AN2125 APPLICATION NOTE

uPSD3300 Series Design Guide for DK3300-ELCD Using KEIL

INTRODUCTION

This application note is based upon AN1943 and has been updated for the DK3300-ELCD Development Kit and, in addition, this document focuses on the use of the DK3300-ELCD with Keil's uVision2 Software Tools. It provides guidelines for the creation and development of applications for the Turbo uPSD Family of devices and shows a number of key steps to follow for creating a design based on the DK3300-ELCD Development Kit. The Kit has all the requirements for using the sample code and examples supplied.

Here, the basic flow is provided for creating a project using Keil Software tools. A simple application included in the Kit is demonstrated using Keil and shows the key features of Keil. The key steps in designing an application are enumerated in this document. PSDsoft, a key tool in using Turbo uPSD, is explained in detail by illustrating the Design used for the demonstration section. PSDsoft has been upgraded to connect to ST's JTAG programming cable (FlashLINK) or Raisonance's JTAG programming cable (RLINK-ST)

As shown in Figure 1, the uPSD3300 family is a series of 8051-class microcontrollers (MCUs) containing a new fast Turbo 8032 core with a large dual-bank flash memory, a large SRAM, many peripherals, programmable logic, and JTAG In-System Programming (ISP). It is important to make the correct selection in PSDsoft, using the HW Setup configuration menu.

Please see the uPSD on-line resources page for latest documentation and other referenced User guides and Application notes at the URL listed below.

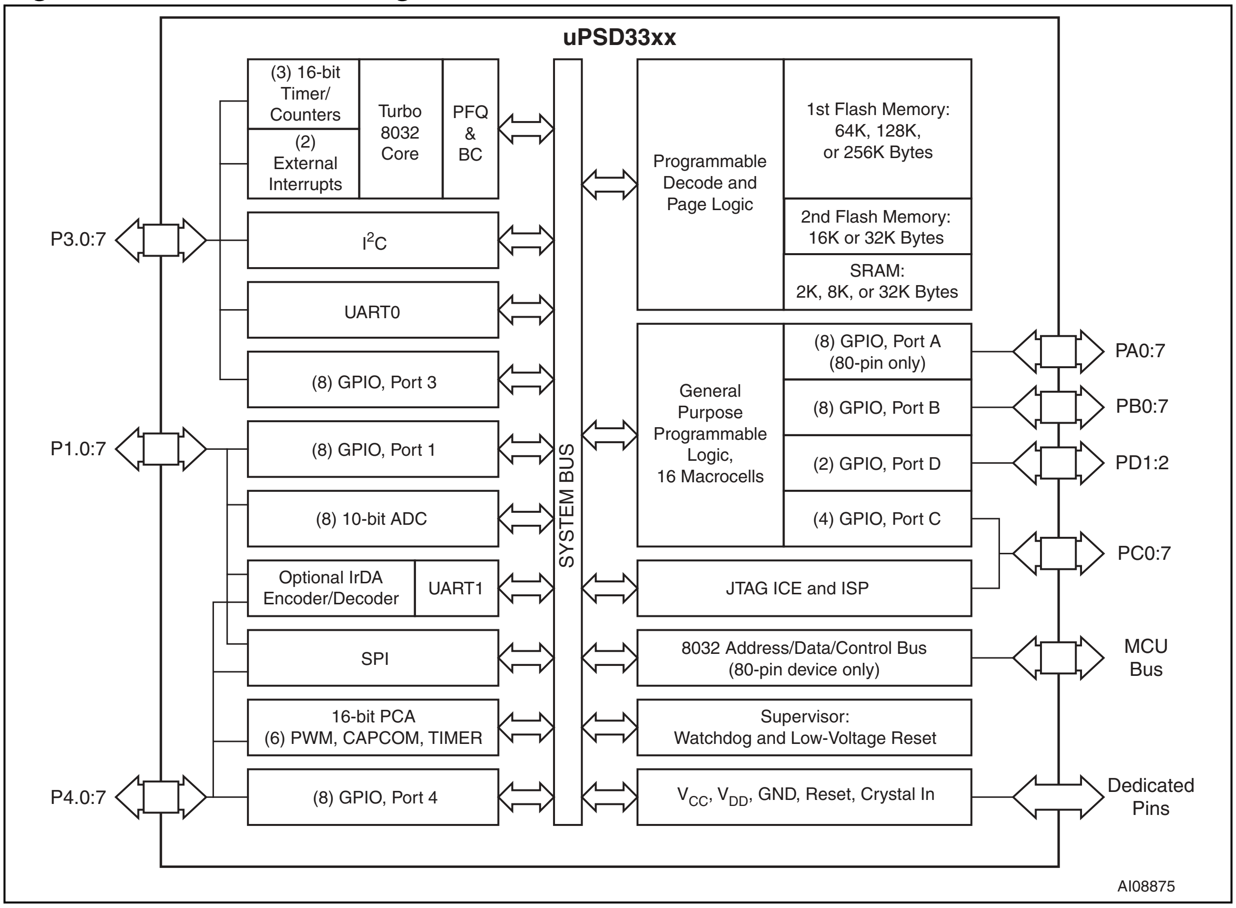
http://www.st.com/stonline/products/families/memories/psm/soft_c2.htm

Rev. 1.0

TABLE OF CONTENTS

1 UPSD3300 FAMILY

Figure 1. General Block Diagram of the uPSD33xx


uPSD33xx
(3) 16-bit Timer/ Counters
(2) External Interrupts
Turbo 8032 Core
PFQ & BC
I2C
UART0
(8) GPIO, Port 3
(8) GPIO, Port 1
(8) 10-bit ADC
Optional IrDA Encoder/Decoder
UART1
SPI
16-bit PCA (6) PWM, CAPCOM, TIMER
(8) GPIO, Port 4
SYSTEM BUS
Programmable Decode and Page Logic
1st Flash Memory: 64K, 128K, or 256K Bytes
2nd Flash Memory: 16K or 32K Bytes
SRAM: 2K, 8K, or 32K Bytes
General Purpose Programmable Logic, 16 Macrocells
(8) GPIO, Port A (80-pin only)
(8) GPIO, Port B
(2) GPIO, Port D
(4) GPIO, Port C
JTAG ICE and ISP
8032 Address/Data/Control Bus (80-pin device only)
Supervisor: Watchdog and Low-Voltage Reset
VCC, VDD, GND, Reset, Crystal In
P3.0:7
P1.0:7
P4.0:7
PA0:7
PB0:7
PD1:2
PC0:7
MCU Bus
Dedicated Pins
AI08875

1.1 uPSD3300 Family Overview

The uPSD3300 family is a turbo 4-clock per instruction 8032 MCU capable of being clocked up to 40MHz at 3.3V or 5.0V at industrial operating temperature range. Currently there are twelve family members that contain different combinations of flash memory size, operating voltage, and packaging (please see the full datasheet). In this Application Note, uPSD3334D-40U6 is used as the example. The term "Turbo uPSD" is used throughout the remainder of the document for brevity (see the Turbo uPSD3334 block diagram shown in Figure 2).

The Turbo uPSD family has a unique memory structure that includes two independent flash memory arrays (Main and Secondary) capable of read-while-write operation. This is ideal for In-Application Programming (IAP) because the 8032 can fetch instructions from one flash array while erasing/writing the other array. Individual sectors of each flash memory array can be mapped to virtually any 8032 address by the Decode PLD (DPLD) for total flexibility. The Turbo uPSD also contains a Page Register whose outputs feed the inputs of the DPLD. This allows paging (or banking) of flash memory to break the 8032's inherent limit of 64 Kbyte addresses. The 8032 may write to the Page Register at runtime.

For more complex designs, the Turbo uPSD is capable of placing each of the flash memory arrays (Main or Secondary) into 8032 code address space, into 8032 data space, or into both code and data space on the fly. Mapping flexibility like this supports IAP because either flash array may be temporarily placed into data space while the firmware is updated, then moved back into code space when finished, all under control of the 8032.

Many peripherals are available in this Turbo uPSD, including: two UART channels, one IrDA channel, one SPI channel, one I2C channel, six PWM channels, eight 10-bit ADC channels, nine Timer/Counters, a watchdog timer, low-VCC detection with reset-out, a general purpose PLD, many GPIO and a USB-JTAG Debugger.

All of the peripherals on Ports 1, 3, and 4 are controlled using 8032 Special Function Registers (SFRs).

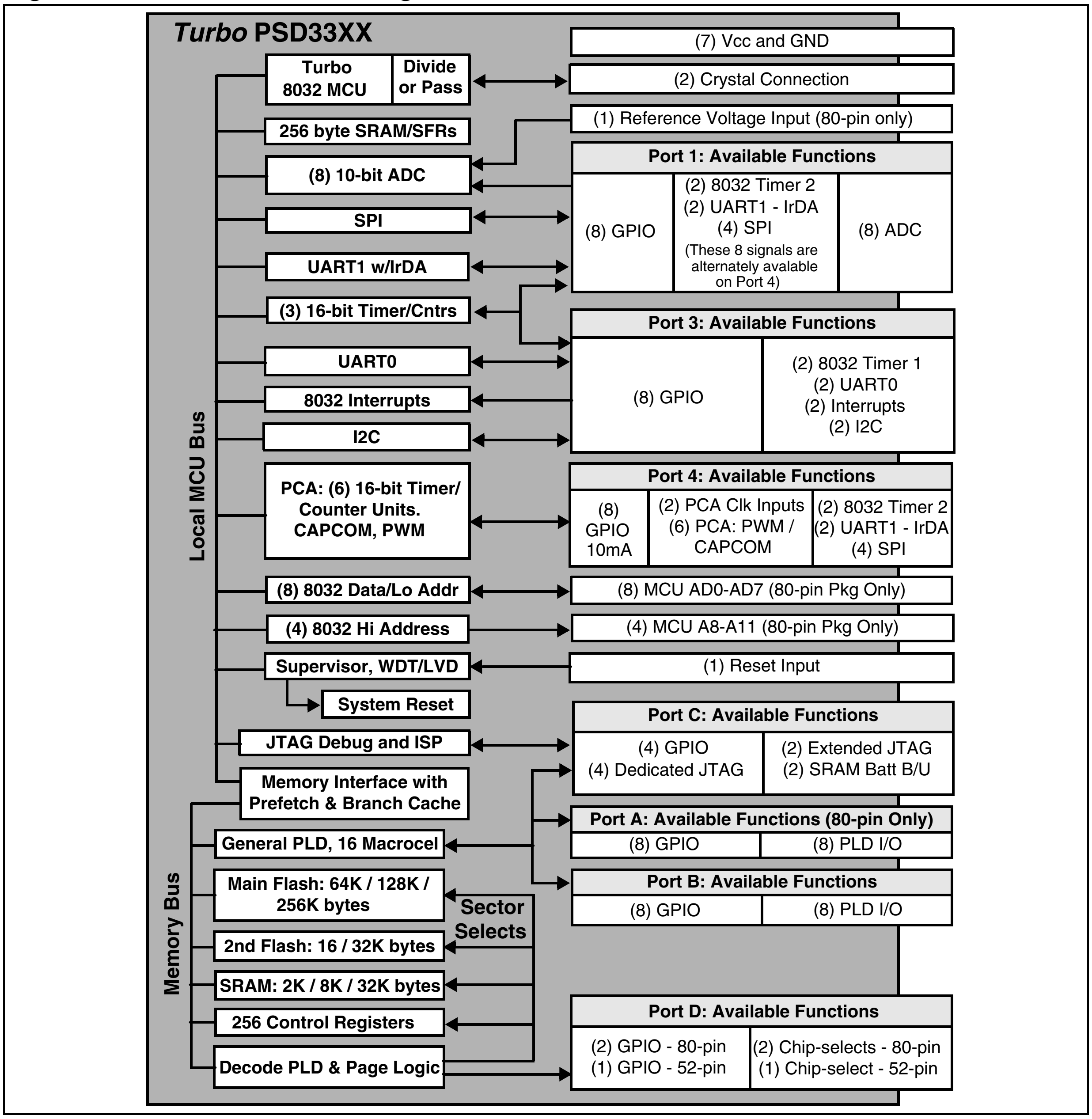
I/O Signals on ports A, B, C, and D are controlled one of two ways:

1. by a block of xdata memory mapped control registers, whose base address (csiop) can be mapped anywhere using the DPLD; and
2. by the programmable logic.

In addition, Turbo uPSD offers a Cross-Bar I/O, which means that Peripheral functions on Port 1 are also available on Port 4 (cross-bar switch), providing more flexibility. There is no need to sacrifice one peripheral function when two functions are available on a single pin, just use the other port.

The JTAG channel on Port C is used for ISP and debug of the 8032 MCU core. ISP is ideal for rapid code iterations during firmware development and for Just-In-Time inventory management during manufacturing. JTAG ISP eliminates the need for sockets and pre-programmed devices, and requires no participation of the 8032. JTAG debug eliminates the need for expensive and intrusive hardware In-Circuit Emulator (ICE).

Figure 2. uPSD3334D Block Diagram



2 DK3300-ELCD DEVELOPMENT KIT

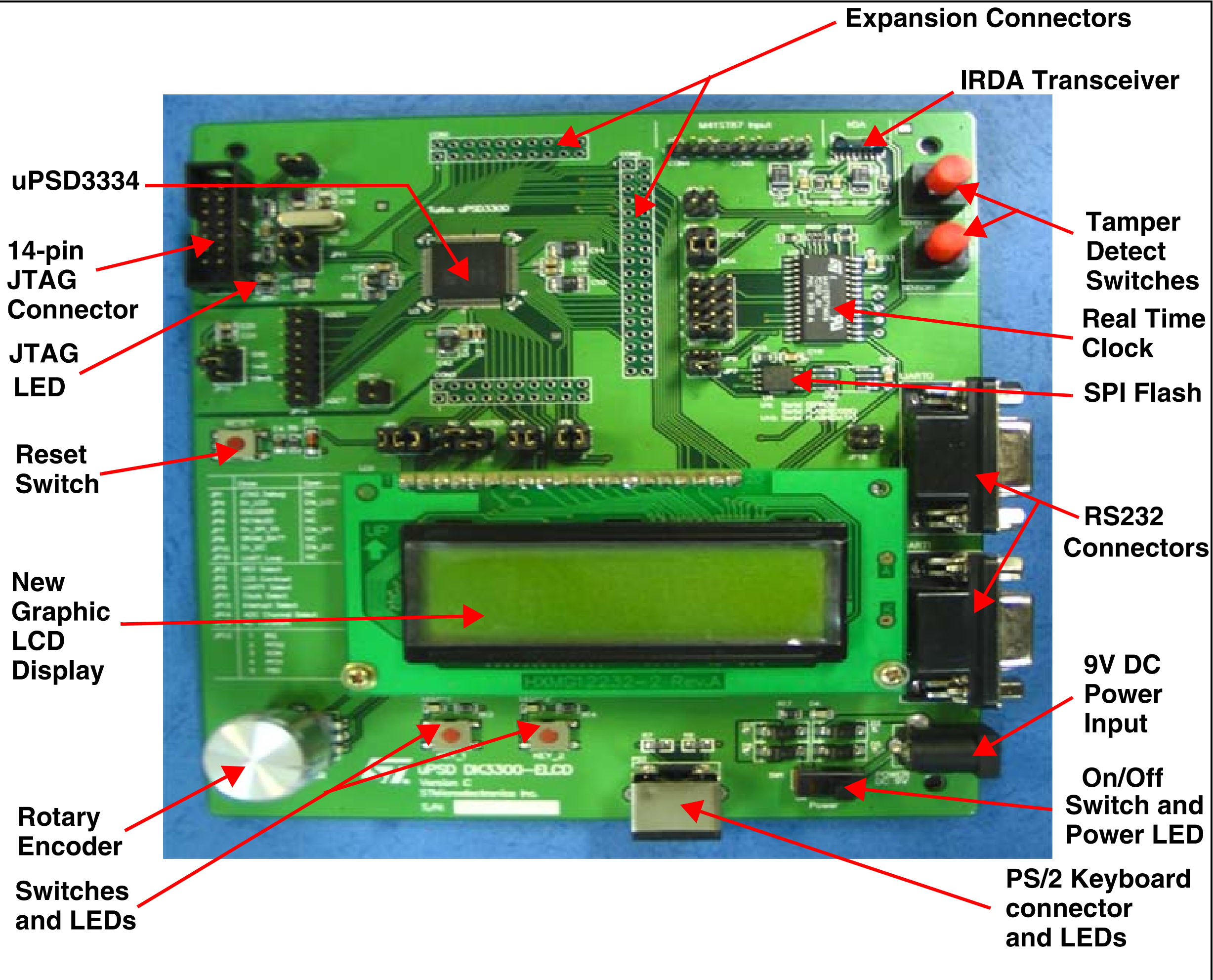
2.1 Overview

A picture of the DK3300-ELCD board, including the important connections, is shown in Figure 3. A list of jumpers JP0 - JP16 and their functions can be found on the DK3300-ELCD board's silk screen. For more detailed information on these jumpers, please refer to APPENDIX A., DK3300-ELCD JUMPERS SELECTION AND DEFAULTS or the DK3300-ELCD User's Guide. Board layout and schematics are available. Connectors CON1, CON2, and CON3 provide easy access to all Turbo uPSD signals for expansion or testing. UARTs are available on connectors marked UART0 and UART1. The FlashLINK/ RLINK-ST/ ULINK JTAG ISP cable connects at the connector, JTAG. The DK3300-ELCD has a graphical LCD interface and a fully-featured real-time clock with a back-up battery, a serial EEPROM, IrDA transceiver, PWM giving control over the LCD brightness, and a rotary encoder knob for selection of various demo applications. Board layout and schematics are available.

The sample design example code used for this application note is a Keil based project which shows use of the PWM and ADC function blocks within the uPSD3300 device. A pulse width modulated signal output from the PWM circuit is tied to an RC circuit resulting in a DC voltage that is proportional to the pulse width. This DC voltage is input to an ADC channel and is read after each time the pulse width out of the PWM is changed. The PWM setting and the ADC value read is displayed on the LCD.

The purpose of using this simple design project is to illustrate and demonstrate the use of Keil software and tools with the ULINK adapter on a uPSD development board. The Keil tools provide many features for editing, compiling, programming, and debugging a uPSD3300 MCU Series from STMicroelectronics. In the following sections, some of the main features are described to give you a feel for the simplicity and capabilities of the tools used for this sample design.

Figure 3. DK3300-ELCD Development Board Components



2.2 Contents of DK3300 Kit

STMicroelectronics provides a DK3300-ELCD Development Kit which is shipped with the following contents:

– Circuit Board: uPSD DK3300-ELCD Development Board- with a uPSD3334D-40U6 MCU with Enhanced Graphic LCD

- RLINK-ST, a USB-based JTAG adapter from Raisonance for debugging with Raisonance Integrated Development Environment (RIDE)
- ULINK, a USB-based JTAG adapter from Keil for debugging with Keil's uVision Tools
- USB Cables and RS232 Cables
- 110/220V Universal Power Supply Adapter
- Software and Tools CD's:

– RKit Development Suite CD from Raisonance contains:
  - RIDE C-Compiler and Assembler (limited to 4 Kbytes code size)
  - RIDE Debugger Utility (no code size limit)
  - uPSD3300 sample projects and Application Note
  - Also includes ST's PSDsoft Express software for configuring the Programmable Logic inside the uPSD3300

– DK3300-ELCD CD from STMicroelectronics contains:
  - STMicroelectronics Datasheets, Tools, Software, uPSD3300 sample projects
  - User Manual and Application Notes
  - Keil uVision 2 Software and support Tools (Demo Version) for uPSD - (Limited to 2 Kbytes code size)

Note: This Application Note assumes that you have access to the Development Kit and software tools.

3 PROJECT CREATION AND SAMPLE DESIGN DEVELOPMENT PROCESS

In the sections below you will be introduced to the process of using Keil for creating application Code using the Development board and the associated tools supplied with the Kit. The key steps for creating a new project with Keil are described. This is followed by section that uses the Keil environment and DK3300-ELCD board to demonstrate the sample design. The main features of Keil and its usage are then shown by loading and debugging the sample application. This way you gain familiarity with the board and the development to design your own applications. The unique feature of Turbo uPSD requires understanding of PSDsoft Express tool from STMicroelectronics. Any Turbo uPSD-based design requires having a good understanding and features of this tool. Following a detailed description of the process and features of PSDsoft used in this simple design, the design block diagram and the memory map are then explained where PSDsoft is used to set up set up the various components, interfaces and the programmable logic.

3.1 Key Design Development Steps

Design and development of applications using Turbo uPSD Family of products require use of both Development Boards from STMicroelectronics or hardware developed by the user in conjunction with Software and Tools that support uPSD Devices. It is important to follow some simple steps and guidelines for successful implementation of the project.

STMicroelectronics provides full support with Hardware Development Kits and Software Tools, utilities and support through the Support Website. The key design development steps for using Keil tools are as follows:

- Identify and select the right development Kits and Tools
- Design a Block Diagram of your Application in relation to the Turbo uPSD
- Design the Logic and connections to be used for the PLD available in uPSD
- Create Memory Maps and inputs for programming devices using PSDsoft Express tools
- Develop your application Code for the chosen Compiler (the Keil uVision2 is used here)
- Verify the project needs and match with the device used
- Compile and create the firmware and applications Code
- Enter data from the Block diagram and memory maps using PSDsoft Express Design flow
- Merge MCU and PSDsoft project to get the object file
- Flash the Code and Data using Tools including JTAG - ISP (RLINK-ST / Flashlink)
- Debug, make changes, reprogram and finalize the project
- Test and qualify the design

This application note provides guidelines for design by showing the key steps as mentioned above. The document has been divided into sections that cover various areas. It is expected that the reader has previous experience of programming and applications development including the use of compilers.

First, an introduction to the Turbo uPSD. The family basic block diagram and features are introduced, followed by an overview of the DK3300-ELCD Development Kit (Board). This hardware is used with the Keil Tools to demonstrate ideas for project creation and design development. A simple example is used for demonstration and explained in detail to provide an understanding of the Keil tools and the DK3300-ELCD Development Kit. It is hoped that with this information and other supporting documents available from STMicroelectronics, you can design and develop your application/project using Turbo uPSD.

3.2 Requirements

In order to follow the examples and processes described here you will need a DK3300-ELCD Development Kit as described above (Refer to section 2.2, Contents of DK3300 Kit).

Note: The DK3300-ELCD Development Kit meets all the requirements for use in the example described here.

To use RLINK-ST, you will need a USB port and a Windows Operating System supporting the USB (Win98SE, Win2000, Me, XP).

RLINK-ST is a Full-Speed device (12Mbit/s). It is delivered with a standard USB cable.

Note: Win95, Win98 First Edition and NT4.0 do NOT support USB.

IMPORTANT:

You must install PSDsoft Express 8.10 or later to have the capability to program a uPSD device.

3.3 Software Installation and Connections:

3.3.1 Software Installation

- Insert the Raisonance disc supplied with DK3300-ELCD Development Kit in your CD drive. The auto-run will bring up the program installation menu.
- First Install PSDsoft Express, taking all the default choices.
- Remove the Raisonance Disc and then insert the DK3300-ELCD Disc from STMicroelectronics in the CD drive.
- Navigate the screens and bring up the Menu page with "Install ST and 3rd Party Tools". Click on it and then select ‘Keil IDE (Eval Version)’. Note this version is limited to 2KB of Code size. Install Keil uVision 2 Compiler.
- Next, go back to the Home menu on the CD and then navigate to bring up Menu page with "Copy Device Drivers and Demo Code". Click on it and then:
- Unzip the file pointing to C:\ root directory in your drive.

3.3.2 Physical Connections

- Connect RLINK-ST to your PC/Laptop using the supplied USB cable and let the USB driver install on Windows.
- Connect RLINK-ST ribbon cable to the JTAG connector on the DK3300-ELCD circuit board.
- Make sure that the Board is powered up using the Universal Adapter supplied with the kit and ensure that it is functioning. (See that the LCD display shows the text).
- Make sure that the Jumpers are set correctly. (Refer to APPENDIX A. at the end of the document for Jumper settings).

3.3.3 Installing Keil uVision

Note: the PSDsoft software must be installed first.

ST provides the Evaluation/Demo version of Keil with the DK3300-ELCD which is fully functional and licensed but limits Code size to 2KB. Users need to contact Keil (http://www.keil.com) for acquiring a full version of uVision tools without limitations.

In this application note the simple design example used can be used with the demo version.

- Use the executable uVision file to install the demo version of Keil on your computer. The Keil uVision tools provide an Integrated Development Environment that combines project management, source code editing, and program debugging in one single powerful environment.

The Keil ULINK connects directly to the JTAG pins of the STMicroelectronics (Turbo) uPSD device. There is only a JTAG connector required on the hardware.

The JTAG interface provides a broad range of debug features on the standard production chip. It allows you to debug the application which is running on an unmodified target system with minimal hardware overhead. In addition, the JTAG interface may be used as high-speed interface for programming the on-chip Flash.

JTAG debugging provides several features that are fully supported by the Keil uVision Debugger:

- Start / Stop the microcontroller.
- Up to 4 code or access breakpoints. Even an address range is supported.
- Read/write access to whole address space (data, code, xdata).
- Single step or procedure step program execution.
- Instruction Trace

Additional features will be shown with the simple demonstration program used with this application note in the next section.

With the uVision IDE you can download and test your applications in the same environment. This shortens the development cycles.

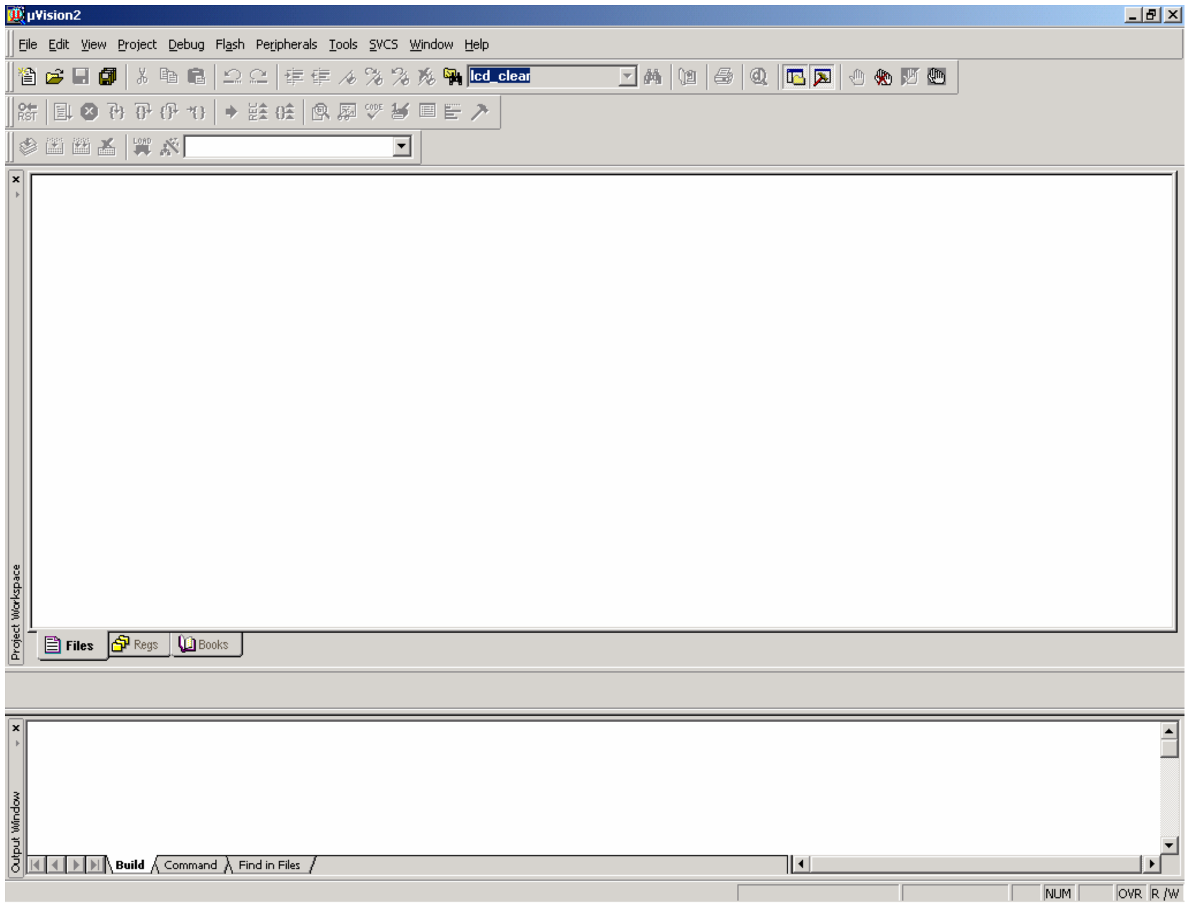
4 USING KEIL AND ULINK FOR CREATING A NEW PROJECT

Keil's IDE environment provides a very simple way to create a new project by adding files and building the project. The Hex code generated is merged with a PSDsoft project to generate the object code. This merged object code is then 'flashed' into the target device using Keil IDE Tools and the ULINK JTAG Adapter. Below are shown the basic Screenshots and steps involved in this process.

Launch KEIL from the Windows programs menu or from the KEIL icon on your desktop.

