University of Florida
Department of Electrical and Computer Engineering
EEL 5666
Intelligent Machines Design Laboratory

‘Thing’

The Robotic Hand in a Box

Date: 4/25/01
Student Name: Chung Chov
TA: Scott Nortman
Rand Chandler
Instructor: A. A. Arroyo

Table of Contents

Abstract

‘Thing’ is an autonomous robot that roams around until it finds white blocks to pick up. It will appear to be nothing more than just a box on wheels. Once it finds a block, it will determine if it is black or white. If it is white, a door will open and a hand will come out to pick the object and go back in. If the block is black, it will move them out of the way or it will try and place a white block next to it if it has one. ‘Thing’ is also able to determine if the object is small enough to pick up. ‘Thing’ also has collision avoidance for large objects that it cannot pick up. The wheels will be controlled by hacked servos. The arm and hand will be controlled by servos. The IR sensors are used for the collision avoidance. The block detector will be a combination of IRs and photoresistors. The sensors, motors, and behaviors are all controlled by a PIC Chip microcontroller.

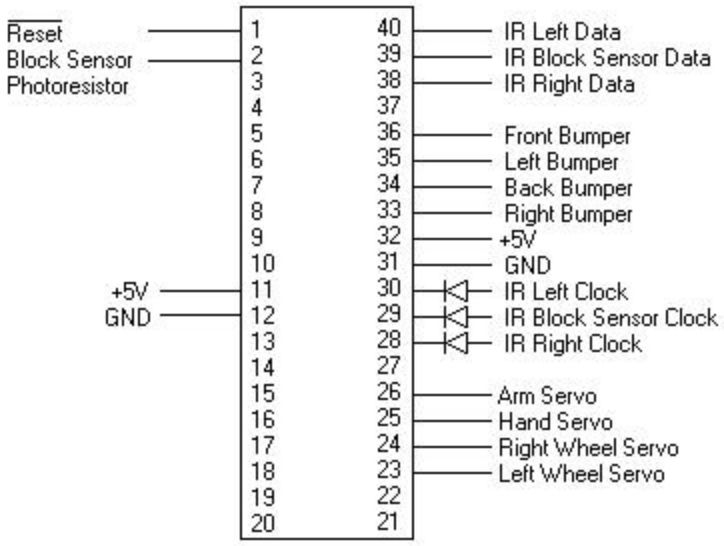
Introduction

The main purpose of any robot is to make life easier for humans. In the home, robots can clean, take care of the lawn, or provide security for the home. Since I am a messy person and I have a tendency to leave things lying around, I wanted a robot to pick things up for me and put them in place.

Integrated System

The microcontroller for ‘Thing’ is the Microchip PIC16F877. The PIC chip will have input coming in from the IR sensors, bump switches, and photoresistors. The PIC chip will determine which sensors are used. For instance, the photoresistor will not need to be on until ‘Thing’ has found an object and needs to determine what color it is. The PIC chip

controls when the servos (that moves the hand and arm around) are activated. ‘Thing’ will roam in its environment avoid objects that are bigger than the objects that it can pick up. The data from the IRs are used to determine the whether the object is small enough. Once it has found the object it can pick up, photoresistors determine wither or not the object is black or white. If it has found a white object a door will open up and hand come out to pick it up and retract with the door closing behind it. Bump switches surround the robot since the IRs are only on the front of the robot. Because of the ease of the PIC Chip, the sensors where directed connected to the PIC Chip with the exception of the three diodes need for the clock of the IRs.



Microchip PIC16F877

Mobile Platform

‘Thing’ will be made out of balsa wood cut out from the ‘T-tech’ machine in lab. Since ‘Thing’ is a hand in a box, the hand had to be designed first. To do this, prototype was created out of Popsicle sticks. This allowed for any mechanical problems to be worked out. Later the hand was drawn in AutoCAD and cut out using the T-Tech machine. Then the ‘box’ was designed around the hand and arm. This was cut out by hand since it was large enough to do so. Two problems occurred when the hand and arm were mounted on to the body. One problem was that the arm would not extend or retract with out hitting the front of the robot. This was solved by using a rubber band to hold the arm above the robot while it was retract or extended by the servo. However the rubber band only last a couple of stretches. A permanent fix to this would be a spring. Another problem was that when fully extended the robot tipped over. To solve this, the battery pack was placed at the back of the robot to counter balance the arm. Another problem that arose was when ‘Thing’ actually tried to pick something up. It did not have enough gripping force to hold onto anything. This was because the hand had fishing line to contract the joints of the finger. A partial fix to this was replacing the fishing line (which had some elasticity) with wires.

Actuation

‘Thing’ has two servos and two hacked servos. The hacked servos provide ‘Thing’ with mobility. One servo is connected to control wires attached to the fingers in the hand. When the servo turns it will shorten the control wires causing the fingers of the hand to curl, thus grab. The other servo is attached to the arm at the where it is connected to the

base of the robot. This servo extends and retracts the arm and hand. The PIC Chip uses one timer to control all the servos.

Sensors

The bump sensors are momentary switches connected to ground on pins on the PIC Chip that have internal pull up resistors. The IRs used on ‘Thing’ are the SHARP GP2D02. This is an all-in-one unit. It contains both the emitter and detector. Its range is 80 cm to 6 cm and about 100mm wide (figure 1). The pins for it are power, ground, clock, and data. The clock is pulled low for 70ms and then pulsed to read the 8 bit serial data. Two of these IRs will be used for collision avoidance. A third IR is used in the ‘block detector’ to find objects that are small enough to pick. By placing the collision avoidance IRs higher that IR in the block detector, small objects lower than the collision avoidance IRs will be missed but still detected by the block detector. The block detector is main sensor for ‘Thing.’ It is a combination of SHARP GP2D02, photoresistor, and two incandescent lamps used to detect white or black blocks. The photoresistor is in series with a 1kΩ resistor. The output from that is attached the analog input of the PIC Chip. At approximately 8cm away, the photoresistors can read the light reflecting off of the object. When the object is black, the light is about half the light read in the room. Because of this, it may be possible to read the light in any given room and still be able to detect if the block is white or not (e.g. self calibrating sensor).

