Processor Support Libraries (PSL)

1 Introduction

We have been programming using direct manipulation of register contents. For example we to set a GPIO pin as an output, we have been writing an appropriate bit pattern to the DIR0 register of the GPIO peripheral. For simple tasks, this can be the best way. However, if we wish to configure more complex peripheral devices, such as communication, direct anipulation of registers may be more difficult and time consuming to perform.

Almost all microcontroller manufacturers provide a complete set of library functions for each microcontroller they market. The Processor Support Library **(PSL)** consists of high level functions to initialize and manage the peripheral devices on the processor. They also contain a set of examples and templates, as well as an Application Programming Interface **API** document that describes in detail what library functions are available and how they work. For the LPC824 processor, the API is in Ninova, under:
`\Documents\Processor Documents\MCUXpresso_SDK_API_Reference_Manual_MKL43Z4.pdf`
You should have downloaded and installed the library itself during the set-up process. It is the file:
`\Software\Processor Support Libraries\Xpresso_SDK.zip`
SDK stands for software development kit which is another name for the library.

2 Library Functions vs. Register Bit Manipulation

The main advantage of using a processor support library is that it uses high level functions to initialize and manage the peripheral devices. For example, to set up a GPIO pin for output and to write a logic '0' to that pin is done using the following code:

```
GPIO_PortInit(GPIO, GRN_LED_PORT);

GPIO_PinInit(GPIO, GRN_LED_PORT, GRN_LED_PIN, &led_config);

GPIO_PinWrite(GPIO, GRN_LED_PORT, GRN_LED_PIN, 0);
```

It is quite clear what each function call does: The first library function call initializes the GPIO Port 0, the second sets the port uwing the `led_config` struct (see the code for more detail) and the last call writes a '0' to the port.

For comparison, the direct register manipulation code looks like:

```
SYSCON_SYSAHBCLKCTRL |= 0x400C0; // Enable clocks for IOCON, SWM &
 GPIO.

SYSCON_PRESETCTRL &= ~(0x400);  // Assert  Reset of GPIO peripheral.
SYSCON_PRESETCTRL |=   0x400;   // Release Reset of GPIO peripheral.
// See 5.6.2 Peripheral reset control register in User Manual.

//Make Pin PIO0_16 an output. On Alakart, PIO0_16 is the blue LED:
GPIO_DIR0 |= (1<<16);
```

Clearly, the direct register manipulation is more compact, but you need to explicitly calculate the values that are used. However, the processor support library code is not much more straightforward either, because we will need to find the functions and understand them before using them in our code.

The advantages and disadvantages of the PSLcan be summarized as follows:

- The PSL is more straightforward to understand and use.
- The PSL can simplify the initialization and management of complex peripherals.
- PSL source code contains a lot of extra functions which must be included in the project. Even though the extra functions are not used, they still get compiled and therefore the final code size is much larger.
- The library may contain errors or ambiguities.

Most importantly, **PSL functions are not an alternative** to the `User Manual` or the `Datasheet` of the processor. Those two documents are always the "go-to" in microcontroller programming.

3 Getting Started

How can we start using the PSL functions? There are three things that we can do:

1. See the API document. It has been uploaded to Ninova:
   `\Documents\Processor Documents\MCUXpresso_SDK_API_Reference_Manual_MKL43Z4.pdf`
   It is a reference document and lists all of the available PSL functions and how they are used. However, the information can be a bit sparse.

2. Use the examples supplied in the PSL directory. There are two example folders containing examples of usage for most peripheral devices. They do not *explain* how to use the PSL functions, but show how they can be used in several scenarios. The first location is:
   `/Xpresso_SDK/boards/lpcxpresso824max/demo_apps`

It gives some examples to get you started. The second location is:
`/Xpresso_SDK/boards/lpcxpresso824max/driver_examples`
This location gives several examples of PSL usage for many peripherals. Most of the examples for this class have their starting point as these two locations.

3. **"Read the source Luke."** All of the PSL functions are supplied as open source files that you can read. In most cases the source files contain comments that explain the functions. The source code of the PSL files are located in the folder:
   `/Xpresso_SDK/devices/LPC824/drivers$`

4 How to Use the Libraries

For your convenience, **we provide a sample LED blink program called `blink_PSL` that blinks the blue LED as before, but it uses proessor support library functions** instead of direct register bit manipulation. The program was heavily modified from the example supplied in the processor support libraries:
`Xpresso_SDK/boards/lpcxpresso824max/driver_examples/gpio/led_output`

To compile it, use the supplied Makefile. As usual, in the terminal window, go to the directory where `blink_PSL` is and type `make`. It will compile your source code and also find, compile and link the library files automatically. However, there is **one change you must make in the Makefile**: Edit the Makefile with a text editor and find the line that says:

```
# Main source directories (Insert your own installation directory):
LIBPATH=/home/onat/work/Ders/KON309E_Microcontroller/NXP284/Xpresso_SDK/
```

Here, `LIBPATH`, must be set to the folder that you have extracted the PSL files `Xpresso_SDK.zip` in your computer. Locate and copy the path of the directory. You should then paste this path into the `Makefile` as shown above.

After obtaining the `.hex` or `.bin` files, you can use the flash programmer software to program Alakart with your new source code.