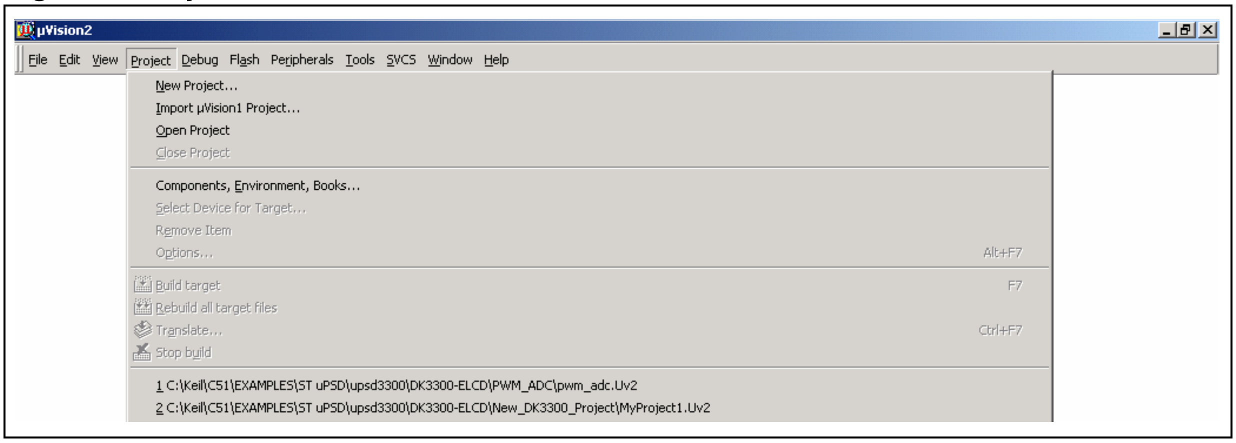
Next, you'll see a blank work area with the Keil uVision2 title menu bar as shown in Figure 4 below:

Figure 4. uVision2



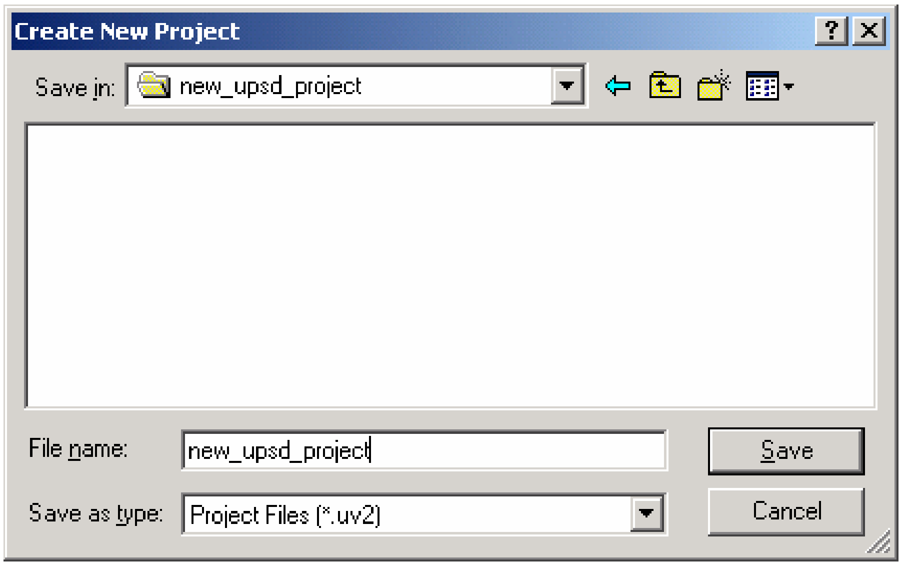
Next from the Project Pull down menu as shown, select new project:

Figure 5. Project Pull-down Menu


µVision2
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
New Project...
Import µVision1 Project...
Open Project
Close Project
Components, Environment, Books...
Select Device for Target...
Remove Item
Options... Alt+F7
Build target F7
Rebuild all target files
Translate... Ctrl+F7
Stop build
1 C:\Keil\C51\EXAMPLES\ST uPSD\upsd3300\DK3300-ELCD\PWM_ADC\pwm_adc.Uv2
2 C:\Keil\C51\EXAMPLES\ST uPSD\upsd3300\DK3300-ELCD\New_DK3300_Project\MyProject1.Uv2

You will get the following Screen. Enter your Project Folder and uvision project you want to create.

Figure 6. Create a New Project

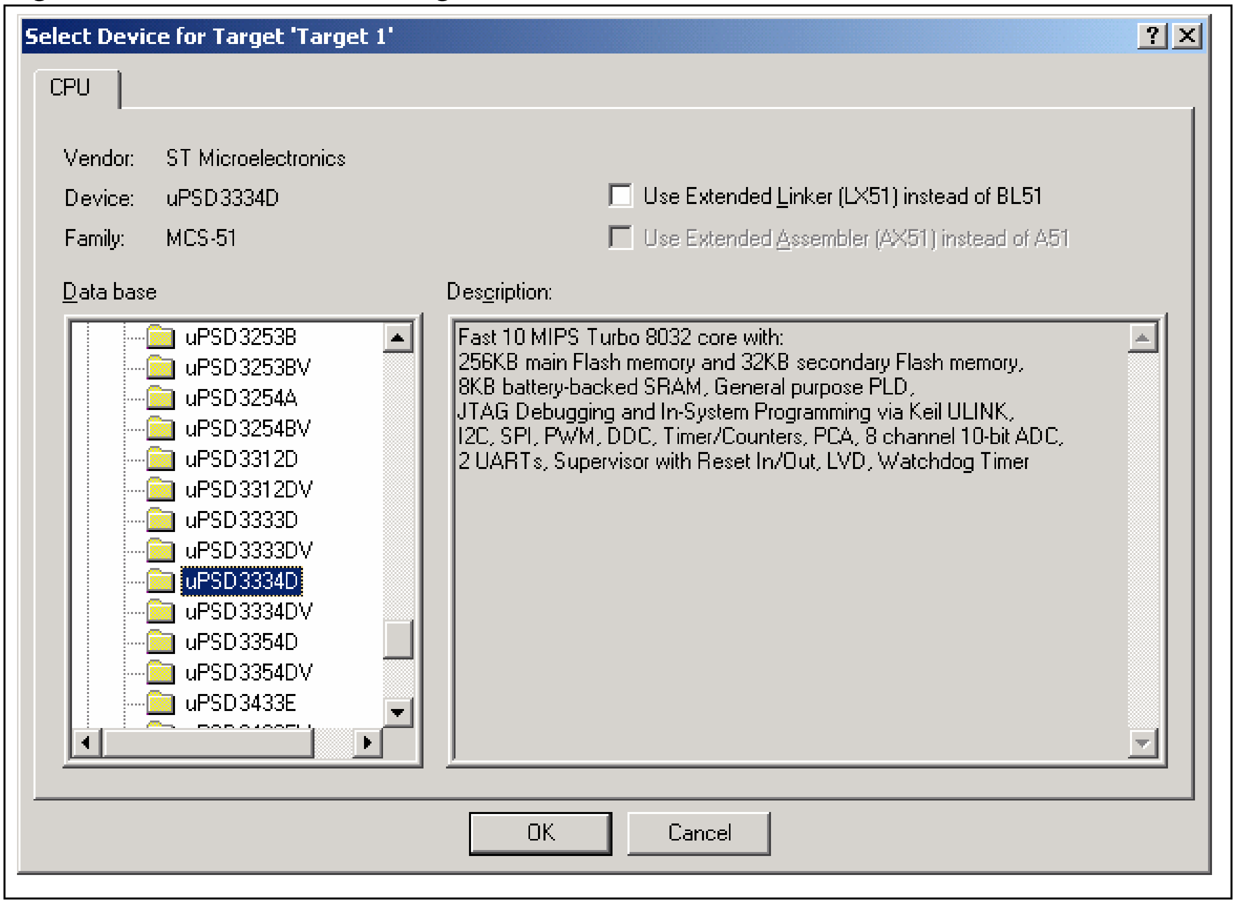


File name: new_upsd_project

Save as type: Project Files (*.uv2)

The name "new_upsd_project" is being used as the new project folder and project file name in this example. Save the file name and you will automatically be asked via a screen (Figure 7) to select the device for the target.

Figure 7. Select Device for Target



Vendor: ST Microelectronics

Device: uPSD3334D

Family: MCS-51

☐ Use Extended Linker (LX51) instead of BL51

☐ Use Extended Assembler (AX51) instead of A51

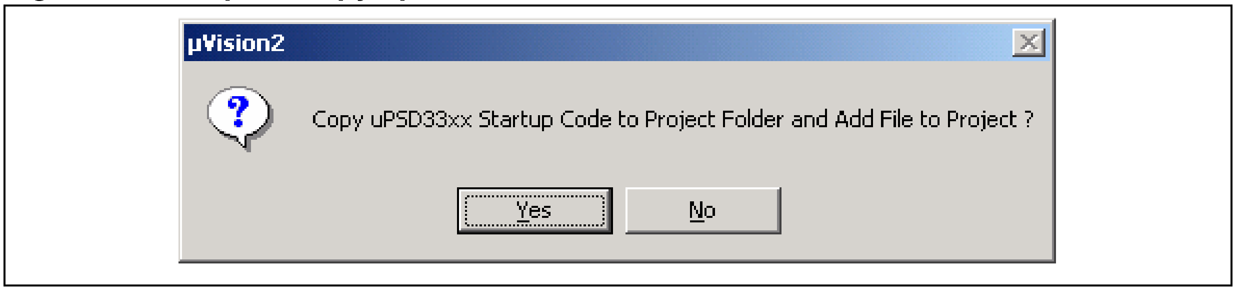
uPSD3253B
uPSD3253BV
uPSD3254A
uPSD3254BV
uPSD3312D
uPSD3312DV
uPSD3333D
uPSD3333DV
uPSD3334D
uPSD3334DV
uPSD3354D
uPSD3354DV
uPSD3433E

Fast 10 MIPS Turbo 8032 core with:
256KB main Flash memory and 32KB secondary Flash memory,
8KB battery-backed SRAM, General purpose PLD,
JTAG Debugging and In-System Programming via Keil ULINK,
I2C, SPI, PWM, DDC, Timer/Counters, PCA, 8 channel 10-bit ADC,
2 UARTs, Supervisor with Reset In/Out, LVD, Watchdog Timer

Select the device for the DK3300_ELCD Board. It is uPSD3334D.

The next screen provides you with the option to copy the appropriate startup files for this device as shown in Figure 8.

Figure 8. Startup file copy option



Copy uPSD33xx Startup Code to Project Folder and Add File to Project ?

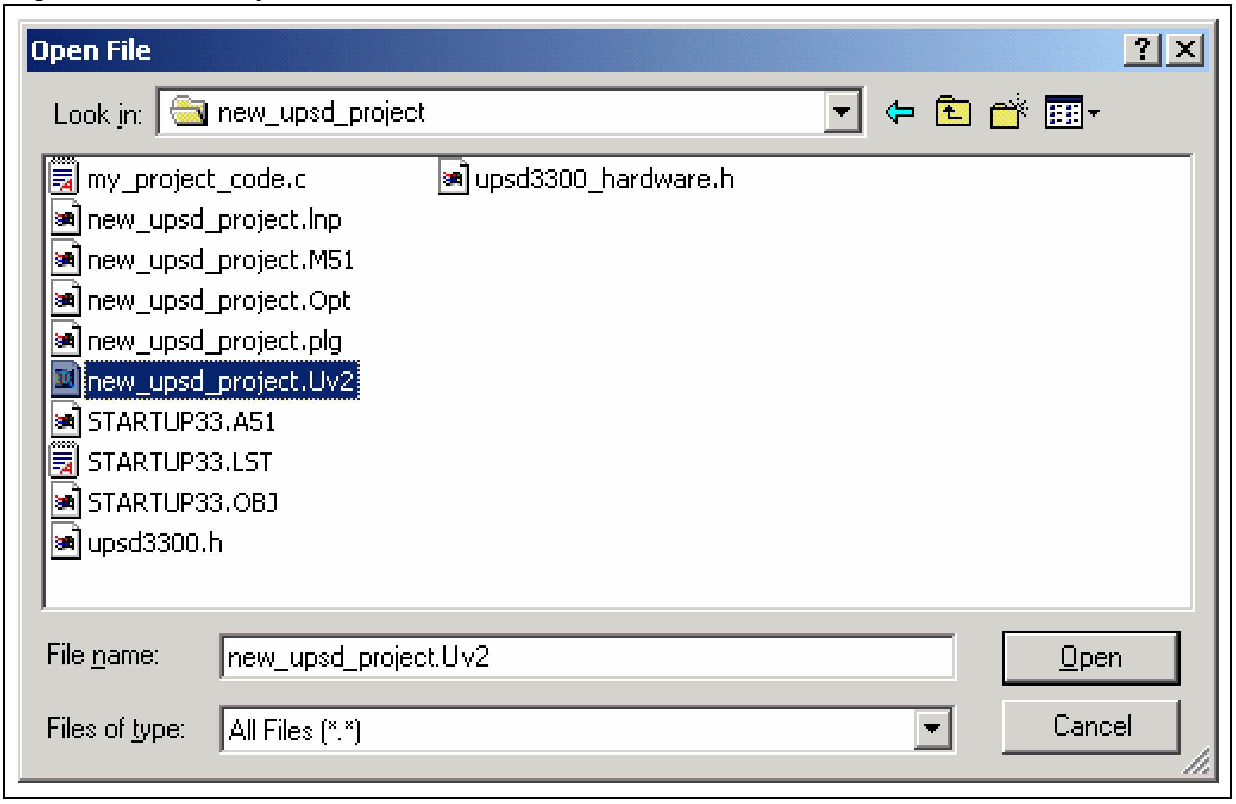
Select “Yes”: Your IDE environment will be updated to reflect the added file.

Next, you must copy the appropriate header files for the device chosen. These files can be taken from other sample projects or examples files.

After this you can create your own code files and add them to the source group, e.g. the file “my_project_code.c” file.

When you check your project folder you should have your required header and code files as well as startup files. Your folder may have files along the lines shown in Figure 9.

Figure 9. New Project folder Basic files



Open File

Look in: new_upsd_project

my_project_code.c
new_upsd_project.lnp
new_upsd_project.M51
new_upsd_project.Opt
new_upsd_project.plg
new_upsd_project.Uv2
STARTUP33.A51
STARTUP33.LST
STARTUP33.OBJ
upsd3300.h
upsd3300_hardware.h

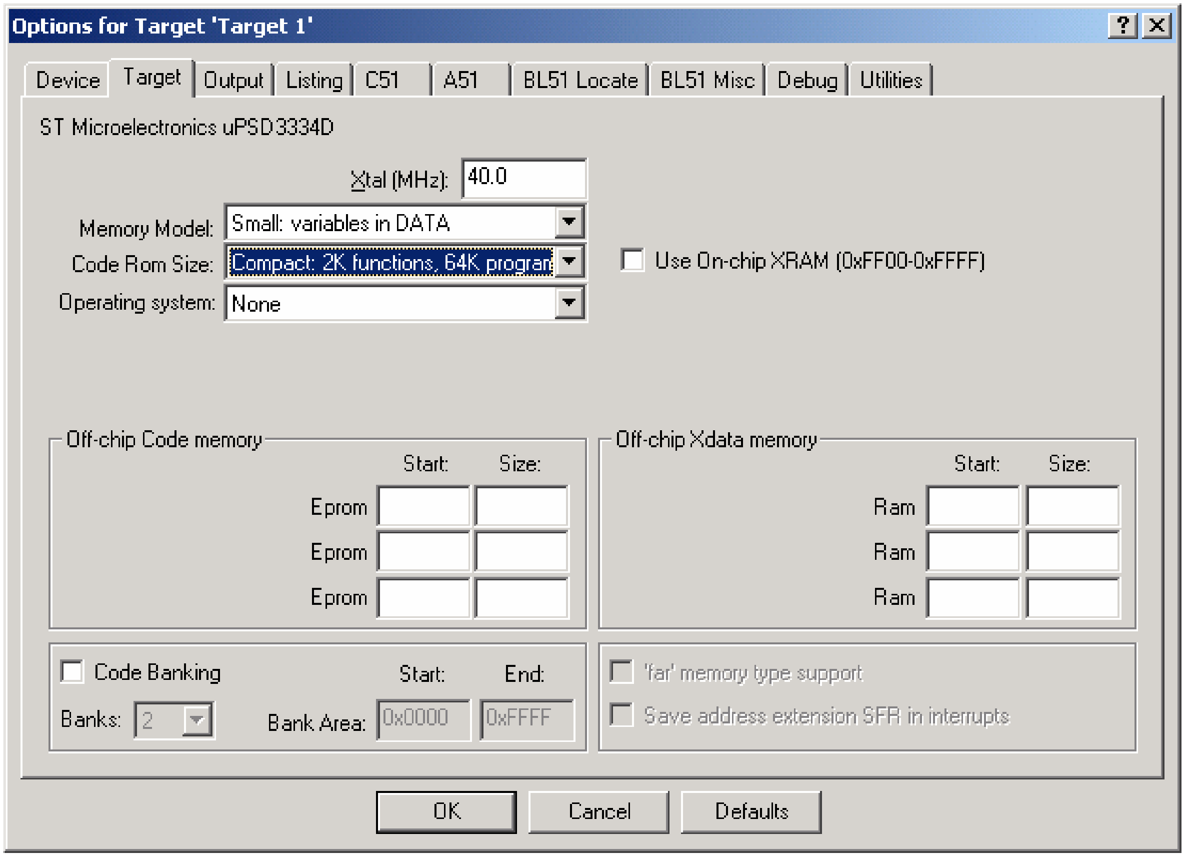
File name: new_upsd_project.Uv2

Files of type: All Files (*.*)

4.1 Device/target Setup Environment Options

After you have added all your code and ensured that all PSDsoft project files related to this project have been created and saved; you should click on the project options screen to set the environment. The following screens show the options to be taken:

Figure 10. Options for Target



ST Microelectronics uPSD3334D

Xtal (MHz): 40.0

Memory Model: Small: variables in DATA

Code Rom Size: Compact: 2K functions, 64K program

Operating system: None

[ ] Use On-chip XRAM (0xFF00-0xFFFF)

Off-chip Code memory

	Start:	Size:
Eprom		
Eprom		
Eprom		

Off-chip Xdata memory

	Start:	Size:
Ram		
Ram		
Ram		

[ ] Code Banking Start: End:

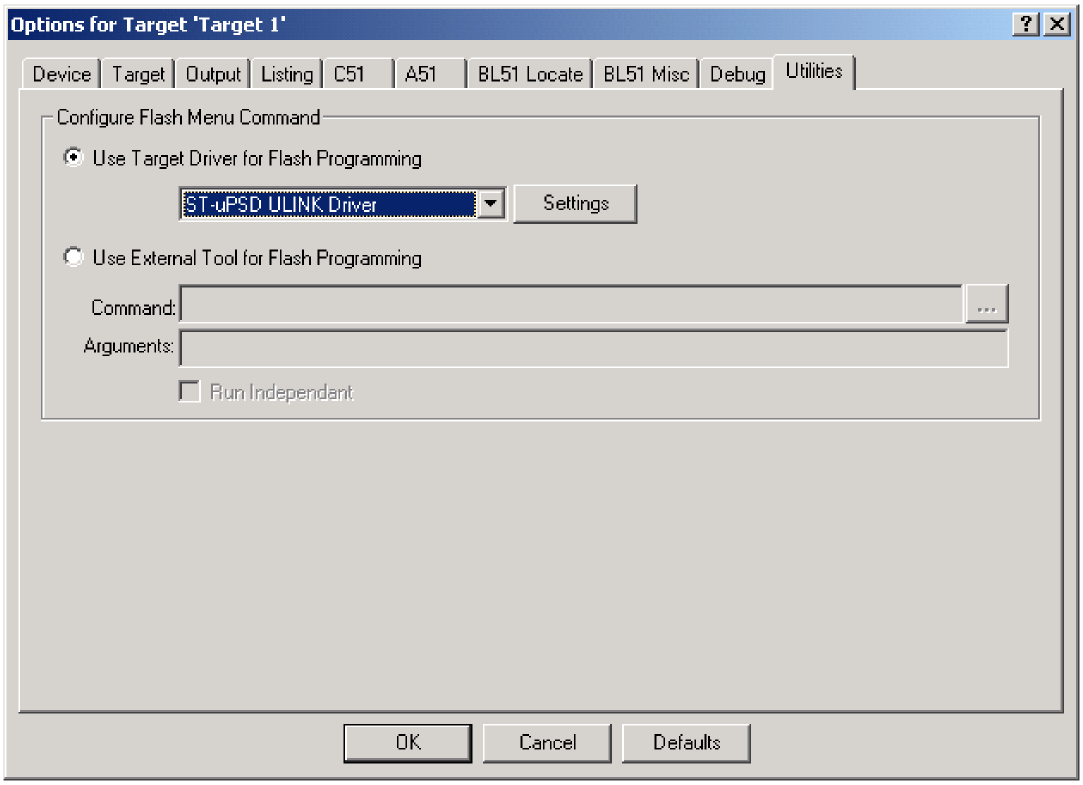
Banks: 2 Bank Area: 0x0000 0xFFFF

[ ] 'far' memory type support

[ ] Save address extension SFR in interrupts

Note: It is important that the selection for Code Rom Size= Small Program or 2K or less should not be used with the Evaluation/Demo version of Keil Tools for this Application Note.

Figure 11. Configure ULINK



Configure Flash Menu Command

(●) Use Target Driver for Flash Programming

ST-uPSD ULINK Driver

( ) Use External Tool for Flash Programming

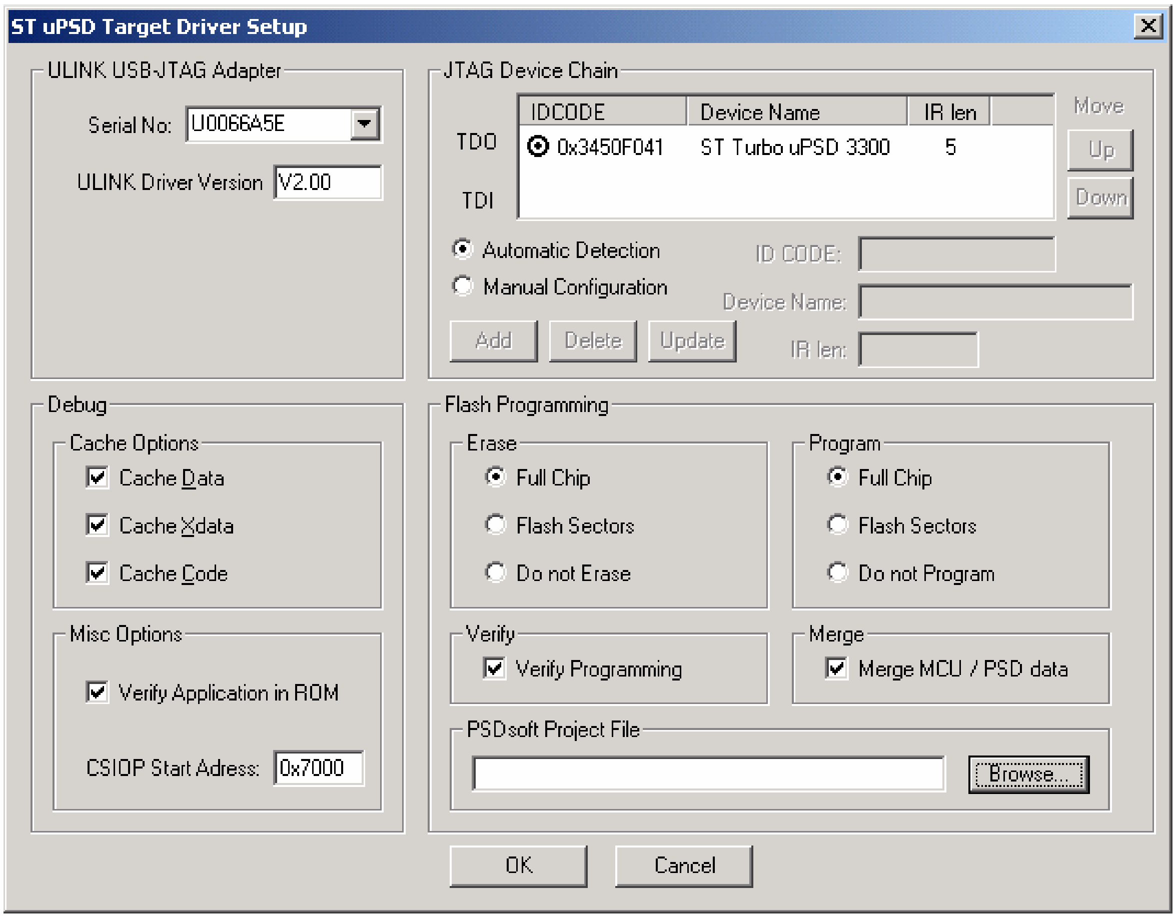
Command:

Arguments:

☐ Run Independant

Make sure you select the right ULINK drivers

Figure 12. uPSD ULINK JTAG Setup Options



ST uPSD Target Driver Setup

ULINK USB-JTAG Adapter
- Serial No: U0066A5E
- ULINK Driver Version: V2.00

JTAG Device Chain

	IDCODE	Device Name	IR len
TDO	● 0x3450F041	ST Turbo uPSD 3300	5
TDI			

[✓ Automatic Detection] Manual Configuration

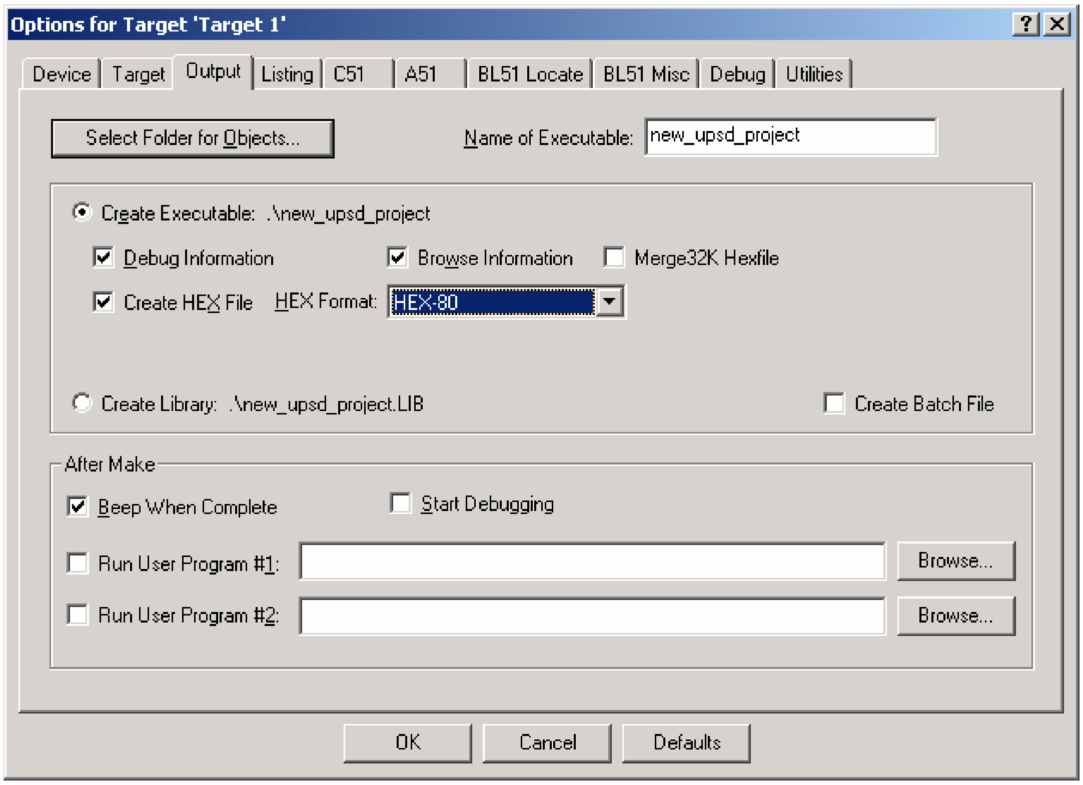
ID CODE: Device Name: IR len:

Debug
- Cache Options: [✓ Cache Data] [✓ Cache Xdata] [✓ Cache Code]
- Misc Options: [✓ Verify Application in ROM]
- CSIOP Start Adress: 0x7000

Flash Programming
- Erase: [✓ Full Chip] Flash Sectors Do not Erase
- Program: [✓ Full Chip] Flash Sectors Do not Program
- Verify: [✓ Verify Programming]
- Merge: [✓ Merge MCU / PSD data]
- PSDsoft Project File:

Note: ensure that you have the correct PSDsoft Project file (*.ini) for your device and application in this setting:

Figure 13. Target Output Options setting



Options for Target 'Target 1'

Device | Target | Output | Listing | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities

Select Folder for Objects...   Name of Executable: new_upsd_project

[✓ ●] Create Executable: .\new_upsd_project

[✓] Debug Information   [✓] Browse Information   [ ] Merge32K Hexfile

[✓] Create HEX File   HEX Format: HEX-80

○ Create Library: .\new_upsd_project.LIB   [ ] Create Batch File

After Make

[✓] Beep When Complete   [ ] Start Debugging

[ ] Run User Program #1:   Browse...