Behavior

The robots main behavior is to avoid obstacles. While it is avoid obstacles, it searches for objects small enough to pick up. Once it finds an object it will go and pick it up off the ground if it is white. If the object is black it will try and move the object away. If however 'Thing' already has picked up an object it will place it next to the black object.

Experimental Layout and Results

To determine what kind of data the block detector would send, black and white objects were placed at varies distances and in different lighting environments.

Lighting	Normal		Dark		Bright	
Distance	Black	White	Black	White	Black	White
No object in front of sensor	**.27**	**.27**	**.19**	**.19**	**.72**	**.72**
30cm	.22	.25	.19	.18	.63	.75
25cm	.21	.24	.18	.16	.57	.75
20cm	.19	.22	.17	.15	.58	.76
15cm	.20	.21	.13	.10	.50	.75
10cm	.13	.22	.10	.14	.43	.70
8cm	**.14**	**.25**	**.08**	**.14**	**.37**	**.67**
5cm	.10	.31	.07	.23	.31	.66
1cm	.10	1.18	.09	1.04	.15	1.77

* Measurements in volts

80 cm ----------

30 cm----------

6 cm----------

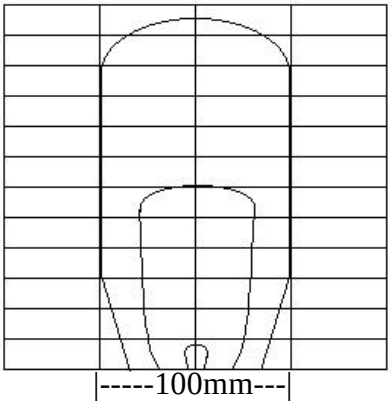


Fig 1

Conclusion

I found that it is one thing to say that your robot will do this and that and another for it to do so. Much of what my robot does was dictated to me by my robot. Instead on it doing this, it actually does that. Since I was using a different microprocessor then the most the class, I wrote most the code by myself with the help of Joshua Philips since had experience with this processor. I glad I used this chip since it provided something new for both the class and myself. One of the main problems I had with software was trying to have it actively search for objects instead of randomly running into them. I believe can fix this with time.

Documentation

Fred Martin, *The 6.270 Robot Builder's Guide* 2nd edition, MIT Media Lab, Cambridge, MA, 1992.

IMDL class: Instruction from Dr. Arroyo, Dr. Schwartz, Rand Chandler, and Scott Nortman

