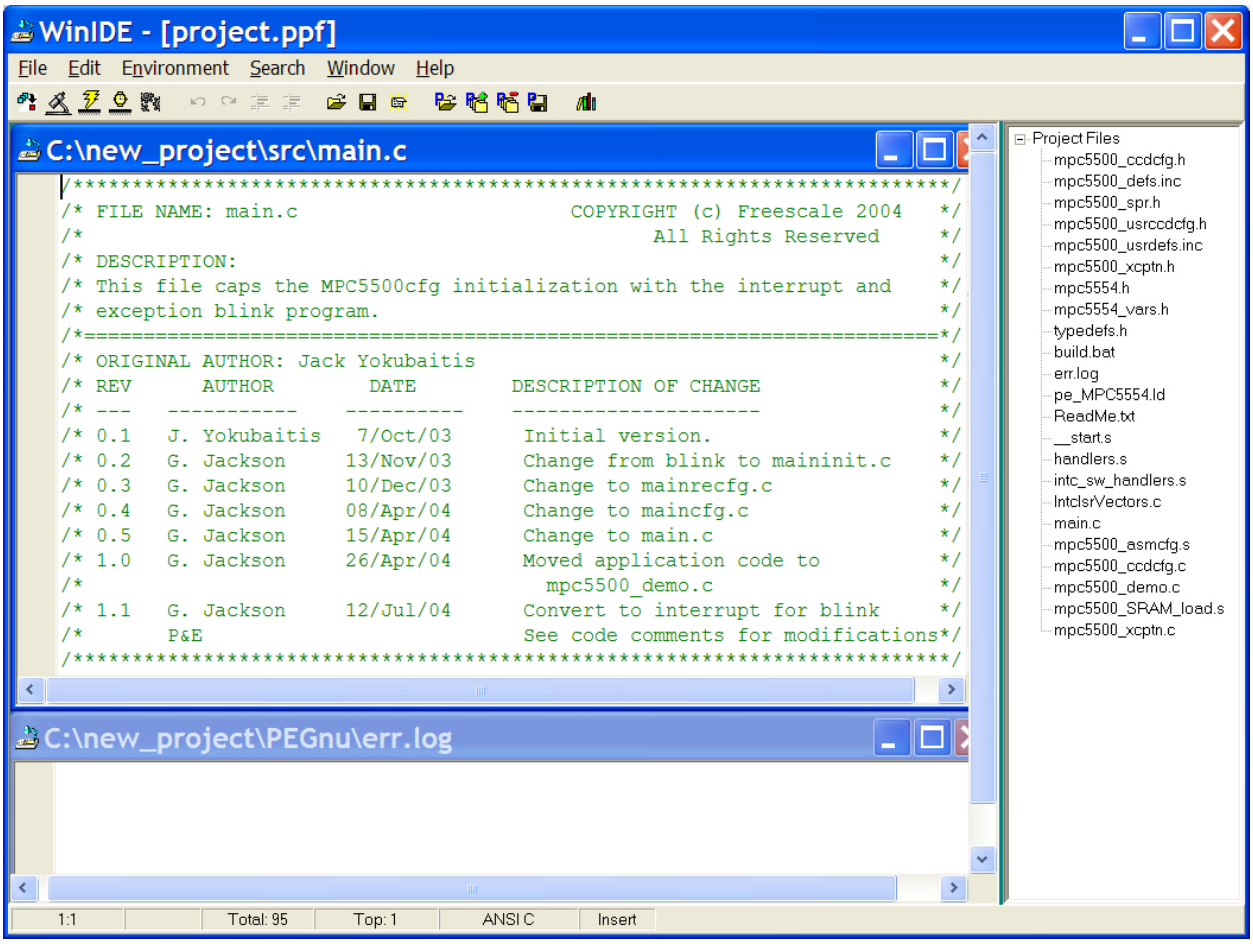
4.5 Windows Layout for C-compilation

After creating your new project with a template, WinIDE will open the project file. ***Close*** the initial file, **ReadMe.txt,** which appears. WinIDE's window contains two areas, a left window pane to display open project files and a right window pane to display the names of the files in the current WinIDE project. Open the file, ***main.c***, whose file name appears on the right pane. Click on the file name to open ***main.c***, maximize the file, and WinIDE will appear as below:



```
/**************************************************************************/
/* FILE NAME: main.c                       COPYRIGHT (c) Freescale 2004   */
/*                                                All Rights Reserved     */
/* DESCRIPTION:                                                           */
/* This file caps the MPC5500cfg initialization with the interrupt and    */
/* exception blink program.                                               */
/*========================================================================*/
/* ORIGINAL AUTHOR: Jack Yokubaitis                                       */
/* REV      AUTHOR        DATE        DESCRIPTION OF CHANGE               */
/* ---   -----------    ----------    ---------------------               */
/* 0.1   J. Yokubaitis   7/Oct/03      Initial version.                   */
/* 0.2   G. Jackson     13/Nov/03      Change from blink to maininit.c    */
/* 0.3   G. Jackson     10/Dec/03      Change to mainrecfg.c              */
/* 0.4   G. Jackson     08/Apr/04      Change to maincfg.c                */
/* 0.5   G. Jackson     15/Apr/04      Change to main.c                   */
/* 1.0   G. Jackson     26/Apr/04      Moved application code to          */
/*                                       mpc5500_demo.c                   */
/* 1.1   G. Jackson     12/Jul/04      Convert to interrupt for blink     */
/*       P&E                           See code comments for modifications*/
/**************************************************************************/


#include "mpc5554.h"
#include "mpc5500_ccdcfg.h"
#include "mpc5500_xcptn.h"

/* P&E modification */
#include "mpc5500_usrccdcfg.h"

/**********************************************************/
/*   Prototype for testing ".data" and ".sdata" data      */
```

The file, ***err.log***, will contain the output messages of the GNU compiler, including warnings and errors. If this file is not already open during compilation, WinIDE will automatically open ***err.log.*** Opening the error log before running GCC will maintain the log's state, its window size and location. Open the error log and arrange the windows, as pictured below:



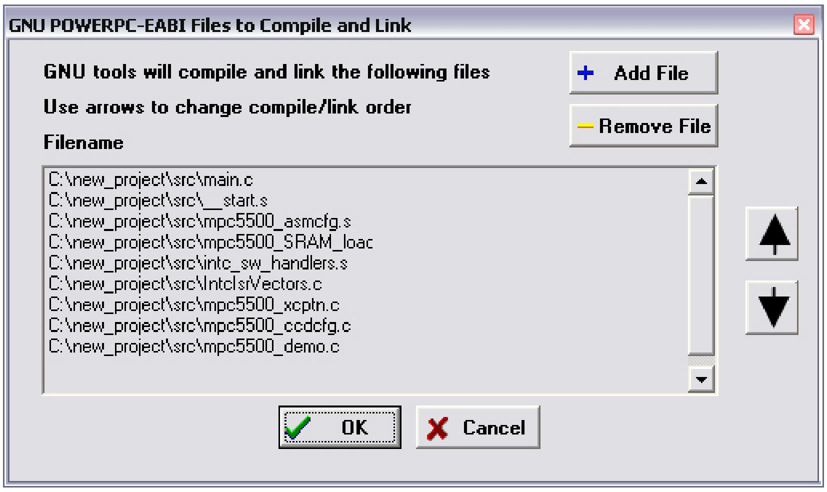
```
/*************************************************************************/
/* FILE NAME: main.c                          COPYRIGHT (c) Freescale 2004 */
/*                                                   All Rights Reserved  */
/* DESCRIPTION:                                                           */
/* This file caps the MPC5500cfg initialization with the interrupt and    */
/* exception blink program.                                               */
/*========================================================================*/
/* ORIGINAL AUTHOR: Jack Yokubaitis                                       */
/* REV      AUTHOR         DATE         DESCRIPTION OF CHANGE             */
/* ---    -----------    ----------    ---------------------              */
/* 0.1    J. Yokubaitis   7/Oct/03      Initial version.                  */
/* 0.2    G. Jackson     13/Nov/03      Change from blink to maininit.c   */
/* 0.3    G. Jackson     10/Dec/03      Change to mainrecfg.c             */
/* 0.4    G. Jackson     08/Apr/04      Change to maincfg.c               */
/* 0.5    G. Jackson     15/Apr/04      Change to main.c                  */
/* 1.0    G. Jackson     26/Apr/04      Moved application code to         */
/*                                        mpc5500_demo.c                  */
/* 1.1    G. Jackson     12/Jul/04      Convert to interrupt for blink    */
/*        P&E                           See code comments for modifications*/
/*************************************************************************/
```

- Project Files
  - mpc5500_ccdcfg.h
  - mpc5500_defs.inc
  - mpc5500_spr.h
  - mpc5500_usrccdcfg.h
  - mpc5500_usrdefs.inc
  - mpc5500_xcptn.h
  - mpc5554.h
  - mpc5554_vars.h
  - typedefs.h
  - build.bat
  - err.log
  - pe_MPC5554.ld
  - ReadMe.txt
  - __start.s
  - handlers.s
  - intc_sw_handlers.s
  - IntcIsrVectors.c
  - main.c
  - mpc5500_asmcfg.s
  - mpc5500_ccdcfg.c
  - mpc5500_demo.c
  - mpc5500_SRAM_load.s
  - mpc5500_xcptn.c

1:1 | Total: 95 | Top: 1 | ANSI C | Insert

4.6 Viewing Source Files to Compile - The Build Dialog

WinIDE project files appear in WinIDE's right window pane. These files are not automatically passed to the GNU compiler. To view the files that will be compiled, select WinIDE's Environment menu, Environment Settings, Assembler/Compiler tab, Build Dialog. Add files, remove files, or change the compilation order from within this dialog. It is not necessary to alter the Build Dialog for this example.



GNU POWERPC-EABI Files to Compile and Link

GNU tools will compile and link the following files

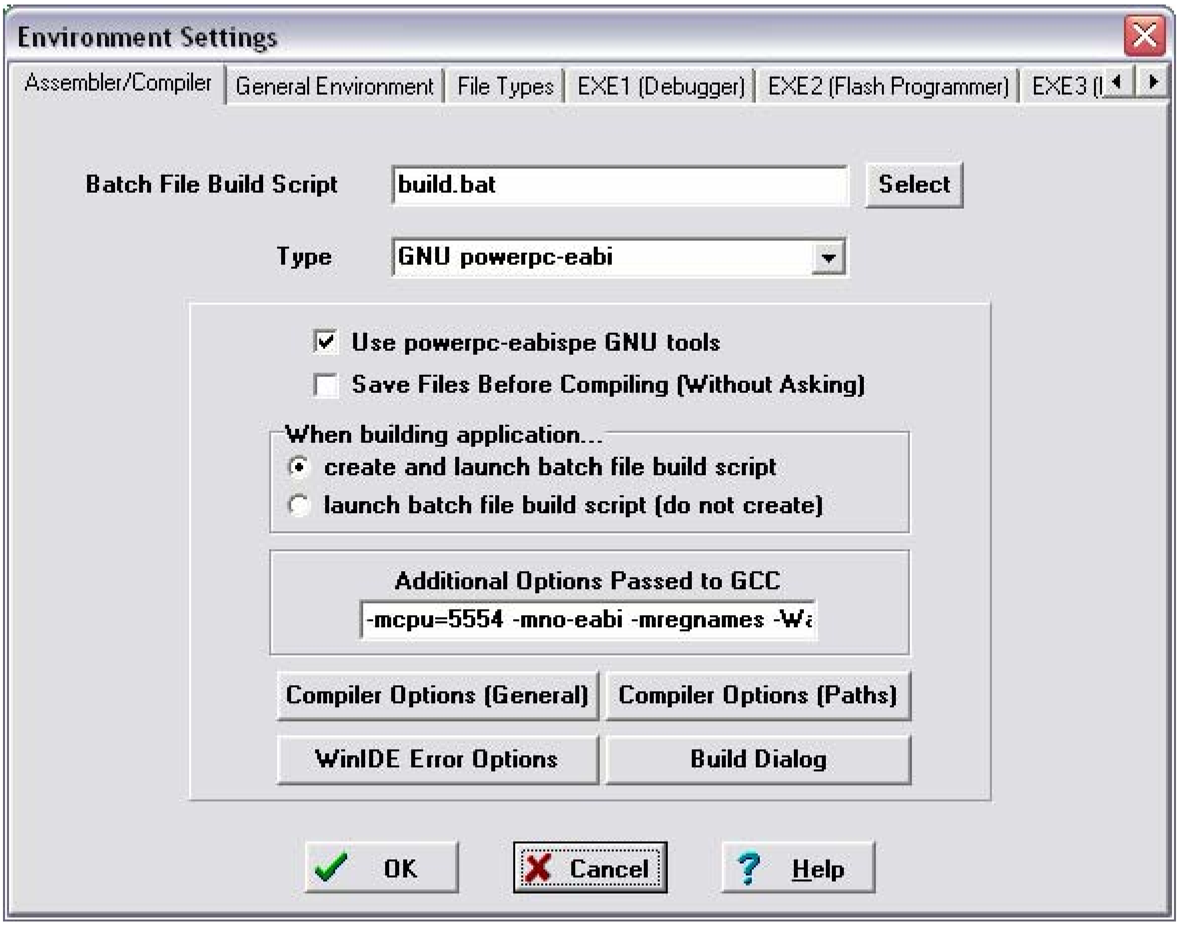
Use arrows to change compile/link order

Filename

```
C:\new_project\src\main.c
C:\new_project\src\__start.s
C:\new_project\src\mpc5500_asmcfg.s
C:\new_project\src\mpc5500_SRAM_loac
C:\new_project\src\intc_sw_handlers.s
C:\new_project\src\IntcIsrVectors.c
C:\new_project\src\mpc5500_xcptn.c
C:\new_project\src\mpc5500_ccdcfg.c
C:\new_project\src\mpc5500_demo.c
```