[ ] Run User Program #2:   Browse...

OK   Cancel   Defaults

Select the correct HEX file format

5 SAMPLE DESIGN AND DEMONSTRATION CODE FOR KEIL

5.1 Purpose

The demonstration application code is a Keil-based project that shows the use of the PWM and ADC function blocks within the uPSD3300 device. A pulse-width modulated signal output from the PWM circuit is tied to an RC circuit resulting in a DC voltage that is proportional to the pulse width. This DC voltage is input to an ADC channel and is read after each time the pulse-width signal from the PWM is changed. The PWM setting and the read ADC value is displayed on the LCD.

This simple demonstration project will illustrate the powerful software development tools based upon Keil software, and the ULINK JTAG adapter, which provides many features for editing, compiling, programming, and debugging a uPSD3300 MCU Series from STMicroelectronics. This demo will quickly illustrate the specific features below to give you a feel for their simplicity and capability:

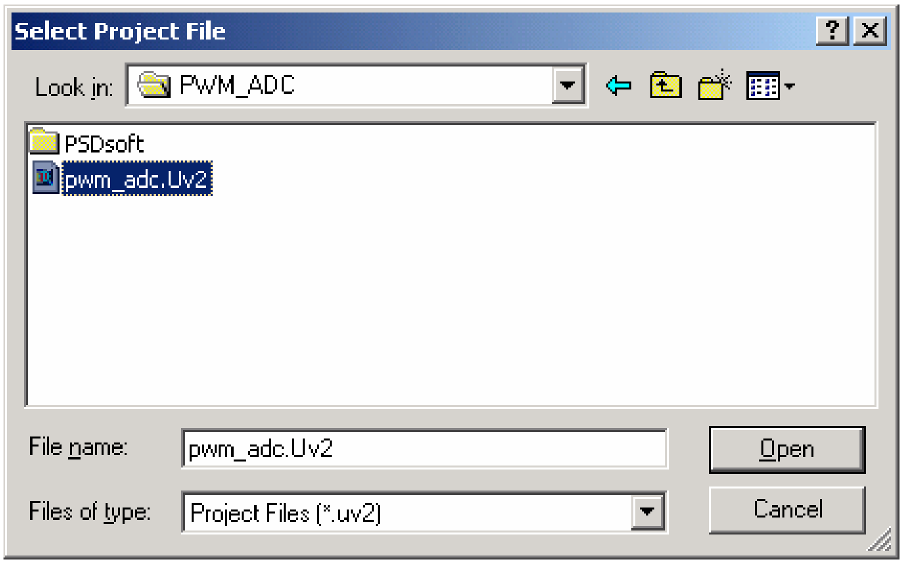
- Project Compilation and Flash Programming
- Single-Step Execution and Source-Level Debugging
- Device-Specific Formatted Displays
- Breakpoints
- Symbolic Debugging and Variables Watch
- Code Iteration
- Instruction Tracing approaching Real-Time performance

The ULINK kit, and the DK3300-ELCD Development Board or your own designed circuit board with a uPSD3300 MCU is all that is needed to develop code. Keil's debugger utility can be used to symbolically debug 8051 code generated. The demo version limits code size and usage. Check (http://www.keil.com) for more information on Keil Software and upgrades.

5.2 Compile Project

Launch KEIL from the Windows programs menu or from KEIL icon. Select Open Project and follow the path to select the project file as shown in Figure 14

Figure 14. Select Project File



- Double- click on ***pwm_adc.uv2***,:
  \KEIL\C51\examples\st_upsd\upsd3300\DK3300-ELCD\PWM_ADC\pwm_adc.uv2
  The Keil environment will display with multiple windows and icons. Refer to Figure 48 in **APPENDIX B.** for help as you follow along.

- Click "Options" to verify that PSDsoft project paths and other target options are shown correctly.

- Click "Build All". This compiles and builds the project. Messages are displayed in the output window.

5.3 Flash Programming

- Click "LOAD". This operation will flash the merged object code in the target device and for use by the debugger.

- Start the Debugger by clicking "Start/stop debug". This starts the debuuging and refreshes the Keil environment showing actions in the "Output Window". Refer to Figure 48 in **APPENDIX B.**

5.4 Single-Step Execution

- Click "Run" to see that program runs full speed with the DK3300-ELCD Display showing the following lines of text:
  PWM to ADC DEMO

PWM = *xx* ADC = *yyy* where *xx* and *yyy* are changing with each execution of the program loop.

- Click "RST" and the program returns to first line of the main program.
- Click "Step-Over" to execute one line of C code. You can scroll down to see more lines of code in the program.
- Click "Step-Over" again, which passes you over, and continue until **timer0_init()**.
- When the yellow pointer is on **timer0_init()**, click "Step-In". The debugger is now in the called function, **timer0_init()**.
- Double click "Disassembly". This opens a display named, "Disassembly" showing both C and Assembly code source instructions.
- Click "Step-Over" a few times to see that code execution can be stepped one assembly instruction at a time.
- Click "Reset" to return to main program, ***pwm_adc.c***

5.5 Breakpoints

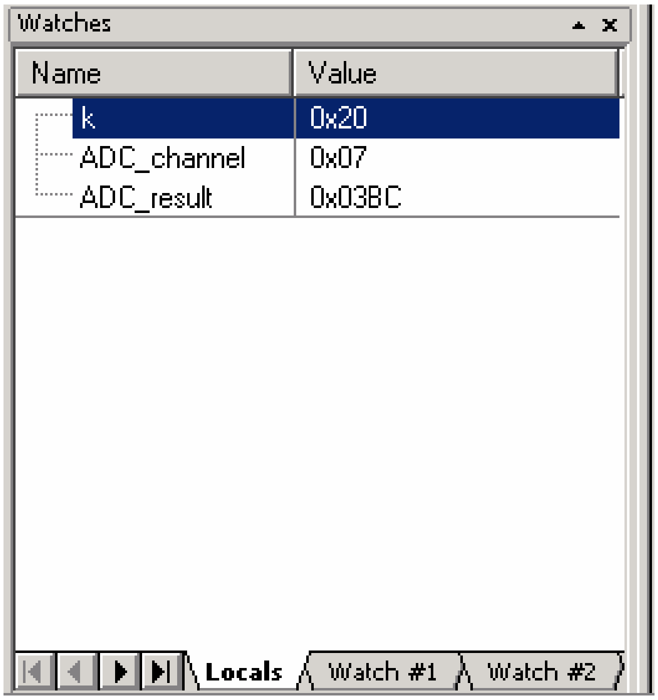
Four hardware breakpoints are available on uPSD3300.

- Set one breakpoint by setting the cursor on the left of the line of code (printfLCD(msg_buff);) and clicking on "Insert Breakpoint" ; a red dot at the beginning of line of code appears as shown in Figure 48 in **APPENDIX B.**
- Click "Run", the program will run until hitting breakpoint. The yellow pointer indicates where MCU execution has stopped in the main program, now waiting for your debugging command.
- Click "Run" repeatedly. See that the display value changes for PWM and ADC after each cycle. You will see increasing values for PWM and decreasing values for ADC each time one loop of code is executed within the **while(1)** loop construct.

5.6 Debugging and Variables Watch

- For this project watch variables were selected during Target device set up aand they are shown in the watch Window which appears at bottom of screen (Refer to Figure 15). See how at each stepping the variables change.
- Click "Run" repeatedly. See that the display value changes for PWM and ADC after each cycle.

Figure 15. Variable Watch Window



Name	Value
k	0x20
ADC_channel	0x07
ADC_result	0x03BC

5.7 Code Iteration

- Make this code change in Flash memory.
- Close the Debugger by clicking on (this is same icon that "starts" the Debugger).
- Now you are in the editor. Go to file ***pwm_adc.c*** by clicking on its file tab and change the following C code statement from Ö.
  printfLCD("PWM to ADC DEMO"); to
  printfLCD("PWM - ADC DEMO");
- Click "Build All" to recompile and rebuild the program.
- Start the Debugger by clicking "Start/stop Debugger " to re-program this new code into Flash memory.
- Click "Run" and see that the LCD display now shows "PWM - ADC DEMO "
- Go back to the editor. Go to file ***pwm_adc.c*** and change text back to original printfLCD("PWM to ADC DEMO")
- Click "Build All" to recompile and rebuild the program.
- Load the program in device by clicking on "LOAD" to re-program this new code into Flash memory

- Click "Run" and see that LCD display now shows original text "PWM to ADC DEMO"
- Click "Reset".
- Go back to the file **pwm_adc.c** by clicking on its file tab.
- Set one breakpoint at line of code (printfLCD(msg_buff);) as shown in Figure 48 in **APPENDIX B.**

Note: the red dot indicates where the breakpoint is set and the yellow arrow indicates the next instruction to execute.

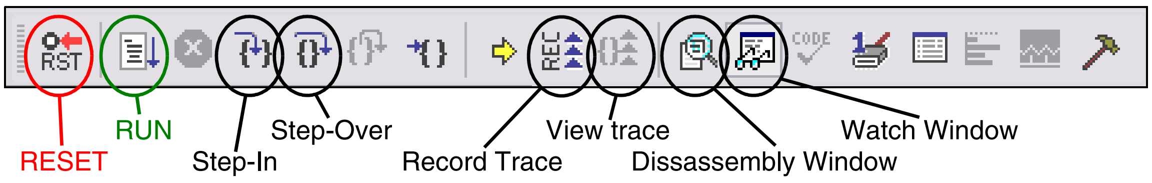
- Go back to the file **pwm_adc.c** and remove the breakpoint by clicking on the icon at the top so that the red dot is gone
- Click "Run " and notice that the LCD display shows continuous changes in values of PWM and ADC as the loop is executed multiple times . This is seen by changing vales of PWM and ADC in the LCD display.
- After about 10 seconds click. "Halt".

5.7.1 Instruction Tracing

The uPSD will rapidly stream a record of all the MCU instruction steps out to the ULINK adaptor. From this data, uVision will create a formatted file to help you find even the most stubborn bugs, showing a history instruction steps that the MCU has last executed.

- •To enable Trace, select from the debug bar (Figure 16) .

Figure 16. Debug Bar



A Trace Display file can display program source code in both C and Assembly formats.

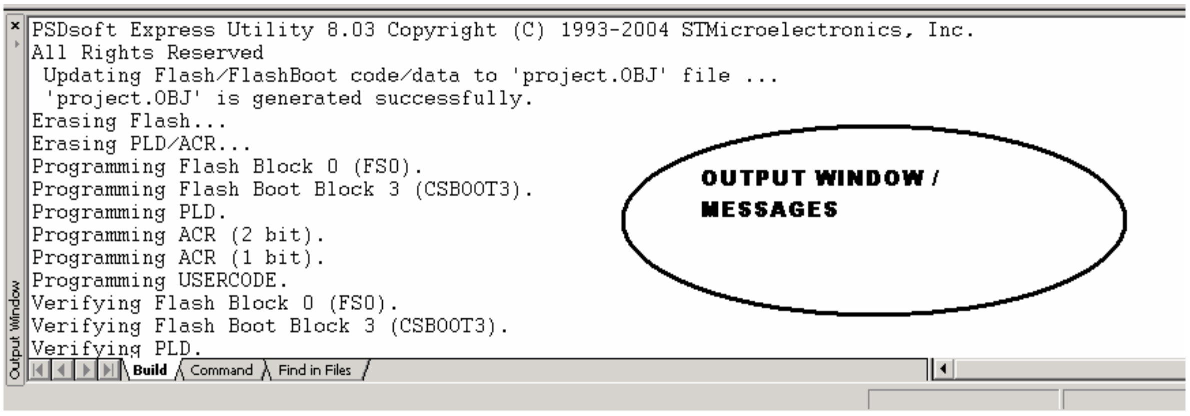
- Go back to the file **pwm_adc.c** by clicking on its file tab.
- Set one breakpoint at line of code (printfLCD(msg_buff);) as shown in Figure 48 in **APPENDIX B.** by moving the pointer on the left of the line of code and clicking on . A red dot is seen on left of line of code.
- Click "Run", and the MCU will run until hitting the breakpoint, then a window will open showing the Assembly source code, as shown in Figure 48 in **APPENDIX B.**

Note: The Program Counter in Register and in the Trace Dissassembly window

- Go back to the file **pwm_adc.c** and remove the breakpoint by clicking on the at the top.

- Click “Run” and notice that the LCD display shows continuous changes in values of PWM and ADC as the loop is executed multiple times while tracing is occurring in the background. This is seen by changing vales of PWM and ADC in the LCD display.
- After about 10 seconds click. ”Halt”.
- See the messages in the windows which are shown in the output window recording the actions taken (Figure 17).

Figure 17. Output Window (Messages)



```
PSDsoft Express Utility 8.03 Copyright (C) 1993-2004 STMicroelectronics, Inc.
All Rights Reserved
 Updating Flash/FlashBoot code/data to 'project.OBJ' file ...
 'project.OBJ' is generated successfully.
Erasing Flash...
Erasing PLD/ACR...
Programming Flash Block 0 (FS0).
Programming Flash Boot Block 3 (CSBOOT3).
Programming PLD.
Programming ACR (2 bit).
Programming ACR (1 bit).
Programming USERCODE.
Verifying Flash Block 0 (FS0).
Verifying Flash Boot Block 3 (CSBOOT3).
Verifying PLD.
```

Note: the output window provides action / error messages from the Integrated Development Environment.

Now that you have had a feel for the sample design example by running the demonstration application on DK3300-ELCD and manipulating some of the features of Keil compiler and debugger, the Design itself is examined by describing the details of implementing it using PSDsoft.

During the steps, the whole PSDsoft design flow process is fully detailed, explaining the features and its relevance to the design. Also, the key points are highlighted to ensure you consider and match with your own applications using DK3300, or your own designed board with the RIDE environment and PSDsoft tools.

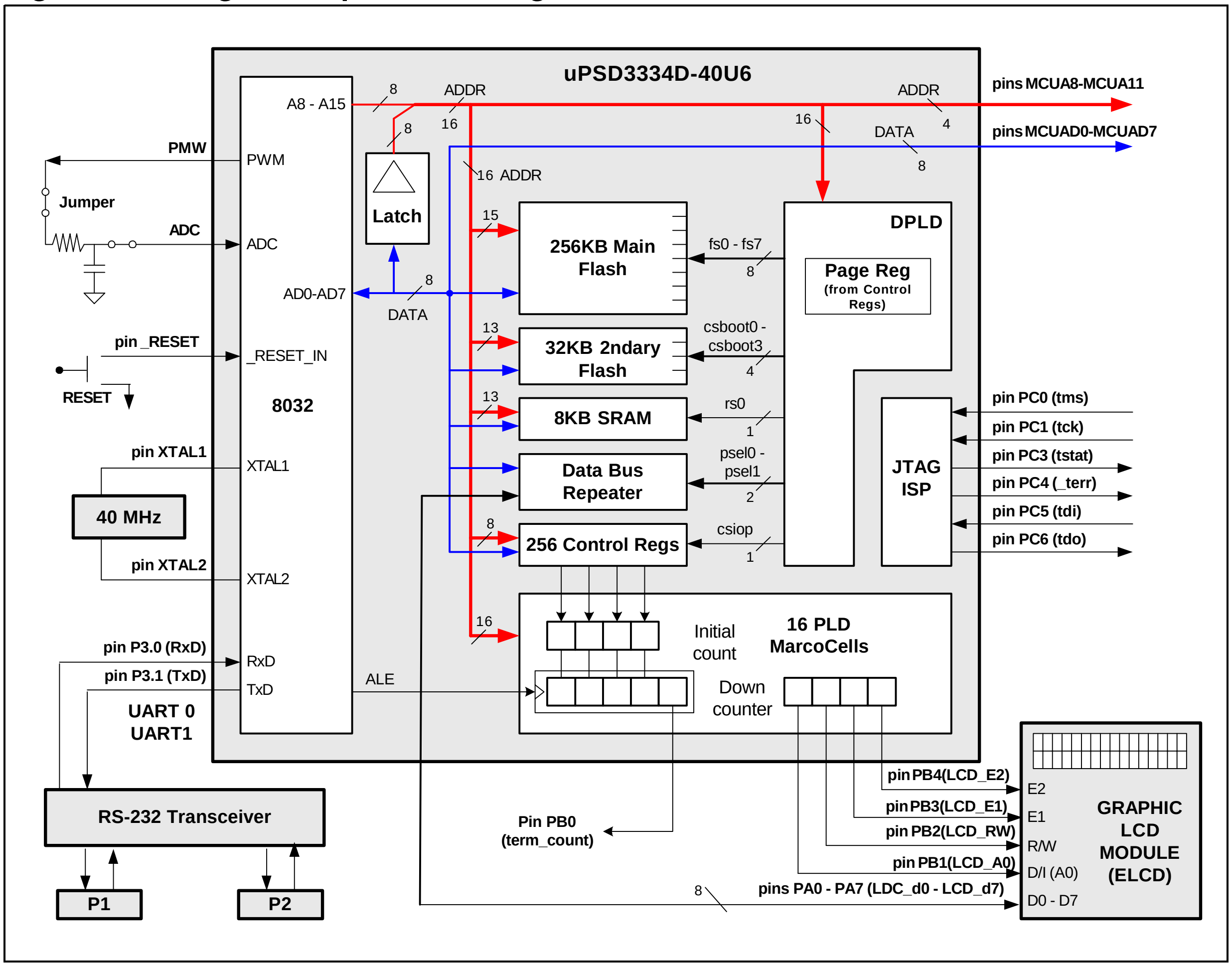
6 SAMPLE DESIGN EXAMPLE

6.1 Description

This simple design chosen for this example and demonstrated earlier uses the Code and the memory maps for the PWM-ADC application supplied as a demo with the DK3300-ELCD. As explained in the Design flow the key step is to first develop the application relationships and configurations. Here the example demonstration is represented by the block diagram in Figure 18 and the associated memory map is shown in Figure 19. The main Flash memory is paged, and few of the 8032 interfaces (e.g., ADC, PWM) are configured and used. The idea is to touch several aspects of the uPSD that may be unfamiliar to a typical 8032 user and to give you an idea of how to use the design tools as well as giving an overview of the Turbo uPSD3334 architecture. The design is based upon DK3300-ELCD Development Board. And the application code is developed and compiled using Keil.

Please refer to the general Users Guide for Keil, the DK3300-ELCD Quick Start, the DK3300-ELCD User Guide and the User Guide for PSDsoft as needed.

Figure 18. Design Example Block Diagram

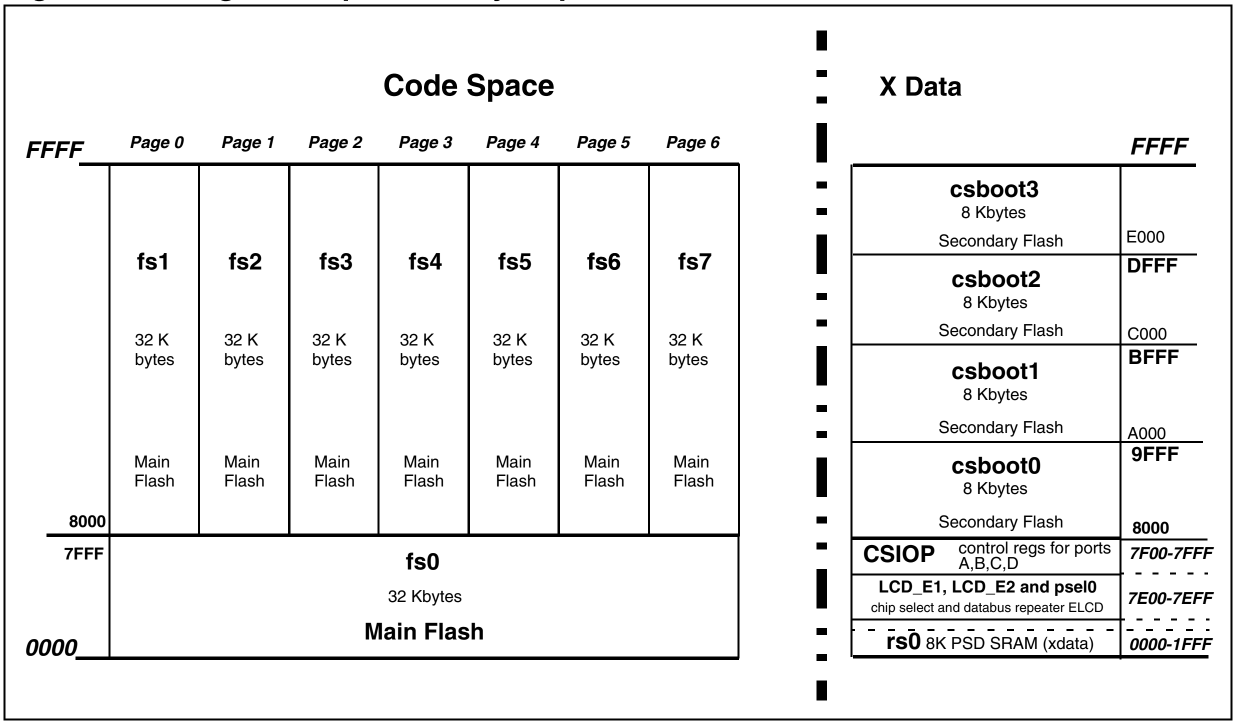


The 8032 outputs a repetitive PWM pulse train with a slowly varying pulse-width to an RC network which converts the pulse train into a slowly sweeping DC voltage (0V to 3.3V). This DC signal is looped back into an ADC input. The 8032 will write the resulting HEX ADC conversion value to the LCD so you can watch the results. The RC network and loop back is implemented with two jumper blocks (JP14 and JP15) on the DK3300-ELCD board.

Additionally and independently, a 4-bit, auto-reloading down-counter is created using PLD MicroCells. The 8032 directly loads the initial count value into four MicroCells, and that count is automatically loaded into another four MicroCells that create the 4-bit down-counter. Reloading occurs each time the counter reaches the terminal count of zero. Terminal count is indicated externally by a pulse on a Turbo uPSD output pin. The down-counter is clocked by an ALE signal (although in this example ALE was a random choice, it could be any signal). The 8032 may load a different initial count at anytime, creating a variable divider of the ALE signal.

The Graphic LCD module (ELCD) is connected to the Turbo uPSD3334 via Port A for data and Port B for some glue logic and chip-select signals. Port A operates in a special data bus repeater mode for this example, called Peripheral I/O mode. MCU8032 data will pass through Port A only for a given address range. The details for entering into PSDsoft and the equations used are described later into this document. (Refer to AN2028 Application Note for further details of the Graphic LCD Driver and the Hardware Interface with Turbo uPSD.).

Figure 19. Design Example Memory Map


Code Space
X Data
FFFF
Page 0
Page 1
Page 2
Page 3
Page 4
Page 5
Page 6
fs1 32 K bytes Main Flash
fs2 32 K bytes Main Flash
fs3 32 K bytes Main Flash
fs4 32 K bytes Main Flash
fs5 32 K bytes Main Flash
fs6 32 K bytes Main Flash
fs7 32 K bytes Main Flash
8000
7FFF
fs0 32 Kbytes Main Flash
0000
FFFF
csboot3 8 Kbytes Secondary Flash
E000
DFFF
csboot2 8 Kbytes Secondary Flash
C000
BFFF
csboot1 8 Kbytes Secondary Flash
A000
9FFF
csboot0 8 Kbytes Secondary Flash
8000
CSIOP control regs for ports A,B,C,D
7F00-7FFF
LCD_E1, LCD_E2 and psel0 chip select and databus repeater ELCD
7E00-7EFF
rs0 8K PSD SRAM (xdata)
0000-1FFF

The memory map in Figure 19 shows that in this design example, 32K byte secondary flash memory is used for Xdata space, and the 256K byte main flash memory is mapped into the Code space with fs1-fs7, banked over 7 pages. The nomenclature fsx, csbootx, rs0, csiop, and psel in Figure 19 refer to the individual internal Turbo uPSD memory segments. The Turbo uPSD main flash memory has a total of eight 32 Kbyte segments (fs0..fs7) (256Kbyte-Total).

The Turbo uPSD secondary Flash memory has a total of four 8 Kbyte segments (csboot0 - csboot3). All segments are being used in the example code supplied for this design. The Turbo uPSD 8 Kbyte SRAM has a single segment (rs0). A group of uPSD control registers which control I/O ports A, B, C, and D lie in a 256-byte xdata address space whose base address is named csiop. The Turbo uPSD has a data bus repeater feature that is enabled over a given address range as specified by psel. Figure 18 also shows external memory select signals (LCD_E1 and LCD_E2) required by the ELCD Module. This memory map is specified using the software tool PSDsoft Express. Each memory segment can be placed at virtually any address, which provides an infinite number of mapping schemes. In a later section all the equations used are listed and the mapping for the various signals as used by PSDsoft. The key PSDsoft screens are shown and explained for this example.

For simplicity in this particular application note, the 8032 will "boot" and run code contained completely within the 32 Kbyte main flash segment in Code space (fs0) and the 256 Kbyte main flash is treated as Code space paged into multiple pages. The font code required resides is in secondary flash segment csboot3 of the xdata mapped memory space. However, you can define alternate memory maps to meet the needs of your particular project. The uPSD memory space can be re-configured to use the secondary as Code space and the primary 256KB flash as Xdata space. (Refer to Turbo uPSD Datasheets and related documentation for further details).

A special register, called the VM Register within the csiop register block, is used to "reclassify" the main flash memory from code space to data space. The VM Register can be accessed by the 8032 at runtime to perform a variety of manipulations. PSDsoft is used to set the initial value of the VM Register upon power-up.

7 ENTERING DESIGN IN PSDSOFT EXPRESS

As shown earlier the key to a successful implementation is defining the desired memory map and then entering the design into PSDsoft. A demonstration design is used to navigate through the steps in the PSDsoft express tool and highlights the key items in this particular example. This way, it is possible to illustrate both the PSDsoft capabilities as well as provide a better understanding and use of PSDsoft's key features in the implementation of the design. Emphasis on setting up the various pin configurations and the associated equations for the PLD portions will be displayed. PSDsoft generates many reports that are very useful for review and analysis. In APPENDIX C. some of the key reports have been given for this project and can be easily correlated to the Design's block diagram and Memory map (Refer to Figures 18 & 19).

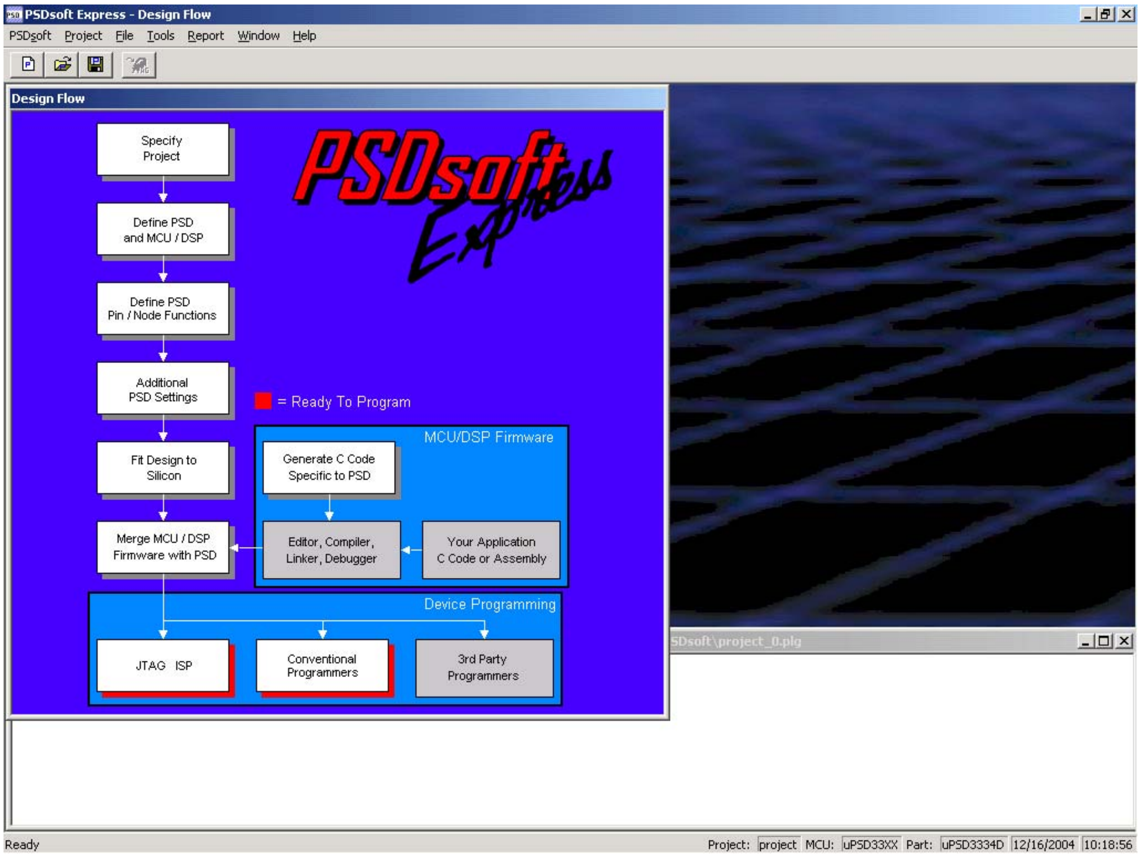
PSDsoft Express is a free tool available from STMicroelectronics, and you will need to download and install the latest version (v8.10 or later) from STMicroelectronics' website at www.st.com/psd, then look for "Software Downloads". In the following, the key inputs of the sample design are highlighted by showing the Design flow and then by integrating with application code (hex files) to generate the object file which will be programmed using the PSDsoft JTAG.