Appendices

```
************************************8chung.asm*********************************
       LIST   P=PIC16F877    ;
       include "p16f877.inc"  ;
       include"def_equ.inc"  ;       personal defines and equates
;*****************************************************************************
       org    0x00           ;       Set RESET vector
       goto   INIT           ;       to beginning of program.
       org    0x04           ;       set INTERRUPT vector to
       goto   INTRUPT        ;       beginning of INTERRUPT service routine
       org    0x05           ;       start of program
```

```
INIT    bcf     STATUS,RP0      ;       select bank 0
        bcf     STATUS,RP1      ;       select bank 0
        include"init01.inc"     ;       all reg and port settings in this file
;*****************************************************************************
;       MAIN PROGRAM
;*****************************************************************************
        bsf     IRFCLK          ;
        bsf     IRLCLK          ;
        bsf     IRRCLK          ;

        clrf    IRTEST          ;

        movlw   0x06            ;
        clrf    WHITE           ;
        bsf     IRR             ;
        bcf     IRF             ;

        movlw   0x00            ;
        movwf   IRLDATA         ;
        movwf   IRRDATA         ;
        movwf   IRFDATA         ;


        movlw   HANDOP          ;
        movwf   HANDPOS         ;

        movlw   RETRACT         ;
        movwf   ARMPOS          ;
        movlw   PWM2            ; initialize to move forward
        movwf   SPEEDL          ;
        movlw   PWM1            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     FORWARD         ;


        call    WAIT            ; wait 5 secs

        call    CALIB           ;
        bsf     INTCON,GIE      ; enable unmasked interrupts
;-----------------------------------------------------------------------------
;
;       LEFT MOTOR (FORWARD = SPEEDL  = 14 / SPEEDL = 0A)
;       RIGHT MOTOR (FORWARD = SPEEDR = 0A / SPEEDR = 14)
;
;-----------------------------------------------------------------------------


HERE    btfss   STOP            ;
        goto    FOR_C           ;
        goto    STOP_R          ;

FOR_C   btfss   FORWARD         ;
        goto    REV_C           ;
        goto    FORW_R          ;

REV_C   btfss   REVERSE         ;
        goto    LT_C            ;
        goto    REV_R           ;

LT_C    btfss   LTURN           ;
        goto    RT_C            ;
        goto    LT_R            ;

RT_C    btfss   RTURN           ;
        goto    ERR             ;
        goto    RT_R            ;

ERR     movlw   0x00            ;
        movwf   SPEEDR          ;
        movwf   SPEEDL          ;
```

```
        clrf    DIRECT          ;
        bsf     STOP            ;
        goto    HERE            ;
;-------------------------------------------------------------------------------
STOP_R  call    WAIT            ;
        call    CHCK            ;
        btfsc   WORB,0x00       ; 1 = white / 0 = black
        goto    BIB             ;
        goto    BIW             ;

BIB     movlw   HANDCL          ;
        movwf   HANDPOS         ;
        call    WAIT            ;
        movlw   EXTEND          ;
        movwf   ARMPOS          ;
        call    WAIT            ;
        movlw   HANDOP          ;
        movwf   HANDPOS         ;
        call    WAIT            ;
        movlw   RETRACT         ;
        movwf   ARMPOS          ;
        movlw   PWM1            ; if ball is black go in reverse
        movlw   SPEEDL          ;
        movlw   PWM2            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     REVERSE         ;
        goto    HERE            ;

BIW     movlw   EXTEND          ;
        movwf   ARMPOS          ;
        call    WAIT
        movlw   HANDCL          ; if ball is white make right turn
        movwf   HANDPOS         ;
        call    WAIT            ;
        movlw   RETRACT         ;
        movwf   ARMPOS          ;
        movlw   PWM2            ;
        movwf   SPEEDL          ;
        movlw   PWM1            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     RTURN           ;
        goto    HERE            ;
;-------------------------------------------------------------------------------

FORW_R  btfss   FBUMP           ;
        goto    BUMPEDF         ;

        btfss   LBUMP           ;
        goto    BUMPEDL         ;

        btfss   RBUMP           ;
        goto    BUMPEDR         ;

        btfss   BBUMP           ;
        goto    BUMPEDB         ;

        movlw   0x6e            ; 9inches
        subwf   IRLDATA, 0      ;

        btfsc   STATUS, 0x00    ;
        goto    FLNEAR          ; IRLDATA - 8C > 0
        goto    FLFAR           ; IRLDATA - 8C < 0

FLNEAR  movlw   0x8c            ; to get here the Left IR sensed an object
        subwf   IRRDATA, 0      ;

        btfsc   STATUS, 0x00    ;
        goto    BOTH_IR         ; IRRDATA - 8C > 0
```

```
        goto    OBJECTL         ; IRRDATA - 8C < 0

FLFAR   movlw   0x8c            ; to get here the Left IR did not sense an object
        subwf   IRRDATA, 0      ;

        btfsc   STATUS, 0x00    ;
        goto    OBJECTR         ; IRRDATA - 8C > 0
        goto    NOOBJ           ; IRRDATA - 8C < 0

BOTH_IR movlw   PWM1            ;
        movwf   SPEEDL          ;
        movlw   PWM2            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     REVERSE         ;
        goto    HERE            ;

OBJECTL movlw   PWM2            ;
        movwf   SPEEDL          ;
        movlw   PWM2            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     RTURN           ;
        goto    HERE            ;

OBJECTR movlw   PWM1            ;
        movwf   SPEEDL          ;
        movlw   PWM1            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     LTURN           ;
        goto    HERE            ;

NOOBJ   movlw   d8cm            ;
        subwf   IRFDATA, 0      ;

        btfsc   STATUS, 0x00    ;
        goto    BALL            ; IRFDATA - d8cm > 0
        goto    NOBALL          ; IRFDATA - d8cm < 0

BALL    clrf    SPEEDR          ;
        clrf    SPEEDL          ;
        clrf    DIRECT          ;
        bsf     STOP            ;
        goto    HERE            ;

NOBALL  movlw   PWM2            ;
        movwf   SPEEDL          ;
        movlw   PWM1            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     FORWARD         ;
