

## Lab Two: PowerPC Instruction Set and MPC875 Port Registers

In this lab you will further explore the PowerPC instruction set architecture and learn how to access MPC875 port registers that control parallel I/O ports. You will also learn how to implement functions in assembly files and link them with C code.

### 1 Accessing port registers in assembly functions

In this exercise, we blink two LEDs on the QUICCstart 875 board, which are controlled by pin 29 and pin 30 of parallel I/O port B. The port is mapped to a section of main memory and can be accessed with load/store instructions. The appendix of this lab gives the detailed information about parallel I/O port B.

The objective of this exercise is to implement a function `blink_led` with PowerPC assembly language. This function repeatedly turns an LED on and off, in an infinite loop, by setting a proper pin of parallel I/O port B to 0 or 1. The function you implement should take two parameters that decide which LED blinks and how fast it blinks. For example, in your C code, you can call the function to blink an LED by a statement like:

```
blink_led(LED1, RATE1);
```

where `LED1` and `RATE1` are macros you have defined.

You may start from the sample code you had in Project 1. You may want to rename the project file to “lab2.mcp” to avoid confusions. Then you can create an assembly file, e.g., `led.asm`, under directory `C:\{Your Working Directory}\Source\`. You need to add the file into the “Source” folder in your project. Right click the Source folder in the project window (see the figure on the next page). Click “Add Files ...” on the popup menu and then “Select files to add ...”. A dialog box will appear for you to locate `led.asm` and add it into the source folder.

In the function, you configure pin 29 and pin 30 of parallel I/O port B first: set them in the general-purpose I/O mode for output. This is done by setting control registers `PBP` and `PBDIR` to proper values (see Appendix and MPC875 reference manual for details). Then you can change the value of bit 29 (or bit 30) of `PBDAT` to turn on/off the corresponding LED. Be careful. You should change only the bit you intend to change and not disturb other bits in these registers.

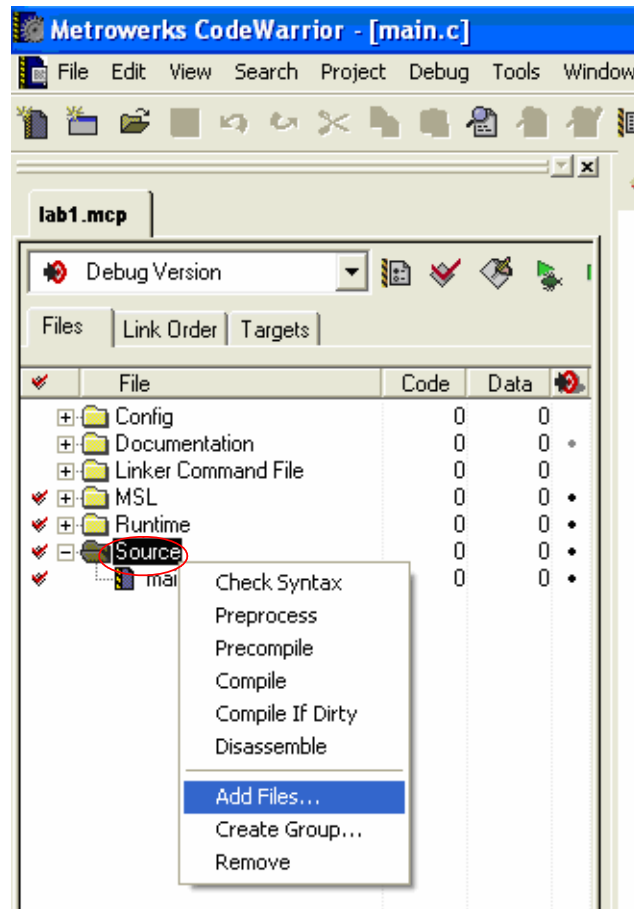
### 2 Accessing port registers in C

You need to develop a C program to perform the same tasks: repeat the first exercise, but use C language this time. You can only use embedded instructions when getting the value of `IMMR`. You CANNOT use embedded instructions to set `PBP`, `PBZDIR`, and `PBDAT`, i.e., reading

and writing parallel port B registers have to be done with C statements. The function can be placed in the same file as the main function. You are encouraged to study the assembly code generated from your C program and to compare it with your code in the first exercise. When the same parameters are provided to the functions, which of them blinks LED faster?

### 3 Deliverables

You will write a report adhering to the lab report requirements. The report will include documentation for each of the tasks. You may be asked to demonstrate your programs.



## Appendix: Parallel I/O Port B

There are five parallel I/O ports in the system. We will deal with port B only in this project. A data port can be used for multiple purposes, which are determined by control registers associated with each port. Port B has four control registers: PBODR, PBDAT, PBDIR, and PBPAR. In this project, we will access *PBPAR*, *PBDIR*, and *PBDAT* only.

The port B pin assignment register (PBPAR) configures pins as general-purpose I/O or dedicated for use with a peripheral. When a bit is cleared (set to 0), its corresponding pin is configured for general-purpose I/O. Otherwise, the pin is configured for dedicated peripheral function. The address of PBPAR is  $\text{IMMR} + 0x\text{ABE}$ , where IMMR is the value in the IMMR register *with lower 16 bits set to 0*. To control LED, pin 29 and pin 30 should be in the general-purpose I/O mode.

If a pin is programmed for general-purpose I/O, the corresponding bit of port B data direction register (PBDIR) decides whether the pin is used for input or output. If a pin is not programmed for general-purpose I/O, PBDIR selects the peripheral function to be performed. So you need to look at both PBDIR and PBPAR to decide the function of a bit. When a bit in PBDIR is set to 0, the corresponding pin will be used for general-purpose input, or for peripheral function 0 (which is determined by the value in PBPAR). If the bit is set to 1, the pin will be used for general-purpose output, or for peripheral function 1. The address of PBDIR is  $\text{IMMR} + 0x\text{ABA}$ . In this project, pin 29 and pin 30 should work in the output mode.

Reading the port B data register (PBDAT) returns data stored in the data register, regardless of whether a bit is an input or an output. The address of PBDAT is  $\text{IMMR} + 0x\text{AC6}$ . Data written to PBDAT is latched. If a PBDIR bit is configured as an output, the latched value will be sent to the corresponding pin. The board we have uses pin 29 and pin 30 to control two LEDs. PBDAT can be read or written at any time.

For more information about the parallel I/O port B, read Section 34.3.1 of *the MPC885 PowerQUICC Family Reference Manual*, which is available on the course website.