4.7 Viewing Compiler Options

WinIDE configures the GNU compiler to build the PowerPC application. To view the compiler settings, select WinIDE's Environment menu, Environment Settings, Assembler/Compiler tab.



Batch File Build Script: build.bat

Type: GNU powerpc-eabi

☑ Use powerpc-eabispe GNU tools

□ Save Files Before Compiling (Without Asking)

When building application...

● create and launch batch file build script

○ launch batch file build script (do not create)

Additional Options Passed to GCC: -mcpu=5554 -mno-eabi -mregnames -W

Use this screen, including the Compiler Options (General), Compiler Options (Paths), WinIDE Error Options, and Build Dialog, to configure the GNU compiler. It is not necessary to change the GNU compiler settings for this example.

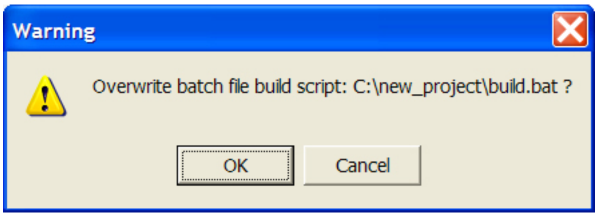
4.8 Compiling the Application, Interpreting the Result

The default memory map for our example uses on-chip Flash memory, internal SRAM, and cache. For this example, use the default memory map, making no changes to the P&E linker script or the Freescale source files.

To compile the PowerPC Nexus application, either press the <F4> function key or click the compilation button on the tool bar at the top of the WinIDE screen.

When compiling, WinIDE will generate a Windows batch file to control the GNU compiler. The batch file will contain the compiler settings from WinIDE.

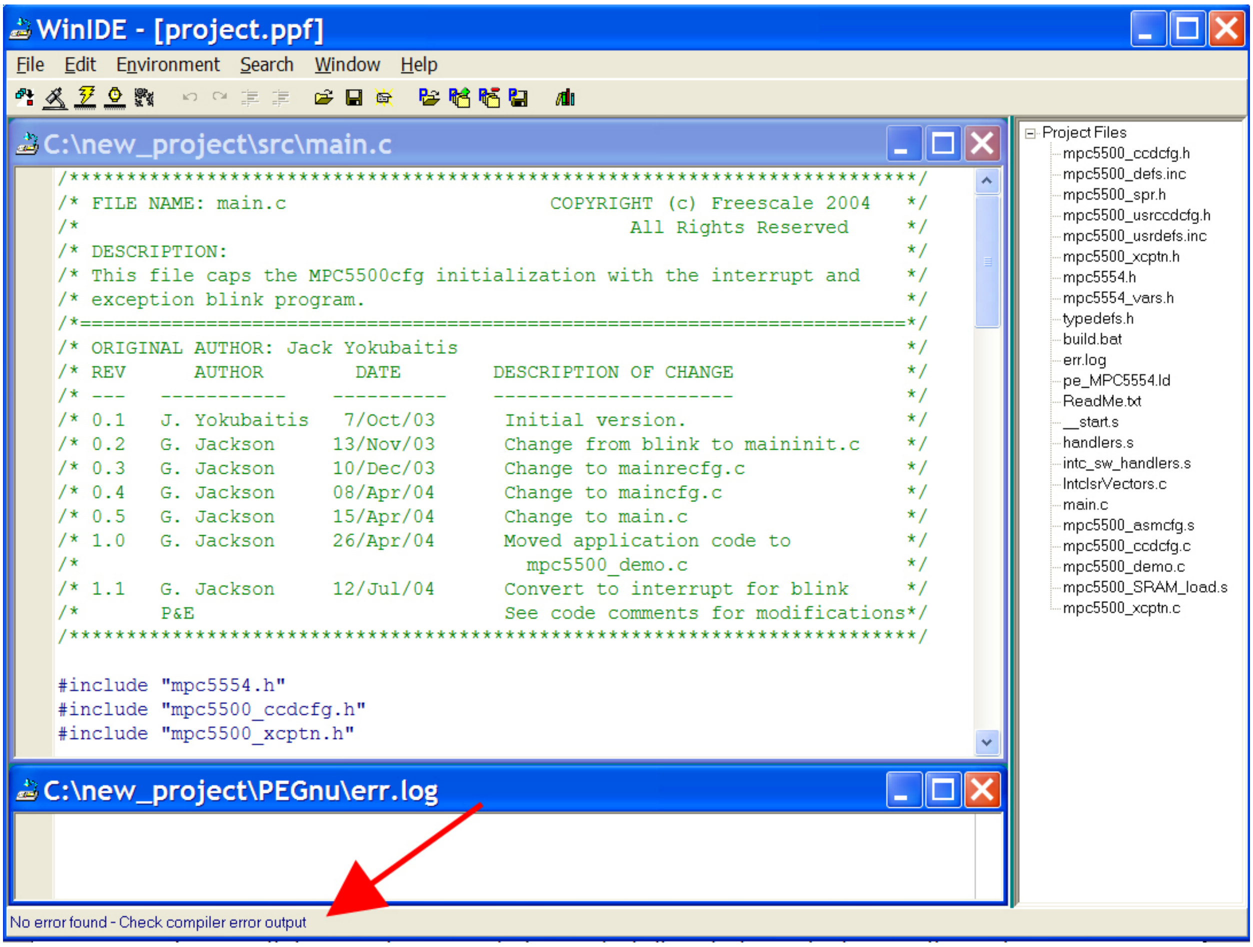
The user may be prompted to confirm that the existing batch file should be overwritten.



Overwrite batch file build script: C:\new_project\build.bat ?

Click ***OK***. WinIDE will execute the batch file, and the batch file will execute the GNU compiler. WinIDE will read the results of the compilation and display the compiler messages in the file, ***err.log.***

If there are no errors during compilation, WinIDE's status bar will indicate that WinIDE detects no errors.

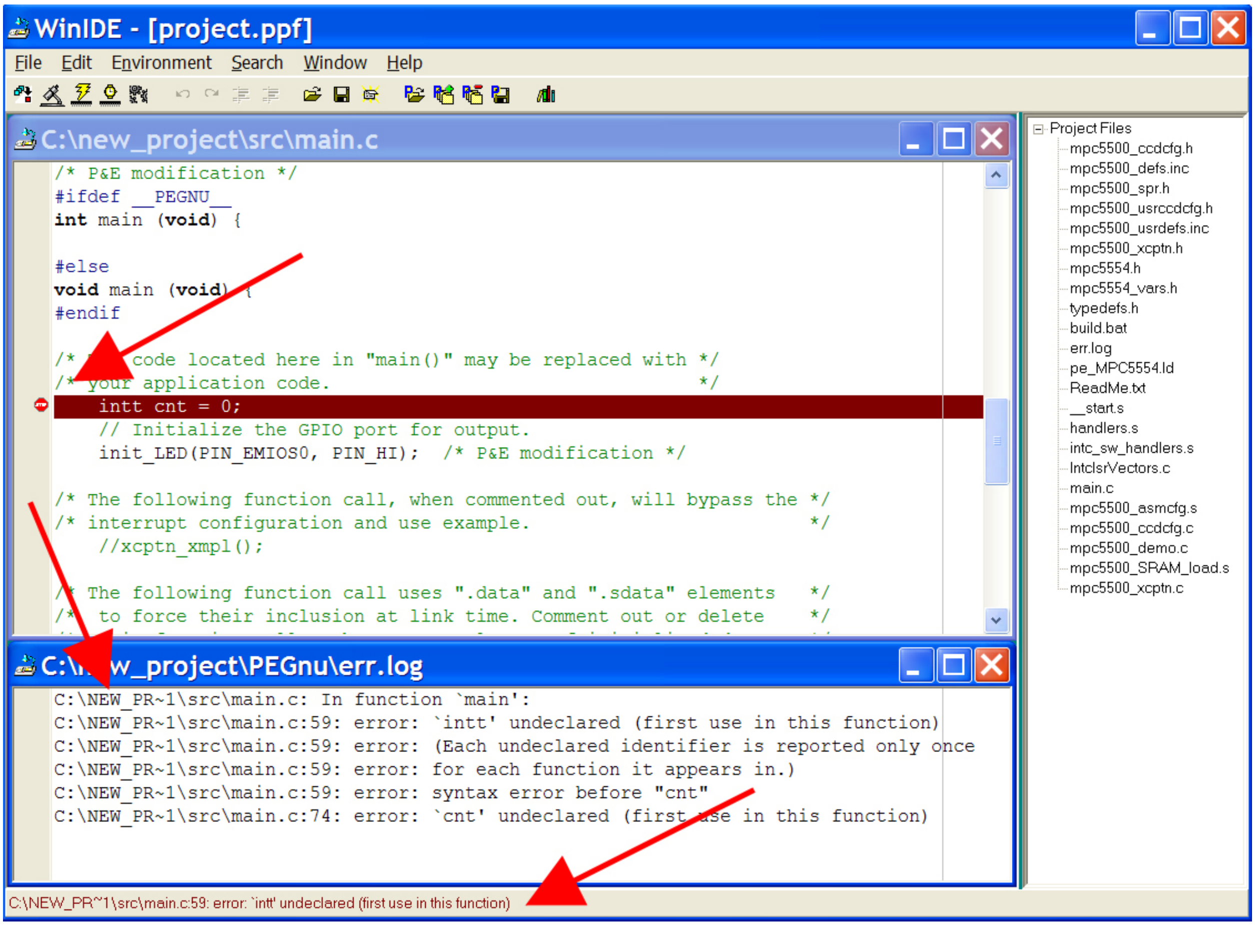


```
/************************************************************************/
/* FILE NAME: main.c                          COPYRIGHT (c) Freescale 2004 */
/*                                                   All Rights Reserved   */
/* DESCRIPTION:                                                          */
/* This file caps the MPC5500cfg initialization with the interrupt and  */
/* exception blink program.                                              */
/*======================================================================*/
/* ORIGINAL AUTHOR: Jack Yokubaitis                                      */
/* REV      AUTHOR         DATE        DESCRIPTION OF CHANGE             */
/* ---    -----------    ----------    ---------------------             */
/* 0.1    J. Yokubaitis   7/Oct/03      Initial version.                  */
/* 0.2    G. Jackson     13/Nov/03      Change from blink to maininit.c   */
/* 0.3    G. Jackson     10/Dec/03      Change to mainrecfg.c             */
/* 0.4    G. Jackson     08/Apr/04      Change to maincfg.c               */
/* 0.5    G. Jackson     15/Apr/04      Change to main.c                  */
/* 1.0    G. Jackson     26/Apr/04      Moved application code to         */
/*                                        mpc5500_demo.c                  */
/* 1.1    G. Jackson     12/Jul/04      Convert to interrupt for blink    */
/*        P&E                           See code comments for modifications*/
/************************************************************************/

#include "mpc5554.h"
#include "mpc5500_ccdcfg.h"
#include "mpc5500_xcptn.h"
```

No error found - Check compiler error output

To ensure that no error has occurred, always refer to the output in ***err.log*** to view all compiler messages.