        goto    HERE            ;
;-------------------------------------------------------------------------------------
REV_R   movf    TMR0,0x00       ;
        movwf   DELAYT          ;
        bcf     STATUS,  0x00   ;
        rrf     DELAYT, 1       ;
        bcf     STATUS,  0x00   ;
        rrf     DELAYT, 0       ;
        movwf   TEMP1           ;
        movwf   TEMP2           ;
        movwf   TEMP3           ;

DELAYB  btfss   BBUMP           ;
        goto    BUMPEDB         ;

        btfss   LBUMP           ;
        goto    BUMPEDL         ;

        btfss   RBUMP           ;
```

```
        goto    BUMPEDR         ;

        btfss   FBUMP           ;
        goto    BUMPEDF         ;

        decfsz  TEMP3,0x01      ;
        goto    DELAYB          ;
        movwf   TEMP3           ;
        decfsz  TEMP2,0x01      ;
        goto    DELAYB          ;
        movwf   TEMP2           ;
        decfsz  TEMP1,0x01      ;
        goto    DELAYB          ;

        movf    TMR0,0x00       ; get a random # from TIMER 0
        sublw   0x7F            ;
        btfss   STATUS,0x00     ;
        goto    RP_RL           ;
        goto    RP_RR           ;

RP_RL   movlw   PWM1            ;
        movwf   SPEEDL          ;
        movlw   PWM1            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     LTURN           ;
        goto    HERE            ;

RP_RR   movlw   PWM2            ;
        movwf   SPEEDL          ;
        movlw   PWM2            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     RTURN           ;
        goto    HERE            ;
;-------------------------------------------------------------------------------------
LT_R    movf    TMR0,0x00       ;
        movwf   DELAYT          ;
        bcf     STATUS,  0x00   ;
        rrf     DELAYT, 1       ;
        bcf     STATUS,  0x00   ;
        rrf     DELAYT, 0       ;
        movwf   TEMP1           ;
        movwf   TEMP2           ;
        movwf   TEMP3           ;

DELAYL  btfss   RBUMP           ;
        goto    BUMPEDR         ;

        btfss   FBUMP           ;
        goto    BUMPEDF         ;

        btfss   BBUMP           ;
        goto    BUMPEDB         ;

        btfss   LBUMP           ;
        goto    BUMPEDL         ;

        decfsz  TEMP3,0x01      ;
        goto    DELAYL          ;
        movwf   TEMP3           ;
        decfsz  TEMP2,0x01      ;
        goto    DELAYL          ;
        movwf   TEMP2           ;
        decfsz  TEMP1,0x01      ;
        goto    DELAYL          ;

        movlw   PWM2            ;
        movwf   SPEEDL          ;
        movlw   PWM1            ;
        movwf   SPEEDR          ;
```

```
        clrf    DIRECT          ;
        bsf     FORWARD         ;
        goto    HERE            ;


;-------------------------------------------------------------------------------------
RT_R    movf    TMR0,0x00       ;
        movwf   DELAYT          ;
        bcf     STATUS,  0x00   ;
        rrf     DELAYT, 1       ;
        bcf     STATUS,  0x00   ;
        rrf     DELAYT, 0       ;
        movwf   TEMP1           ;
        movwf   TEMP2           ;
        movwf   TEMP3           ;

DELAYR  btfss   LBUMP           ;
        goto    BUMPEDL         ;

        btfss   FBUMP           ;
        goto    BUMPEDF         ;

        btfss   BBUMP           ;
        goto    BUMPEDB         ;

        btfss   RBUMP           ;
        goto    BUMPEDR         ;

        decfsz  TEMP3,0x01      ;
        goto    DELAYR          ;
        movwf   TEMP3           ;
        decfsz  TEMP2,0x01      ;
        goto    DELAYR          ;
        movwf   TEMP2           ;
        decfsz  TEMP1,0x01      ;
        goto    DELAYR          ;

        movlw   PWM2            ;
        movwf   SPEEDL          ;
        movlw   PWM1            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     FORWARD         ;
        goto    HERE            ;
;-------------------------------------------------------------------------------------
;       BUMP ROUNTINES
;-------------------------------------------------------------------------------------
BUMPEDRmovlw    PWM2            ; turn right slightly
        movwf   SPEEDL          ;
        movlw   PWM2            ;
        movwf   SPEEDR          ;
        movlw   0x41            ;
        movwf   DELAYT          ;
        call    DELAY2          ;
        movlw   PWM1            ; go reverse
        movwf   SPEEDL          ;
        movlw   PWM2            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     REVERSE         ;
        goto    HERE            ;

BUMPEDLmovlw    PWM1            ; turn left slightly
        movwf   SPEEDL          ;
        movlw   PWM1            ;
        movwf   SPEEDR          ;
        movlw   0x41            ;
        movwf   DELAYT          ;
        call    DELAY2          ;
        movlw   PWM1            ; go reverse
        movwf   SPEEDL          ;
```

```
        movlw   PWM2            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     REVERSE         ;
        goto    HERE            ;

BUMPEDF movlw   PWM1            ; go reverse
        movwf   SPEEDL          ;
        movlw   PWM2            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     REVERSE         ;
        goto    HERE            ;

BUMPEDB movlw   PWM2            ; go forward
        movwf   SPEEDL          ;
        movlw   PWM1            ;
        movwf   SPEEDR          ;
        clrf    DIRECT          ;
        bsf     FORWARD         ;
        goto    HERE            ;


;-------------------------------------------------------------------------------
;       INTERRUPT ROUNTINES
;-------------------------------------------------------------------------------
        include "intrupt.inc"  ; interrupt rountine
;-------------------------------------------------------------------------------
;       SUBROUTINES
;-------------------------------------------------------------------------------
        include "subrtn.inc"   ; subroutines
;-------------------------------------------------------------------------------
        end;



************************************def_equ.inc*********************************

;*****************************************************************************
;
;       LABELS
;
;*****************************************************************************
#define  IRLDAT PORTB,0x07     ;1
#define  IRFDAT PORTB,0x06     ;1