PSDsoft Design Express is shown and explained below:

Start PSDsoft Express. - You will get the main design flow screen as shown in Figure 20.

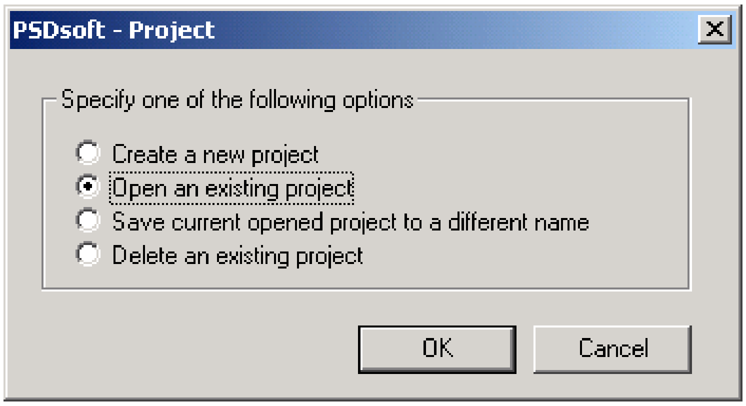
This is your starting point. Refer to PSDsoft Users guides for more details

Figure 20. PSDsoft Express Design Flow


PSDsoft Express - Design Flow
PSDsoft Project File Tools Report Window Help
Design Flow
Specify Project
Define PSD and MCU / DSP
Define PSD Pin / Node Functions
Additional PSD Settings
Fit Design to Silicon
Merge MCU / DSP Firmware with PSD
PSDsoft Express
🟥 = Ready To Program
MCU/DSP Firmware
Generate C Code Specific to PSD
Editor, Compiler, Linker, Debugger
Your Application C Code or Assembly
Device Programming
JTAG ISP
Conventional Programmers
3rd Party Programmers
SDsoft\project_0.plg
Ready
Project: project MCU: uPSD33XX Part: uPSD3334D 12/16/2004 10:18:56

Click on "Specify Project". You will get the screen shown in Figure 21.

Figure 21. PSDsoft Specify Project Screen

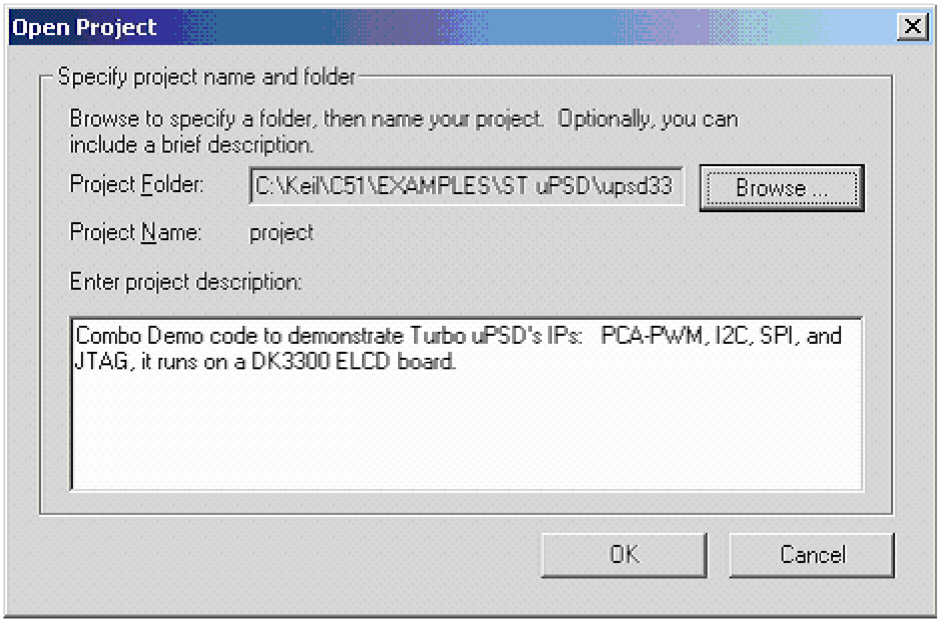

PSDsoft - Project

Specify one of the following options

- ( ) Create a new project
- (●) Open an existing project
- ( ) Save current opened project to a different name
- ( ) Delete an existing project

Select "Open an existing project" (as the demonstration example is being used) and a number of entries will be shown. Next, select the correct PSDsoft project folder and you will be shown this screen. Note the filename used by PSDsoft for this example is project.ini

Figure 22. PSDsoft OPen Project Screen



Specify project name and folder

Browse to specify a folder, then name your project. Optionally, you can include a brief description.

Project Folder: C:\Keil\C51\EXAMPLES\ST uPSD\upsd33

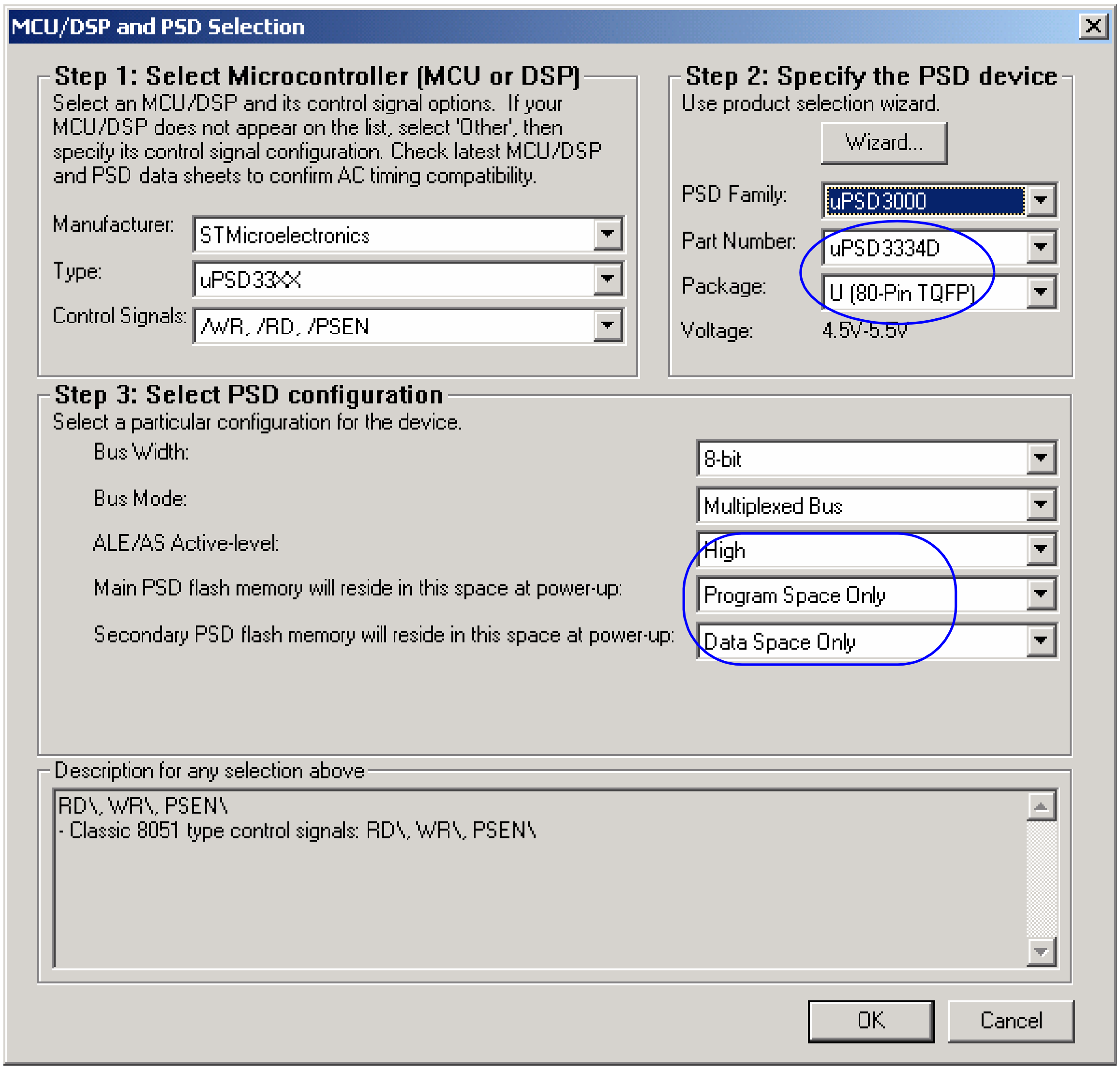
Project Name: project

Enter project description:

Combo Demo code to demonstrate Turbo uPSD's IPs: PCA-PWM, I2C, SPI, and JTAG, it runs on a DK3300 ELCD board.

This verifies that you have made the right selection. Next go to the second step in the design flow "Define PSD and MCU/DSP". This step defines the MCU selected. As Keil is used, all the necessary parameters are identified in the project.ini file. This step is performed to verify that the entry is correct. You should see the following screen:

Figure 23. Select MCU and Initial Placement of Flash in Code Space or Data Space



Step 1: Select Microcontroller (MCU or DSP)

Select an MCU/DSP and its control signal options. If your MCU/DSP does not appear on the list, select 'Other', then specify its control signal configuration. Check latest MCU/DSP and PSD data sheets to confirm AC timing compatibility.

Manufacturer: STMicroelectronics

Type: uPSD33XX

Control Signals: /WR, /RD, /PSEN

Step 2: Specify the PSD device

Use product selection wizard.

PSD Family: uPSD3000

Part Number: uPSD3334D

Package: U (80-Pin TQFP)

Voltage: 4.5V-5.5V

Step 3: Select PSD configuration

Select a particular configuration for the device.

Bus Width: 8-bit

Bus Mode: Multiplexed Bus

ALE/AS Active-level: High

Main PSD flash memory will reside in this space at power-up: Program Space Only

Secondary PSD flash memory will reside in this space at power-up: Data Space Only

Description for any selection above

RD\, WR\, PSEN\
- Classic 8051 type control signals: RD\, WR\, PSEN\

Notice that all items compare with the design block diagram. The key items are circled.

Here you can set the parameters if your design and memory map are different. The key items (fields) for this screen are:

1. Select the MCU. In this case it is STMicroelectronics, then uPSD33xx, then uPSD3334D.
2. Select the main flash memory to reside in 8032 program space at power-up (this means that the 8032 _PSEN signal is routed to the secondary flash array).
3. Select the secondary flash to reside in 8032 code space at power-up (this means that the 8032 _RD and _WR signals are routed to secondary flash array). Make sure that screen looks like as shown above.
4. Click OK. Now and then go to the next stepin the Design Flow "Define PSD/Pin/Node Functions". This is described in detail in the next section.

7.1 Pin Definitions

You will be shown the Pin Definitions screen. The entry is normally a 3-step process:

1. Select the pin based upon the selected device output.
2. Add or update the pin function and features, name as shown the second portion.
3. Complete all pin definitions and then continue to the next step.

Since the example design is completed, this will show all the pin definitions as shown in the block diagram (Figure 18).

Click through the pins and see how they are configured and how they relate to the design.

You will notice that upon clicking on the radial button the associated pin name is shown in the Step-2 portion and identifies how the pin function is configured (whether as CPLD input/ output or as other mode.)

For the sample design, the key signal names will be shown as related to interfacing with ELCD screen and their relationship to Design block diagram. (Refer to the Block diagram and the LCD pin connections: Figure 18 and Figure 19. See that they are similar for the design)

You will also notice that you cannot change the definition of some pins because they have a fixed function. PSDsoft already take cares of this.

In this design the pin configurations for Port A and Port B[1..4] are shown below.

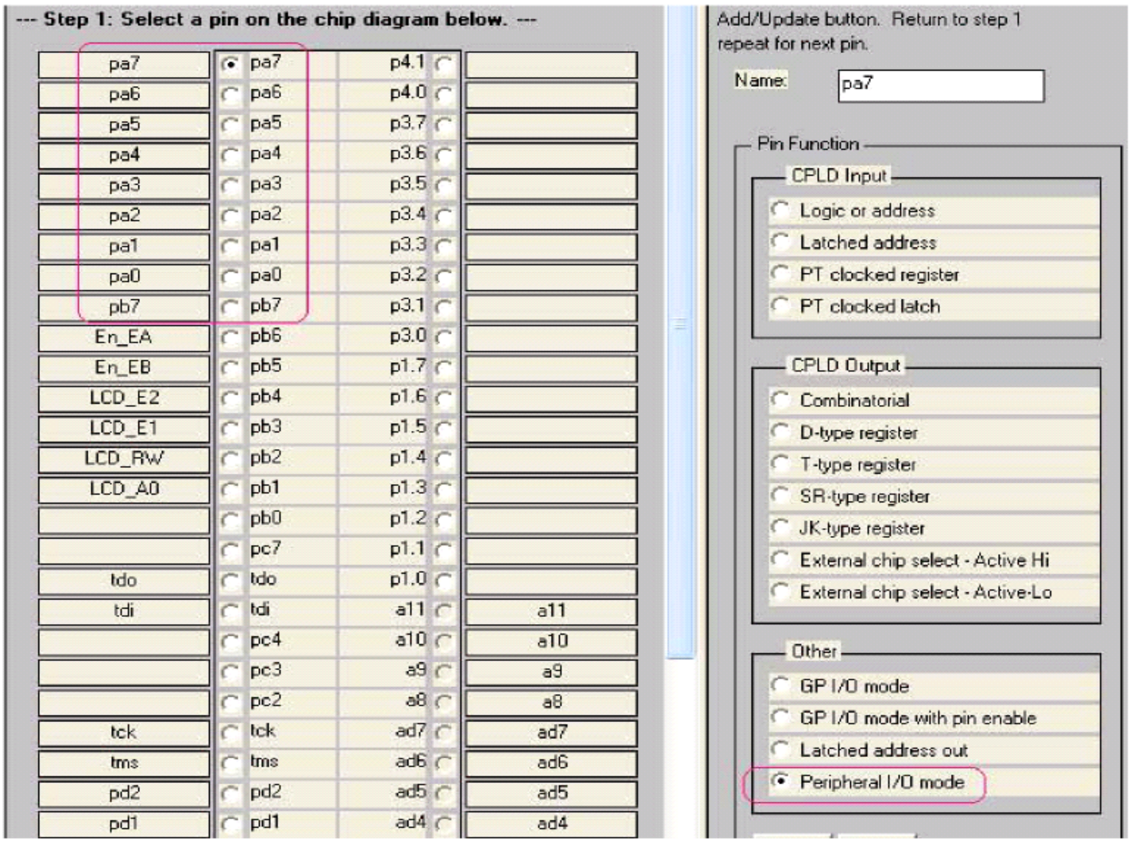
Port A should be in the Peripheral mode; (Figure 24)

Port B, pins 1 and 2 should be in the Combinational Logic Output mode (Figure 25)

and Port B, pins 3 and 4 should be in the External Chip-Select, Active-High mode (Figure 26).

Here is a sample of each:

Figure 24. Pin definitions for Port A



--- Step 1: Select a pin on the chip diagram below. ---

pa7	[✓ pa7]	p4.1	
pa6	pa6	p4.0	
pa5	pa5	p3.7	
pa4	pa4	p3.6	
pa3	pa3	p3.5	
pa2	pa2	p3.4	
pa1	pa1	p3.3	
pa0	pa0	p3.2	
pb7	pb7	p3.1	
En_EA	pb6	p3.0	
En_EB	pb5	p1.7	
LCD_E2	pb4	p1.6	
LCD_E1	pb3	p1.5	
LCD_RW	pb2	p1.4	
LCD_A0	pb1	p1.3	
	pb0	p1.2	
	pc7	p1.1	
tdo	tdo	p1.0	
tdi	tdi	a11	a11
	pc4	a10	a10
	pc3	a9	a9
	pc2	a8	a8
tck	tck	ad7	ad7
tms	tms	ad6	ad6
pd2	pd2	ad5	ad5
pd1	pd1	ad4	ad4

Add/Update button. Return to step 1 repeat for next pin.

Name: pa7

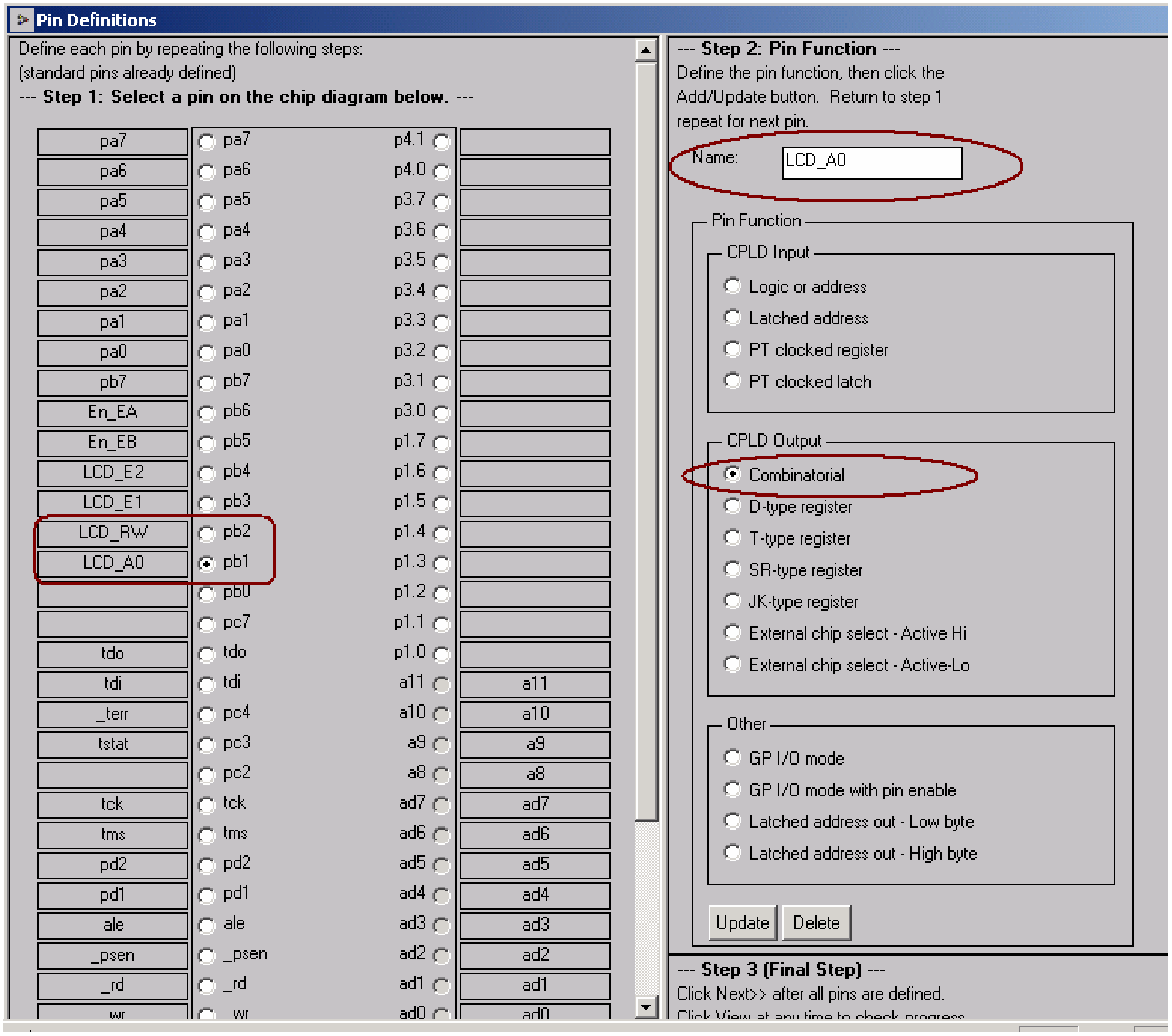
Pin Function

CPLD Input
- Logic or address
- Latched address
- PT clocked register
- PT clocked latch

CPLD Output
- Combinatorial
- D-type register
- T-type register
- SR-type register
- JK-type register
- External chip select - Active Hi
- External chip select - Active-Lo

Other
- GP I/O mode
- GP I/O mode with pin enable
- Latched address out
- [✓ Peripheral I/O mode]

Figure 25. Pin definitions for Port B (Pins 1-2)



Pin Definitions

Define each pin by repeating the following steps:
(standard pins already defined)
--- **Step 1: Select a pin on the chip diagram below.** ---

pa7	pa7	p4.1	
pa6	pa6	p4.0	
pa5	pa5	p3.7	
pa4	pa4	p3.6	
pa3	pa3	p3.5	
pa2	pa2	p3.4	
pa1	pa1	p3.3	
pa0	pa0	p3.2	
pb7	pb7	p3.1	
En_EA	pb6	p3.0	
En_EB	pb5	p1.7	
LCD_E2	pb4	p1.6	
LCD_E1	pb3	p1.5	
LCD_RW	pb2	p1.4	
LCD_A0	[✓ pb1]	p1.3	
	pb0	p1.2	
	pc7	p1.1	
tdo	tdo	p1.0	
tdi	tdi	a11	a11
_terr	pc4	a10	a10
tstat	pc3	a9	a9
	pc2	a8	a8
tck	tck	ad7	ad7
tms	tms	ad6	ad6
pd2	pd2	ad5	ad5
pd1	pd1	ad4	ad4
ale	ale	ad3	ad3
_psen	_psen	ad2	ad2
_rd	_rd	ad1	ad1
wr	wr	ad0	ad0

--- **Step 2: Pin Function** ---
Define the pin function, then click the Add/Update button. Return to step 1 repeat for next pin.

Name: LCD_A0

Pin Function

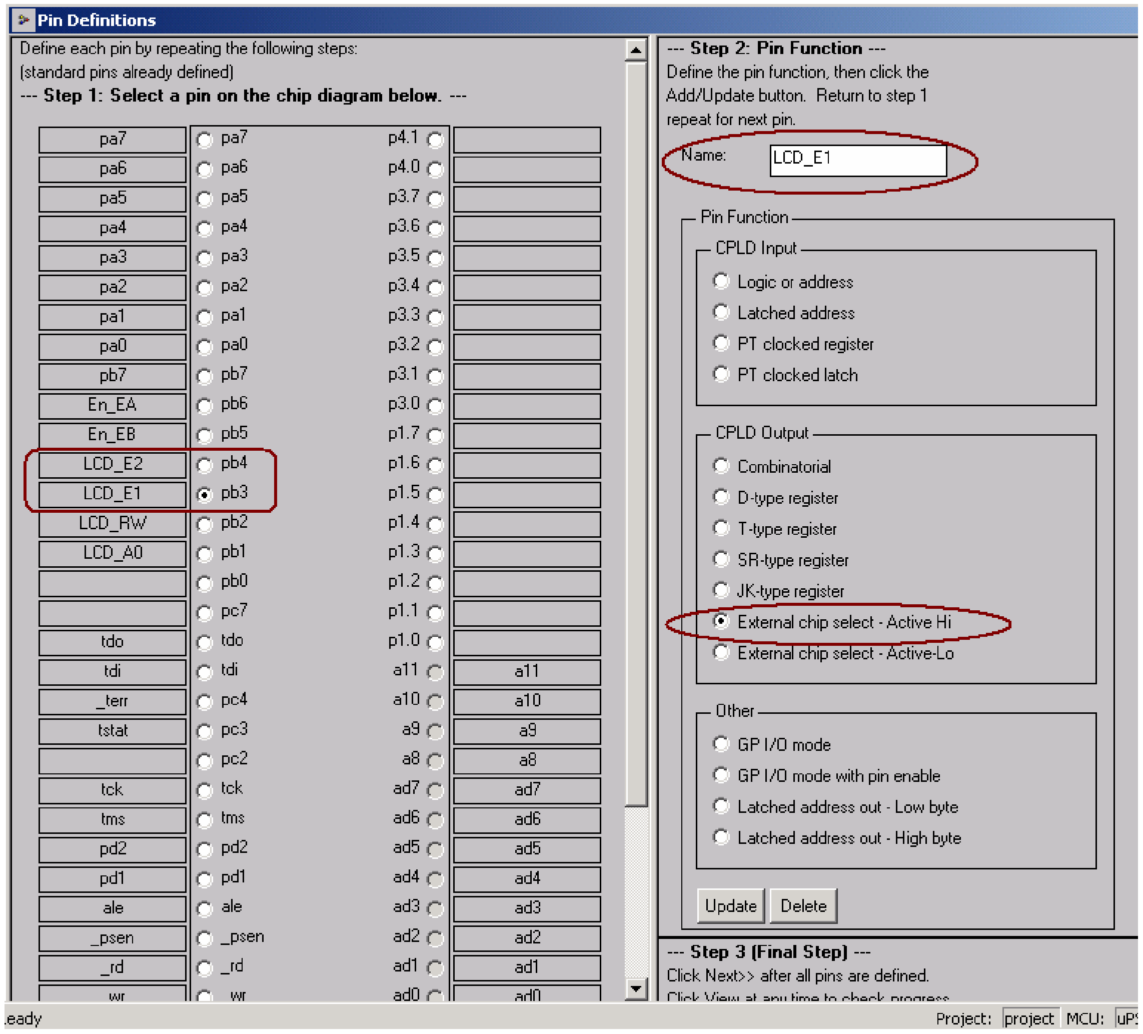
CPLD Input
- Logic or address
- Latched address
- PT clocked register
- PT clocked latch

CPLD Output
- [✓ Combinatorial]
- D-type register
- T-type register
- SR-type register
- JK-type register
- External chip select - Active Hi
- External chip select - Active-Lo

Other
- GP I/O mode
- GP I/O mode with pin enable
- Latched address out - Low byte
- Latched address out - High byte

--- **Step 3 (Final Step)** ---
Click Next>> after all pins are defined.

Figure 26. Pin definitions for Port B (Pins 3-4)



Pin Definitions

Define each pin by repeating the following steps:
(standard pins already defined)
--- **Step 1: Select a pin on the chip diagram below.** ---

pa7	pa7	p4.1	
pa6	pa6	p4.0	
pa5	pa5	p3.7	
pa4	pa4	p3.6	
pa3	pa3	p3.5	
pa2	pa2	p3.4	
pa1	pa1	p3.3	
pa0	pa0	p3.2	
pb7	pb7	p3.1	
En_EA	pb6	p3.0	
En_EB	pb5	p1.7	
LCD_E2	pb4	p1.6	
LCD_E1	[✓ pb3]	p1.5	
LCD_RW	pb2	p1.4	
LCD_A0	pb1	p1.3	
	pb0	p1.2	
	pc7	p1.1	
tdo	tdo	p1.0	
tdi	tdi	a11	a11
_terr	pc4	a10	a10
tstat	pc3	a9	a9
	pc2	a8	a8
tck	tck	ad7	ad7
tms	tms	ad6	ad6
pd2	pd2	ad5	ad5
pd1	pd1	ad4	ad4
ale	ale	ad3	ad3
_psen	_psen	ad2	ad2
_rd	_rd	ad1	ad1
wr	wr	ad0	ad0

--- **Step 2: Pin Function** ---
Define the pin function, then click the Add/Update button. Return to step 1 repeat for next pin.

Name: LCD_E1

Pin Function

CPLD Input
- Logic or address
- Latched address
- PT clocked register
- PT clocked latch

CPLD Output
- Combinatorial
- D-type register
- T-type register
- SR-type register
- JK-type register
- [✓ External chip select - Active Hi]
- External chip select - Active-Lo

Other
- GP I/O mode
- GP I/O mode with pin enable
- Latched address out - Low byte
- Latched address out - High byte

Update Delete

--- **Step 3 (Final Step)** ---
Click Next>> after all pins are defined.

eady Project: project MCU: uP

Walk through some of the pins to verify each pin definitions and get an overview of the design implementation.

Click "Next" in step-3 (final step) to move on to the Design Assistant for memory mapping and logic equations. You will see the Design Assistant Screen with the following tabs:

- Page Register definition
- Chip Select Equations
- I/O Logic Equations
- User-defined Node Equations

This is a key part of the design process and requires careful entry for the pin definitions and associated memory maps and logic equations. Since this determines how PSDsoft maps the memory Address space and makes the PLD connections it is imperative that the memory map matches the chip selects for individual memory elements of the Turbo uPSD (memory external to the 8032 core). Definition of the use of the Turbo uPSD Page Register is also required.

Four memory blocks (Main flash, Secondary flash, SRAM, and Control Registers) external to the 8032 core are available and are individually selected segment-by-segment when 8032 addresses are presented to the Decode PLD (DPLD). Each of these memory segments has its own chip-select name (fs0, fs1.. csboot1, rs0, csiop, and so forth). Equations for these chip-selects, and for any external chip-selects, must be specified using PSDsoft Express. For this example, chip-selects are defined to match the memory map.