If there is an error during compilation, WinIDE automatically opens the file containing the first error and attempts to highlight the line with the error.



```
/* P&E modification */
#ifdef __PEGNU__
int main (void) {

#else
void main (void) {
#endif

/* The code located here in "main()" may be replaced with */
/* your application code.                                  */
    intt cnt = 0;
    // Initialize the GPIO port for output.
    init_LED(PIN_EMIOS0, PIN_HI);  /* P&E modification */

/* The following function call, when commented out, will bypass the */
/* interrupt configuration and use example.                         */
    //xcptn_xmpl();

/* The following function call uses ".data" and ".sdata" elements   */
/*  to force their inclusion at link time. Comment out or delete    */
```

```
C:\NEW_PR~1\src\main.c: In function `main':
C:\NEW_PR~1\src\main.c:59: error: `intt' undeclared (first use in this function)
C:\NEW_PR~1\src\main.c:59: error: (Each undeclared identifier is reported only once
C:\NEW_PR~1\src\main.c:59: error: for each function it appears in.)
C:\NEW_PR~1\src\main.c:59: error: syntax error before "cnt"
C:\NEW_PR~1\src\main.c:74: error: `cnt' undeclared (first use in this function)
```

```
C:\NEW_PR~1\src\main.c:59: error: `intt' undeclared (first use in this function)
```

WinIDE's status bar, at the bottom of the window, displays the first compiler error message. WinIDE recovers error messages from the file ***err.log***. *For more information on a specific error, refer to the compiler output in **err.log.*** Certain errors, such as those that occur during linking, may not have a corresponding source line. To be sure that no error has occurred during the build process, *always refer to the log file.*

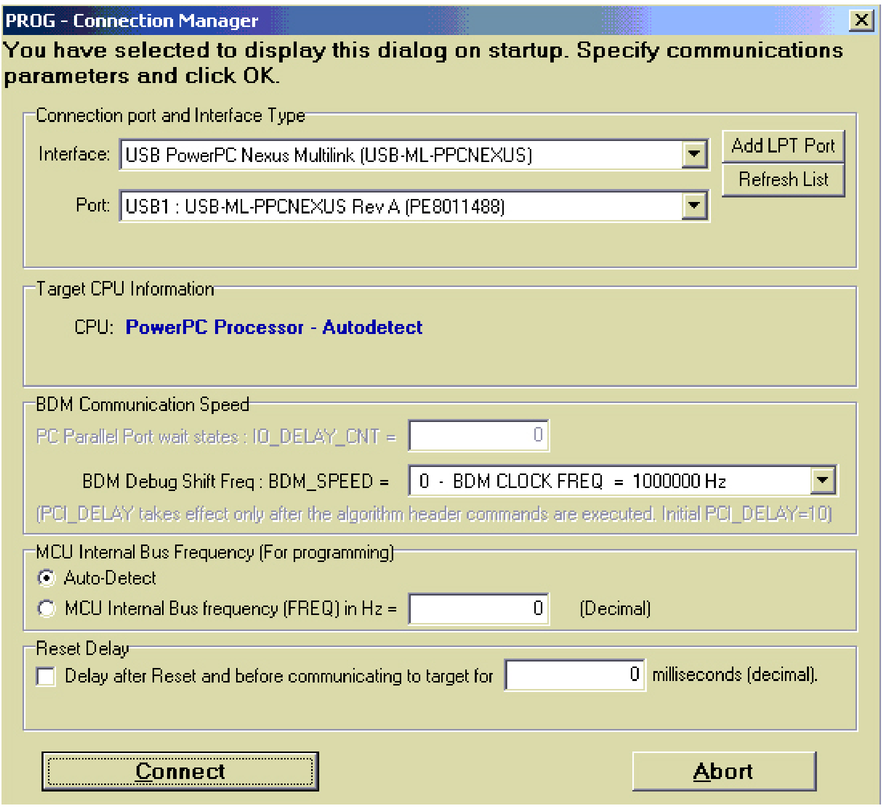
4.9 Programming Flash Using PROGPPCNEXUS

The template project produces a PowerPC Nexus application that will execute in the on-chip Flash memory. The PowerPC target application uses on-chip SRAM for global variables and on-chip cache for stack space. Use the default memory map for this example. The application base address, in Flash, is 0x00000000.

After compiling the application, program the application to on-chip Flash memory. Power down the target board. Before launching the Flash programming software, connect the host PC to the target board using a P&E hardware interface, such as the Cyclone MAX (USB port), USB-ML-PPCNEXUS (USB port), or the CABPPCNEXUS (parallel port). Power up the target board.

Launch the Flash programming software from within WinIDE by using the <F7> hotkey or the programming button on the tool bar. Alternatively, there is an icon for PROGPPCNEXUS in the Windows Start menu in the program group for PKGPPCNEXUS.

After starting the Flash programmer, the connection assistant dialog will appear.



PROG - Connection Manager

You have selected to display this dialog on startup. Specify communications parameters and click OK.

Connection port and Interface Type

Interface: USB PowerPC Nexus Multilink (USB-ML-PPCNEXUS)

Port: USB1 : USB-ML-PPCNEXUS Rev A (PE8011488)

Target CPU Information

CPU: **PowerPC Processor - Autodetect**

BDM Communication Speed

PC Parallel Port wait states : IO_DELAY_CNT = 0

BDM Debug Shift Freq : BDM_SPEED = 0 - BDM CLOCK FREQ = 1000000 Hz

(PCI_DELAY takes effect only after the algorithm header commands are executed. Initial PCI_DELAY=10)

MCU Internal Bus Frequency (For programming)

● Auto-Detect

○ MCU Internal Bus frequency (FREQ) in Hz = 0 (Decimal)

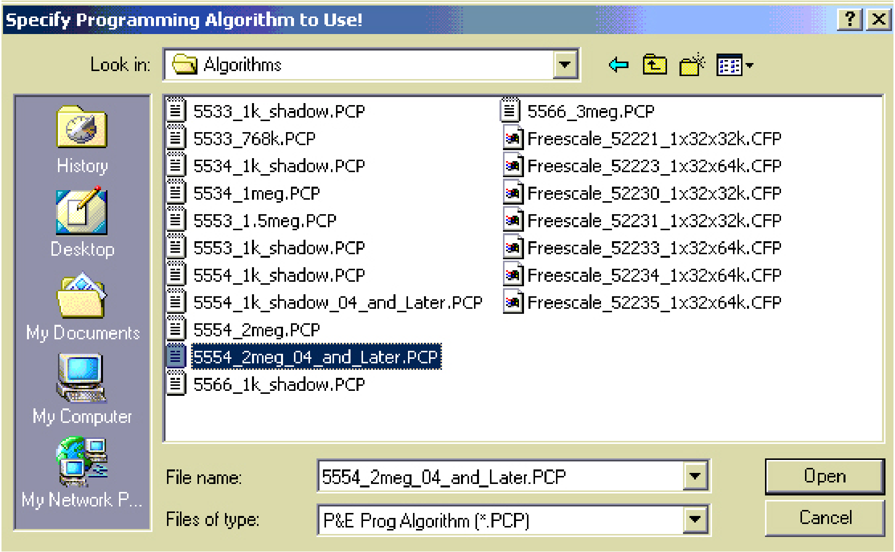
Reset Delay

□ Delay after Reset and before communicating to target for 0 milliseconds (decimal).

Use the drop-down menu labeled "Port" to choose the proper PC connection to your target board.

Click the Connect button, and PROGPPCNEXUS will attempt to contact the target processor. PROGPPCNEXUS will attempt to establish communications and to reset the processor.

Next, the Flash programmer will prompt you for a programming algorithm file.



Specify Programming Algorithm to Use!

Look in: Algorithms

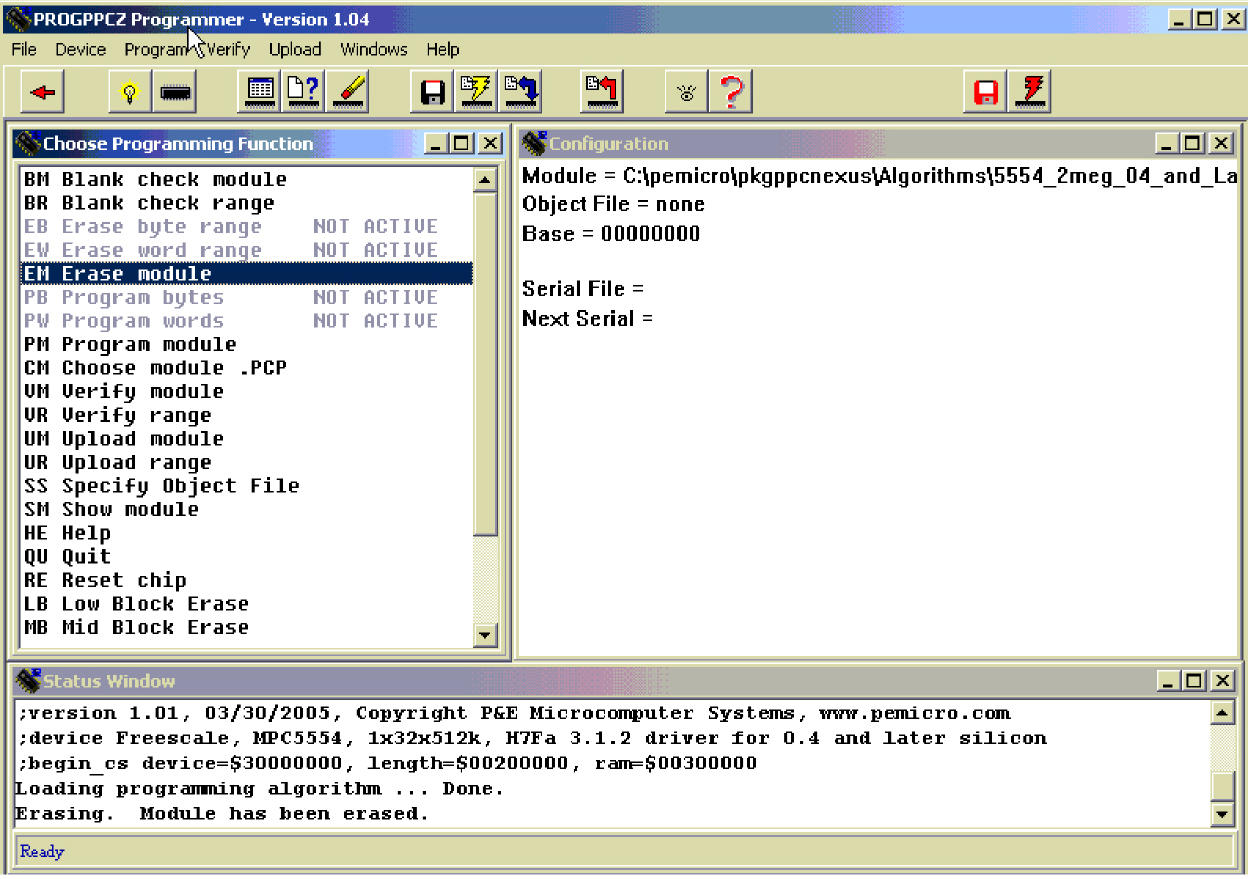
5533_1k_shadow.PCP
5533_768k.PCP
5534_1k_shadow.PCP
5534_1meg.PCP
5553_1.5meg.PCP
5553_1k_shadow.PCP
5554_1k_shadow.PCP
5554_1k_shadow_04_and_Later.PCP
5554_2meg.PCP
5554_2meg_04_and_Later.PCP
5566_1k_shadow.PCP
5566_3meg.PCP
Freescale_52221_1x32x32k.CFP
Freescale_52223_1x32x64k.CFP
Freescale_52230_1x32x32k.CFP
Freescale_52231_1x32x32k.CFP
Freescale_52233_1x32x64k.CFP
Freescale_52234_1x32x64k.CFP
Freescale_52235_1x32x64k.CFP

File name: 5554_2meg_04_and_Later.PCP

Files of type: P&E Prog Algorithm (*.PCP)

Select the appropriate file in the Algorithms folder of the installation directory, and click Open. For this example, select ***5554_2meg_04_and_Later.PCP***.