#define  IRRDAT PORTB,0x05     ;1

#define  FBUMP  PORTB,0x03     ;1
#define  LBUMP  PORTB,0x02     ;1
#define  BBUMP  PORTB,0x01     ;1
#define RBUMP   PORTB,0x00     ;1

#define IRLCLK  PORTD,0x07     ;0
#define IRFCLK  PORTD,0x06     ;0
#define IRRCLK  PORTD,0x05     ;0


#define EYEL    PORTA,0x00     ;
#define EYEM    PORTA,0x01     ;
#define EYER    PORTA,0x02     ;

#define FOUND   LOOK,0x00      ;



#define SERVOL  PORTC,0x04     ;
#define SERVOR  PORTC,0x05     ;
#define HAND    PORTC,0x06     ;
```

```
#define ARM     PORTC,0x07



#define STOP   DIRECT, 0x00   ; 0000 0001
#define FORWARD        DIRECT, 0x01   ; 0000 0010
#define REVERSE        DIRECT, 0x02   ; 0000 0100
#define LTURN  DIRECT, 0x03   ; 0000 1000
#define RTURN  DIRECT, 0x04   ; 0001 0000
#define SYNC5  DIRECT, 0x05   ; 0010 0000
#define SYNC6  DIRECT, 0x06   ; 0100 0000
#define NEWSPD DIRECT, 0x07   ; 1000 0000

#define LWHEEL SPDCTRL, 0x00  ; 0000 0001
#define RWHEEL  SPDCTRL, 0x01  ; 0000 0010
#define HANDCTL        SPDCTRL, 0x02  ; 0000 0100
#define ARMCTL  SPDCTRL, 0x03  ; 0000 1000

#define IRREAD  IRTEST, 0x00   ; 0000 0001
#define IRR     IRTEST, 0x01   ; 0000 0010
#define IRF     IRTEST, 0x02   ; 0000 0100
#define IRL     IRTEST, 0x03   ; 0000 1000

;-----------------------------------timer 1
#define PWM2   0x0a           ; 1.022ms at 1:2 prescalar -100%
#define PWM125 0x0d           ; -50%
#define PWM15  0x0f           ; 1.533ms at 1:2 prescalar 0%
#define PWM175  0x11          ; 50%
#define PWM1   0x14           ; 2.044ms at 1:2 prescalar 100%
#define PWM18  0xb0           ; 17.9872ms at 1:2 prescalar
#define PWM185 0xb5           ; 18.4982ms at 1:2 prescalar
#define PWM19  0xba           ; 19.0092ms at 1:2 prescalar
#define PWM20   0xc4          ; 20.0312ms

;-----------------------------------DISTANCE DEFINATION
#define d8cm    0xe6          ; 8cm
#define VFAR    0x8c          ; 30cm
#define FAR     0x92
#define NEAR    0xa5
#define VNEAR   0xb9
#define VVNEAR  0xc5

#define HANDOP  0x0a          ;
#define HANDCL  0x14          ;
#define EXTEND  0x05          ;
#define RETRACT 0x19          ;

COUNT   equ     0x24          ;

ARMPOS  equ     0x28          ;
ARMCNT  equ     0x29          ;
HANDCNT equ     0x2a          ;
HANDPOS equ     0x2b          ;
TEMP1   equ     0x2c          ;
TEMP2   equ     0x2d          ;
TEMP3   equ     0x2e          ;
TEMP4   equ     0x2f          ;
RWT     equ     0x30          ;
LWTH    equ     0x31          ;
LWTL    equ     0x32          ;

TM2CNT  equ     0x33          ;
IRRDATA equ     0x34          ;
IRLDATA equ     0x35          ;
IRFDATA equ     0x36          ;
IRTEST  equ     0x37          ;

DIRECT  equ     0x38          ;
```

```
SPDCTRL equ     0x39            ;
SPEEDL  equ     0x3a            ;
SPEEDR  equ     0x3b            ;
PERIOD  equ     0x3c            ;


SPDTL   equ     0x3d            ;
SPDTR   equ     0x3e            ;
LOOKR   equ     0x3f            ;
LOOKLT  equ     0x40            ;
LOOKMT  equ     0x41            ;
LOOKRT  equ     0x42            ;
WHITE   equ     0x43            ;
WORB    equ     0x44            ;



LOOK    equ     0x45            ;
DELAYT  equ     0x46

TABLE   equ     0x50            ;
ENDTBL  equ     0x68            ;

W_TEMP  equ     0x70            ; temporary register for w available in all banks
S_TEMP  equ     0x71            ; temp reg for the status reg (all banks)


************************************init01.inc*************************************


;*********************************************************************************
;
;       INTIALIZE REGISTERS / PORTS / INTERUPTS
;
;*********************************************************************************
;*********************************************************************************
;
;       INTERRUPT CONTROL REGISTER
;
;       initialize to:
;
;       GLOBAL INTERRUPT ENABLE              : DISABLED     0
;       PERIPHERAL INTERRUPT ENABLE   : ENABLED      1
;       TIMER0 OVERFLOW INTERRUPT     : ENABLED      1
;       EXTERNAL INTERRUPT PIN        : DISABLED     0
;       PORTB INTERRUPT ON CHANGE     : DISABLED     0
;       LOWER 3 BITS ARE FLAGS               : DONT CARE    xxx
;
;*********************************************************************************
        movlw   0x60            ;       initialize interrupt control reister
        movwf   INTCON          ;
;*********************************************************************************
;*********************************************************************************
;
;       OPTION REGISTER
;
;       initialize to:
;
;       PORTB INTERNAL PULLUP RESISTORS      : ENABLED      0
;       EXTERNAL INTERRUPT EDGE SELECT       : RISING EDGE  1
;       TIMER0 CLOCK SOURCE SELECT    : Fosc/4       0
;       TIMER0 SOURCE EDGE SELECT     : low to high  0
;       PRESCALER ASSIGNMENT          : TIMER0       0
;       LOWER 3 BITS ARE PRESCALER    : 1/2          000
;
;*********************************************************************************
        bsf     STATUS,RP0      ;       select bank 1
        bcf     STATUS,RP1      ;       select bank 1
        movlw   0x40            ;
        movwf   OPTION_REG      ;       initialize option register
```

```
;*******************************************************************************
;
;       PERIPHERAL INTERRUPT REGISTER 1
;
;       initialize to:
;
;       PARALLEL SLAVE PORT R/W INTERRUPT ENABLE     : DISABLED     0
;       A/D CONVERTER INTERRUPT ENABLE                      : ENABLED      1
;       USART RECIEVE INTERRUPT ENABLE                      : DISABLED     0
;       USART TRANSMIT INTERRUPT ENABLE                     : DISABLED     0
;       SYNCHRONOUS SERIAL PORT INTERRUPT ENABLE     : DISABLED     0
;       CAPTURE/COMPARE 1 INTERRUPT ENABLE           : DISABLED     0
;       TIMER2 TO PR2 MATCH INTERRUPT                : ENABLED      1
;       TIMER1 OVERFLOW INTERRUPT ENABLE             : ENABLED      1
;
;*******************************************************************************
       movlw   0x43    ;
       movwf   PIE1    ; initialize PIE1 register
;*******************************************************************************
;
;       PERIPHERAL INTERRUPT REGISTER 2
;
;       initialize to:
;
;       BIT 7-5                              : ALWAYS 000
;       EEPROM WRITE INTERRUPT ENABLE        : DISABLED 0
;       BUS COLLISION INTERRUPT ENABLE               : DISABLED 0
