Embedded Systems Module. 6EJ505

C Tutorial 3: using the ICD3

rev. 27.9.16 tjw

Images are reproduced from Reference 1. Microchip permits the use of its images for educational purposes.

Main Learning Points

C Programming	Using MPLABX and dev. tools
18F2420 peripheral initialization through SFRs	Use of the ICD3
Simple use of Pulse Width Modulation	

0. Preliminaries

This introduction assumes you have MPLABX running on a PC, an ICD 3, and a Derbot board populated to Build Stage 1 (i.e. a "House Derbot"). If you are running from your own laptop, you may need to install the USB device driver for the ICD3.

1. Introducing the Microchip in-circuit debugger (ICD 3)

The features of the Microchip ICD 3 system are illustrated in Figures 1 and 2; the host computer is linked to the target system via the ICD 3 pod. The ICD is controlled from the host computer, which must be running MPLAB X. The ICD pod links to the host computer via a USB cable. It then connects to the target system via another cable, having five interconnections. These are shown in Figure 2.

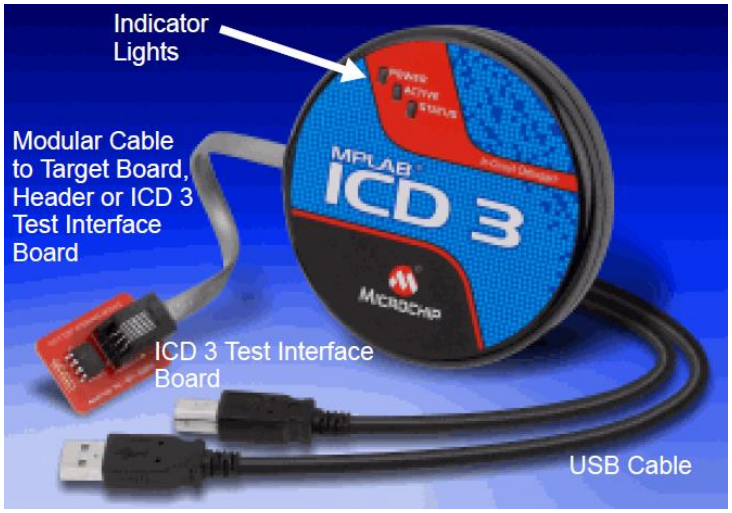

Figure 1

The ICD 3 can be used as a *debugger*, in which case it can program the microcontroller, at the same time downloading its own 'debug executive', which is loaded into the high end of program memory. Running from MPLAB X it can then execute all the functions of the MPLAB X simulator, except that now the program is actually running in the hardware. Running in this mode, the target has to remain connected to the ICD and the host computer. The ICD 3 can also be used simply as a *programmer*, downloading the program into the microcontroller. In this mode, once programmed the target must be disconnected from the ICD 3, and then runs independently.

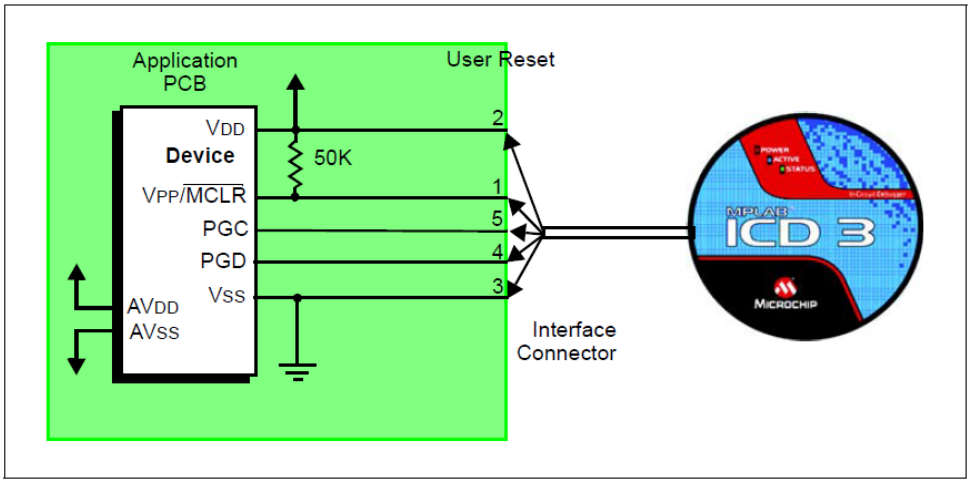

Figure 2: The required ICD3 connections, as provided on the Derbot board

The led functions on the ICD 3 are shown in Table 1.