At this point, double click the ***"EM Erase module"*** command and allow the command to complete.



PROGPPCZ Programmer - Version 1.04
File Device Program Verify Upload Windows Help
Choose Programming Function
Configuration
Status Window
Ready

```
BM Blank check module
BR Blank check range
EB Erase byte range     NOT ACTIVE
EW Erase word range     NOT ACTIVE
EM Erase module
PB Program bytes        NOT ACTIVE
PW Program words        NOT ACTIVE
PM Program module
CM Choose module .PCP
VM Verify module
VR Verify range
UM Upload module
UR Upload range
SS Specify Object File
SM Show module
HE Help
QU Quit
RE Reset chip
LB Low Block Erase
MB Mid Block Erase
```

```
Module = C:\pemicro\pkgppcnexus\Algorithms\5554_2meg_04_and_La
Object File = none
Base = 00000000

Serial File =
Next Serial =
```

```
;version 1.01, 03/30/2005, Copyright P&E Microcomputer Systems, www.pemicro.com
;device Freescale, MPC5554, 1x32x512k, H7Fa 3.1.2 driver for 0.4 and later silicon
;begin_cs device=$30000000, length=$00200000, ram=$00300000
Loading programming algorithm ... Done.
Erasing.  Module has been erased.
```

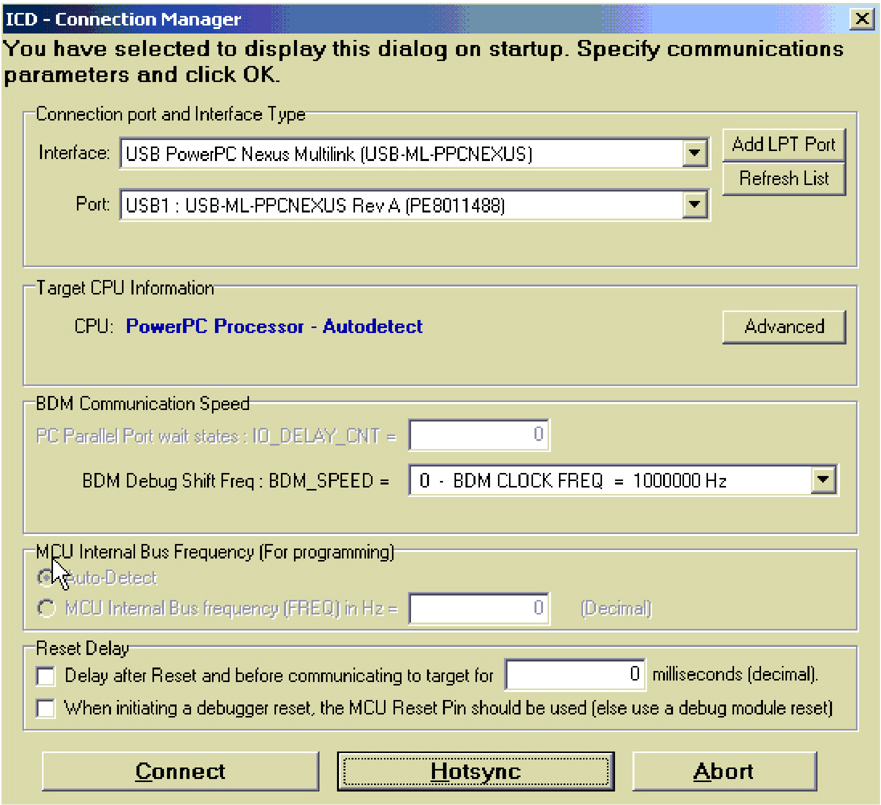
Issue the "***BM Blank check module***" command, and wait for the command to complete. Next, issue the "***SS Specify Object File***" command. The software will prompt for the S19 file. Select the file *main.s19* in the directory ***C:\new_project\PEGnu***. Next, execute the commands "***PM Program module***" followed by "***VM Verify module***". If PROGPPCNEXUS reports no errors, then the Flash memory on the processor has been programmed.

Exit the Flash programmer.

4.10 Starting the Debugger

After compiling and programming the PowerPC Nexus application to Flash, launch the debugger from WinIDE by using the <F6> key or the “run debugger” button on the tool bar. Alternately, use the debugger icon in the program group for PKGPPCNEXUS, located in the Windows Start menu.

Upon starting the debugger, the connection assistant dialog appears.



ICD - Connection Manager

You have selected to display this dialog on startup. Specify communications parameters and click OK.

Connection port and Interface Type

Interface: USB PowerPC Nexus Multilink (USB-ML-PPCNEXUS)

Port: USB1 : USB-ML-PPCNEXUS Rev A (PE8011488)

Target CPU Information

CPU: PowerPC Processor - Autodetect

BDM Communication Speed

PC Parallel Port wait states : IO_DELAY_CNT = 0

BDM Debug Shift Freq : BDM_SPEED = 0 - BDM CLOCK FREQ = 1000000 Hz

MCU Internal Bus Frequency (For programming)

● Auto-Detect

○ MCU Internal Bus frequency (FREQ) in Hz = 0 (Decimal)

Reset Delay

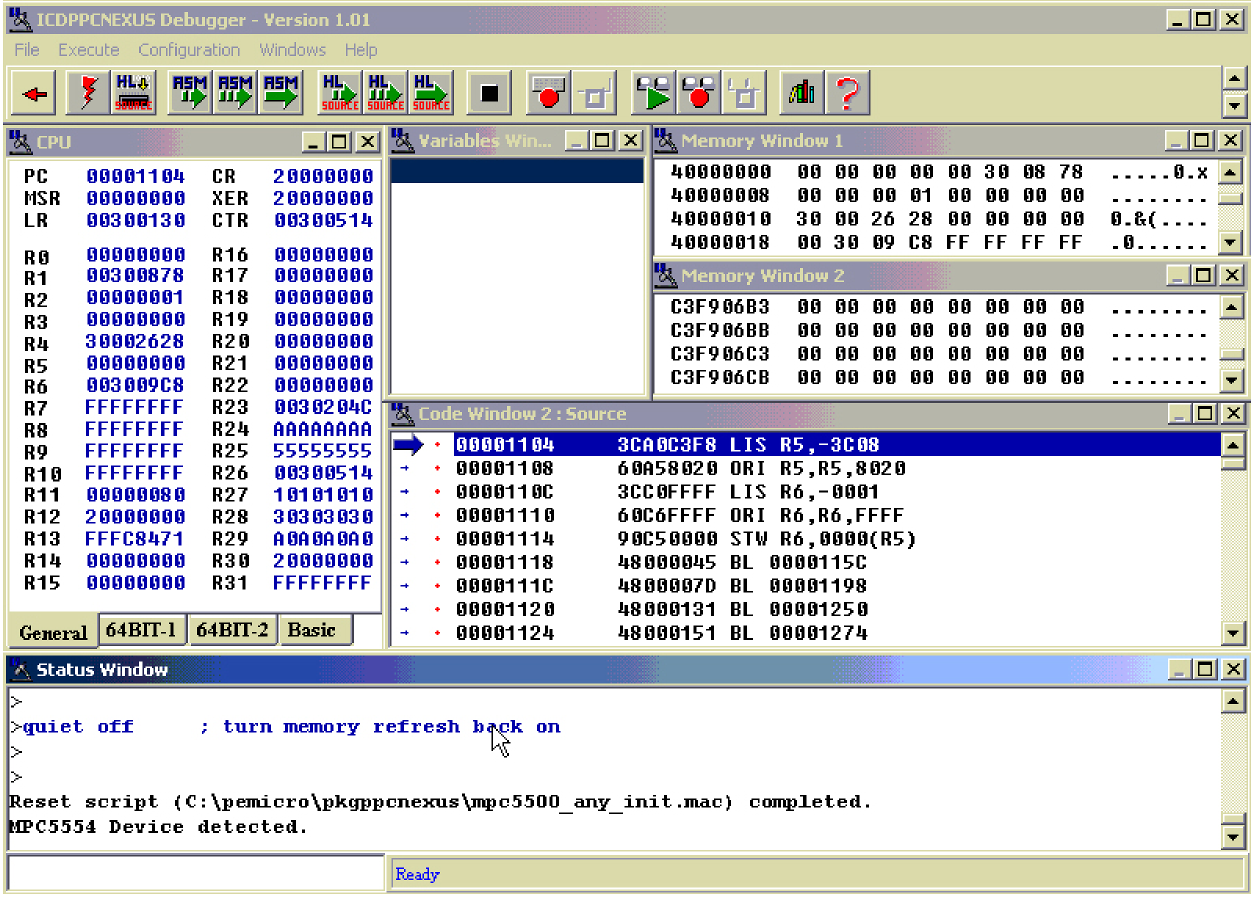
□ Delay after Reset and before communicating to target for 0 milliseconds (decimal).

□ When initiating a debugger reset, the MCU Reset Pin should be used (else use a debug module reset)

Use the drop-down menu labeled "***Port***" to choose the proper PC connection to your target board.

Click the **Connect** button, and ICDPPCNEXUS will attempt to contact the processor. Using the default debugger settings, ICDPPCNEXUS will establish communications and reset the processor.

After establishing communications, the main debugger screen will appear, and a debugger macro should automatically execute and complete. If you do not see the debugger macro executing, issue the MACRO command in the debugger and select the file ***C:\pemicro\pkgppcnexus\MPC5500_any_init.mac.*** See the next section for an explanation of this debugger macro file.



```
PC    00001104   CR    20000000
MSR   00000000   XER   20000000
LR    00300130   CTR   00300514

R0    00000000   R16   00000000
R1    00300878   R17   00000000
R2    00000001   R18   00000000
R3    00000000   R19   00000000
R4    30002628   R20   00000000
R5    00000000   R21   00000000
R6    003009C8   R22   00000000
R7    FFFFFFFF   R23   0030204C
R8    FFFFFFFF   R24   AAAAAAAA
R9    FFFFFFFF   R25   55555555
R10   FFFFFFFF   R26   00300514
R11   00000080   R27   10101010
R12   20000000   R28   30303030
R13   FFFC8471   R29   A0A0A0A0
R14   00000000   R30   20000000
R15   00000000   R31   FFFFFFFF
```

```
40000000  00 00 00 00 00 30 08 78  .....0.x
40000008  00 00 00 01 00 00 00 00  ........
40000010  30 00 26 28 00 00 00 00  0.&(....
40000018  00 30 09 C8 FF FF FF FF  .0......
```

```
C3F906B3  00 00 00 00 00 00 00 00  ........
C3F906BB  00 00 00 00 00 00 00 00  ........
C3F906C3  00 00 00 00 00 00 00 00  ........
C3F906CB  00 00 00 00 00 00 00 00  ........
```

```
00001104     3CA0C3F8 LIS R5,-3C08
00001108     60A58020 ORI R5,R5,8020
0000110C     3CC0FFFF LIS R6,-0001
00001110     60C6FFFF ORI R6,R6,FFFF
00001114     90C50000 STW R6,0000(R5)
00001118     48000045 BL  0000115C
0000111C     4800007D BL  00001198
00001120     48000131 BL  00001250
00001124     48000151 BL  00001274
```

```
>
>quiet off     ; turn memory refresh back on
>
>
Reset script (C:\pemicro\pkgppcnexus\mpc5500_any_init.mac) completed.
MPC5554 Device detected.
```

4.11 Boot Assist Module Considerations

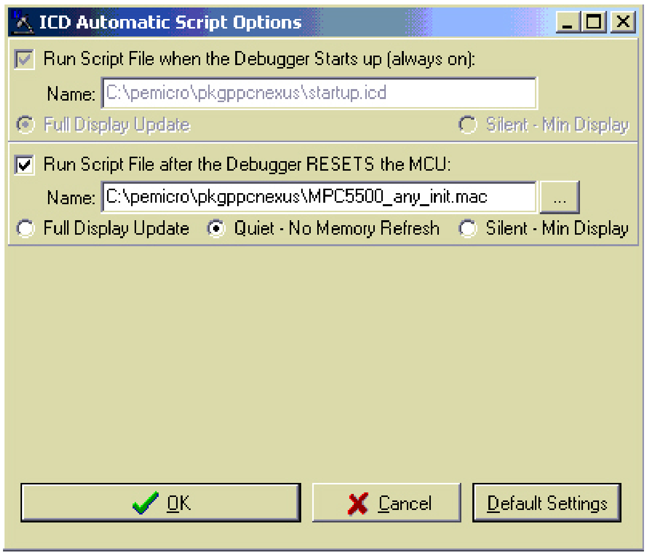
This section does not contain any steps for the user to complete, it simply explains the initialization that the debugger performs on the processor. A debugger initialization script controls the processor initialization. The user can view and modify all the script's initialization tasks.

