Chapter 2
Walking in Circles

Programming Loops in C
The Plan

- The Main Loop
- WHILE loops
- Logic Expressions
  - Of TRUE and FALSE in C
- DO loops
- First peripheral: Timer1 (TMR1)
- Debugging with MPLAB SIM
  - Animations
  - The Logic Analyzer
The Structure of an (Embedded) C Program

- The `main()` function is where we place our application code.
- At power up or after a reset, special initialization code (`crt0`) is executed first before anything else.
- When the `main()` function returns, the `_exit()` function is called.
- Both `crt0` and `_exit()` contain default code inserted automatically by the C32 compiler.
WHILE Loops

while (x) // x is Boolean
{
   // your code
}

Di Jasio - Programming 32-bit Microcontrollers in C
Logic Expressions

- true and false are integers in C
- false is zero
- true is everything else!
- Logic operators produce true/false results:
  - || the “logic OR” operator,
  - && the “logic AND” operator,
  - ! the “logic NOT” operator
  - == the “equal-to” operator
  - != the “NOT-equal to” operator.
  - > the “greater-than” operator.
  - >= the “greater-or-equal to” operator.
  - < the “less-than” operator.
  - <= the “less-or-equal to” operator.
Strange Loops

```c
while (0) {
    // your code
}

while (1) // not so strange
{
    // your code
}
```
“Day 2” Program: Loops.c

```c
#include <p32xxxx.h>
main()
{
    // initialization
    DDPCONbits.JTAGEN = 0;  // disable the JTAG port
    TRISA = 0xff00;          // PORTA pins 0..7 as output

    // application main loop
    while(1)
    {
        PORTA = 0xff;       // turn PORTA pins 0..7 on
        PORTA = 0;          // turn all PORTA pins off
    }
}
```
MPLAB SIM: Animation

- Use the **Project Build checklist** to compile and link the “loops.c” program.
- Also use the **MPLAB SIM simulator Setup checklist** to prepare the software simulator.
- Use the *Animate* mode (**Debugger>Animate**). In this mode, the simulator executes one C program line at a time, pausing shortly after each one to give us the time to observe the immediate results.
- Control the simulation speed with the **Debug>Settings** dialog box, selecting the **Animation/Real Time Updates** tab, and modifying the **Animation Step Time** parameter.
Timer1 (TMR1)
Timer1

- A 16-bit timer
- Compatible with 8-bit PIC microcontrollers and 16-bit microcontrollers
- Operates off the internal (peripheral) clock or an external input signal
- Has a dedicated low power (secondary) oscillator that can also be used to provide a clock signal to the entire system
- Can be gated via a separate input pin
- A 16-bit register for cyclical operation
Using Timer1 for Simple Delays

while (TMR1 < DELAY)
{
    // wait for \texttt{Tdelay} time
}

- \texttt{Tdelay} = \frac{1}{Fpb} \times \text{Prescaler} \times \text{DELAY}
- \texttt{Fpb}: \text{36-MHz peripheral bus}
- \text{Prescaler}: \text{TMR1 allows 1, 8, 64, 256}
- \texttt{DELAY}: \text{a unit-less user-defined constant}
- See pp. 31\textendash34 of Di Jasio book for details
Loops.c

/*
 ** Loops
 */
#include <p32xxx.h>
#define DELAY 36000        // delay time = 256 ms => DELAY = 36000
main()
{
    // 0. initialization
    DDPCONbits.JTAGEN = 0;  // disable JTAGport to free up PORTA
    TRISA = 0xff00;         // PORTA pins 0..7 as output
    T1CON = 0x8030;         // TMR1 on, prescale 1:256 PB=36MHz
    PR1 = 0xFFFF;           // set timer reset period to maximum
    // 1. main loop
    while(1)
    {
        // 1.1 turn all LEDs ON
        PORTA = 0xff;
        TMR1 = 0;
        while (TMR1 < DELAY)
        {
            // wait for 256 ms
        }
        // 1.2 turn all LEDs OFF
        PORTA = 0;
        TMR1 = 0;
        while (TMR1 < DELAY)
        {
            // wait for 256 ms
        }
    } // while loop
} // main()
MPLAB SIM: Logic Analyzer

- Select the **Debug>Settings** dialog box and then choose the **Osc/Trace** tab.
- In the **Tracing** options section, check the **Trace All** box.
- Now you can open the Analyzer window, from the **View->Simulator Logic Analyzer** menu.
Adding Channels

- Click on the **Channels** button, to bring up the channel selection dialog box.
- select **RA0** and click **Add** =>
- Click on **OK**
Di Jasio - Programming 32-bit Microcontrollers in C
Summary

In “Day 2” we learned:

- Function of the main()
- The WHILE loop
- Logic expressions in C
- Timer1 (TMR1)
- The MPLAB SIM simulator:
  - Animate mode
  - Logic Analyzer window
Advanced Material

Let’s learn it anyway.
Timer0 is gone.
All timers are now 16-bit wide.
Each timer has a 16-bit period register.
A new 32-bit mode timer-pairing mechanism is available for Timer2/3 and Timer4/5 (more on this later).
A new external clock gating feature has been added on Timer1 via separate input pin.
More Notes

- **Notes for 16-bit PIC Experts**
  - The *peripheral bus* and the *system bus* are now separated to allow better power control when executing code at higher clock speeds.

- **Notes for C experts**
  - While several *Real Time Operating Systems* (RTOSs) are available for the PIC32, a large number of applications won’t need and won’t use one. By default, the C32 compiler assumes there is no operating system to return control to.

- **Notes for MIPS experts**
  - The *core (32-bit)* *timer* is available as an additional resource to the set of five peripheral timers compatible with the 16-bit PIC microcontrollers.
More Notes

- PIC32 decoupled the core bus clock and the peripheral bus clock.

- This freed the MIPS core of PIC32 from the limitations of the flash memory array and of the peripheral modules for an improved processor performance.

- There are totally 6 timers inside the PIC32!
Tips and Tricks

- When writing an application that is “always ON”, consider the impossible:
  - Refresh the most important control registers of the essential peripherals used by the application.
  - Group the sequence of initialization instructions in one or more functions.
  - Call these functions once at power up, before entering the main loop.
  - Make sure that inside the main loop the initialization functions are called when idling and no other critical task is pending so that every control register is re-initialized periodically.
Peripheral Libraries

- Using the peripheral libraries
  
  ```
  #include <plib.h> and <timer.h>
  ```

- We can replace the current code:
  ```
  TMR1 = 0;
  T1CON = 0x8030;
  PR1 = 0xFFFF;
  ```

- With this (more portable) code:
  ```
  WriteTimer1(0);
  OpenTimer1(T1_ON | T1_PS_1_256, 0xFFFF);
  ```
Suggested Exercises

- Output a counter on PORTA pins instead of the alternating on and off patterns.
  - Use PORTD if you have PIC32 Starter Kit.

- Use a rotating pattern instead of alternating on and off.

- Re-write the “loops” project using exclusively peripheral library functions to:
  - Control the PORTA pins
  - Configure and read the timer
  - Disable the JTAG port if necessary