LED	Color	Description
Power	Green	Lit when powered.
Active	Blue	Lit when power is first applied or when target is connected.
Status	Green	Lit when the debugger is operating normally – standby.
	Red	Lit when an operation has failed.
	Orange	Lit when the debugger is busy.

Table 1: LED Functions on the ICD 3

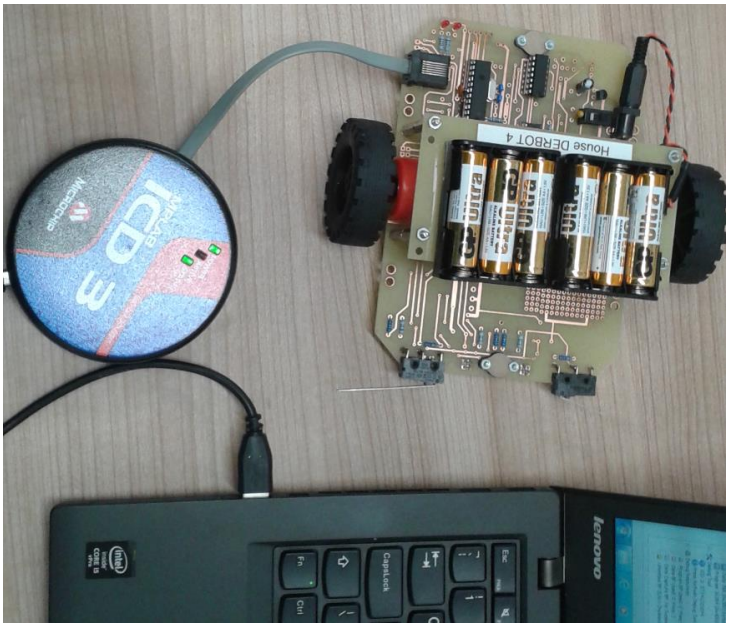

Figure 3: A "House Derbot" Connected to the ICD 3

2. Applying the ICD 3 as debugger

Use the “Blind Navigation” Program Example below. Create this as a project, selecting ICD 3 under **Select Tool**, and build in the normal way.

Connect the ICD 3 between computer and Derbot board, as shown in Figure 3; the board should be powered up and switched on. Set the ICD to operate initially in debug mode, by clicking **Debug > Debug Project**. This forces a new project build, and then automatically proceeds to download to the microcontroller. This takes a bit of time, and progress is reported as usual in the Output window. If the target system is powered and working, the ICD 3 should be able to identify the microcontroller, reporting its version number. The drop-down menu under **Debug** then becomes active, as does the toolbar, as seen in Tutorial Sheet 1.

You can now test the operation of the downloaded program, exactly as you did with the simulator in Tutorial Sheet 2. The big difference is that now you are actually controlling the program as it runs in the hardware. This is powerful indeed! Instead of simulating inputs however you can of course press the microswitches, and observe the response of the leds. If you single-step in the main loop, try pressing the microswitches in turn, and see the response of the Port values in the Watch window as you step. Unlike when using the simulator, it is actual values from the hardware that are being transmitted back to the display you are seeing. As with the simulator, breakpoints can be set in the program by clicking on any line.

3. Applying the ICD 3 as programmer

To program a product, the ICD 3 should be used in simple programmer mode. Exit the Debug session, and click **Run > Run Project.** Again MPLAB X will force a project Build, and will automatically connect to the ICD 3 and download. Now if you disconnect from the ICD 3, you should find your program runs – you have a stand-alone system!

Follow this with the “Blind Navigation” Program, appendix to this tutorial, and download and trial on one of the House Derbots.

4. Making changes to the Program

Try varying:

- The length of the delays,
- The number of times the leds flash in the Diagnostic function,
- The response to microswitch pushes, e.g. make an led light as long as a switch is pressed, instead of just holding a steady state.

Download the revised program each time to the Derbot board.

5. Line Following

Note the physical location of the reflective opto-sensors on the Derbot, and from the circuit diagram check their connection to the microcontroller. When a reflective object is paced in front of them, the sensors output a logic 0 to the microcontroller, when there is no object they output a logic 1.

Program a House Derbot to follow a white line on the trial track. You should be able to do this, with care, by adapting the Blind Navigation program.