The processor Boot Assist Module (BAM) would normally initialize the memory of the processor. However, when running the target application from the debugger, the BAM functionality is disabled. To account for this, the debugger must run a script file on reset. The script initializes the memory of the processor similar to the way in which the BAM would initialize the processor.

The appropriate macro file should automatically run on reset. To run the script file manually, type the MACRO command in the debugger status window or click the play macro button on the debugger toolbar. Select the macro file ***MPC5500_any_init.mac***, located in the PKGPPCNEXUS installation directory as well as ***C:\new_project\PEGnu*** in this example. When the script file runs, if the debugger has not been configured to run in silent mode, you will see a significant number of status messages scroll down the status window of the debugger.

The macro file MPC5500_any_init.mac does not initialize the external bus. You may run the macro files ***MPC5500_ebi_32.mac*** or ***MPC5500_ebi_16.mac*** to configure the external bus for 32-bit or 16-bit memory.

To configure the debugger macro to execute automatically on reset, select the debugger Configuration menu, Automated Script Options dialog, shown here:



ICD Automatic Script Options

☑ Run Script File when the Debugger Starts up (always on):

Name: C:\pemicro\pkgppcnexus\startup.icd

● Full Display Update ○ Silent - Min Display

☑ Run Script File after the Debugger RESETS the MCU:

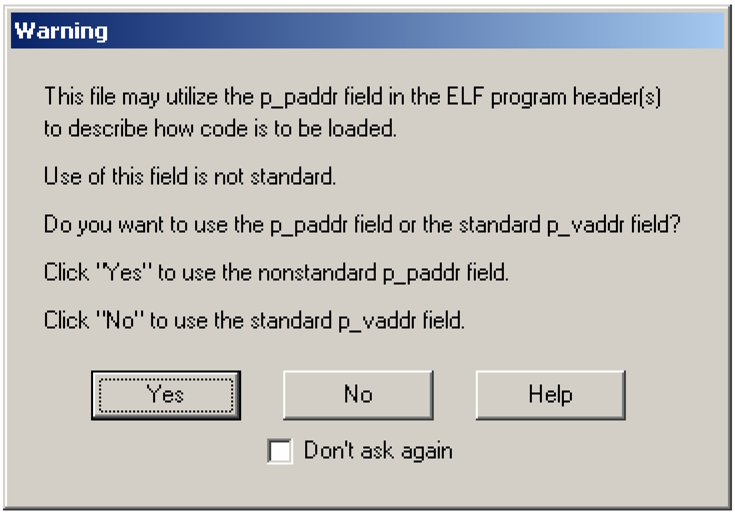
Name: C:\pemicro\pkgppcnexus\MPC5500_any_init.mac

○ Full Display Update ● Quiet - No Memory Refresh ○ Silent - Min Display

4.12 Loading the Debug Information

After executing the initialization script, it is time to load the debugging information into the debugger. Issue the HLOADMAP command (type in HLOADMAP and hit enter) in the debugger status window, and select the Elf/Dwarf file, ***C:\new_project\PEGnu\main.elf***. This will load the debug information that corresponds to the PowerPC application, now located in Flash.

Depending on the version of ICDPPCNEXUS, the debugger may display the following dialog box:



Warning

This file may utilize the p_paddr field in the ELF program header(s) to describe how code is to be loaded.

Use of this field is not standard.

Do you want to use the p_paddr field or the standard p_vaddr field?

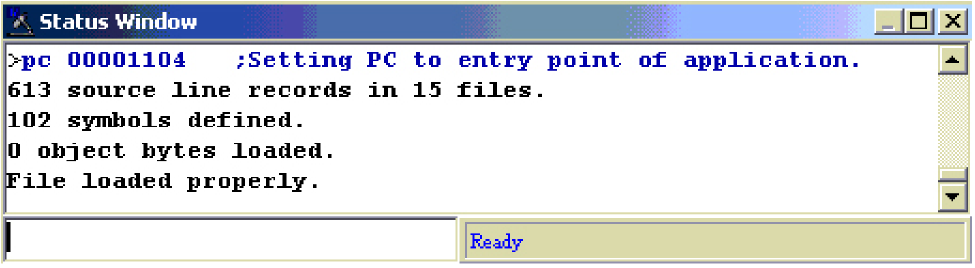
Click "Yes" to use the nonstandard p_paddr field.

Click "No" to use the standard p_vaddr field.

□ Don't ask again

This warning relates to the format of the GNU Elf/Dwarf file. *For more information, see the document "P&E, GNU, and Elf/Dwarf"*. When using GNU GCC, always select "**Yes**".

After loading the Elf/Dwarf 2.0 file, the debugger status window will display the amount of debug and object information loaded from the file.

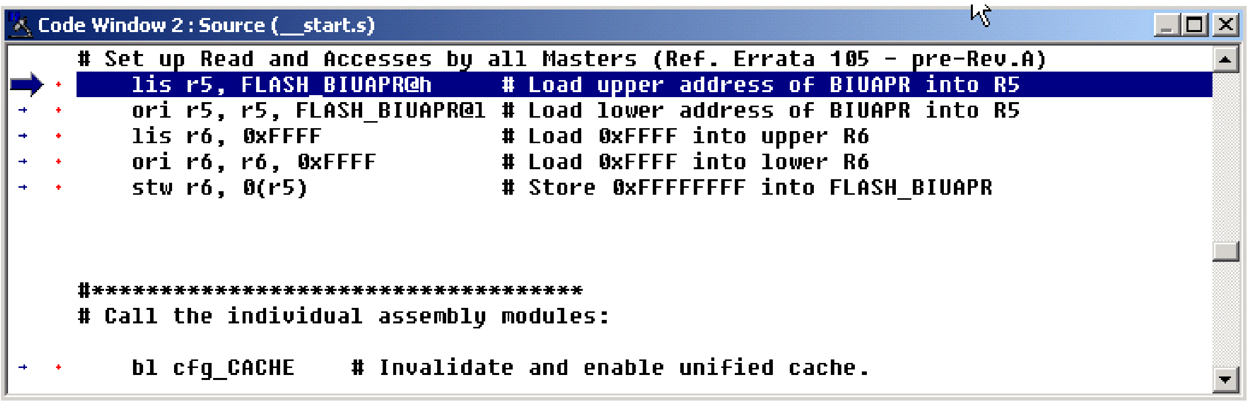


```
>pc 00001104    ;Setting PC to entry point of application.
613 source line records in 15 files.
102 symbols defined.
0 object bytes loaded.
File loaded properly.
```

By default, when the debugger loads an Elf/Dwarf 2.0 file, the debugger sets the program counter (PC) to the start address of the target application.

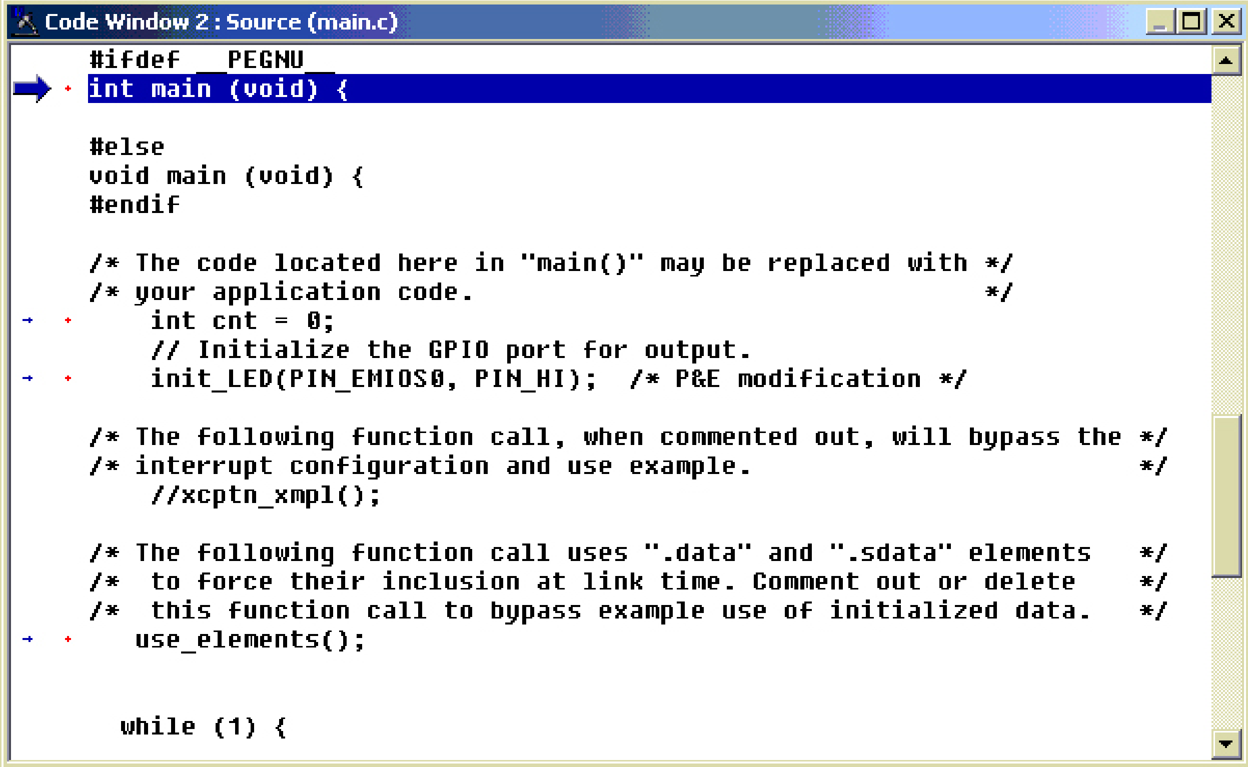
4.13 Stepping through C Instructions

After loading the Elf/Dwarf file, the debugger code window displays assembly language startup code. After loading the file *main.elf,* the code window should show this source:



```
# Set up Read and Accesses by all Masters (Ref. Errata 105 - pre-Rev.A)
    lis r5, FLASH_BIUAPR@h       # Load upper address of BIUAPR into R5
    ori r5, r5, FLASH_BIUAPR@l # Load lower address of BIUAPR into R5
    lis r6, 0xFFFF               # Load 0xFFFF into upper R6
    ori r6, r6, 0xFFFF           # Load 0xFFFF into lower R6
    stw r6, 0(r5)                # Store 0xFFFFFFFF into FLASH_BIUAPR



#***************************************
# Call the individual assembly modules:

    bl cfg_CACHE     # Invalidate and enable unified cache.
```

To run past the startup code, issue the debugger command "***GOTIL main***" in the status window. Note that the labels are case sensitive and that the “main” label should be lowercase. This will set a breakpoint at the beginning of the main function in ***main.c*** and start the processor running. Execution should stop almost immediately, and the program counter will be pointing to valid C source code.