;       BIT 2-1                              : ALWAYS 00
;       CAPTURE COMPARE 2 INTERRUPT ENABLE   : DISABLED 0
;
;*******************************************************************************
       movlw   0x00;
       movwf   PIE2; initialize PIE2 register
;*******************************************************************************
;
;       PORT A (6 bits wide)
;
;       For all ports data direction registers (TRISx) a 1 makes the pin an input
;
;       Analog input 0 thru 4 are on this port along with
;       TIMER0's external clock input pin
;       Assign analog pins here or in port e
;       All other pins are inputs
;
;*******************************************************************************
       movlw   0xff            ;
       movwf   TRISA           ; data direction register for PORTA
;*******************************************************************************
;
;       PORT B (8 bits wide)
;
;       General purpose in-out.
;       All pins will be used as inputs.
;
;*******************************************************************************
       movlw   0xff            ;1111 1111
       movwf   TRISB           ; data direction register for PORTB
;*******************************************************************************
;
;       PORT C (8 bits wide)
;
;       General purpose in-out, PWM output, SPI, USART, CAPTURE 1&2
;
;
;*******************************************************************************
       movlw   0x0f            ; 1110 0000
       movwf   TRISC           ; data direction register for PORTC
;*******************************************************************************
;
;       PORT D (8 bits wide)
;
```

```
;       Port D can be used as a parallel slave port or general in-out
;
;*****************************************************************************
        movlw   0x00            ; 0000 0000
        movwf   TRISD           ; data direction register for PORTD
;*****************************************************************************
;
;       PORT E (3 bits wide)
;
;       Port E can be used as general purpose in-out, Analog input 7 thru 5,
;       or as the control bits for parallel slave port mode.
;
;*****************************************************************************
        movlw   0x0F            ;
        movwf   TRISE           ; data direction register for PORTE
;*****************************************************************************
;
;       ANALOG-TO-DIGITAL REGISTER 1
;
;       This register selects the port configurations for analog or digital
;       input and selects the values for Vref+
;       This register also right/left justifies the A/D result register
;       The result is 10 bits wide in a 16 bit wide register.
;
;       The pins will be used as follows:
;
;       PORT E A/D pins = Digital I/O
;       PORT A  PIN0    = Digital I/O
;               PIN1    = Digital I/O
;               PIN2    = Digital I/O
;               PIN3    = Digital I/O
;               PIN4    = Digital I/O
;               PIN5    = Digital I/O
;*****************************************************************************
        movlw   0x00            ;
        movwf   ADCON1          ; config A/D
;*****************************************************************************
;
;       ANALOG-TO-DIGITAL REGISTER 0
;
;       This register is mainly used to start and stop the conversions
;       and select which analog input is to be used for the next conversion.
;
;       note:   The required pause before the next acquisition can begin is
;               2*the value selected in bits 7 and 6 - in this case Fosc/32
;               would yield about .8us @ 20Mhz
;
;*****************************************************************************
        bcf     STATUS,RP0      ;
        bcf     STATUS,RP1      ; SELECT BANK 0
        bsf     ADCON0,0x07     ;
        bcf     ADCON0,0x06     ; conversion clock set to Fosc/32
        bcf     ADCON0,0x05     ;
        bcf     ADCON0,0x04     ;
        bcf     ADCON0,0x03     ; select analog input 0
        bcf     ADCON0,0x00     ; turn off the A/D converter (set this bit to enable)


;*****************************************************************************
;
;       TIMER2  Setup
;
;*****************************************************************************
        movlw   0x7F            ; prescale of 16
        movwf   T2CON           ; post scale of 16
;*****************************************************************************
;
;       TIMER1 SETUP
;
;       Timer1 will be used to generate the pulses for the motors
;
```

```
;       prescale = 1:1
;
;*****************************************************************************
        movlw   0x01            ;       0011 0101
        movwf   T1CON           ;


*****************************************subrtn.inc*******************************

;*****************************************************************************
;*****************************************************************************
;
;       SUBROUTINES
;
;*****************************************************************************
;*****************************************************************************


DRIVE   movf    SPEEDL, 0       ;
        movwf   SPDTL           ;

        movf    SPEEDR, 0       ;
        movwf   SPDTR           ;

        movf    HANDPOS, 0      ;
        movwf   HANDCNT         ;

        movlw   PWM20           ;
        movwf   PERIOD          ;

        movf    ARMPOS, 0       ;
        movwf   ARMCNT          ;

        return                  ;



;-----------------------------------------------------------------------------
DELAY   movlw   0xFA            ;
        movwf   COUNT           ; 250 cycles
        decfsz  COUNT,0x01      ;
        goto    $ - 1           ;
        return                  ;
;-----------------------------------------------------------------------------


CALIB   bcf     ADCON0, 0x03    ;
        bcf     ADCON0, 0x04    ;
        bcf     ADCON0, 0x05    ;

        bsf     ADCON0, 0x00    ;
        movlw   0x64            ;
        movwf   TEMP1           ;
        decfsz  TEMP1, 1        ;
        goto    $ - 1           ;

        bsf     ADCON0, 0x02    ;

        btfsc   ADCON0, 0x02    ;
        goto    $ - 1           ;

        movf    ADRESH, 0       ;
        movwf   WHITE           ;

        return                  ;
;-------------------------------------------------------------
