ECE 2534

A short introduction to the

PIC32 Peripheral Libraries (Plib)

- So far, my code examples have interfaced to low-level Special Function Registers

```
TRISGCLR = 0x1000;   // Configure PortG bit . . .
ODCGCLR  = 0x1000;   // Configure PortG bit . . .
TRISGSET = 0x40;     // Set PortG bit   . . .
LATGCLR  = 1 << 12;  // LED1 off
```

- The above does <u>not</u> use the Plib
- Interfacing at such a low-level has advantages, and disadvantages

Plib = PIC32 Peripheral Libraries

- Allows code to be written at a (slightly) higher level
- Can write code with or without the Plib
- Without Plib:

```
#include <p32xxxx.h>

PORTGCLR = (1 << 12);
```

- With Plib:

```
#include <plib.h>

PORTClearBits(IOPORT_G, BIT_12);
```

An example without Plib:

```
#define _PLIB_DISABLE_LEGACY
#include <p32xxxx.h>

// Digilent board configuration
#pragma config ICESEL       = ICS_PGx1  // ICE/ICD Comm Channel Select
#pragma config DEBUG        = OFF       // Debugger Disabled Kit
#pragma config FNOSC        = PRIPLL    // Oscillator selection
#pragma config POSCMOD      = XT        // Primary oscillator mode
#pragma config FPLLIDIV     = DIV_2     // PLL input divider
#pragma config FPLLMUL      = MUL_20    // PLL multiplier
#pragma config FPLLODIV     = DIV_1     // PLL output divider
#pragma config FPBDIV       = DIV_8     // Peripheral bus clock divider
#pragma config FSOSCEN      = OFF       // Secondary oscillator enable

int main()
{
    TRISGCLR = (1 << 12) | (1 << 13);

    while (1) {
      if (PORTG & (1 << 6))
        PORTGSET = (1 << 12);
      else
        PORTGCLR = (1 << 12);
      if (PORTG & (1 << 7))
        PORTGSET = (1 << 13);
      else
        PORTGCLR = (1 << 13);
    }
    return 0;
}
```

Correction:

should use

LATGSET and

LATGCLR

Same example, now with Plib:

```
#define _PLIB_DISABLE_LEGACY
#include <plib.h>

// Digilent board configuration
#pragma config ICESEL       = ICS_PGx1  // ICE/ICD Comm Channel Select
#pragma config DEBUG        = OFF       // Debugger Disabled Kit
#pragma config FNOSC        = PRIPLL    // Oscillator selection
#pragma config POSCMOD      = XT        // Primary oscillator mode
#pragma config FPLLIDIV     = DIV_2     // PLL input divider
#pragma config FPLLMUL      = MUL_20    // PLL multiplier
#pragma config FPLLODIV     = DIV_1     // PLL output divider
#pragma config FPBDIV       = DIV_8     // Peripheral bus clock divider
#pragma config FSOSCEN      = OFF       // Secondary oscillator enable

int main()
{
   PORTSetPinsDigitalOut(IOPORT_G, BIT_12 | BIT_13);

   while (1) {
     if (PORTRead(IOPORT_G) & 0x40)
       PORTSetBits(IOPORT_G, BIT_12);
     else
       PORTClearBits(IOPORT_G, BIT_12);

     if (PORTRead(IOPORT_G) & 0x80)
       PORTSetBits(IOPORT_G, BIT_13);
     else
       PORTClearBits(IOPORT_G, BIT_13);
   }
   return 0;
}
```

From the Plib manual…

32-BIT LANGUAGE TOOLS LIBRARIES

10.8 WRITE OPERATIONS

Macros	Functions
mPORTAWrite( ) ... *mPORTGWrite( )*	*PORTWrite( )*
mPORTASetBits( ) ... *mPORTGSetBits( )*	*PORTSetBits( )*
mPORTAClearBits( ) ... *mPORTGClearBits( )*	*PORTClearBits( )*
mPORTAToggleBits( ) ... *mPORTGToggleBits( )*	*PORTToggleBits( )*

- **Description**

  Before reading and writing to any I/O port, the data direction of a desired pin must be properly configured as digital input or digital output. Some port I/O pins share digital and analog features and require the analog feature to be disabled when configuring the I/O port pin for digital mode.

- **Useage**

  These functions are typically used early in the program execution to establish the proper mode and state of the general purpose IO pins.

PORTSetBits

Description: This function sets the selected PORT pins.

Include: `plib.h`

Prototype:

```
void PORTSetBits(IO_PORT_ID port, unsigned int
bits);
```

Arguments:

port This argument is an IO_PORT_ID which specifies the desired port. Select only one mask from the mask set defined below.

IO PORT ID

```
IOPORT_A
IOPORT_B
IOPORT_C
IOPORT_D
IOPORT_E
IOPORT_F
IOPORT_G
```

bits This argument contains one or more bit masks bitwise OR'd together. Select one or more masks from the mask set defined below. Note: An absent mask symbol assumes corresponding bit(s) are disabled, or default value, and will be set = 0.

IO Pin Bit Masks

```
BIT_0
BIT_1
BIT_2
...
BIT_15
```

Return Value: None

PROBLEM: Parts of the Plib manual are *out of date*

- *SOLUTION*: Look at the source code (primarily the *.h files)

- Find your MPLAB XC32 compiler installation directory, e.g.,

  c:\Program Files (x86)\Microchip\xc32\v1.31

- It contains the following subdirectories with library-related files

  - …\pic32mx\include\plib.h - master include file for all APIs
  - …\pic32mx\include\peripheral*.h - API header files
  - …\pic32-libs\peripheral\... - library source files

PLIB functions for GPIO ports

c:

→ Program Files (x86)

→ Microchip

→ xc32

→ v1.31

→ pic32mx

→ include

→ peripheral

→ ports.h

Example for PORTSetBits()

```
/*****************************************************************************
 * Sets digital port bits
 *
 * Function:        void    PORTSetBits(IoPortId portId, unsigned int bits);
 *
 * Description:     Writes selected bits to PORTSET register
 *
 * PreCondition:    None
 *
 * Inputs:          portID: Enumerated PORT Identifier - see ports.h
 *
 *                  bits:   Mask of bits to be written
 *
 * Output:          None
 *
 * Example:         PORTSetBits(IOPORT_B, BIT_2 | BIT_4);
 *
 *****************************************************************************/
void    PORTSetBits(IoPortId portId, unsigned int bits);
```

Summary

- Using the Plib can be very helpful when working with complicated peripherals

- Refer to the Plib manual, but with caution
- Refer to the *.h files for up-to-date information