7.2 Page Register

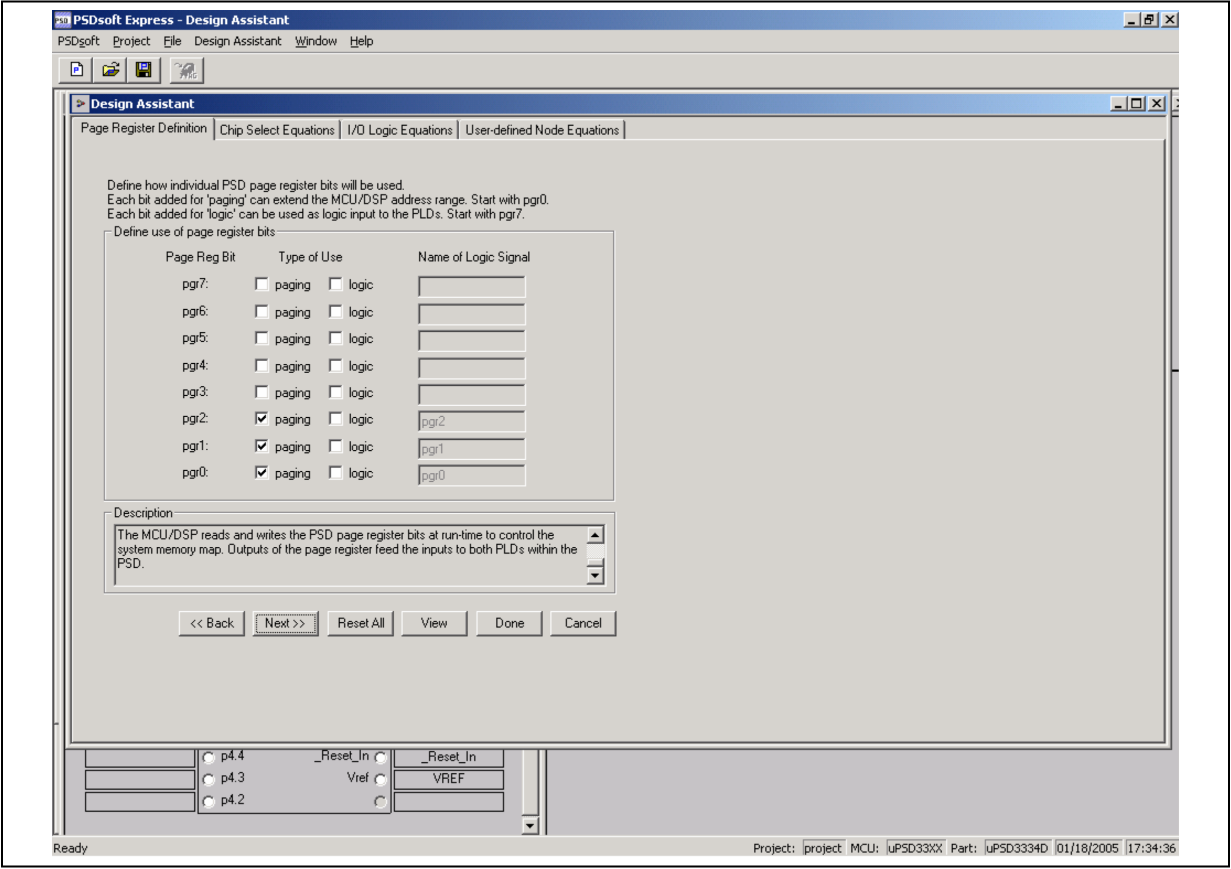
Paging bits may be used for other types of memory manipulation (such as memory swapping), but that will be discussed in other application notes.

Select this bit for memory paging. Use one bit to define two memory pages, use two bits to define four pages, three bits for eight pages and so on. Select enough bits to cover the number required pages. Always start with pgr0 and add more bits going upward, as these bits are an extension of the MCU's or DSP's natural address bits.

The MCU/DSP reads and writes the PSD page register bits at run-time to control the system memory map. Outputs of the page register feed the inputs to both PLDs within the PSD.

Since eight memory pages (or banks) are needed as shown in the memory map diagram of the design (Figure 19), three paging bits ($2^3 = 8$) are specified in the Screen below (Figure 27)

Figure 27. Page Register Definitions



Define how individual PSD page register bits will be used.
Each bit added for 'paging' can extend the MCU/DSP address range. Start with pgr0.
Each bit added for 'logic' can be used as logic input to the PLDs. Start with pgr7.

Define use of page register bits

Page Reg Bit	Type of Use		Name of Logic Signal
pgr7:	☐ paging	☐ logic	
pgr6:	☐ paging	☐ logic	
pgr5:	☐ paging	☐ logic	
pgr4:	☐ paging	☐ logic	
pgr3:	☐ paging	☐ logic	
pgr2:	✓ paging	☐ logic	pgr2
pgr1:	✓ paging	☐ logic	pgr1
pgr0:	✓ paging	☐ logic	pgr0

Description

The MCU/DSP reads and writes the PSD page register bits at run-time to control the system memory map. Outputs of the page register feed the inputs to both PLDs within the PSD.

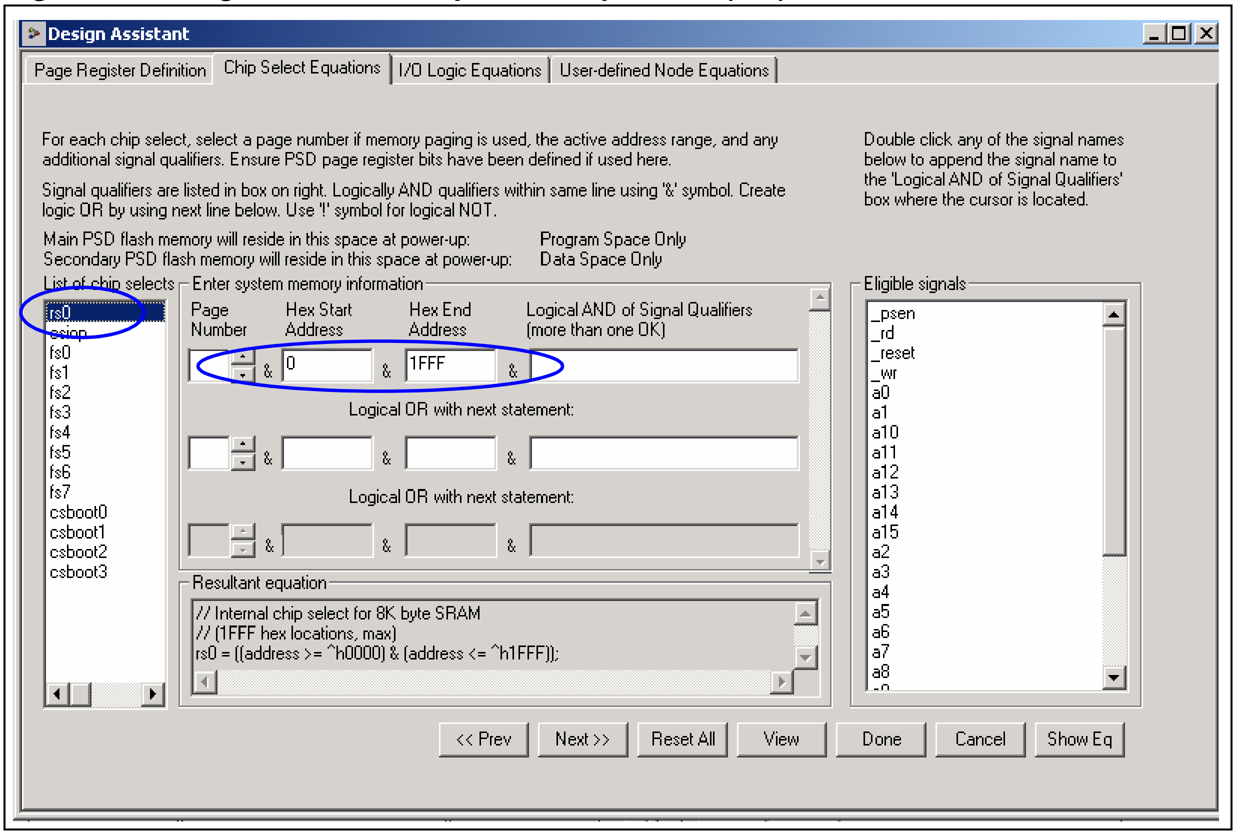
Ready   Project: project MCU: uPSD33XX Part: uPSD3334D 01/18/2005 17:34:36

Click "Next" to move on to the "Chip Select Equations"Definition Screen.

7.3 Chip-Select Equations

Now you will see the Chip-Select definition screen (Figure 28). Some of the signals will be shown in this application note for understanding the different types. You can go through each one to get the pin definitions. The key signals based upon the Block diagram (Figure 18) are shown in the following screens:.

Figure 28. Design Assistant Chip select equations (rs0)



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

Main PSD flash memory will reside in this space at power-up:	Program Space Only
Secondary PSD flash memory will reside in this space at power-up:	Data Space Only

Enter system memory information

Page Number		Hex Start Address		Hex End Address		Logical AND of Signal Qualifiers (more than one OK)
	&	0	&	1FFF	&	

Logical OR with next statement:

Logical OR with next statement:

Resultant equation

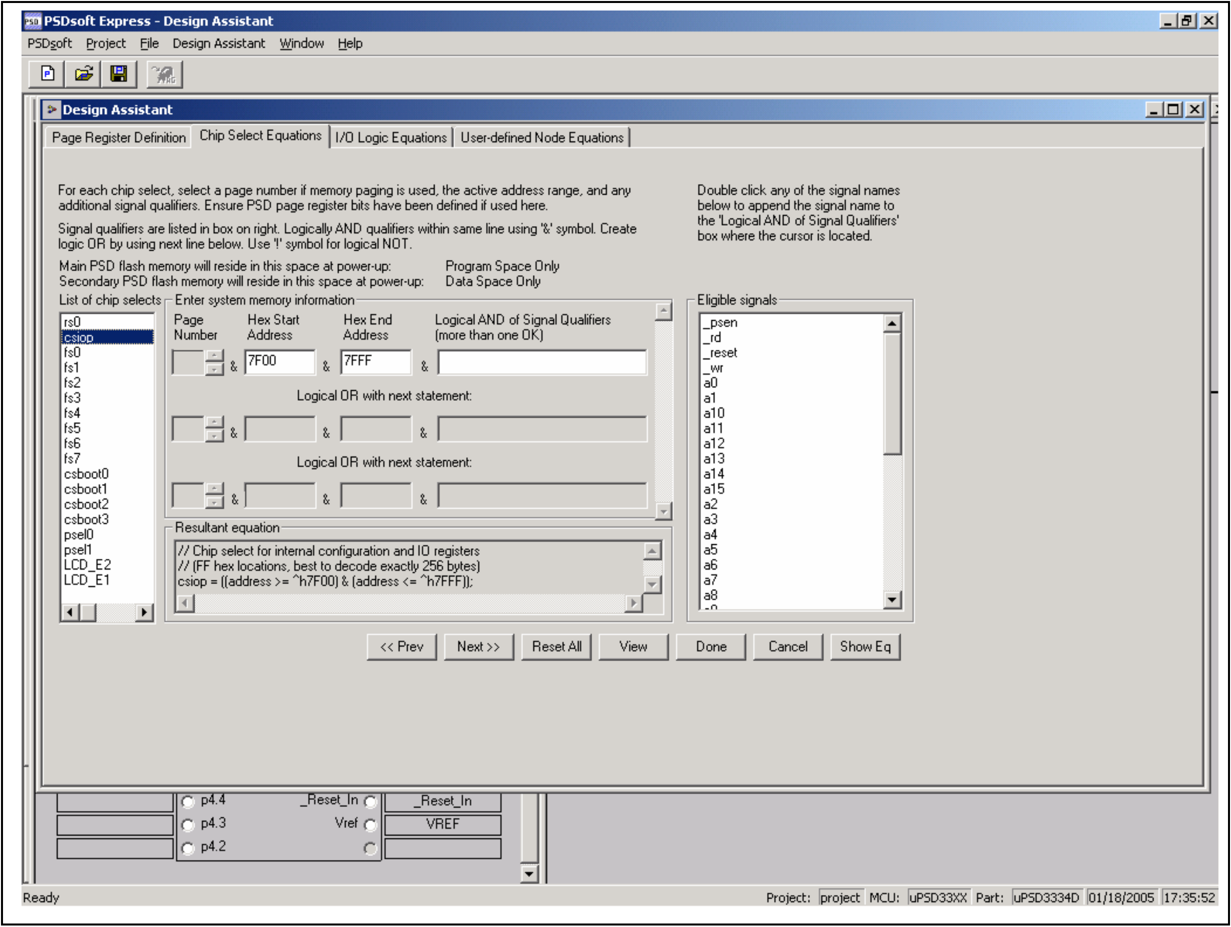
```
// Internal chip select for 8K byte SRAM
// (1FFF hex locations, max)
rs0 = ((address >= ^h0000) & (address <= ^h1FFF));
```

1. Click the chip-select signal rs0 for the 8 Kbyte xdata SRAM (see Figure 28)
2. Make sure that its definition matches the memory map in Figure 19.

   Note: No page number is specified for rs0 since the SRAM is common to all pages (page independent). Additional signal qualifiers (8032 control signals _rd, _wr, _psen, and ale) are NOT needed for internal uPSD chip-selects as this is taken care of in silicon. The SRAM always defaults to 8032 data space. At any time, you may click the "View" button to see how you are doing, and a summary will appear.
3. Click on the chip-select csiop (Chip Select I/O Port). This is a band of 256 xdata registers used to control Turbo uPSD Ports A, B, C, D, the Page Register, power management, and other functions.

40 of the 256 registers are used (see the complete Turbo uPSD datasheet for register definitions and their address offset from the csiop base address). There is no need to specify additional signal qualifiers for csiop, and it is not allowed to place csiop on a particular memory page.

Figure 29. Design Assistant Chip select equations (csiop)



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Main PSD flash memory will reside in this space at power-up: Program Space Only
Secondary PSD flash memory will reside in this space at power-up: Data Space Only

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

List of chip selects: rs0, csiop, fs0, fs1, fs2, fs3, fs4, fs5, fs6, fs7, csboot0, csboot1, csboot2, csboot3, psel0, psel1, LCD_E2, LCD_E1

Enter system memory information

Page Number	Hex Start Address	Hex End Address	Logical AND of Signal Qualifiers (more than one OK)
	7F00	7FFF	

Logical OR with next statement:

Logical OR with next statement:

Resultant equation

```
// Chip select for internal configuration and IO registers
// (FF hex locations, best to decode exactly 256 bytes)
csiop = ((address >= ^h7F00) & (address <= ^h7FFF));
```

Eligible signals: _psen, _rd, _reset, _wr, a0, a1, a10, a11, a12, a13, a14, a15, a2, a3, a4, a5, a6, a7, a8

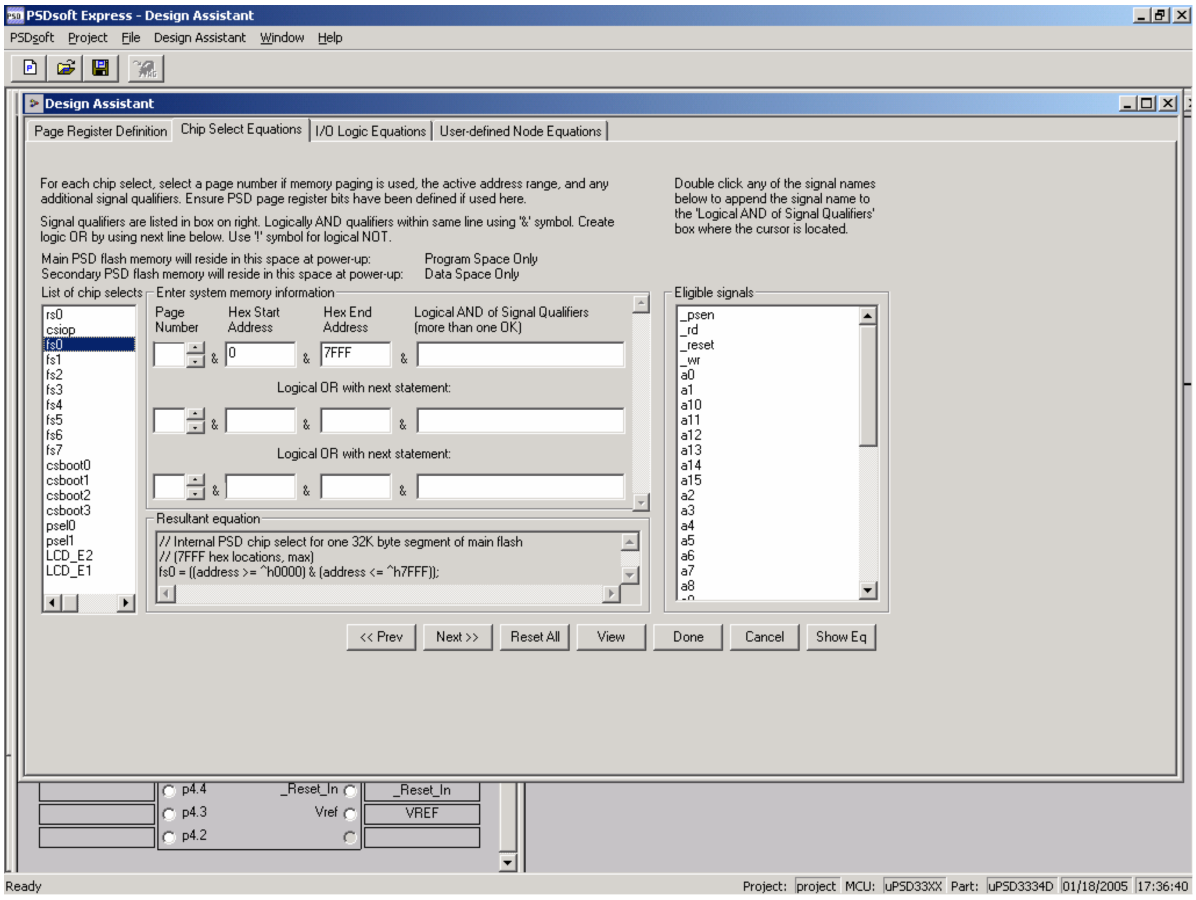
	p4.4	_Reset_In	_Reset_In
	p4.3	Vref	VREF
	p4.2		

Project: project MCU: uPSD33XX Part: uPSD3334D 01/18/2005 17:35:52

4. Click on fs0, fs1…fs7, which are chip-selects for the eight 32 Kbyte segments of Turbo uPSD Main flash. Figure 30 below shows fs0.

   Note: the address range of fs0 matches with the memory match, and the address range is 0000 - 7FFF as shown in memory map of Figure 19.

Figure 30. Design Assistant Chip select equations (fs0)



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

Main PSD flash memory will reside in this space at power-up: Program Space Only
Secondary PSD flash memory will reside in this space at power-up: Data Space Only

List of chip selects: rs0, csiop, fs0, fs1, fs2, fs3, fs4, fs5, fs6, fs7, csboot0, csboot1, csboot2, csboot3, psel0, psel1, LCD_E2, LCD_E1

Enter system memory information

Page Number	Hex Start Address	Hex End Address	Logical AND of Signal Qualifiers (more than one OK)
	0	7FFF	

Logical OR with next statement:

Logical OR with next statement:

Eligible signals: _psen, _rd, _reset, _wr, a0, a1, a10, a11, a12, a13, a14, a15, a2, a3, a4, a5, a6, a7, a8

Resultant equation

```
// Internal PSD chip select for one 32K byte segment of main flash
// (7FFF hex locations, max)
fs0 = ((address >= ^h0000) & (address <= ^h7FFF));
```

p4.4 _Reset_In _Reset_In
p4.3 Vref VREF
p4.2

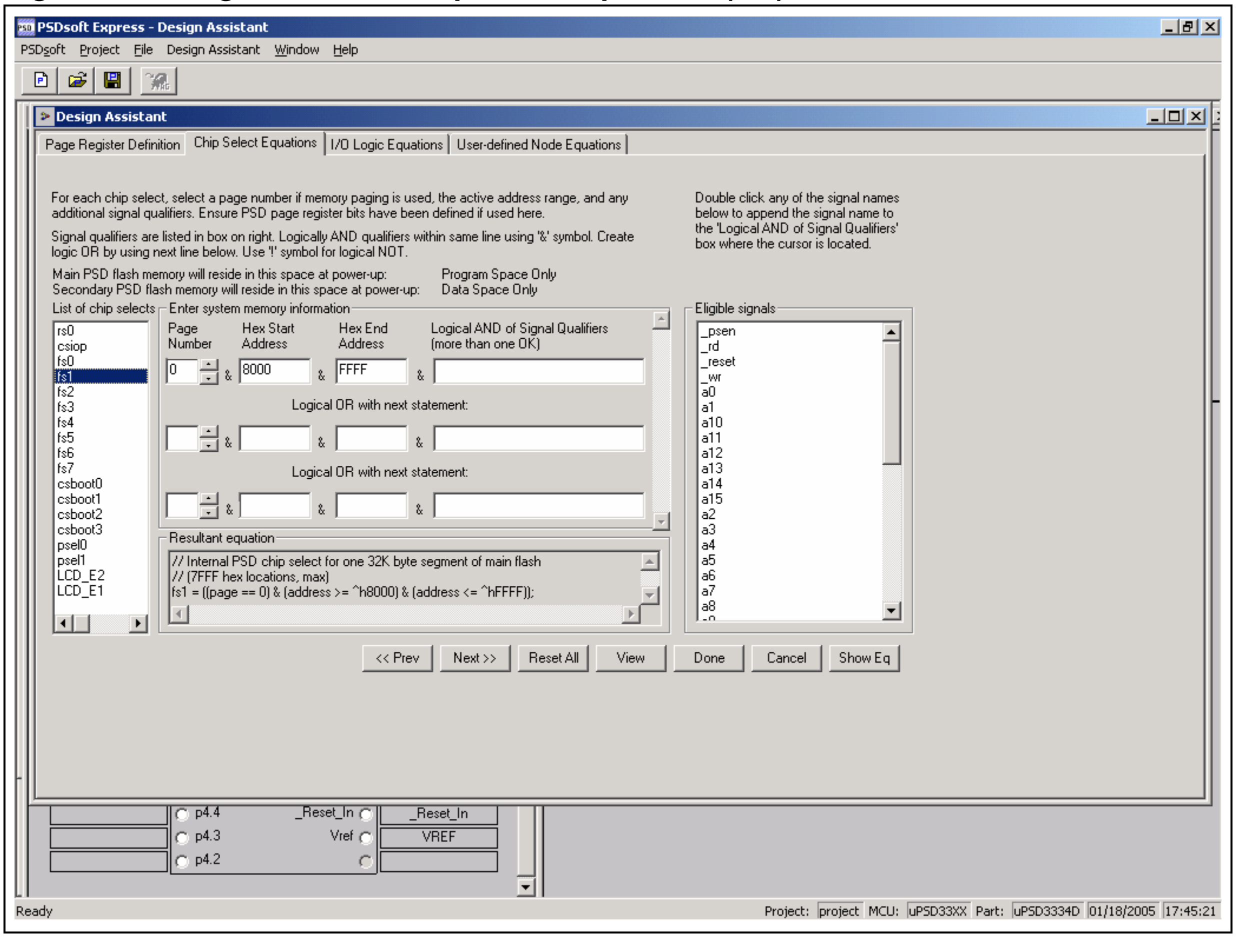
Ready   Project: project MCU: uPSD33XX Part: uPSD3334D 01/18/2005 17:36:40

5. Click on fs1. fs2 .. fs7, which are chip-selects for the other 7 32 Kbyte segments of Turbo uPSD Main flash.

   Notes: The page number is 0 for fs1, and the address range is 8000 - FFFF as shown in the memory map (Figure 19)

   No additional qualifiers are needed for the page number assignments. In Figures 31,32 below, fs1 and fs7 are shown.

Figure 31. Design Assistant Chip select equations (fs1)



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

Main PSD flash memory will reside in this space at power-up: Program Space Only
Secondary PSD flash memory will reside in this space at power-up: Data Space Only

Enter system memory information

Page Number		Hex Start Address		Hex End Address		Logical AND of Signal Qualifiers (more than one OK)
0	&	8000	&	FFFF	&	

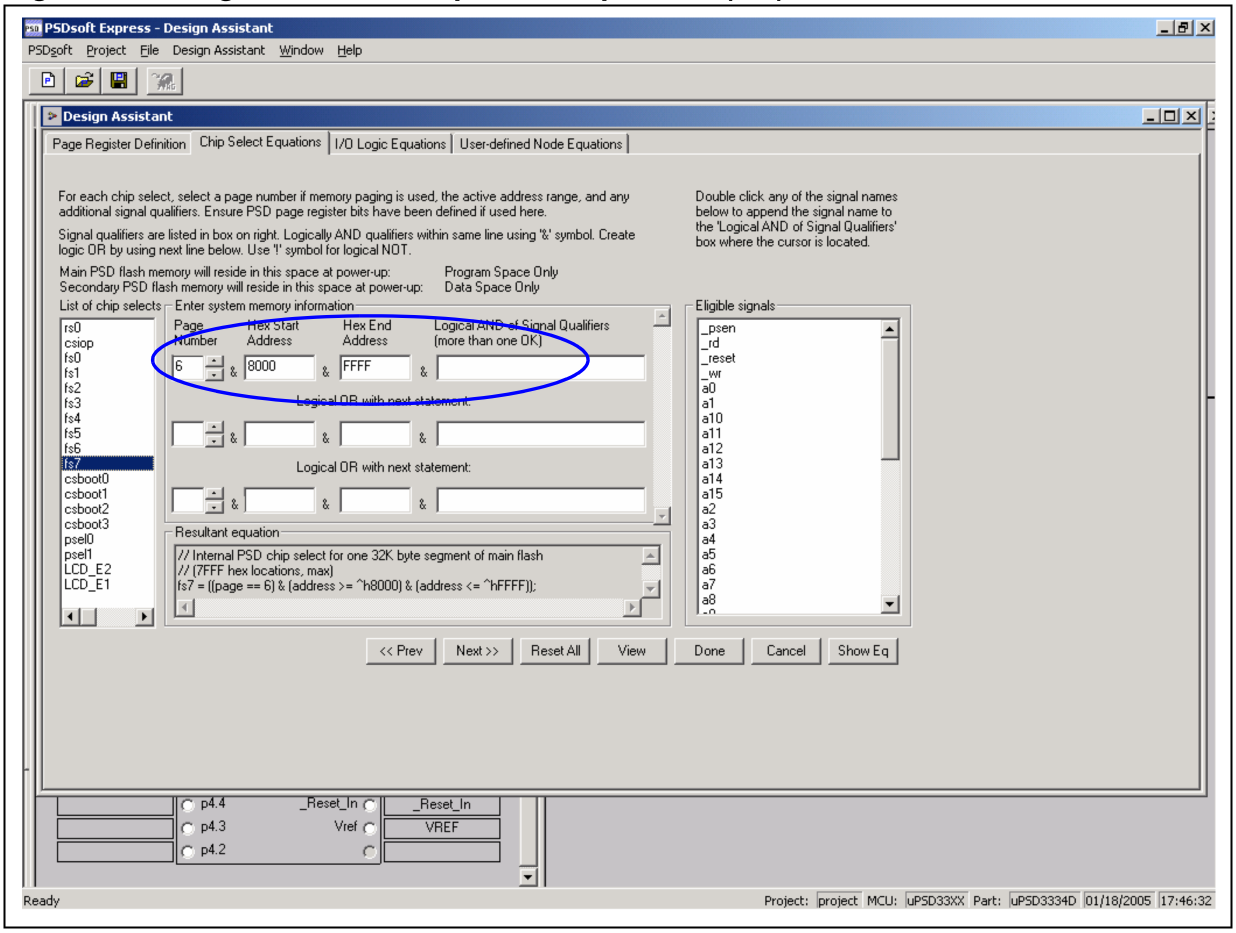
Logical OR with next statement:

Logical OR with next statement:

Resultant equation

```
// Internal PSD chip select for one 32K byte segment of main flash
// (7FFF hex locations, max)
fs1 = ((page == 0) & (address >= ^h8000) & (address <= ^hFFFF));
```

Figure 32. Design Assistant Chip select equations (fs7)



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

Main PSD flash memory will reside in this space at power-up: Program Space Only
Secondary PSD flash memory will reside in this space at power-up: Data Space Only

Page Number	Hex Start Address	Hex End Address	Logical AND of Signal Qualifiers (more than one OK)
6	8000	FFFF	

Resultant equation

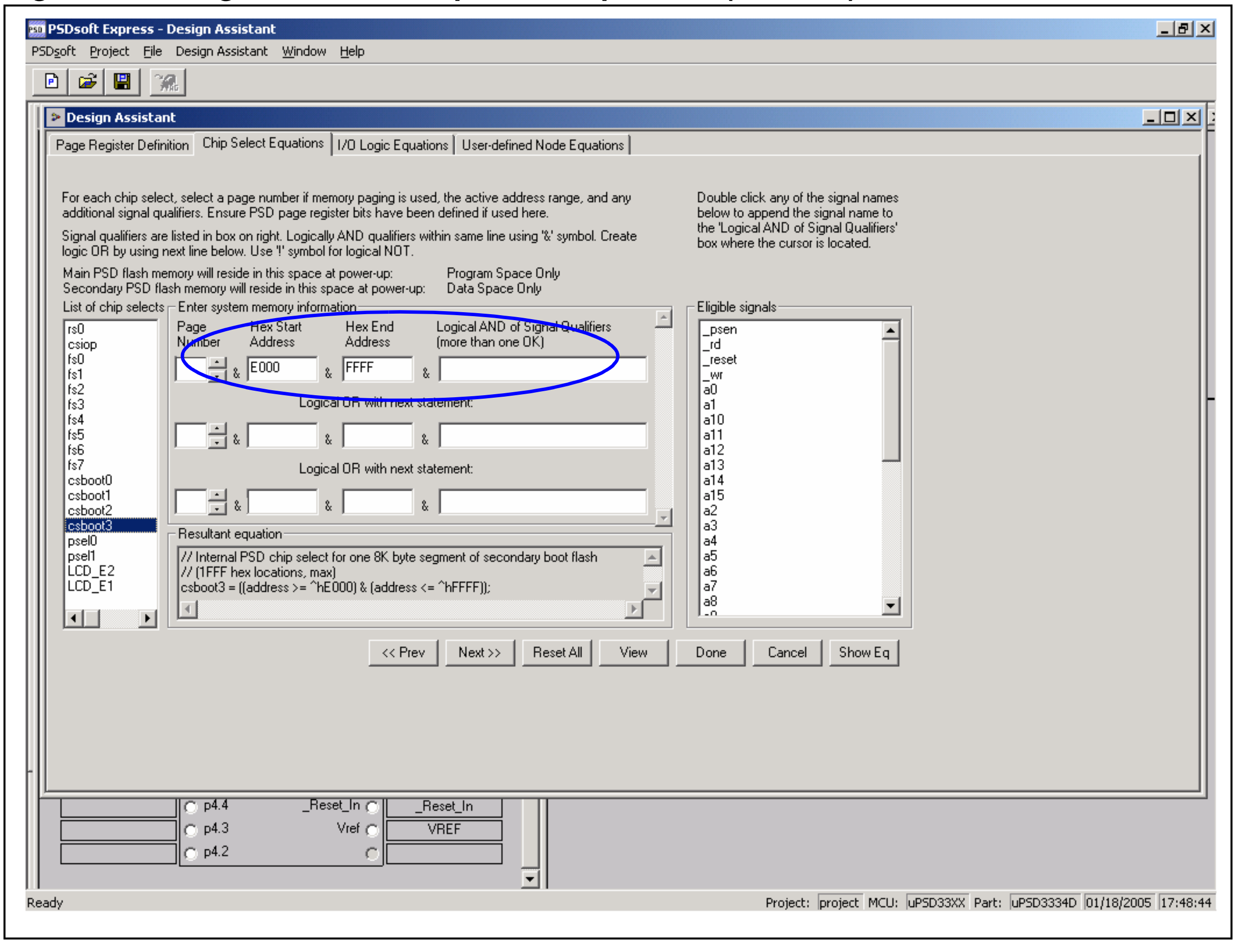
```
// Internal PSD chip select for one 32K byte segment of main flash
// (7FFF hex locations, max)
fs7 = ((page == 6) & (address >= ^h8000) & (address <= ^hFFFF));
```

Project: project MCU: uPSD33XX Part: uPSD3334D 01/18/2005 17:46:32

6. Click on csboot0, csboot1, csboot2, csboot3, which are chip-selects for the four 8 Kbyte segments of Turbo uPSD secondary flash memory (see Figure 33) - shown for csboot3.
7. Check the address assignments for each of these chip-selects.