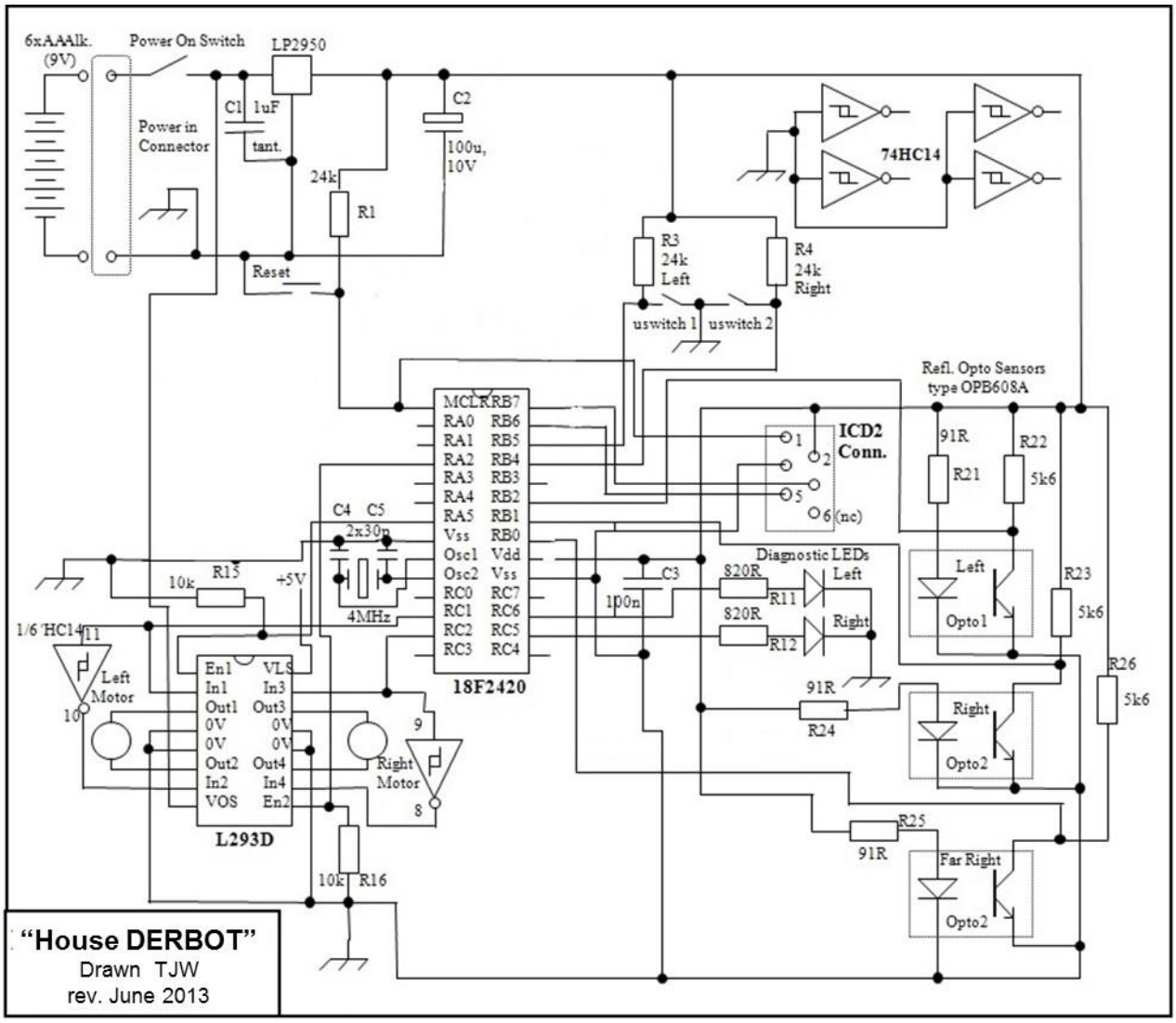

Figure 4: Derbot "Build Stage 2"

6. Troubleshooting

Refer to Reference 1. This also tells you, in Part 4 of its reference section, how you can use the little Test Interface Board (as seen in Figure 1) to conduct a self-test of the ICD 3.