CHCK    bcf     ADCON0, 0x03    ;
        bcf     ADCON0, 0x04    ;
```

```
        bcf     ADCON0, 0x05   ; select a/d 0

        bsf     ADCON0, 0x02   ; start a conversion
        btfsc   ADCON0, 0x02   ; poll for conversion finished
        goto    $ - 1          ;

        movf    ADRESH,0x00    ; grab conversion results
        subwf   WHITE, 0       ;
        btfss   STATUS,0x00    ; if a borrow occurs then ball is white (WHITE - W_reg < 0)
        goto    NOTIT          ; this opcode is skipped if the ball is white
        bsf     WORB,0x00      ;this is executed is ball is white
        return                 ;
NOTIT   bcf     WORB,0x00      ; ball is not white
        return                 ;
;-------------------------------------------------------



WAIT    movlw   0x92           ; WAIT 5 secs
        movwf   TEMP1          ;
        movwf   TEMP2          ;
        movwf   TEMP3          ;

FOR1    decfsz  TEMP3,0x01     ;
        goto    $ -1           ;
        movwf   TEMP3          ;
        decfsz  TEMP2,0x01     ;
        goto    FOR1           ;
        movwf   TEMP2          ;
        decfsz  TEMP1,0x01     ;
        goto    FOR1           ;
        return                 ;
;-------------------------------------------------------

DELAY2  movf    DELAYT,0x00    ; WAIT VARIABLE TIME
        movwf   TEMP1          ;
        movwf   TEMP2          ;
        movwf   TEMP3          ;

DELAYF  decfsz  TEMP3,0x01     ;
        goto    $ -1           ;
        movwf   TEMP3          ;
        decfsz  TEMP2,0x01     ;
        goto    DELAYF         ;
        movwf   TEMP2          ;
        decfsz  TEMP1,0x01     ;
        goto    DELAYF         ;
        return                 ;


**********************************intrupt.inc*****************************************

;*******************************************************************************
;
;       INTERRUPT ROUTINE
;
;*******************************************************************************
INTRUPT movwf   W_TEMP         ; save w reg contents

;-------------------------------------------------------------------------------------------
---------

        btfss   PIR1,TMR1IF    ; Check if Timer 1 interrupt
        goto    NEXT1          ;
        bcf     PIR1,TMR1IF    ; clear flag

; code for timer 1 int goes here

        goto    INT_END        ;
```

```
;-------------------------------------------------------------------------------------
---------

NEXT1   btfss   INTCON,0x02    ; Check if Timer 0 interrupt
        goto    NEXT2           ;
        bcf     INTCON,0x02    ; clear flag


LWHL    btfss   LWHEEL          ;
        decfsz  SPDTL, 1        ;
        goto    RWHL            ;
        bcf     SERVOL          ;
        bsf     LWHEEL          ;
        goto    RWHL            ;

RWHL    btfss   RWHEEL          ;
        decfsz  SPDTR, 1        ;
        goto    GRAB            ;
        bcf     SERVOR          ;
        bsf     RWHEEL          ;
        goto    SERCYC          ;

GRAB    btfss   HANDCTL         ;
        decfsz  HANDCNT, 1      ;
        goto    REACH           ;
        bcf     HAND            ;
        bsf     HANDCTL         ;
        goto    SERCYC          ;

REACH   btfss   ARMCTL          ;
        decfsz  ARMCNT, 1       ;
        goto    SERCYC          ;
        bcf     ARM             ;
        bsf     ARMCTL          ;
        goto    SERCYC          ;


SERCYC  decfsz  PERIOD, 1       ;
        goto    INT_END         ;
        call    DRIVE
        bsf     SERVOL          ;
        bsf     SERVOR          ;
        bsf     HAND            ;
        bsf     ARM             ;
        bcf     LWHEEL          ;
        bcf     RWHEEL          ;
        bcf     HANDCTL         ;
        bcf     ARMCTL          ;









        goto    INT_END         ;

;-------------------------------------------------------------------------------------
---------

NEXT2   btfss   PIR1,TMR2IF     ; Check if Timer 2 interrupt
        goto    NEXT3           ;
        bcf     PIR1,TMR2IF     ; clear flag

        btfsc   IRREAD          ; check if measurement has been initiated
        goto    TESTOK          ; if yes go check if measurement is finished else proceed
with init

        decfsz  TM2CNT,0x01     ; wait for 70 ms before reading I.R.
```

```
        goto    INT_END         ;

        movlw   0x06            ; reset 70 ms counter
        movwf   TM2CNT          ;



        btfsc   IRR             ; Check which IR to start read
        goto    RIGHTIR         ;
        btfsc   IRF             ;
        goto    FRONTIR         ;
        goto    LEFTIR          ;

TESTOK  btfsc   IRR             ; Check which IR to begin read
        goto    RIGHTOK         ;
        btfsc   IRF             ;
        goto    FRONTOK         ;
        goto    LEFTOK          ;

;_____________________________________________________________________________________
RIGHT IR

RIGHTIR bcf     IRRCLK          ; make clock signal low to initiate a measurement
        btfsc   IRRDAT          ; check for data signal to go low (ack that a measurement
is in progress)
        goto    $ - 1           ;

        bsf     IRREAD          ; set flag to indicate a measurement has been started
        goto    INT_END         ;

RIGHTOK btfss   IRRDAT          ; check for measurement complete
        goto    INT_END         ; else end interrupt so other routines may continue
processing

        bsf     IRRCLK          ; start bit
        call    DELAY           ;
        bcf     IRRCLK          ;
        call    DELAY           ;

RIR7    bsf     IRRCLK          ; bit 7
        btfss   IRRDAT          ;
        goto    RIRLO7          ;
        goto    RIRHI7          ;
RIRLO7  bcf     IRRDATA,0x07    ;
        goto    RIR6            ;
RIRHI7  bsf     IRRDATA,0x07    ;
        goto    RIR6            ;

RIR6    call    DELAY           ; bit 6
        bcf     IRRCLK          ;
        call    DELAY           ;
        bsf     IRRCLK          ;
        btfss   IRRDAT          ;
        goto    RIRLO6          ;
        goto    RIRHI6          ;
RIRLO6  bcf     IRRDATA,0x06    ;
        goto    RIR5            ;
RIRHI6  bsf     IRRDATA,0x06    ;
        goto    RIR5            ;

RIR5    call    DELAY           ; bit 5
        bcf     IRRCLK          ;
        call    DELAY           ;
        bsf     IRRCLK          ;
        btfss   IRRDAT          ;
        goto    RIRLO5          ;
        goto    RIRHI5          ;