   Note: There are no page numbers assigned; the secondary flash is common to all pages.

Figure 33. Design Assistant Chip select equations (csboot3)



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Main PSD flash memory will reside in this space at power-up: Program Space Only
Secondary PSD flash memory will reside in this space at power-up: Data Space Only

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

List of chip selects: rs0, csiop, fs0, fs1, fs2, fs3, fs4, fs5, fs6, fs7, csboot0, csboot1, csboot2, csboot3, psel0, psel1, LCD_E2, LCD_E1

Page Number	Hex Start Address	Hex End Address	Logical AND of Signal Qualifiers (more than one OK)
	E000	FFFF	

Eligible signals: _psen, _rd, _reset, _wr, a0, a1, a10, a11, a12, a13, a14, a15, a2, a3, a4, a5, a6, a7, a8

Resultant equation:

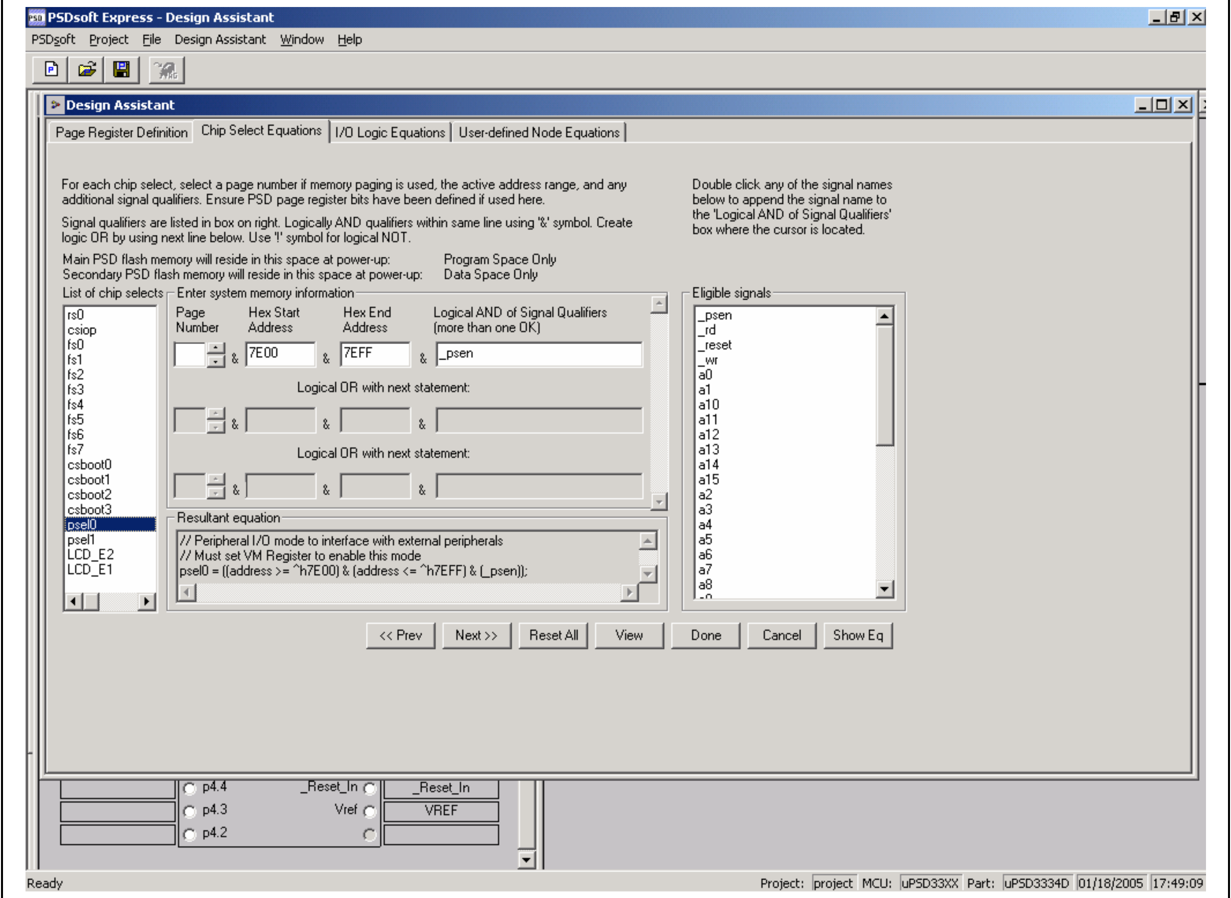
```
// Internal PSD chip select for one 8K byte segment of secondary boot flash
// (1FFF hex locations, max)
csboot3 = ((address >= ^hE000) & (address <= ^hFFFF));
```

p4.4 _Reset_In _Reset_In
p4.3 Vref VREF
p4.2

Project: project MCU: uPSD33XX Part: uPSD3334D 01/18/2005 17:48:44

8. Click on psel0. This address range specifies when Port A pins behaves like a data bus repeater in Peripheral I/O Mode to drive the ELCD module. Port A pins were earlier specified a "Peripheral I/O Mode" which acts like a 245 bus transceiver chip connecting the 8032 data bus to external peripherals over a given address range specified by the label psel0 or psel1. The direction of this transceiver function is controlled automatically in silicon by the 8032 _rd and _wr signals (see the full uPSD datasheet for details). All you have to do is click on psel0 and enter the address range 7E00-7EFF to enable this feature for that address range as shown in Figure 19, with no Page Number assignment. psel1 is not needed because the Peripheral I/O feature is active for the logical OR of psel0 or psel1.

Figure 34. Design Assistant Chip select equations (psel0)



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

Main PSD flash memory will reside in this space at power-up: Program Space Only
Secondary PSD flash memory will reside in this space at power-up: Data Space Only

List of chip selects: rs0, csiop, fs0, fs1, fs2, fs3, fs4, fs5, fs6, fs7, csboot0, csboot1, csboot2, csboot3, psel0, psel1, LCD_E2, LCD_E1

Enter system memory information

Page Number	Hex Start Address	Hex End Address	Logical AND of Signal Qualifiers (more than one OK)
	7E00	7EFF	_psen

Logical OR with next statement:

Logical OR with next statement:

Eligible signals: _psen, _rd, _reset, _wr, a0, a1, a10, a11, a12, a13, a14, a15, a2, a3, a4, a5, a6, a7, a8

Resultant equation

```
// Peripheral I/O mode to interface with external peripherals
// Must set VM Register to enable this mode
psel0 = ((address >= ^h7E00) & (address <= ^h7EFF) & (_psen));
```

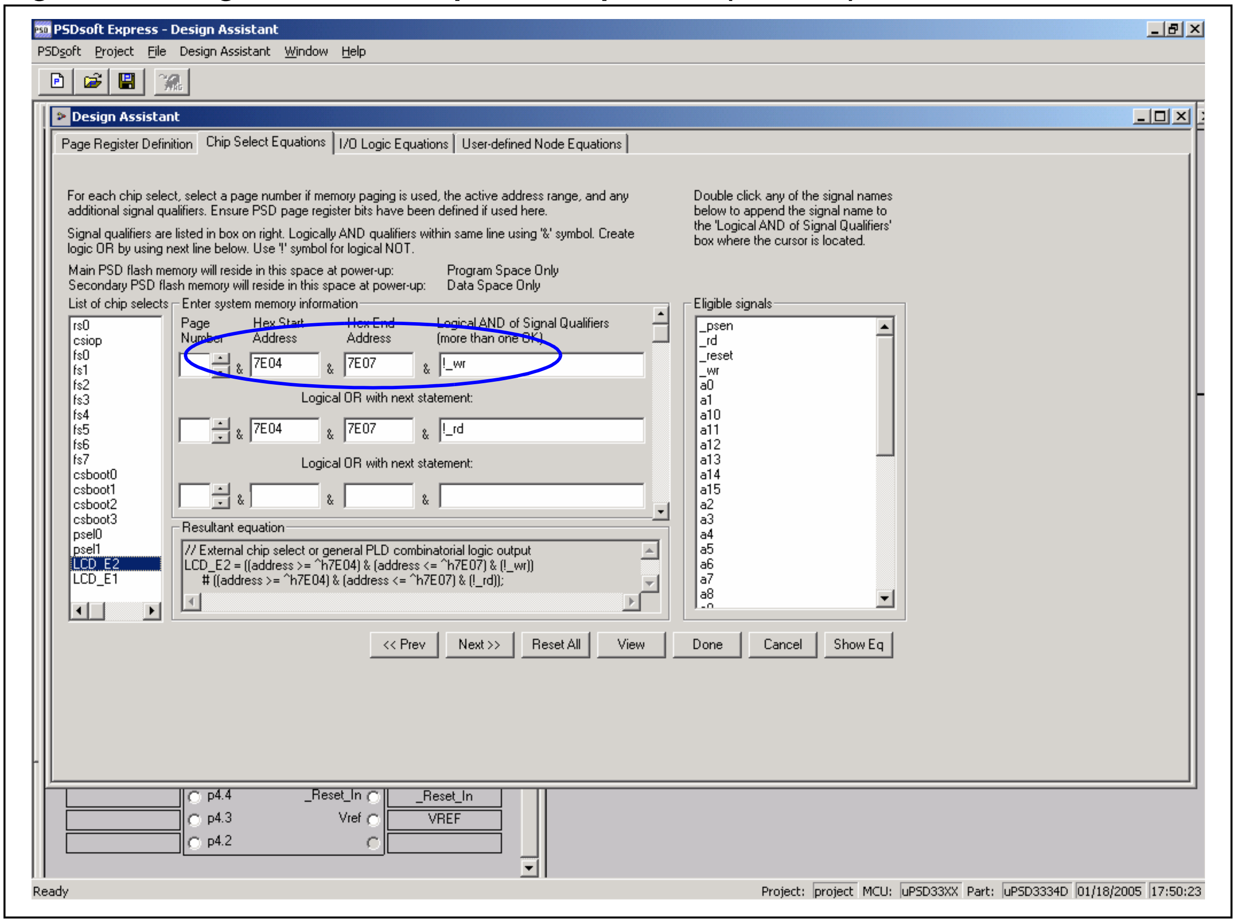
p4.4 _Reset_In _Reset_In
p4.3 Vref VREF
p4.2

Ready   Project: project MCU: uPSD33XX Part: uPSD3334D 01/18/2005 17:49:09

9. Click on LCD_E2. This is an external chip-select for the LCD module. Since this is an external chip select, you must include signal qualifiers _rd and _wr. In this design, LCD_E is true (active-high) only when the 8032 presents an address in the range of 7E04-7E07 AND when either 8032 control signal _rd is true, OR when 8032 control signal _wr is true, as shown in Figure 35

   Note: Signal qualifiers may be added by setting the cursor where you want the signal name to go, then just double-clicking on the signal name in the list of eligible qualifiers.

Figure 35. Design Assistant Chip select equations (LCD_E2)



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

Main PSD flash memory will reside in this space at power-up: Program Space Only
Secondary PSD flash memory will reside in this space at power-up: Data Space Only

Enter system memory information

Page Number		Hex Start Address		Hex End Address		Logical AND of Signal Qualifiers (more than one OK)
	&	7E04	&	7E07	&	!_wr
Logical OR with next statement:						
	&	7E04	&	7E07	&	!_rd
Logical OR with next statement:						
	&		&		&	

Resultant equation

```
// External chip select or general PLD combinatorial logic output
LCD_E2 = ((address >= ^h7E04) & (address <= ^h7E07) & (!_wr))
    # ((address >= ^h7E04) & (address <= ^h7E07) & (!_rd));
```

Project: project MCU: uPSD33XX Part: uPSD3334D 01/18/2005 17:50:23

Click on the remaining chip-selects for main flash

10. Click "Next" to move on to I/OLogic Definitions.

7.4 I/O Logic Equations

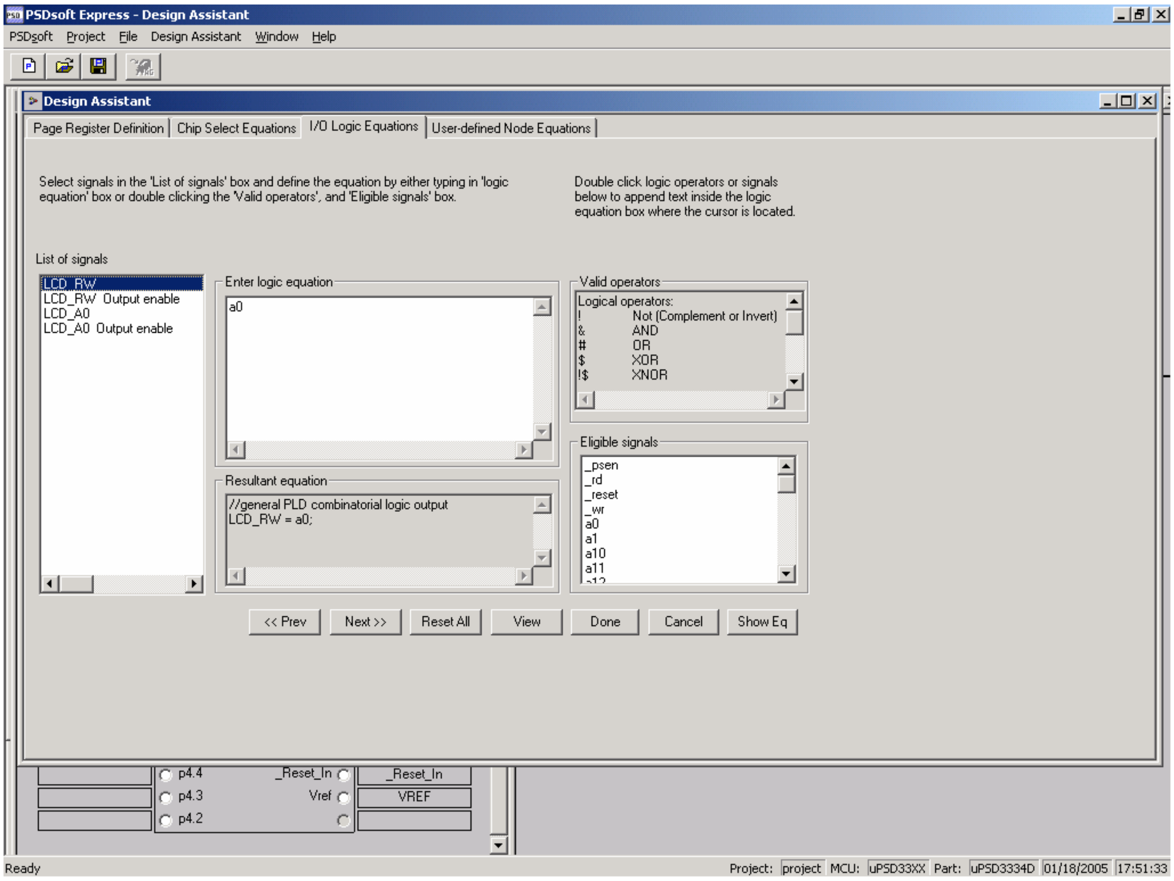
Defined here are equations for PLD outputs for the LCD interface signals. The Design Assistant (DA) will create HDL logic statements using the ABEL language in the background after you enter logic in this point-and-click design entry environment. The DA will also create all the declaration statements in ABEL. This saves much typing and reduces the chance of error. For more complicated logic PSDsoft allows you to edit the ABEL statements directly.

1. Click on "LCD_rw" as shown in Figure 36

   Note: The internal signal a0 is assigned to drive the output signal "LCD_rw". Although this was a very simple logic equation, AND, OR, XOR, NOT, and other logic operators are also available for general purpose logic.
2. Click through the remaining signal names and observe the logic assigned.

   Note: There is no logic equation assigned to term_count because that assignment will be made by editing the ABEL file directly.

Figure 36. Design Assistant I/O Logic equations (LCD_rw)



Select signals in the 'List of signals' box and define the equation by either typing in 'logic equation' box or double clicking the 'Valid operators', and 'Eligible signals' box.

Double click logic operators or signals below to append text inside the logic equation box where the cursor is located.

List of signals
- LCD_RW
- LCD_RW Output enable
- LCD_A0
- LCD_A0 Output enable

Enter logic equation

```
a0
```

Resultant equation

```
//general PLD combinatorial logic output
LCD_RW = a0;
```

Valid operators

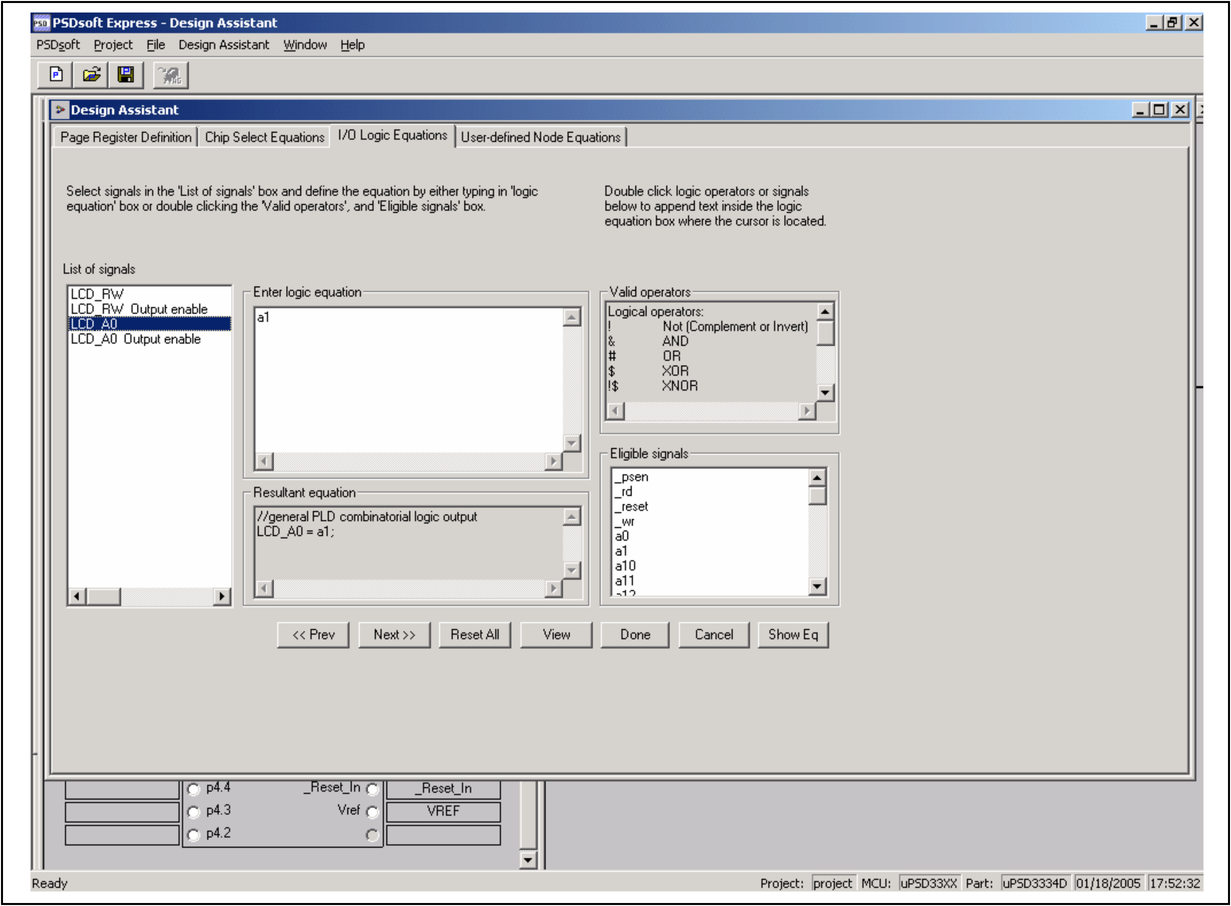
```
Logical operators:
!       Not (Complement or Invert)
&       AND
#       OR
$       XOR
!$      XNOR
```

Eligible signals
- _psen
- _rd
- _reset
- _wr
- a0
- a1
- a10
- a11

p4.4 _Reset_In _Reset_In
p4.3 Vref VREF
p4.2

Ready Project: project MCU: uPSD33XX Part: uPSD3334D 01/18/2005 17:51:33

Figure 37. Design Assistant I/O Logic equations (LCD_A0)



Select signals in the 'List of signals' box and define the equation by either typing in 'logic equation' box or double clicking the 'Valid operators', and 'Eligible signals' box.

Double click logic operators or signals below to append text inside the logic equation box where the cursor is located.

List of signals: LCD_RW, LCD_RW Output enable, LCD_A0, LCD_A0 Output enable

Enter logic equation: a1

Logical operators:

!	Not (Complement or Invert)
&	AND
#	OR
$	XOR
!$	XNOR

Eligible signals: _psen, _rd, _reset, _wr, a0, a1, a10, a11

Resultant equation:

```
//general PLD combinatorial logic output
LCD_A0 = a1;
```

p4.4 _Reset_In _Reset_In
p4.3 Vref VREF
p4.2

Project: project MCU: uPSD33XX Part: uPSD3334D 01/18/2005 17:52:32

3. Click "Next" to move on to User-Defined Node Equations.

7.5 User-Defined Node Equations

Here you will see how internal logic nodes are created. In this example, there are four registers (or nodes) to hold the initial count of the 4-bit down-counter, and four additional registers to create the actual 4-bit down-counter (see Figure 38).

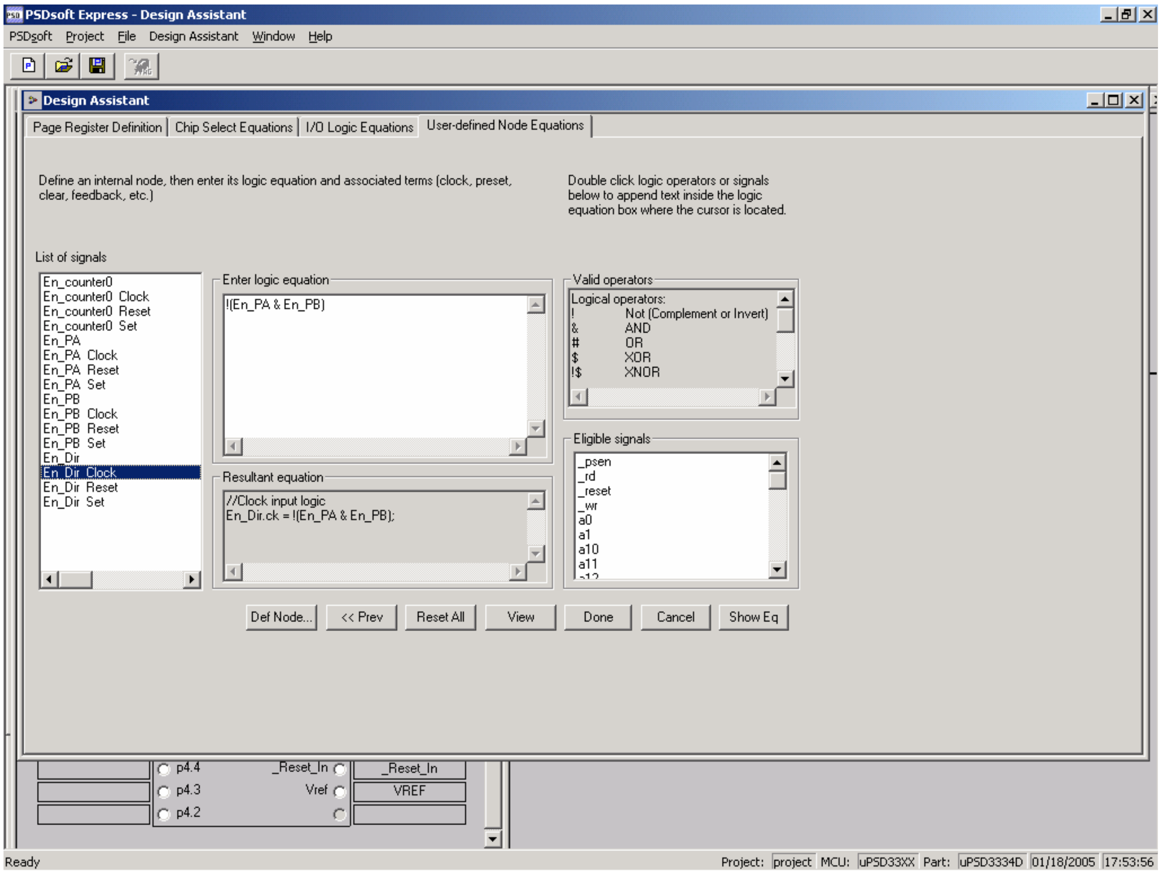
These nodes were created by:

1. Clicking the "Def Node.." button;
2. Naming the node; and
3. Delecting the type node (e.g., combinatorial, D-register, J-K register).
4. Clicking "Done."

Now you will see the main PSDsoft flow diagram that will guide you through the remaining steps.

Note: You may view a summary report at this time by pulling down the "Report" selection in the main menu bar at the top of the screen, then selecting "Design Assistant Summary." Your report should match the one in APPENDIX C.

Figure 38. User Node Equations



Define an internal node, then enter its logic equation and associated terms (clock, preset, clear, feedback, etc.)

Double click logic operators or signals below to append text inside the logic equation box where the cursor is located.