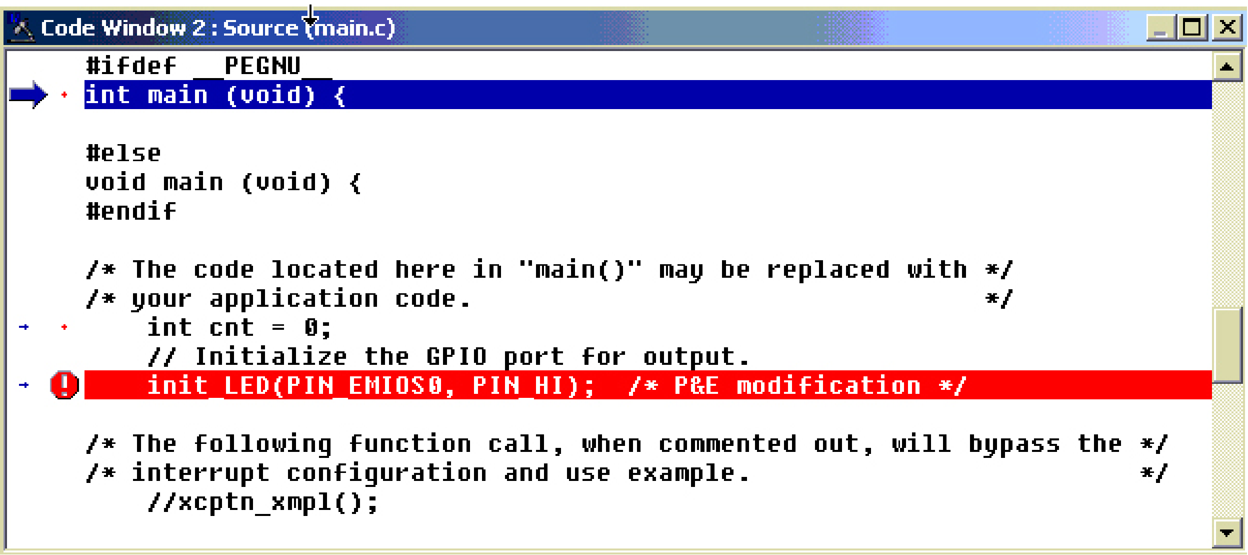
```
#ifdef __PEGNU__
int main (void) {

#else
void main (void) {
#endif

/* The code located here in "main()" may be replaced with */
/* your application code.                                 */
    int cnt = 0;
    // Initialize the GPIO port for output.
    init_LED(PIN_EMIOS0, PIN_HI);  /* P&E modification */

/* The following function call, when commented out, will bypass the */
/* interrupt configuration and use example.                          */
    //xcptn_xmpl();

/* The following function call uses ".data" and ".sdata" elements   */
/*  to force their inclusion at link time. Comment out or delete    */
/*  this function call to bypass example use of initialized data.   */
   use_elements();


  while (1) {
```

Instead of using the GOTIL command, you may step through the startup code to reach the main function. Step through the initialization code, or any source code, using the high-level language source step command. Use this feature by typing HSTEP in the status window or by clicking the high-level step button on the debugger tool bar. Each time the HSTEP command executes, the debugger will rapidly single step assembly instructions until it encounters the next source instruction, at which point target execution will cease. When the debugger reaches the next source instruction, all visible windows will be updated with data from the board.

After reaching the main function, step through several C language instructions. Notice that some instructions will take longer to step through than others because each C instruction may consist of a greater or fewer number of underlying assembly instructions.

4.14 Using Code Window Quick Execution Features

In the source code window, there will be a small red dot and a small blue arrow next to each source instruction that has underlying object code. If a large blue arrow appears on a source line, this indicates that the program counter (PC) points to this instruction. If a large red stop sign appears on the source line, this indicates that a breakpoint exists on this line.

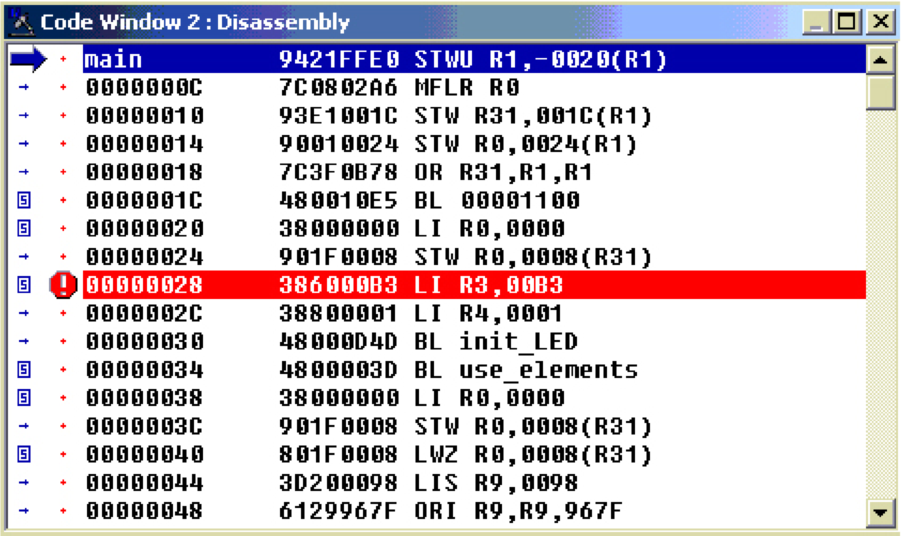


```
#ifdef __PEGNU__
int main (void) {

#else
void main (void) {
#endif

/* The code located here in "main()" may be replaced with */
/* your application code.                                 */
    int cnt = 0;
    // Initialize the GPIO port for output.
    init_LED(PIN_EMIOS0, PIN_HI);  /* P&E modification */

/* The following function call, when commented out, will bypass the */
/* interrupt configuration and use example.                         */
    //xcptn_xmpl();
```

Set a breakpoint at an instruction by double-clicking the tiny red dot. To remove a breakpoint, double-click the large red stop sign. When you issue the HGO command or click the high-level language GO button on the debugger tool bar, execution will begin in real-time. If the debugger encounters a breakpoint, execution will stop on this source line. If it does not encounter a breakpoint, target execution will continue until you press a key or use the stop button on the debugger tool bar.

By double clicking the small blue arrow, you will be issuing a GOTIL command to the address of this source line. A GOTIL command will set a single breakpoint at the desired address, and the processor will begin executing code in real-time from the point of the current program counter (PC). When the debugger encounters the GOTIL address, execution stops. If the debugger does not encounter this location, execution continues until you press a key or use the stop button on the debugger tool bar. Note that all user breakpoints are ignored when the GOTIL command is used.

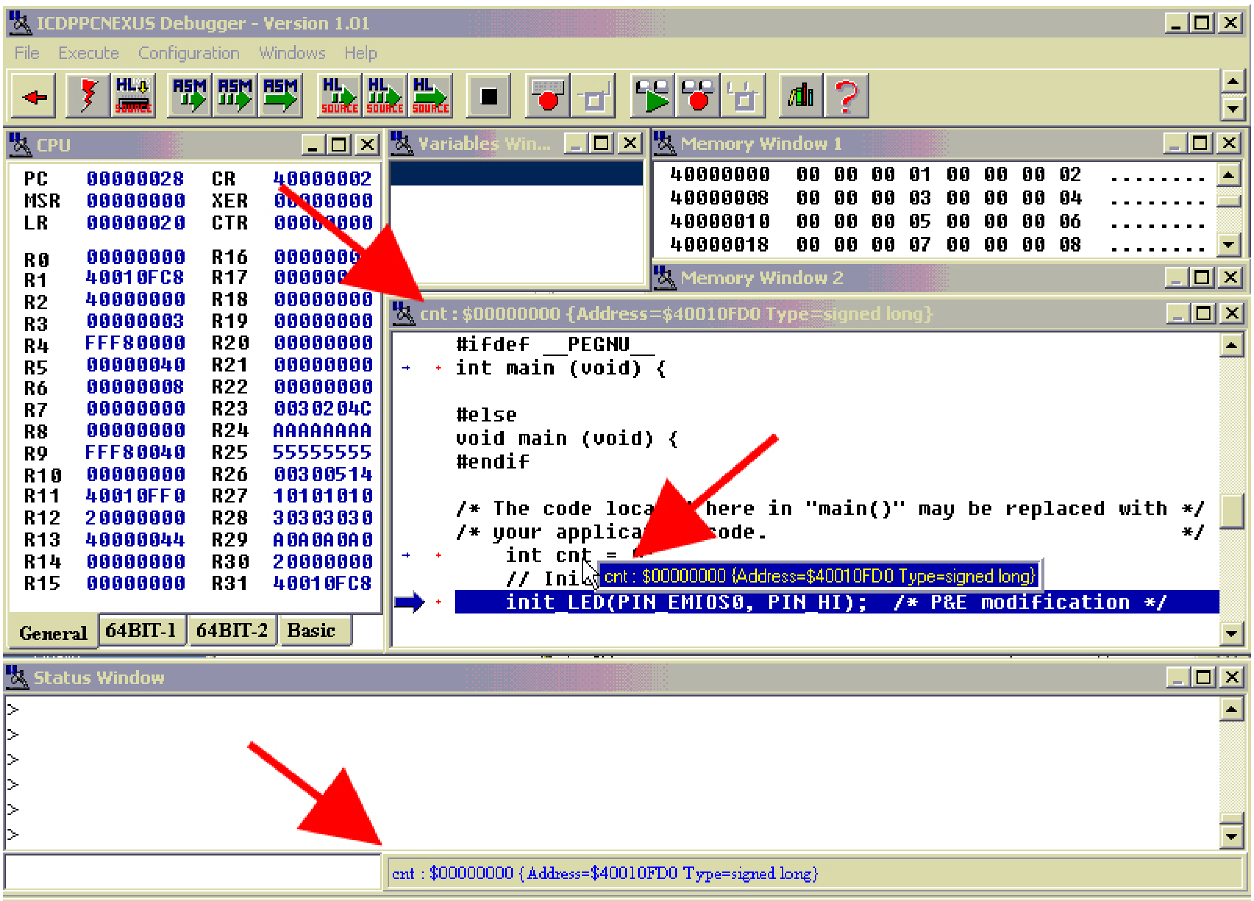
You may also double-click the red and blue symbols in the disassembly window. The disassembly window may display an additional symbol, a small, blue "S" enclosed in a box. This indicates that that a source code instruction begins on this disassembly instruction.



```
main          9421FFE0 STWU R1,-0020(R1)
0000000C      7C0802A6 MFLR R0
00000010      93E1001C STW R31,001C(R1)
00000014      90010024 STW R0,0024(R1)
00000018      7C3F0B78 OR R31,R1,R1
0000001C      480010E5 BL 00001100
00000020      38000000 LI R0,0000
00000024      901F0008 STW R0,0008(R31)
00000028      386000B3 LI R3,00B3
0000002C      38800001 LI R4,0001
00000030      48000D4D BL init_LED
00000034      4800003D BL use_elements
00000038      38000000 LI R0,0000
0000003C      901F0008 STW R0,0008(R31)
00000040      801F0008 LWZ R0,0008(R31)
00000044      3D200098 LIS R9,0098
00000048      6129967F ORI R9,R9,967F
```

4.15 Using Code Window Popup Debug Evaluation Hints

When debugging source code, it is advantageous to view the contents of a variable while viewing your source code. The in-circuit debugger has a feature, debug hints, which displays the value of a variable while the mouse cursor is held over the variable name. The hint may be displayed in any of three locations, as shown below.



```
PC    00000028   CR    40000002
MSR   00000000   XER   00000000
LR    00000020   CTR   00000000

R0    00000000   R16   00000000
R1    40010FC8   R17   00000000
R2    40000000   R18   00000000
R3    00000003   R19   00000000
R4    FFF80000   R20   00000000
R5    00000040   R21   00000000
R6    00000008   R22   00000000
R7    00000000   R23   0030204C
R8    00000000   R24   AAAAAAAA
R9    FFF80040   R25   55555555
R10   00000000   R26   00300514
R11   40010FF0   R27   10101010
R12   20000000   R28   30303030
R13   40000044   R29   A0A0A0A0
R14   00000000   R30   20000000
R15   00000000   R31   40010FC8
```

```
40000000   00 00 00 01 00 00 00 02   ........
40000008   00 00 00 03 00 00 00 04   ........
40000010   00 00 00 05 00 00 00 06   ........
40000018   00 00 00 07 00 00 00 08   ........
```

cnt : $00000000 {Address=$40010FD0 Type=signed long}

```
#ifdef __PEGNU__
int main (void) {

#else
void main (void) {
#endif

/* The code loca   here in "main()" may be replaced with */
/* your applica    ode.                                   */
    int cnt =
    // Ini
    init_LED(PIN_EMIOS0, PIN_HI);  /* P&E modification */
```

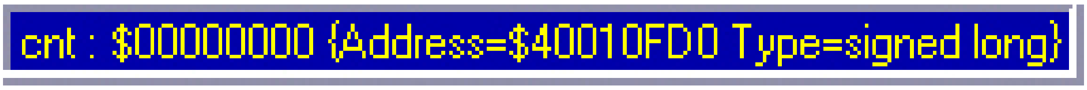
cnt : $00000000 {Address=$40010FD0 Type=signed long}

```
>
>
>
>
>
>
```

cnt : $00000000 {Address=$40010FD0 Type=signed long}

The three locations for the debug hints are the code window title bar, the status window caption bar, and a popup hint that appears over the variable in source code. You can configure the hints to display in any combination. Set the locations in the configuration menu of the debugger.