References

1. MPLAB ICD-3 In-Circuit Debugger User's Guide, for MPLAB X IDE. Microchip. Doc. DS50002081B.

```
/*****************************************************************
Dbt_blind_Nav.                          Program Example 15.3 (rev.)
Derbot moves by "blind" navigation.
Moves forward, and reverses and turns on bump.
Fixed rate PWM applied to set reasonable speeds.
rev. 26.9.16 (for MPLABX)
*****************************************************************
Clock is 4MHz */

#include <xc.h>

//configuration settings. These take action at the moment of power-up, and can't change
#pragma config OSC = HS   //HS oscillator (for crystal use)
#pragma config PWRT = ON, BOREN = OFF  //power-up timer on, brown-out detect off
#pragma config WDT = OFF       //watchdog timer off
#pragma config LVP = OFF       //low voltage programming off
#pragma config PBADEN = OFF    //Port B A to D Enable is off

#define _XTAL_FREQ 4000000   //4 MHz clock frequency defined
//function prototypes
void initialise (void);
void diagnostic (void);
void leftmot_fwd(void);
void rtmot_fwd(void);
void rev_left(void);
void rev_rt(void);
void delay_100ms(char);

//main function
int main(void) {
   initialise ();
   diagnostic ();

//move microswitch states to diag leds
while(1) {
    leftmot_fwd();
    rtmot_fwd();
    if (PORTBbits.RB4 == 0)
        rev_left();
    if (PORTBbits.RB5 == 0)
        rev_rt();
    __delay_ms(100);
    }
}                             //end of main()

void initialise (void){
  //set port bits as ip (1) or op (0).
  TRISA = 0;   //set all Port A to output, pins 2 and 5 are motor enables
  TRISB = 0b00110000;   //bits 5 and 4 of Port B are inputs, microswitches
  TRISC = 0;            //Port C all output, includes PWM and diag leds
  ADCON1 = 0b00001111;  //Port A is set to digital i.o.
  //switch all outputs off, for safety
  PORTA = PORTB = PORTC = 0;
  //set up conditions required for PWM
  T2CON = 0b00000100;  //Timer 2 feeds PWM generator
  PR2 = 0xFF;          //PR2 determines PWM period, this is max value
  CCP1CON = 0b0001100; //select PWM action in "Capture/Compare/PWM" (CCP)
  CCP2CON = 0b0001100;
  }                    //end of function
```

```
void diagnostic (void){
        //flashes leds at start of program, to demonstrate prog launch
     PORTCbits.RC6 = 1;  //switch on led
     PORTCbits.RC5 = 0;  //switch off led
     delay_100ms(10);
     PORTCbits.RC6 = 0;  //switch off led
     PORTCbits.RC5 = 1;  //switch on led
     delay_100ms(10);
     PORTCbits.RC6 = 1;  //switch on led
     PORTCbits.RC5 = 0;  //switch off led
     delay_100ms(10);
     PORTCbits.RC6 = 0;  //switch off led
     PORTCbits.RC5 = 1;  //switch on led
     delay_100ms(10);
     PORTCbits.RC5 = 0;  //switch off led
   }       //end of function

void leftmot_fwd (void){
   CCPR2L = 196;     //this number sets the PWM ratio, to fast forward
   PORTAbits.RA5 = 1;
   }

void rtmot_fwd (void){
   CCPR1L = 196;    //this number sets the PWM ratio, to fast forward
   PORTAbits.RA2 = 1;
   }

void leftmot_rev (void){
   CCPR2L = 60;     //this number sets the PWM ratio, to medium backward
   PORTAbits.RA5 = 1;
   }

void rtmot_rev (void){
   CCPR1L = 60;    //this number sets the PWM ratio, to medium backward
   PORTAbits.RA2 = 1;
   }

void rev_rt (void){
    PORTCbits.RC6 = 1;  //switch on led
    PORTAbits.RA5 = 0;  //switch off motor enable
    delay_100ms(10);
    leftmot_rev();   //run backwards
    rtmot_rev();
    delay_100ms(10);
    leftmot_fwd();  //left forwards, so machine spins
    delay_100ms(10);
    PORTCbits.RC6 = 0; //switch off led
    }
void rev_left (void){
    PORTCbits.RC5 = 1;  //switch on led
    PORTAbits.RA2 = 0;  //switch off motor enable
    delay_100ms(10);
    leftmot_rev();
    rtmot_rev();
    delay_100ms(10);
    rtmot_fwd();
    delay_100ms(10);
    PORTCbits.RC5 = 0;
    }

void delay_100ms(char n) {
    while (n!=0){
      __delay_ms(100);
      n = n-1;
        }
    }
```