Ihsan Doğramacı Bilkent University

CS 224

Computer Organization

Preliminary Design Report
Lab-7

Kerem Ayöz
Section 03 / 21501569

20/12/2017

Part b

The general purpose I/O pins can be considered the simplest of peripherals. They allow the PIC32MX microcontroller to monitor and control other devices. To add flexibility and functionality to a device, some pins are multiplexed with alternate function(s). These functions depend on which peripheral features are on the device. In general, when a peripheral is functioning, that pin may not be used as a general purpose I/O pin.

Following are some of the key features of this module:

- Individual output pin open-drain enable/disable
- Individual input pin pull-up enable/disable
- Monitor select inputs and generate interrupt on mismatch condition
- Operate during CPU SLEEP and IDLE modes
- Fast bit manipulation using CLR, SET and INV registers

The I/O Ports module consists of the following Special Function Registers (SFRs):

- TRISx: Data Direction register for the module ‘x’
- PORTx: PORT register for the module ‘x’
- LATx: Latch register for the module ‘x’
- ODCx: Open-Drain Control register for the module ‘x’
- CNCON: Interrupt-on-Change Control register
- CNEN: Input Change Notification Interrupt Enable register
- CNPUE: Input Change Notification Pull-up Enable register

Part c

```
/*
 * Project name:
     Led_Blinking (The simplest simple example)
 * Copyright:
     (c) Kerem Ayöz, 2017.
 * Information:
     21501569:
       - CS 224 / Section: 3;
 * Description:
     Simple "Hello world" example for the world of PIC32 MCUs;
 * Test configuration:
     MCU:             P32MX795F512L
*/

void main() {
  AD1PCFG = 0xFFFF;      // Configure AN pins as digital I/O
  JTAGEN_bit = 0;        // Disable JTAG

  TRISA = 0;             // Initialize PORTA as output


  LATA = 119;            // Set LATA to 0x10001000
  LATB = 119;            // To save the last value while entering frozen state

  while(1) {
     if (PORTF.F1 == 1) {  // EN==1, rotate and display the value
       if (LATA == 255) {  //If we have entered the frozen state, restore the
last value
          LATA = LATB;     //Display the last value
       }
       else if (PORTF.F2 == 1) {   // DIR==1
           LATA = ((LATA << 1) | (LATA >> (7))); //Rotate left by 1
       }
       else if (PORTF.F2 == 0) {   // DIR==0
           LATA = ((LATA << 7) | (LATA >> 1));  //Rotate right by 1
       }
     }
     else {                // EN=0, frozen state and displaying 0
          if (LATA != 255) //Save the last value in LATB
             LATB = LATA;
          LATA = 255;
     }
    Delay_ms(1000);
  }
}
```

Part d

```
/*
 * Project name:
     Led_Blinking (The simplest simple example)
 * Copyright:
     (c) Kerem Ayöz, 2017.
 * Information:
     21501569:
       - CS 224 / Section: 3;
 * Description:
      A simple example of using the ADC library.
      ADC results are displayed on PORTA.
 * Test configuration:
     MCU:             P32MX795F512L
 */

#include <built_in.h>

unsigned long adc_result = 0;

void main() {
  int x = 0;
  AD1PCFG = 0xFFFE;              // Configure AN pins as digital I/O, PORTB.B0
as analog
  JTAGEN_bit = 0;                // Disable JTAG port
  LATE = 0;
  TRISE = 0;                     // Set PORTD as output

  ADC1_Init();                   // Initialize ADC module
  Delay_ms(100);

  while(1) {
    TRISE = ~TRISE;
    if (TRISE == 1) {
       LATB = PORTEbits.RE0 + (PORTEbits.RE2 << 1) + (PORTEbits.RE4 << 2) +
(PORTEbits.RE6 << 3) + (PORTEbits.RE7 << 4) + (PORTEbits.RE5 << 5) +
(PORTEbits.RE3 << 6) + (PORTEbits.RE1 << 7);
    }
    else {
       LATE = LATB;
    }
    Delay_ms(50);
  }
}
```