The information in the popup hint box is similar to the information displayed in the variables window.

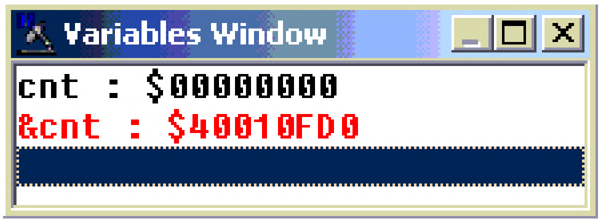


cnt : $00000000 {Address=$40010FD0 Type=signed long}

The information includes the variable name (cnt), value ($0), and type (signed long).

4.16 Using the Variables Window

The variables window displays the current value of application variables. The following window shows a display of variables from the example application.

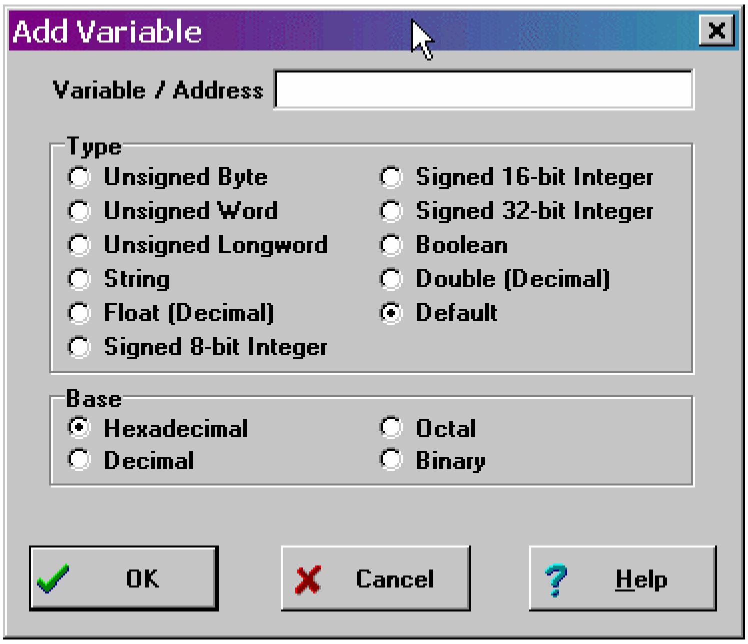


```
cnt : $00000000
&cnt : $40010FD0
```

Variables that are pointer types are displayed in red. Normal variables are displayed in black. Real time variables are displayed in blue. A real-time variable is a variable that is updated while the processor is running. Add real time variables by using the RTVAR *varname* command in the status window. Add normal variables by using the VAR *varname* command.

To view the variables above, type "***var cnt"*** and "***var &cnt***" in the debugger status window. Also, you may add a variable by right clicking the variables window and choosing "***Add a variable***". Delete a variable by selecting it in the variables window, right clicking, and choosing "***Delete variable***".

When adding a variable using the pop-up menu, the debugger displays the following screen.



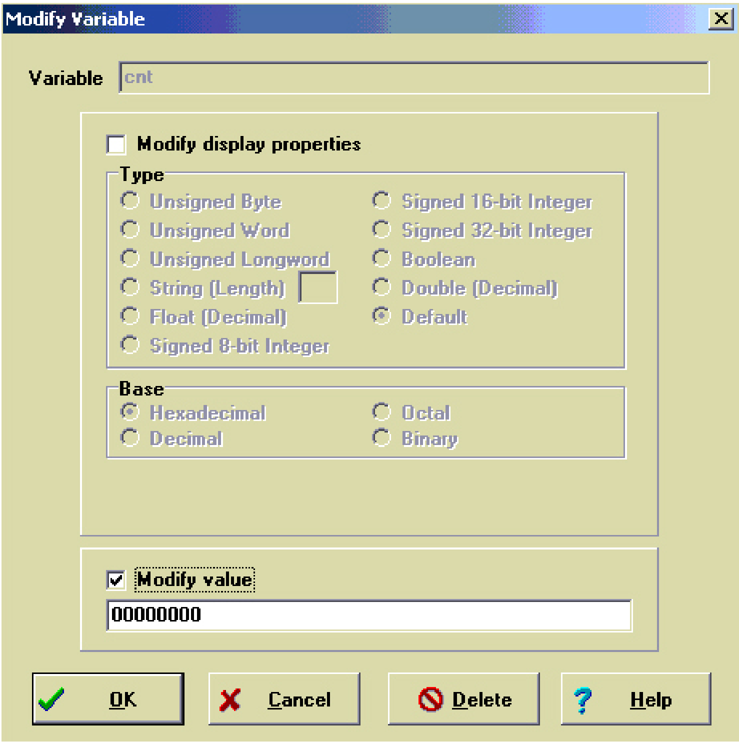
Type
- ○ Unsigned Byte
- ○ Unsigned Word
- ○ Unsigned Longword
- ○ String
- ○ Float (Decimal)
- ○ Signed 8-bit Integer
- ○ Signed 16-bit Integer
- ○ Signed 32-bit Integer
- ○ Boolean
- ○ Double (Decimal)
- ● Default

Base
- ● Hexadecimal
- ○ Decimal
- ○ Octal
- ○ Binary

In the variable field, type the address or name of the variable. Typically, set the type of the variable to "***Default"***, which means that the variable will be displayed as it is defined in the debugging information. When adding a variable, you may specify the numeric display base of the variable.

4.16.1 Modifying a Variable's Value

To modify the current value of a variable, double-click the ***variable name*** in the variables window. If the debugger supports modification of this type of variable, it will display the modify variable dialog. Check the “***Modify value***” checkbox, and type the variable’s new value. After you click the ***OK*** button, the debugger updates the variable value on the target memory, and the debugger refreshes the variable window to display the new value. Note that the debugger will not edit certain user-defined types, such as enumerated types.



Modify Variable

Variable: cnt

☐ Modify display properties

Type

○ Unsigned Byte
○ Unsigned Word
○ Unsigned Longword
○ String (Length)
○ Float (Decimal)
○ Signed 8-bit Integer
○ Signed 16-bit Integer
○ Signed 32-bit Integer
○ Boolean
○ Double (Decimal)
● Default

Base

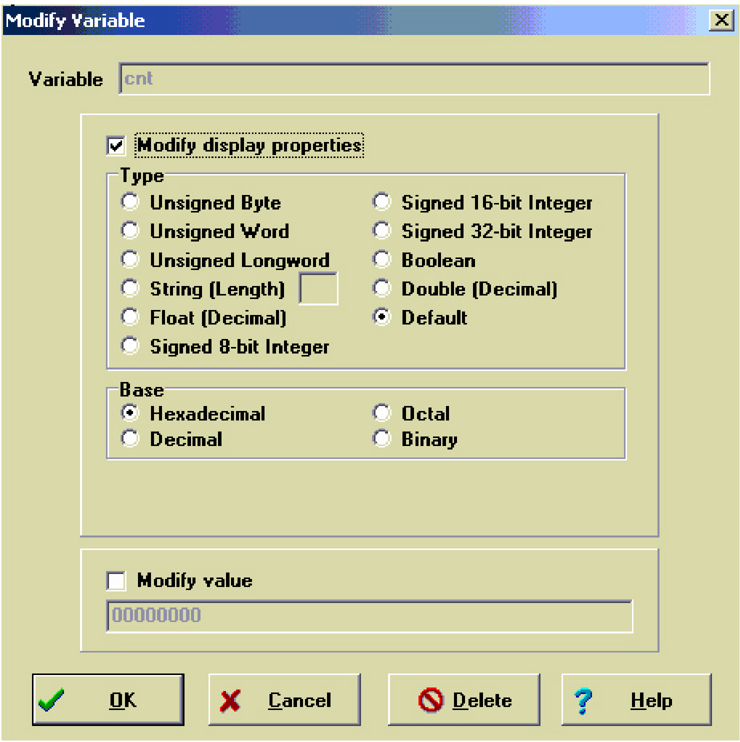
● Hexadecimal
○ Decimal
○ Octal
○ Binary

☑ Modify value

00000000

4.16.2 Modifying a Variable's Properties

To modify a variable’s display properties, such as the type or numeric display base, double-click the ***variable name*** in the variables window. Check “***Modify display properties***” in the modify variable dialog. Change the variable type and base. Click the ***OK*** button, and the debugger will update the variable's display properties in the variables window.



Modify Variable

Variable: cnt

☑ Modify display properties

Type

○ Unsigned Byte
○ Unsigned Word
○ Unsigned Longword
○ String (Length)
○ Float (Decimal)
○ Signed 8-bit Integer
○ Signed 16-bit Integer
○ Signed 32-bit Integer
○ Boolean
○ Double (Decimal)
● Default

Base

● Hexadecimal
○ Decimal
○ Octal
○ Binary

□ Modify value

00000000

4.17 Using the register interpreter

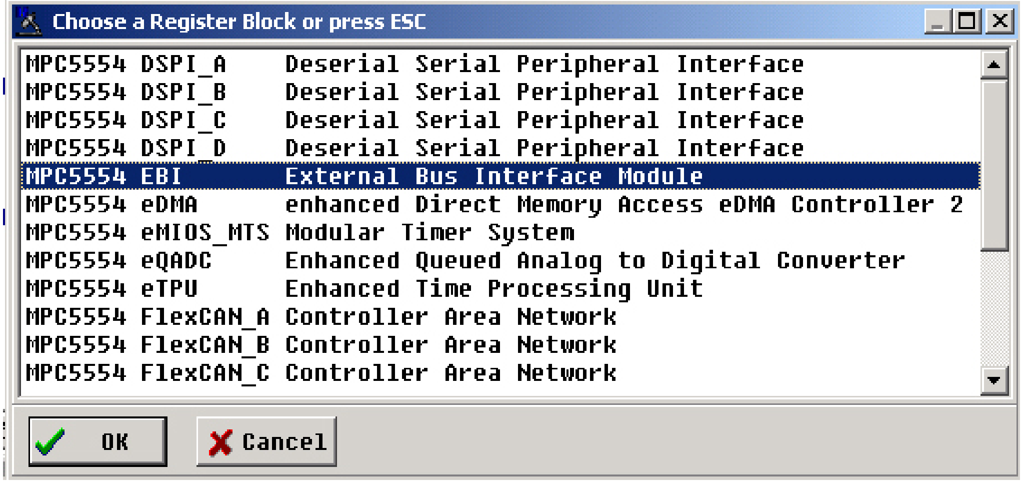
The register interpreter provides a descriptive display of bit fields within the processor's peripheral registers. The register interpreter allows you easily to change the value of these registers. You may quickly check the current state of a peripheral and examine the configuration of the target device.

When you use the register interpreter within the debugger, it reads the current value of the peripheral register, decodes it, and displays it. When using the register interpreter within WinIDE, all the fields are initialized to 0. When the user hits the Enter key, the value is written into the editor at the current cursor location, as opposed to being written back into memory, as the debugger does.

Note that the register interpreter is available for select processors only.