List of signals

En_counter0
En_counter0 Clock
En_counter0 Reset
En_counter0 Set
En_PA
En_PA Clock
En_PA Reset
En_PA Set
En_PB
En_PB Clock
En_PB Reset
En_PB Set
En_Dir
En_Dir Clock
En_Dir Reset
En_Dir Set

Enter logic equation

!(En_PA & En_PB)

Resultant equation

```
//Clock input logic
En_Dir.ck = !(En_PA & En_PB);
```

Valid operators

Logical operators:

!	Not (Complement or Invert)
&	AND
#	OR
$	XOR
!$	XNOR

Eligible signals

_psen
_rd
_reset
_wr
a0
a1
a10
a11

p4.4 _Reset_In _Reset_In
p4.3 Vref VREF
p4.2

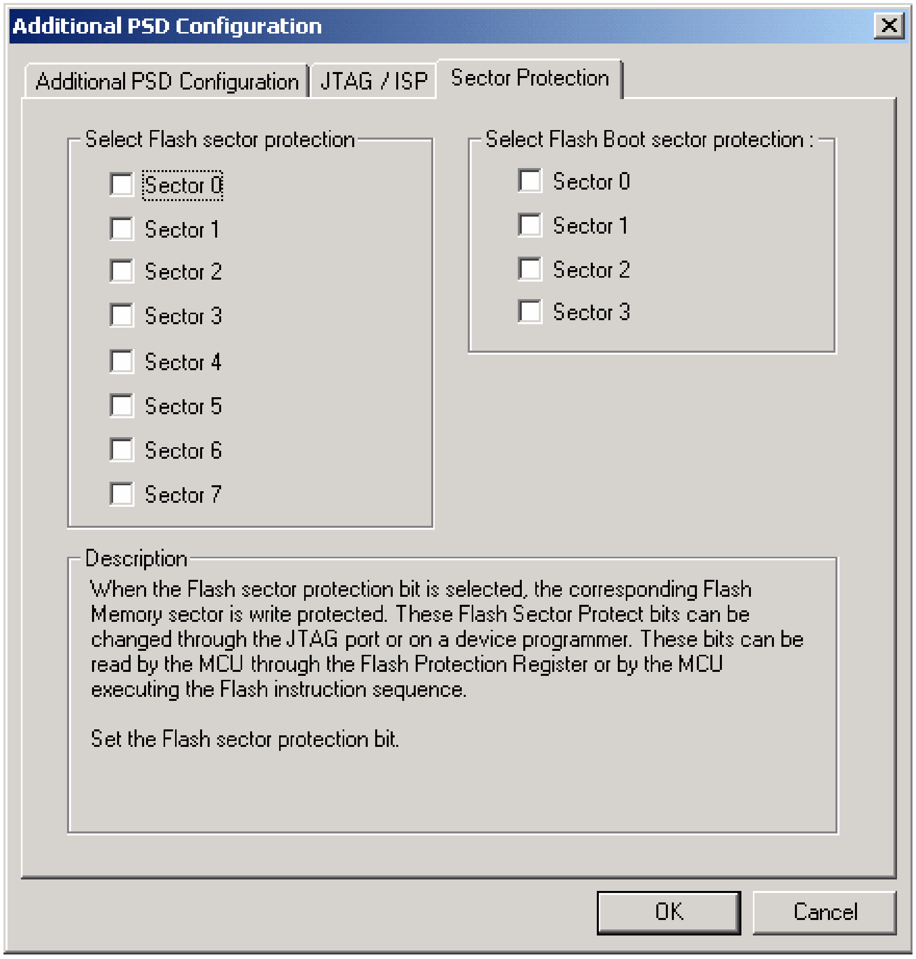
Ready Project: project MCU: uPSD33XX Part: uPSD3334D 01/18/2005 17:53:56

7.6 Additional uPSD Configuration

Click the box "Additional PSD Configuration." This is where you can choose to set the security bit to prevent a device programmer from examining or copying the contents of the Turbo uPSD. The only way to override the security bit is to erase the entire Turbo uPSD, then it can be used again as a blank part.

Note: You may also click through the other sheets on this screen to set the JTAG USERCODE value and set the sector protection on the individual uPSD Non-Volatile memory segments. (Just click "OK" for now.)

Figure 39. Additional PSD Configuration



Additional PSD Configuration

Tabs: Additional PSD Configuration | JTAG / ISP | Sector Protection

Select Flash sector protection
- [ ] Sector 0
- [ ] Sector 1
- [ ] Sector 2
- [ ] Sector 3
- [ ] Sector 4
- [ ] Sector 5
- [ ] Sector 6
- [ ] Sector 7

Select Flash Boot sector protection :
- [ ] Sector 0
- [ ] Sector 1
- [ ] Sector 2
- [ ] Sector 3

Description

When the Flash sector protection bit is selected, the corresponding Flash Memory sector is write protected. These Flash Sector Protect bits can be changed through the JTAG port or on a device programmer. These bits can be read by the MCU through the Flash Protection Register or by the MCU executing the Flash instruction sequence.

Set the Flash sector protection bit.

OK | Cancel

7.7 Fitting Design

Click the next highlighted box in the design flow, "Fit Design to Silicon." PSDsoft will compile all the configuration selections and present a report (also available in APPENDIX C.). The fitter report documents how pins are configured and how the programmable logic is allocated. It also shows how many programmable logic product terms are used, which is needed to estimate power consumption.

7.8 Merging 8032 Firmware with uPSD Configuration

Now that all Turbo uPSD pins and configuration settings have been defined, PSDsoft Express will create a single object file (*.obj) that is a composite of the 8032 firmware (*.hex) and the Turbo uPSD configuration. FlashLINK/R-LINK-ST or third-party programmer tools can use this object file to program a Turbo uPSD device. PSDsoft Express will create project1.obj for this design example. During this merging process, PSDsoft Express will input firmware files from the 8032 compiler/linker in Intel HEX format. It will map the content of these files into the physical memory segments of the Turbo uPSD according to the choices that were made in the 'Chip Select Equations' screen. This mapping process translates the absolute system addresses inside 8032 firmware files into physical internal Turbo uPSD addresses that are used by a programmer device to program the Turbo uPSD. This address translation process is transparent. All you need to do is type (or browse) the file name that was generated from the 8032 linker into the appropriate boxes and PSDsoft Express does the rest. You can specify a single file name for more than one Turbo uPSD chip-select, or a different file name for each Turbo uPSD chipselect. It depends on how the 8032 linker has created the firmware file(s). For each Turbo uPSD chip-select in which you have specified a firmware file name, PSDsoft Express will extract firmware from that file only between the specified start and stop addresses, and ignore firmware outside of the start and stop addresses.

Click on "Merge MCU Firmware" in the main flow diagram. You will see an information window pop up to remind you to be sure you have configured the firmware compiler and linker to support a paged memory mapping scheme. Select "OK" and you will see the screen shown in Figure 39.

In the left column of the "Step 1" area are Turbo uPSD memory segment chip-selects (e.g., FS0, FS1). The next column shows the logic equations for selection of each Turbo uPSD memory segment. These equations reflect the choices that were made while defining Turbo uPSD internal chip-select equations in an earlier step. In the middle of the screen are hexadecimal start and stop addresses that PSDsoft Express has filled in, based on the chip-select equations. On the right are fields to enter (browse) the 8032 firmware files. To select a firmware file:

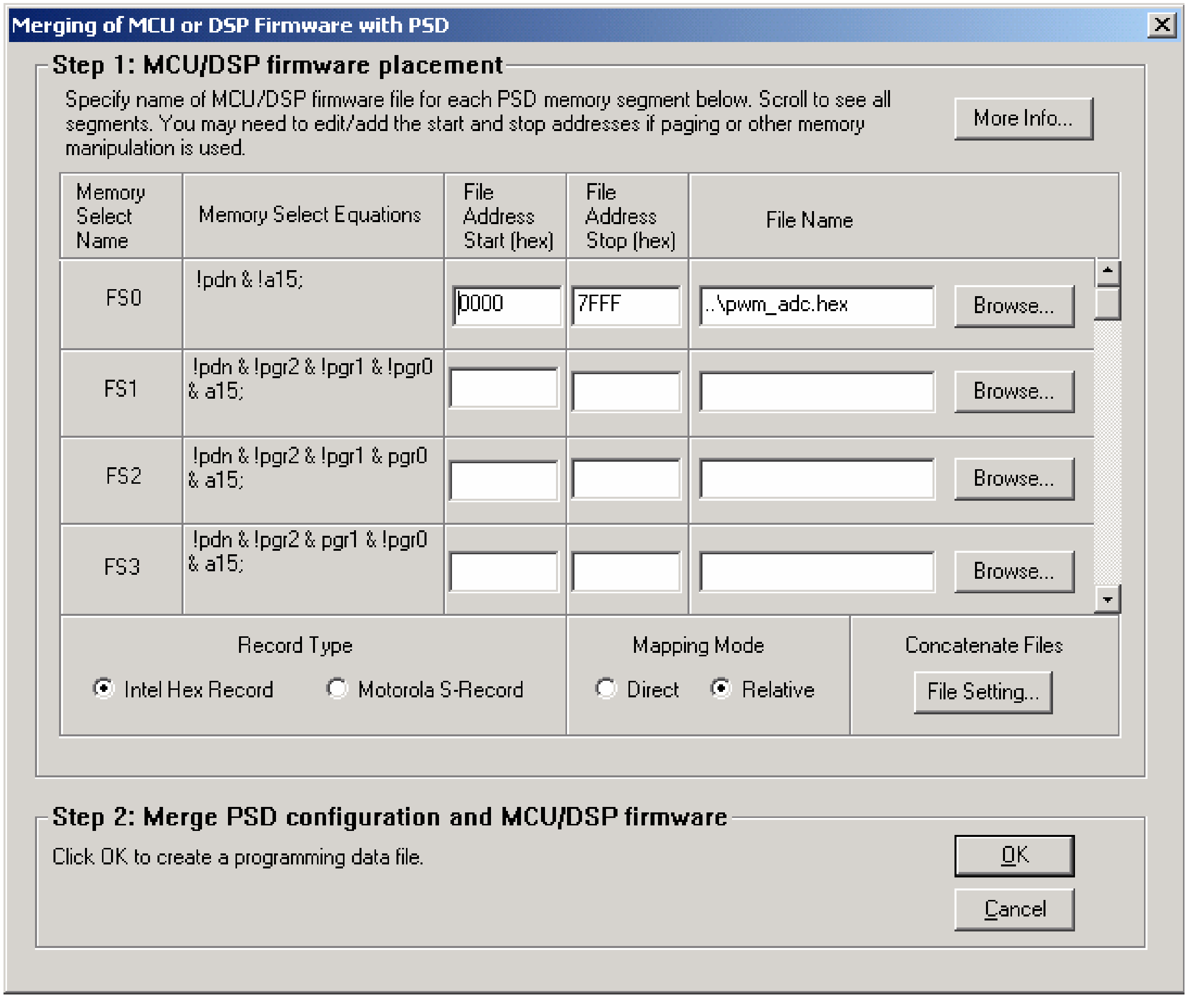
1. Select "Intel Hex Record" for 'Record Type' as shown in Figure 40.
2. Slide the bar on the right side all the way down to the bottom until you see FS0
3. See that the firmware files are in FS0.
   This is a small example program that exercises the PWM and ADC channels of the Turbo uPSD on the DK3300-ELCD board, and this code fits completely within the 32Kbyte flash segment fs0.
4. Slide the bar on the right side all the way down to the bottom until you see CSBOOT3
5. You will see the font .hex file in that segment.

This specification places firmware in primary Turbo uPSD flash memory segment ds0 and the Fonts in the CSBOOT3.

The composite object file used for the demonstration was generated as "project.obj".

You can regenerate it as long as all parameters are same.

Figure 40. Merge Firmware (Fs0)



Merging of MCU or DSP Firmware with PSD

Step 1: MCU/DSP firmware placement

Specify name of MCU/DSP firmware file for each PSD memory segment below. Scroll to see all segments. You may need to edit/add the start and stop addresses if paging or other memory manipulation is used.

Memory Select Name	Memory Select Equations	File Address Start (hex)	File Address Stop (hex)	File Name
FS0	!pdn & !a15;	0000	7FFF	..\pwm_adc.hex
FS1	!pdn & !pgr2 & !pgr1 & !pgr0 & a15;			
FS2	!pdn & !pgr2 & !pgr1 & pgr0 & a15;			
FS3	!pdn & !pgr2 & pgr1 & !pgr0 & a15;			

Record Type: [✓ Intel Hex Record] Motorola S-Record

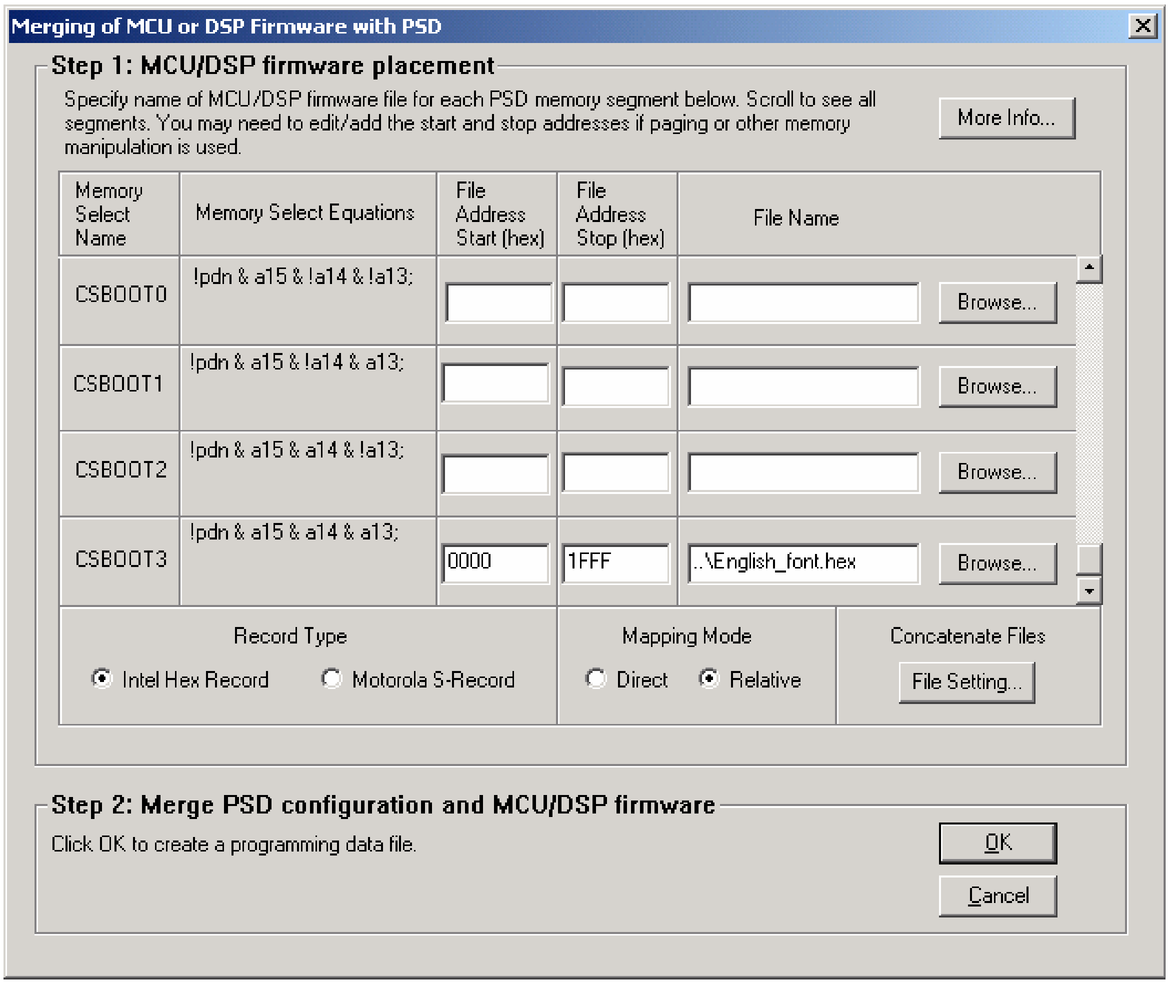
Mapping Mode: Direct [✓ Relative]

Concatenate Files

Step 2: Merge PSD configuration and MCU/DSP firmware

Click OK to create a programming data file.

Figure 41. Merge Firmware (csboot3)



Merging of MCU or DSP Firmware with PSD

Step 1: MCU/DSP firmware placement

Specify name of MCU/DSP firmware file for each PSD memory segment below. Scroll to see all segments. You may need to edit/add the start and stop addresses if paging or other memory manipulation is used.

Memory Select Name	Memory Select Equations	File Address Start (hex)	File Address Stop (hex)	File Name
CSBOOT0	!pdn & a15 & !a14 & !a13;			
CSBOOT1	!pdn & a15 & !a14 & a13;			
CSBOOT2	!pdn & a15 & a14 & !a13;			
CSBOOT3	!pdn & a15 & a14 & a13;	0000	1FFF	..\English_font.hex

Record Type: [✓ Intel Hex Record] Motorola S-Record

Mapping Mode: Direct [✓ Relative]

Concatenate Files

Step 2: Merge PSD configuration and MCU/DSP firmware

Click OK to create a programming data file.

7.9 JTAG Programming

Selection of The Programming Tool (either FlashLINK OR R-LINK) is done in the HW setup window.

1. Click the "STMicroelectronics JTAG/ISP" box to program the Turbo uPSD. You will be asked how many JTAG devices are on the target circuit board.
2. Choose "Only One" to see the screen shown in Figure 42

This window enables you to perform JTAG-ISP operations and also offers a loop back test for your FlashLINK/R-LINK cable. If this is your first use, test your FlashLINK or R-LINK cable and PC parallel or USB port by clicking the "HW Setup" button, then click "LoopTest" button and follow the directions.

To define your JTAG-ISP environment:

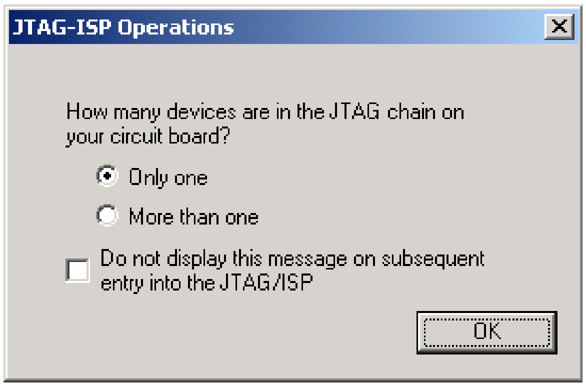
1. Connect the JTAG ribbon cable to the target system
2. Power-up the target system
3. Click 'Execute' on the JTAG screen. The Log window at the bottom of the JTAG screen shows the progress. Programming should just take a few seconds.

   Note: For this example project, PSDsoft Express should have filled in the folder and filename of the object file to program, the uPSD device, and the JTAG-ISP operation, as shown in Figure 43.

   There are optional choices available when the "Properties.." button is clicked. One choice includes setting the state of all pins on port A, B, C, or D during JTAG-ISP operations (make them inputs or outputs). The default state of these pins is "input", which is fine for this design example. The other choice allows you to specify a USERCODE value to compare before any JTAG-ISP operation starts. This is typically used in a manufacturing environment (see on-screen description for details).
4. After JTAG-ISP operations are complete, click on the 'Save' button so that you can save the JTAG setup for this programming session to a file for later use.

   Note: You may restore the setup of a different previous session by clicking the 'Browse..' button.

Figure 42. JTAG-ISP Operation Selection


JTAG-ISP Operations

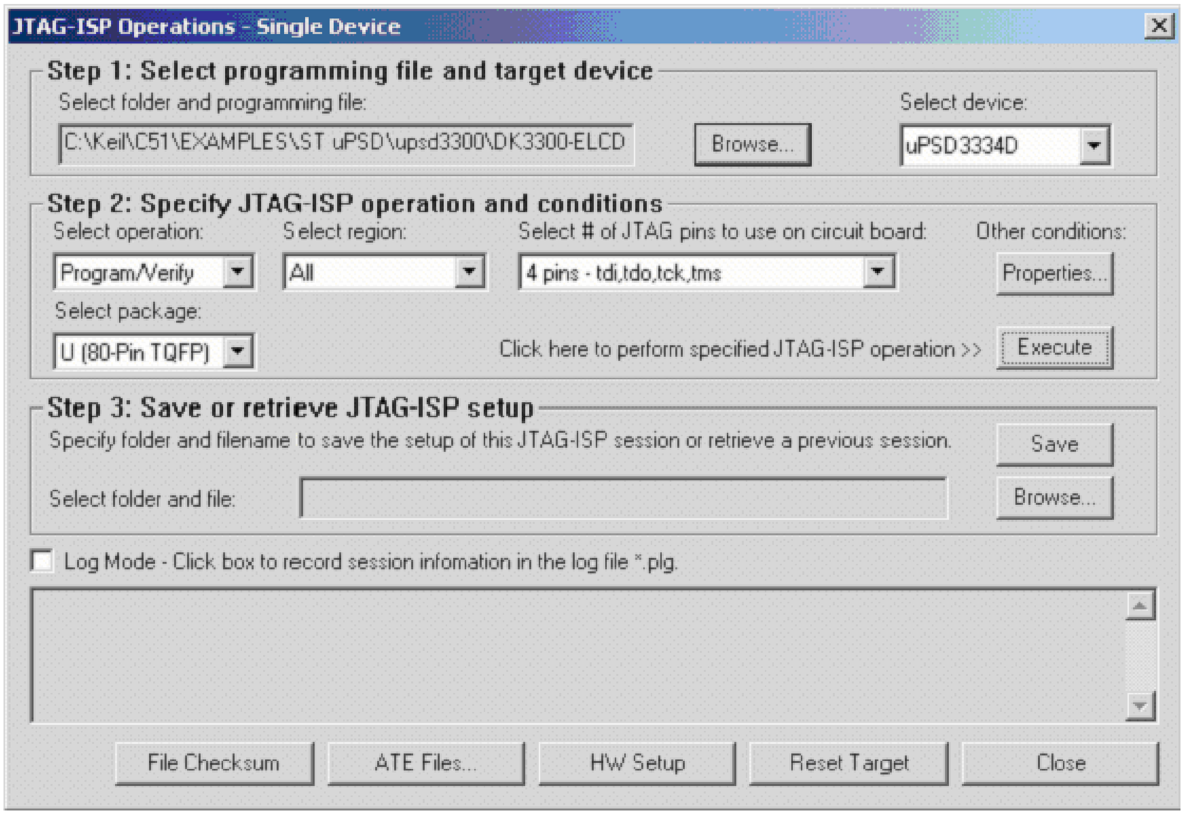
How many devices are in the JTAG chain on your circuit board?

[✓ Only one]

More than one

☐ Do not display this message on subsequent entry into the JTAG/ISP

Figure 43. JTAG-ISP Operations: Programming


JTAG-ISP Operations - Single Device

Step 1: Select programming file and target device

Select folder and programming file: C:\Keil\C51\EXAMPLES\ST uPSD\upsd3300\DK3300-ELCD

Select device: uPSD3334D

Step 2: Specify JTAG-ISP operation and conditions

Select operation: Program/Verify

Select region: All

Select # of JTAG pins to use on circuit board: 4 pins - tdi,tdo,tck,tms

Select package: U (80-Pin TQFP)

Click here to perform specified JTAG-ISP operation >>

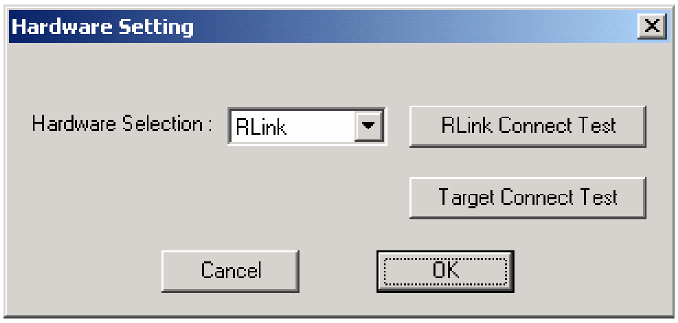
Step 3: Save or retrieve JTAG-ISP setup

Specify folder and filename to save the setup of this JTAG-ISP session or retrieve a previous session.

Select folder and file:

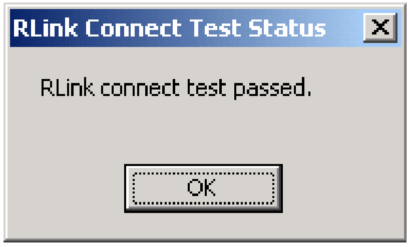
Log Mode - Click box to record session infomation in the log file *.plg.

Figure 44. Hardware Setting


Hardware Setting

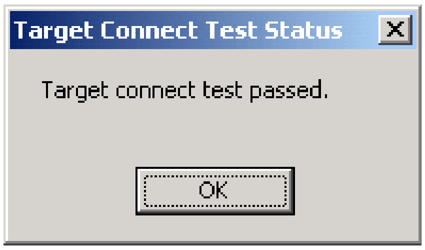
Hardware Selection : RLink

Figure 45. RLINK Test Status


RLink Connect Test Status

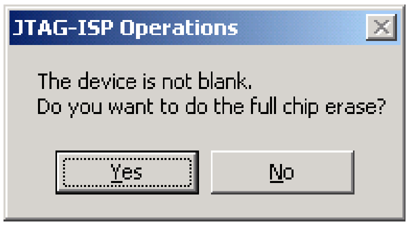
RLink connect test passed.

Figure 46. Target Connect test



Target connect test passed.

Figure 47. JTAG-ISP Message



The device is not blank.
Do you want to do the full chip erase?

7.10 Watch It Run On DK3300-ELCD

After JTAG programming completes in just a few seconds, you should see a message appear on the LCD: "PWM to ADC DEMO"

You will see the HEX value of the ADC conversion sweep up and down between 0x000 and 0x3FF as the PWM pulse width changes. If you do not see the ADC value change, make sure there are two jumpers installed on the DK3300-ELCD board. They are JP14. Remove the jumper and watch the ADC value on the LCD drop to 000 hex.

8 CONCLUSION

Congratulations! You have seen the majority of steps to implement a Turbo uPSD design on the DK3300-ELCD board. This design guide showed the basic steps to pre-configure the memories with PSDsoft, compile, program in Flash and debug with Keil Tools. The process flow diagram steps were described so that the method for creating a new project from scratch was shown and a detailed design and process based upon the PWM-ADC demo has also been described in detail with all the tools required.

You still need to review the relevant documentation in the CD-ROM about the uPSD Turbo architecture and the additional documentation supplied on the CD with the DK3300-ELCD kit as well as from the Website links provided earlier. The supplied tools from Keil limit the Code size to 2KB and any application larger than 2KB would require purchase of the full tools from Keil.

The example code and the steps clearly demonstrate the powerful firmware development and debugging capabilities of the Keil environment with ULINK for uPSD DK3300-ELCD-Development Board.

For more information, please refer to:

- Datasheet of the uPSD33xx MCU at: http://www.st.com/psm
- Application Note AN177 included in Keil Software Tools.
- Keil Software Tools documentation included with Keil Software and at: http://www.keil.com
- Schematic for the DK3300-ELCD circuit board resides at: http://psmdev.st.com/DK3300-ELCD _schematics.pdf

Please see the ST web site for the latest information on uPSD products, tools, application notes, and other documentation:

http://www.st.com/psm

APPENDIX A. DK3300-ELCD JUMPERS SELECTION AND DEFAULTS

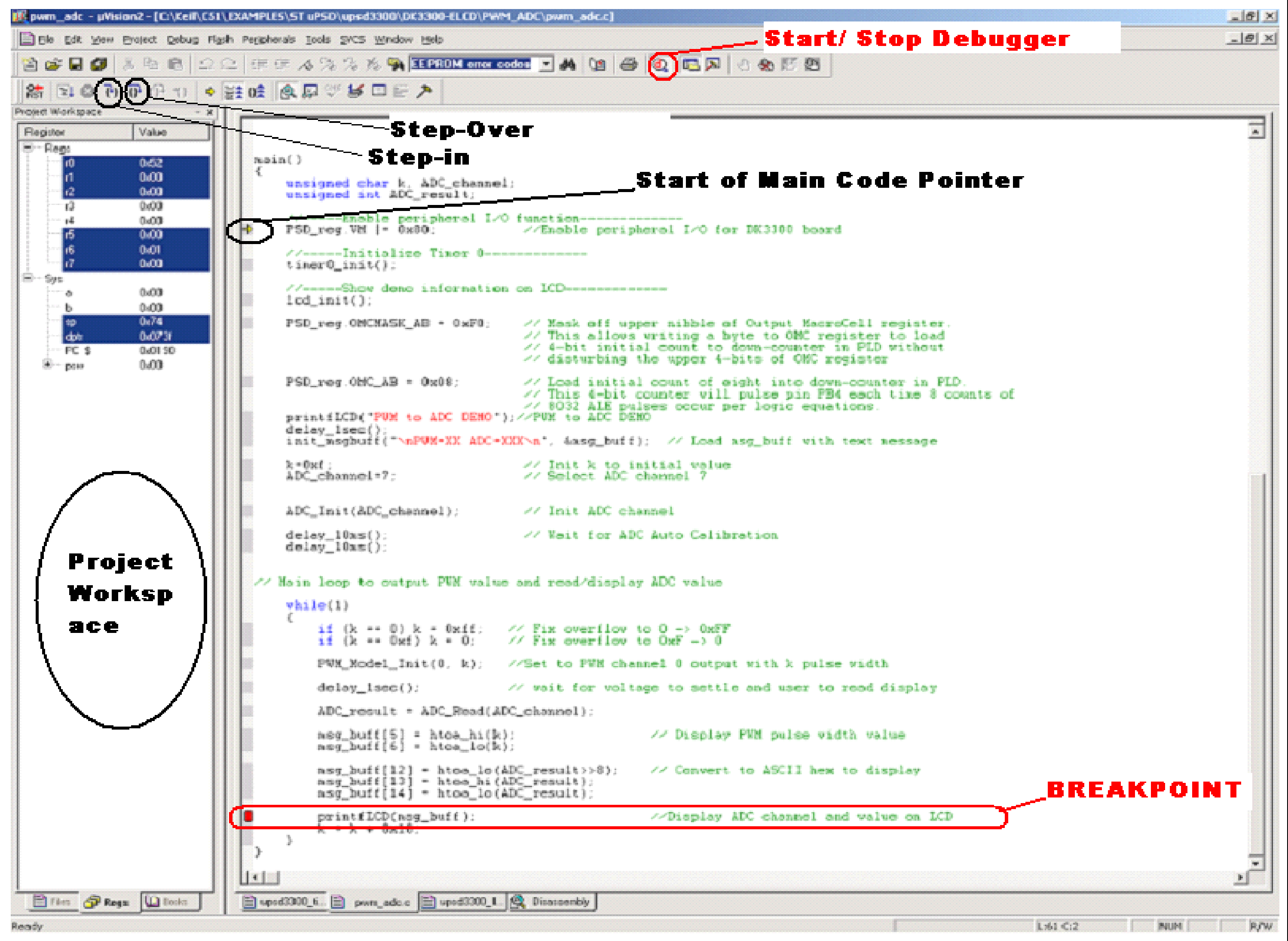
The following Table describes the DK3300-ELCD Jumpers. Verify that in Jumper set JP14 - ADC7 is closed and JP3 is set to Fix. JP5, JP4 and JP6 Jumper sets are all closed for the PW-MADC demo. See the Schematics for more information regarding the jumpers.