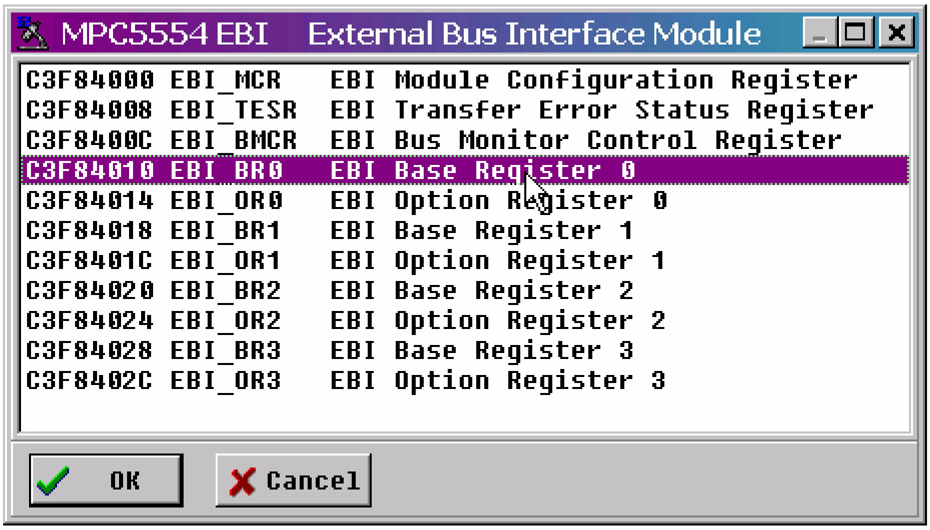
4.17.1 Register interpreter display

To launch the register interpreter in the debugger, either use the "***R***" command or click the view/edit register button on the tool bar. To display the interpreter from WinIDE, use the "***SHIFT-F1***" keystroke combination or the register files button on the tool bar. In either application, a window will appear that allows you to select a peripheral block to examine.



```
MPC5554 DSPI_A    Deserial Serial Peripheral Interface
MPC5554 DSPI_B    Deserial Serial Peripheral Interface
MPC5554 DSPI_C    Deserial Serial Peripheral Interface
MPC5554 DSPI_D    Deserial Serial Peripheral Interface
MPC5554 EBI       External Bus Interface Module
MPC5554 eDMA      enhanced Direct Memory Access eDMA Controller 2
MPC5554 eMIOS_MTS Modular Timer System
MPC5554 eQADC     Enhanced Queued Analog to Digital Converter
MPC5554 eTPU      Enhanced Time Processing Unit
MPC5554 FlexCAN_A Controller Area Network
MPC5554 FlexCAN_B Controller Area Network
MPC5554 FlexCAN_C Controller Area Network
```

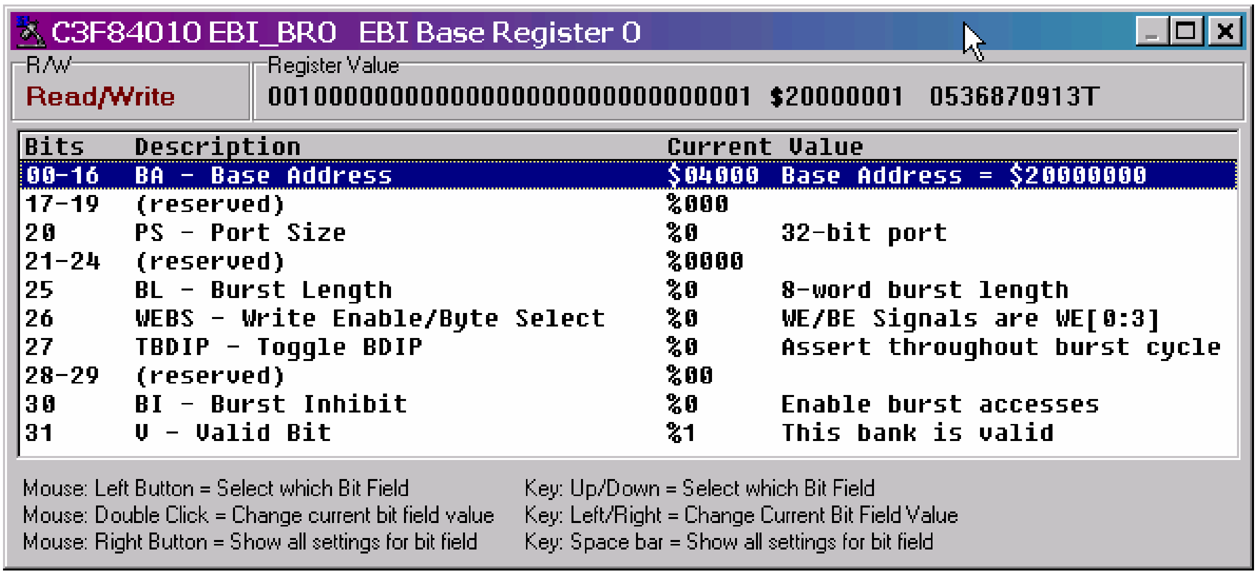
Double clicking the module of choice will launch the register selection window.



MPC5554 EBI External Bus Interface Module

```
C3F84000 EBI_MCR   EBI Module Configuration Register
C3F84008 EBI_TESR  EBI Transfer Error Status Register
C3F8400C EBI_BMCR  EBI Bus Monitor Control Register
C3F84010 EBI_BR0   EBI Base Register 0
C3F84014 EBI_OR0   EBI Option Register 0
C3F84018 EBI_BR1   EBI Base Register 1
C3F8401C EBI_OR1   EBI Option Register 1
C3F84020 EBI_BR2   EBI Base Register 2
C3F84024 EBI_OR2   EBI Option Register 2
C3F84028 EBI_BR3   EBI Base Register 3
C3F8402C EBI_OR3   EBI Option Register 3
```

Double clicking a specific register will launch the edit/display window for that register.



C3F84010 EBI_BR0 EBI Base Register 0

R/W: Read/Write

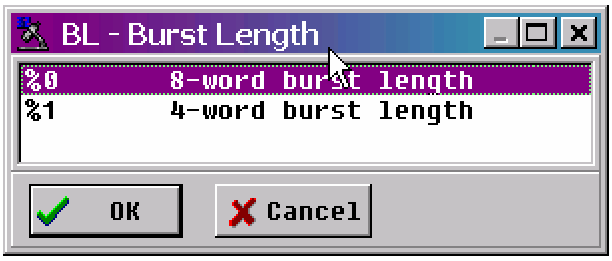
Register Value: 00100000000000000000000000000001 $20000001 0536870913T

```
Bits   Description                         Current Value
00-16  BA - Base Address                   $04000  Base Address = $20000000
17-19  (reserved)                          %000
20     PS - Port Size                      %0      32-bit port
21-24  (reserved)                          %0000
25     BL - Burst Length                   %0      8-word burst length
26     WEBS - Write Enable/Byte Select     %0      WE/BE Signals are WE[0:3]
27     TBDIP - Toggle BDIP                 %0      Assert throughout burst cycle
28-29  (reserved)                          %00
30     BI - Burst Inhibit                  %0      Enable burst accesses
31     V - Valid Bit                       %1      This bank is valid
```

Mouse: Left Button = Select which Bit Field
Mouse: Double Click = Change current bit field value
Mouse: Right Button = Show all settings for bit field

Key: Up/Down = Select which Bit Field
Key: Left/Right = Change Current Bit Field Value
Key: Space bar = Show all settings for bit field

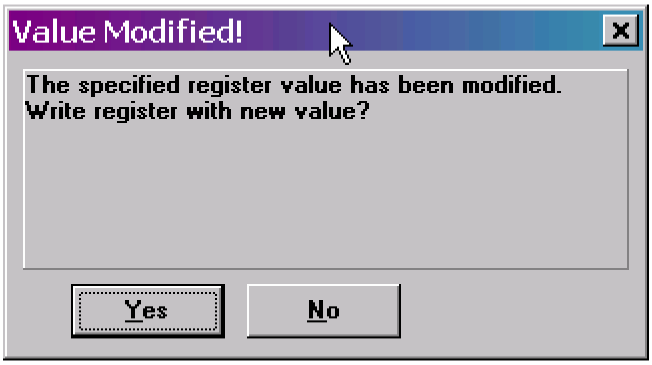
The window lists the keystrokes and mouse actions, allowing you to modify the values of each of the fields. After right clicking on a specific field, the register interpreter will display all options for that field.



BL - Burst Length

```
%0      8-word burst length
%1      4-word burst length
```

In the debugger, when you quit the register view/edit window by hitting the **ESC** key, you will be given the opportunity to write the new value into the register, as shown in the following window.

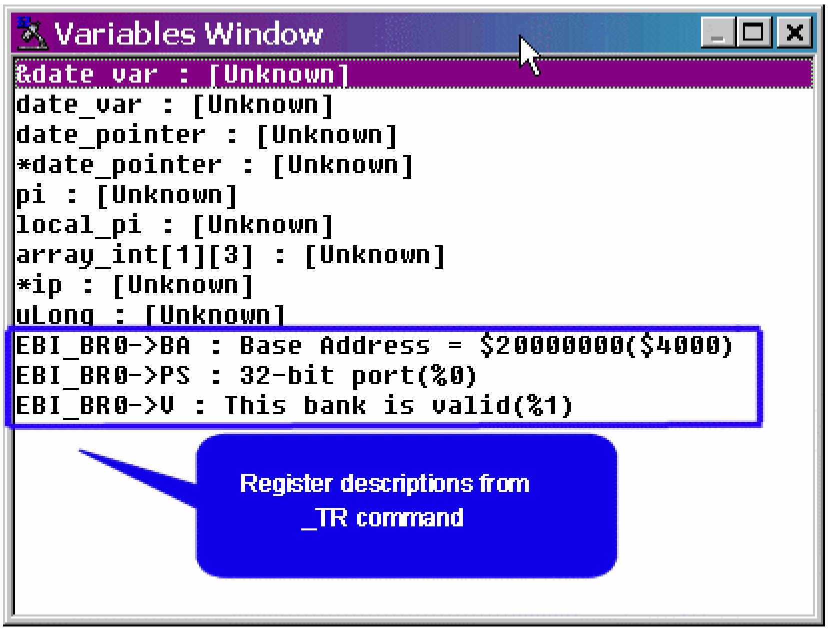


Value Modified!

The specified register value has been modified.
Write register with new value?

4.17.2 Adding Register Field Descriptions to the VAR Window

Add register bit fields to the variables window by using the "_***TR***" command in the debugger. Add bit fields defined within a register description. After selecting the register field, the field appears in the debugger variables window, and the debugger will continually update its value. A variables window with fields from a chip select register may look like this.



Variables Window

```
&date_var : [Unknown]
date_var : [Unknown]
date_pointer : [Unknown]
*date_pointer : [Unknown]
pi : [Unknown]
local_pi : [Unknown]
array_int[1][3] : [Unknown]
*ip : [Unknown]
uLong : [Unknown]
EBI_BR0->BA : Base Address = $20000000($4000)
EBI_BR0->PS : 32-bit port(%0)
EBI_BR0->V : This bank is valid(%1)
```

4.18 Executing the PowerPC Nexus Application

Issue the HGO command in the debugger. The PowerPC Nexus application will execute in real time. The LED on the target board should be blinking.

4.19 Creating an Application for SRAM

You may create an application for the MPC5554 that runs in on-chip SRAM and cache only. The PowerPC application will not use Flash memory.
Follow these steps:

1. In WinIDE, select the ***Environment menu***, New Project from Template. Select ***EVB_PHYTEC_PHYCOREMPC5554_SRAM***. Finish creating the new project and compile using <F4>. The new PowerPC application will execute in on-chip SRAM at address 0x40000000 and use the cache as stack space.

2. In WinIDE, press <F6> to launch the debugger. The debugger should automatically execute macro file ***mpc5500_any_init.mac***. If not, issue the MACRO command in the debugger and select ***C:\pemicro\pkgppcnexus\mpc5500_any_init.mac***.

3. In the debugger, issue the ***HLOAD*** command. Select your *Elf/Dwarf* file, such as ***C:\new_project\PEGnu\main.elf***. This will load the object code to SRAM as well as load the debugging information to the debugger. When you see source code in the debugger, you are ready to debug as described in the previous sections.

Document: **QuickStart Guide for the PHYTEC phyCORE-MPC5554 Rapid Development Kit**
Document number: L-486e_1, 2006

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

2006 Ordering No. L-489e_1