Table 1. DK3300-ELCD Jumpers

Jumper #	Description	Default settings	Comments/
JP1	JTAG Debug I/O Pin	Closed	Should be closed
JP2	Reset Input Select	closed in position 1-2 for reset switch.	Position 2-3 for RTC reset.
JP3	LCD Contrast	2-3 closed (Fix)	Normally closed in position 2-3 Position 1-2 used for PWM control
JP4	Enhanced LCD	Closed	Determines if Enhanced -LCD is on Board
JP5	Encoder Connection	Normally all 3 closed to enable Encoder	This connects Encoder to Port B.
JP6	Key board & LED	Closed	
JP7	Enable SPI	Closed	Normally closed to enable SPI EEPROM
JP8	IrDA / Uart1 Select	Normally 1-3 and 2-4 Closed to select the RS232 connector 1	Else can be set to position 3-5 and 4-6 to select the IrDA transceiver to be connected to Uart1.
JP9	SRAM Battery	Normally Open	
JP10	Enable I2C	Closed	Normally both positions closed to enable I2C access to RTC chip.
JP11	Clock Select	Closed X2 for Crystal	Selects Crystal or Oscillator
JP12	Interrupt Select for MCU)	Normally open. (See DK3300-ELCD schematics)	(Used to map various RTC Interrupt sources to the MCU) 1-IRQ; 2-PFO2; 3-SQW; 4-PFO1; 5-PBO
JP14	ADC Channel Select	ADC7 (Positions 15-16) is Closed	Selects what ADC channel connects the RC circuit on the board.
JP15	PWM RC Constant	Normally (position 1-2) is closed.	Selects PWM RC constant. position 1-2 is 1ms.
JP16	For connecting Uart0 and Uart1 in loop back mode	Normally open	Can be connected positions 1-2 and 3-4 for loop back.
JP18	Headers for M41ST87 Signals	Normally not used	Headers can be used to connect to check signals: 1- ECON 2-TPCLR 3-F32K 4-GND

APPENDIX B. INTERFACE DISPLAY WINDOWS AND CODE VIEW

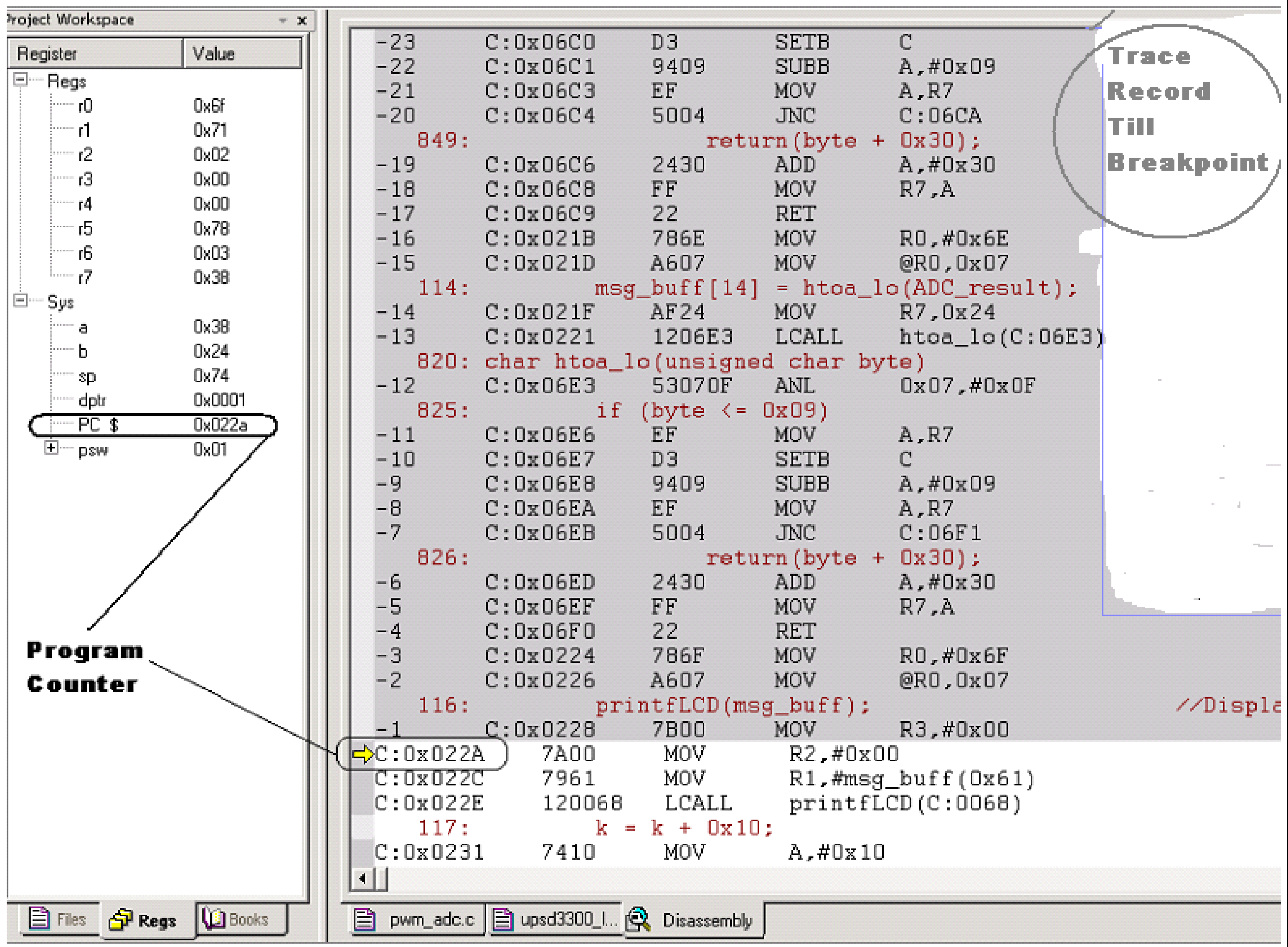
Figure 48. KEIL INTERFACE DISPLAY


pwm_adc - µVision2 - [C:\Keil\C51\EXAMPLES\ST uPSD\upsd3300\DK3300-ELCD\PWM_ADC\pwm_adc.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
EEPROM error codes
Start/ Stop Debugger
Step-Over
Step-in
Start of Main Code Pointer
BREAKPOINT
Project Workspace
Register Value
Regs
Sys
Project Workspace
Files Regs Books
upsd3300_6... pwm_adc.c upsd3300_I... Disassembly
Ready
L:61 C:2 NUM R/W

Register	Value
r0	0x52
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x01
r7	0x00
a	0x00
b	0x00
sp	0x74
dptr	0x073f
PC $	0x01SD
psw	0x00

```
main()
{
     unsigned char k, ADC_channel;
     unsigned int ADC_result;

     //------Enable peripheral I/O function--------------
     PSD_reg.VM |= 0x80;                 //Enable peripheral I/O for DK3300 board

     //------Initialize Timer 0--------------
     timer0_init();

     //------Show demo information on LCD--------------
     lcd_init();

     PSD_reg.OMCMASK_AB = 0xF0;          // Mask off upper nibble of Output MacroCell register.
                                         // This allows writing a byte to OMC register to load
                                         // 4-bit initial count to down-counter in PLD without
                                         // disturbing the upper 4-bits of OMC register

     PSD_reg.OMC_AB = 0x08;              // Load initial count of eight into down-counter in PLD.
                                         // This 4-bit counter will pulse pin PB4 each time 8 counts of
                                         // 8032 ALE pulses occur per logic equations.
     printfLCD("PWM to ADC DEMO");//PWM to ADC DEMO
     delay_1sec();
     init_msgbuff("\nPWM=XX ADC=XXX\n", &msg_buff);   // Load msg_buff with text message

     k=0xf;                              // Init k to initial value
     ADC_channel=7;                      // Select ADC channel 7

     ADC_Init(ADC_channel);              // Init ADC channel

     delay_10ms();                       // Wait for ADC Auto Calibration
     delay_10ms();

// Main loop to output PWM value and read/display ADC value

     while(1)
     {
        if (k == 0) k = 0xff;        // Fix overflow to 0 -> 0xFF
        if (k == 0xf) k = 0;         // Fix overflow to 0xF -> 0

        PWM_Mode1_Init(0, k);        //Set to PWM channel 0 output with k pulse width

        delay_1sec();                // wait for voltage to settle and user to read display

        ADC_result = ADC_Read(ADC_channel);

        msg_buff[5] = htoa_hi(k);                   // Display PWM pulse width value
        msg_buff[6] = htoa_lo(k);

        msg_buff[12] = htoa_lo(ADC_result>>8);      // Convert to ASCII hex to display
        msg_buff[13] = htoa_hi(ADC_result);
        msg_buff[14] = htoa_lo(ADC_result);

        printfLCD(msg_buff);                        //Display ADC channel and value on LCD
        k = k + 0x10;
     }
}
```

Figure 49. TRACE VIEW (Disassembly)



Register	Value
Regs	
r0	0x6f
r1	0x71
r2	0x02
r3	0x00
r4	0x00
r5	0x78
r6	0x03
r7	0x38
Sys	
a	0x38
b	0x24
sp	0x74
dptr	0x0001
PC $	0x022a
psw	0x01

```
-23     C:0x06C0    D3       SETB     C
-22     C:0x06C1    9409     SUBB     A,#0x09
-21     C:0x06C3    EF       MOV      A,R7
-20     C:0x06C4    5004     JNC      C:06CA
   849:                      return(byte + 0x30);
-19     C:0x06C6    2430     ADD      A,#0x30
-18     C:0x06C8    FF       MOV      R7,A
-17     C:0x06C9    22       RET
-16     C:0x021B    786E     MOV      R0,#0x6E
-15     C:0x021D    A607     MOV      @R0,0x07
   114:         msg_buff[14] = htoa_lo(ADC_result);
-14     C:0x021F    AF24     MOV      R7,0x24
-13     C:0x0221    1206E3   LCALL    htoa_lo(C:06E3)
   820: char htoa_lo(unsigned char byte)
-12     C:0x06E3    53070F   ANL      0x07,#0x0F
   825:         if (byte <= 0x09)
-11     C:0x06E6    EF       MOV      A,R7
-10     C:0x06E7    D3       SETB     C
-9      C:0x06E8    9409     SUBB     A,#0x09
-8      C:0x06EA    EF       MOV      A,R7
-7      C:0x06EB    5004     JNC      C:06F1
   826:                      return(byte + 0x30);
-6      C:0x06ED    2430     ADD      A,#0x30
-5      C:0x06EF    FF       MOV      R7,A
-4      C:0x06F0    22       RET
-3      C:0x0224    786F     MOV      R0,#0x6F
-2      C:0x0226    A607     MOV      @R0,0x07
   116:         printfLCD(msg_buff);                    //Displa
-1      C:0x0228    7B00     MOV      R3,#0x00
C:0x022A     7A00      MOV      R2,#0x00
C:0x022C     7961      MOV      R1,#msg_buff(0x61)
C:0x022E     120068    LCALL    printfLCD(C:0068)
   117:         k = k + 0x10;
C:0x0231     7410      MOV      A,#0x10
```

APPENDIX C. PSDSOFT REPORTS

Project.frp

This report is generated by PSDsoft after the Fit design to silicon step and the report for this example is listed here. Some Key points are highlighted for reviewing in relationship to the example design.

```
*****************************************************************************************
                                   PSDsoft Express Version 8.30
                                   Output of PSD Fitter
*****************************************************************************************
  PROJECT    : project                          DATE : 01/18/2005
  DEVICE     : uPSD3334D                        TIME : 17:56:54
  FIT OPTION : Keep Current
  DESCRIPTION: Combo Demo code to demonstrate Turbo uPSD's IPs: PCA-PWM,
               I2C, SPI, and JTAG, it runs on a DK3300 ELCD board.
*****************************************************************************************
     ==== Pin Layout for U (80-Pin TQFP) Package Type ====
-----------------------------
                                              |                             |
                                          pd2 |1 ] pd2             adio4 [41| Address Bus a4/Data Port d4, ad4
                                              |2 ] p3_3             p3_5 [42|
                                          pd1 |3 ] pd1             adio5 [43| Address Bus a5/Data Port d5, ad5
                                          ale |4 ] pd0              p3_6 [44|
                                              |5 ] pc7             adio6 [45| Address Bus a6/Data Port d6, ad6
                                     tdo, TDO |6 ] pc6/TDO          p3_7 [46|
                                     tdi, TDI |7 ] pc5/TDI         adio7 [47| Address Bus a7/Data Port d7, ad7
                               JTAG_debug_pin |8 ] debug           Xtal1 [48| Xtal1
                                  _terr, TERR |9 ] pc4/TERR        Xtal2 [49| Xtal2
                                              |10] 3.3V VCC     5.0V VCC [50|
                                              |11] N/C             adio8 [51| Address Bus a8, a8
                                              |12] 5.0V VCC         p1_0 [52|
                                              |13] GND             adio9 [53| Address Bus a9, a9
                                 tstat, TSTAT |14] pc3/TSTAT        p1_1 [54|
                                              |15] pc2            adio10 [55| Address Bus a10, a10
                                     tck, TCK |16] pc1/TCK          p1_2 [56|
                                              |17] N/C            adio11 [57| Address Bus a11, a11
                                              |18] p4_7             p1_3 [58|
                                              |19] p4_6             p1_4 [59|
                                     tms, TMS |20] pc0/TMS          p1_5 [60|
                     pa7 ,Peripheral I/O Mode |21] pa7              p1_6 [61|
                     pa6 ,Peripheral I/O Mode |22] pa6             cntl0 [62| _wr
                                              |23] p4_5            cntl2 [63| _psen
                     pa5 ,Peripheral I/O Mode |24] pa5              p1_7 [64|
                                              |25] p4_4            cntl1 [65| _rd
                     pa4 ,Peripheral I/O Mode |26] pa4               pb7 [66| pb7
                                              |27] p4_3              pb6 [67| En_EA
                     pa3 ,Peripheral I/O Mode |28] pa3          Reset_In [68| _Reset_In
                                              |29] GND               GND [69|
                                              |30] p4_2             Vref [70| VREF
                                              |31] p4_1              pb5 [71| En_EB
                     pa2 ,Peripheral I/O Mode |32] pa2              AVcc [72|
                                              |33] p4_0              pb4 [73| LCD_E2
                     pa1 ,Peripheral I/O Mode |34] pa1               pb3 [74| LCD_E1
                     pa0 ,Peripheral I/O Mode |35] pa0              p3_0 [75|
             ad0, Address Bus a0/Data Port d0 |36] adio0             pb2 [76| LCD_RW
             ad1, Address Bus a1/Data Port d1 |37] adio1            p3_1 [77|
             ad2, Address Bus a2/Data Port d2 |38] adio2             pb1 [78| LCD_A0
             ad3, Address Bus a3/Data Port d3 |39] adio3            p3_2 [79|
                                              |40] p3_4              pb0 [80|
                                              |                             |
                                               -----------------------------
     ==== Global Configuration ====
Data Bus                                                         : 8-Bit
Address/Data Mode                                                : Multiplexed
ALE/AS Signal                                                    : Active High
Control Signals                                                  : /WR, /RD, /PSEN
Main PSD flash memory will reside in this space at power-up      : Program space
Secondary PSD flash memory will reside in this space at power-up : Data space
Enable Chip-Select Input(/CSI)                                   : OFF
Standby Voltage Input (PC2)                                      : OFF
Standby-on Indicator (PC4)                                       : OFF
RDY/Busy function (PC3)                                          : OFF
Load Micro-Cell on                                               : edge
Security Protection                                              : OFF
==== DataBus_IMC access information ====
                  CSIOP
Location     Address Offset     Register Name           Signals
--------------------------------------------------------

     ===== Resource Usage Summary =====

Total Product Terms Used:  45

Device Resources                    used / total
------------------------------------------------
Port A: (pins 35 34 32 28 26 24 22 21)
I/O Pins :                           8   /  8
   GP I/O or Address Out        :    0
   Peripheral I/O               :    8
   Logic Inputs                 :    0
   Address Latch Inputs         :    0
   PT Dependent Latch Inputs    :    0
   PT Dependent Register Inputs :    0
   Combinatorial Outputs        :    0
   Registered Outputs           :    0
Other Information
   Microcells                   :    6   /  8
     Micro-Cells AB :
      Buried Microcells         :    6
      Output Microcells         :    0
   Product Terms                :    7   /  24
   Control Product Terms        :    12  /  34

Port B: (pins 80 78 76 74 73 71 67 66)
I/O Pins :                           7   /  8
   GP I/O or Address Out        :    1
   Logic Inputs                 :    2
   Address Latch Inputs         :    0
   PT Dependent Latch Inputs    :    0
   PT Dependent Register Inputs :    0
   Combinatorial Outputs        :    4
   Registered Outputs           :    0
Other Information
   Microcells                   :    6   /  8
     Micro-Cells AB :
      Buried Microcells         :    2
      Output Microcells         :    0
     Micro-Cells BC :
      Buried Microcells         :    0
      Output Microcells         :    4
   Product Terms                :    9   /  28
   Control Product Terms        :    7   /  34

Port C: (pins 20 16 15 14 9 7 6 5)
I/O Pins :                           6   /  8
   GP I/O or Address Out        :    0
   Logic Inputs                 :    0
   Address Latch Inputs         :    0
   PT Dependent Latch Inputs    :    0
   PT Dependent Register Inputs :    0
   JTAG signals                 :    6
   Standby Voltage Input        :    0
   Rdy/Bsy signal               :    0
   Standby On Indicator         :    0
   Combinatorial Outputs        :    0
   Registered Outputs           :    0
Other Information
   Microcells                   :    4   /  8
     Micro-Cells BC :
      Buried Microcells         :    4
      Output Microcells         :    0
   Product Terms                :    6   /  32
   Control Product Terms        :    0   /  34

Port D: (pins 4 3 1)
I/O Pins :                           3   /  3
   GP I/O or Address Out        :    2
   Logic Inputs                 :    0
   Chip-Select Input            :    0
   Clock Input                  :    0
   Control Signal Input         :    1
   Fast Decoding Outputs        :    0
Other Information
   Product Terms                :    0   /  3
   Control Product Terms        :    0   /  3
```

```
     ==== OMC Resource Assignment ====

  Resources           PT             User
  Used                Allocation     Name
 ---------------------------------------------------------
Micro-Cell AB :
  Micro-Cells 0        -             En_counter0 => Register
  Micro-Cells 1        -             En_PA => Register
  Micro-Cells 2        -             En_PB => Register
  Micro-Cells 3        -             En_Dir => Register
  Micro-Cells 4        -             En_counter0_C_0 => Combinatorial
  Micro-Cells 5        -             En_Dir_C_0 => Combinatorial

Micro-Cell BC :
  Micro-Cells 1        -             LCD_A0 (mcellbc1)  => Combinatorial
  Micro-Cells 2        -             LCD_RW (mcellbc2)  => Combinatorial
  Micro-Cells 3        -             LCD_E1 (mcellbc3)  => Combinatorial
  Micro-Cells 4        -             LCD_E2 (mcellbc4)  => Combinatorial

External Chip Select :


     ========= Equations =========

DPLD          EQUATIONS :
=======================
     fs0 = !pdn & !a15;

     fs1 = !pdn & !pgr2 & !pgr1 & !pgr0 & a15;

     fs2 = !pdn & !pgr2 & !pgr1 & pgr0 & a15;

     fs3 = !pdn & !pgr2 & pgr1 & !pgr0 & a15;

     fs4 = !pdn & !pgr2 & pgr1 & pgr0 & a15;

     fs5 = !pdn & pgr2 & !pgr1 & !pgr0 & a15;

     fs6 = !pdn & pgr2 & !pgr1 & pgr0 & a15;

     fs7 = !pdn & pgr2 & pgr1 & !pgr0 & a15;

     csboot0 = !pdn & a15 & !a14 & !a13;

     csboot1 = !pdn & a15 & !a14 & a13;

     csboot2 = !pdn & a15 & a14 & !a13;

     csboot3 = !pdn & a15 & a14 & a13;

     csiop = !pdn & !a15 & a14 & a13 & a12 & a11 & a10 & a9 & a8;

     rs0 = !pdn & !a15 & !a14 & !a13;

     psel0 = !pdn & _psen & !a15 & a14 & a13 & a12 & a11 & a10 & a9 & !a8;

     jtagsel = !_reset;

PORTA         EQUATIONS :
=======================
     En_counter0.T := (En_counter0.Q)
          # (!En_counter0.Q);
     En_counter0.PR = 0;
     En_counter0.RE = !_reset;
     En_counter0.C = En_counter0_C_0.FB;

     En_PA.D := 1;
     En_PA.PR = En_EA & !En_EB;
     En_PA.RE = !En_EA & En_EB;
     En_PA.C = 1;

     En_PB.D := 1;
     En_PB.PR = En_EA & En_EB;
     En_PB.RE = !En_EA & !En_EB;
     En_PB.C = 1;

     En_Dir.D := En_EB;
     En_Dir.PR = 0;
     En_Dir.RE = !_reset;
     En_Dir.C = En_Dir_C_0.FB;
     !En_counter0_C_0 = En_PB.Q & En_PA.Q;

     !En_Dir_C_0 = En_PB.Q & En_PA.Q;

PORTB         EQUATIONS :
=======================
     LCD_A0 = a1;
     LCD_A0.OE = 1;

     LCD_RW = a0;
     LCD_RW.OE = 1;

     LCD_E1 = (!_wr & !a15 & a14 & a13 & a12 & a11 & a10 & a9 & !a8 & !a7 & !a6 & !a5 & !a4 & !a3 & !a2)
          # (!_rd & !a15 & a14 & a13 & a12 & a11 & a10 & a9 & !a8 & !a7 & !a6 & !a5 & !a4 & !a3 & !a2);
     LCD_E1.OE = 1;

     LCD_E2 = (!_wr & !a15 & a14 & a13 & a12 & a11 & a10 & a9 & !a8 & !a7 & !a6 & !a5 & !a4 & !a3 & a2)
          # (!_rd & !a15 & a14 & a13 & a12 & a11 & a10 & a9 & !a8 & !a7 & !a6 & !a5 & !a4 & !a3 & a2);
     LCD_E2.OE = 1;

     En_EB.LE = 1;

     En_EA.LE = 1;

PORTC         EQUATIONS :
=======================
PORTD         EQUATIONS :
```

Project.sum

This report generated by PSDsoft provides a Summary of the whole project and has useful information. Some of it also available in Fitter report.

```
**********************************************************************
                         PSDsoft Express Version 8.30
                         Summary of Design Assistant
**********************************************************************
PROJECT    : project                     DATE : 01/18/2005
DEVICE     : uPSD3334D                   TIME : 18:10:17
MCU/DSP    : uPSD33XX
**********************************************************************

Initial setting for Program and Data Space:
===========================================

  Main PSD flash memory will reside in this space at power-up:      Program Space Only
  Secondary PSD flash memory will reside in this space at power-up: Data Space Only

Pin Definitions:
================

Pin          Signal                    Pin
Name         Name                      Type
------------ ------------------------- ------------
pa7          pa7                       Peripheral I/O mode
pa6          pa6                       Peripheral I/O mode
pa5          pa5                       Peripheral I/O mode
pa4          pa4                       Peripheral I/O mode
pa3          pa3                       Peripheral I/O mode
pa2          pa2                       Peripheral I/O mode
pa1          pa1                       Peripheral I/O mode
pa0          pa0                       Peripheral I/O mode
pb7          pb7                       GP I/O mode
pb6          En_EA                     Logic or address
pb5          En_EB                     Logic or address
pb4          LCD_E2                    External chip select - Active Hi
pb3          LCD_E1                    External chip select - Active Hi
pb2          LCD_RW                    Combinatorial
pb1          LCD_A0                    Combinatorial
tdo          tdo                       Dedicated JTAG - TDO
tdi          tdi                       Dedicated JTAG - TDI
pc4          _terr                     Dedicated JTAG - /TERR
pc3          tstat                     Dedicated JTAG - TSTAT
tck          tck                       Dedicated JTAG - TCK
tms          tms                       Dedicated JTAG - TMS
pd2          pd2                       GP I/O mode
pd1          pd1                       GP I/O mode
ale          ale                       ALE output
_psen        _psen                     Bus control output
_rd          _rd                       Bus control output
_wr          _wr                       Bus control output
a11          a11                       Address line
a10          a10                       Address line
a9           a9                        Address line
a8           a8                        Address line
ad7          a7                        Data/Address line
ad6          a6                        Data/Address line
ad5          a5                        Data/Address line
ad4          a4                        Data/Address line
ad3          a3                        Data/Address line
ad2          a2                        Data/Address line
ad1          a1                        Data/Address line
ad0          a0                        Data/Address line
debug        JTAG_debug_pin            JTAG debug pin
Xtal1        Xtal1                     Xtal1
Xtal2        Xtal2                     Xtal2
_Reset_In    _Reset_In                 Reset In
Vref         VREF                      VREF input

User defined nodes:
===================

Node         Node
Name         Type
------------ ------------
En_counter0  D-type register
En_PA        D-type register
En_PB        D-type register
```

```
En_Dir       D-type register

Page Register settings:
=======================

pgr0 is used for paging
pgr1 is used for paging
pgr2 is used for paging
pgr3 is not used
pgr4 is not used
pgr5 is not used
pgr6 is not used
pgr7 is not used

Equations:
==========

rs0 = ((address >= ^h0000) & (address <= ^h1FFF));
csiop = ((address >= ^h7F00) & (address <= ^h7FFF));
fs0 = ((address >= ^h0000) & (address <= ^h7FFF));
fs1 = ((page == 0) & (address >= ^h8000) & (address <= ^hFFFF));
fs2 = ((page == 1) & (address >= ^h8000) & (address <= ^hFFFF));
fs3 = ((page == 2) & (address >= ^h8000) & (address <= ^hFFFF));
fs4 = ((page == 3) & (address >= ^h8000) & (address <= ^hFFFF));
fs5 = ((page == 4) & (address >= ^h8000) & (address <= ^hFFFF));
fs6 = ((page == 5) & (address >= ^h8000) & (address <= ^hFFFF));
fs7 = ((page == 6) & (address >= ^h8000) & (address <= ^hFFFF));
csboot0 = ((address >= ^h8000) & (address <= ^h9FFF));
csboot1 = ((address >= ^hA000) & (address <= ^hBFFF));
csboot2 = ((address >= ^hC000) & (address <= ^hDFFF));
csboot3 = ((address >= ^hE000) & (address <= ^hFFFF));
psel0 = ((address >= ^h7E00) & (address <= ^h7EFF) & (_psen));
LCD_E2 = ((address >= ^h7E04) & (address <= ^h7E07) & (!_wr))
     # ((address >= ^h7E04) & (address <= ^h7E07) & (!_rd));
LCD_E1 = ((address >= ^h7E00) & (address <= ^h7E03) & (!_wr))
     # ((address >= ^h7E00) & (address <= ^h7E03) & (!_rd));
LCD_RW = a0;
LCD_RW.oe = Vcc;
LCD_A0 = a1;
LCD_A0.oe = Vcc;
En_counter0.ck = !(En_PA & En_PB);
En_counter0.re = !_reset;
En_counter0.pr = Gnd;
En_PA := Vcc;
En_PA.ck = Vcc;
En_PA.re = !(En_EA #( !En_EB));
En_PA.pr = En_EA &( !En_EB);
En_PB := Vcc;
En_PB.ck = Vcc;
En_PB.re = !(En_EA # En_EB);
En_PB.pr = En_EA & En_EB;
En_Dir := En_EB;
En_Dir.ck = !(En_PA & En_PB);
En_Dir.re = !_reset;
En_Dir.pr = Gnd;
```

9 REVISION HISTORY

Table 2. Document Revision History

Date	Version	Revision Details
31-Mar-2005	1.0	First Issue