RIRLO5  bcf     IRRDATA,0x05    ;
        goto    RIR4            ;
RIRHI5  bsf     IRRDATA,0x05    ;
        goto    RIR4            ;
```

```
RIR4    call    DELAY           ; bit 4
        bcf     IRRCLK          ;
        call    DELAY           ;
        bsf     IRRCLK          ;
        btfss   IRRDAT          ;
        goto    RIRLO4          ;
        goto    RIRHI4          ;
RIRLO4  bcf     IRRDATA,0x04    ;
        goto    RIR3            ;
RIRHI4  bsf     IRRDATA,0x04    ;
        goto    RIR3            ;

RIR3    call    DELAY           ; bit 3
        bcf     IRRCLK          ;
        call    DELAY           ;
        bsf     IRRCLK          ;
        btfss   IRRDAT          ;
        goto    RIRLO3          ;
        goto    RIRHI3          ;
RIRLO3  bcf     IRRDATA,0x03    ;
        goto    RIR2            ;
RIRHI3  bsf     IRRDATA,0x03    ;
        goto    RIR2            ;

RIR2    call    DELAY           ; bit 2
        bcf     IRRCLK          ;
        call    DELAY           ;
        bsf     IRRCLK          ;
        btfss   IRRDAT          ;
        goto    RIRLO2          ;
        goto    RIRHI2          ;
RIRLO2  bcf     IRRDATA,0x02    ;
        goto    RIR1            ;
RIRHI2  bsf     IRRDATA,0x02    ;
        goto    RIR1            ;

RIR1    call    DELAY           ; bit 1
        bcf     IRRCLK          ;
        call    DELAY           ;
        bsf     IRRCLK          ;
        btfss   IRRDAT          ;
        goto    RIRLO1          ;
        goto    RIRHI1          ;
RIRLO1  bcf     IRRDATA,0x01    ;
        goto    RIR0            ;
RIRHI1  bsf     IRRDATA,0x01    ;
        goto    RIR0            ;

RIR0    call    DELAY           ; bit 0
        bcf     IRRCLK          ;
        call    DELAY           ;
        bsf     IRRCLK          ;
        btfss   IRRDAT          ;
        goto    RIRLO0          ;
        goto    RIRHI0          ;
RIRLO0  bcf     IRRDATA,0x00    ;
        bcf     IRR             ;
        bsf     IRF             ;
        goto    IR_END          ;
RIRHI0  bsf     IRRDATA,0x00    ;
        bcf     IRR             ;
        bsf     IRF             ;
        goto    IR_END          ;


;_____________________________________________________________________________________
FRONT IR
FRONTIR bcf     IRFCLK          ; make clock signal low to initiate a measurement
        btfsc   IRFDAT          ; check for data signal to go low (ack that a measurement
is in progress)
```

```
        goto    $ - 1           ;

        bsf     IRREAD          ; set flag to indicate a measurement has been started
        goto    INT_END         ;

FRONTOK btfss   IRFDAT          ; check for measurement complete
        goto    INT_END         ; else end interrupt so other routines may continue
processing

        bsf     IRFCLK          ; start bit
        call    DELAY           ;
        bcf     IRFCLK          ;
        call    DELAY           ;

FIR7    bsf     IRFCLK          ; bit 7
        btfss   IRFDAT          ;
        goto    FIRLO7          ;
        goto    FIRHI7          ;
FIRLO7  bcf     IRFDATA,0x07    ;
        goto    FIR6            ;
FIRHI7  bsf     IRFDATA,0x07    ;
        goto    FIR6            ;

FIR6    call    DELAY           ; bit 6
        bcf     IRFCLK          ;
        call    DELAY           ;
        bsf     IRFCLK          ;
        btfss   IRFDAT          ;
        goto    FIRLO6          ;
        goto    FIRHI6          ;
FIRLO6  bcf     IRFDATA,0x06    ;
        goto    FIR5            ;
FIRHI6  bsf     IRFDATA,0x06    ;
        goto    FIR5            ;

FIR5    call    DELAY           ; bit 5
        bcf     IRFCLK          ;
        call    DELAY           ;
        bsf     IRFCLK          ;
        btfss   IRFDAT          ;
        goto    FIRLO5          ;
        goto    FIRHI5          ;
FIRLO5  bcf     IRFDATA,0x05    ;
        goto    FIR4            ;
FIRHI5  bsf     IRFDATA,0x05    ;
        goto    FIR4            ;

FIR4    call    DELAY           ; bit 4
        bcf     IRFCLK          ;
        call    DELAY           ;
        bsf     IRFCLK          ;
        btfss   IRFDAT          ;
        goto    FIRLO4          ;
        goto    FIRHI4          ;
FIRLO4  bcf     IRFDATA,0x04    ;
        goto    FIR3            ;
FIRHI4  bsf     IRFDATA,0x04    ;
        goto    FIR3            ;

FIR3    call    DELAY           ; bit 3
        bcf     IRFCLK          ;
        call    DELAY           ;
        bsf     IRFCLK          ;
        btfss   IRFDAT          ;
        goto    FIRLO3          ;
        goto    FIRHI3          ;
FIRLO3  bcf     IRFDATA,0x03    ;
        goto    FIR2            ;
FIRHI3  bsf     IRFDATA,0x03    ;
        goto    FIR2            ;
```

```
FIR2    call    DELAY           ; bit 2
        bcf     IRFCLK          ;
        call    DELAY           ;
        bsf     IRFCLK          ;
        btfss   IRFDAT          ;
        goto    FIRLO2          ;
        goto    FIRHI2          ;
FIRLO2  bcf     IRFDATA,0x02    ;
        goto    FIR1            ;
FIRHI2  bsf     IRFDATA,0x02    ;
        goto    FIR1            ;

FIR1    call    DELAY           ; bit 1
        bcf     IRFCLK          ;
        call    DELAY           ;
        bsf     IRFCLK          ;
        btfss   IRFDAT          ;
        goto    FIRLO1          ;
        goto    FIRHI1          ;
FIRLO1  bcf     IRFDATA,0x01    ;
        goto    FIR0            ;
FIRHI1  bsf     IRFDATA,0x01    ;
        goto    FIR0            ;

FIR0    call    DELAY           ; bit 0
        bcf     IRFCLK          ;
        call    DELAY           ;
        bsf     IRFCLK          ;
        btfss   IRFDAT          ;
        goto    FIRLO0          ;
        goto    FIRHI0          ;
FIRLO0  bcf     IRFDATA,0x00    ;
        bcf     IRR             ;
        bcf     IRF             ;
        goto    IR_END          ;
FIRHI0  bsf     IRFDATA,0x00    ;
        bcf     IRR             ;
        bcf     IRF             ;
        goto    IR_END          ;
;_____________________________________________________________________________left IR

LEFTIR  bcf     IRLCLK          ; make clock signal low to initiate a measurement
        btfsc   IRLDAT          ; check for data signal to go low (ack that a measurement
is in progress)
        goto    $ - 1           ;

        bsf     IRREAD          ; set flag to indicate a measurement has been started
        goto    INT_END         ;

LEFTOK  btfss   IRLDAT          ; check for measurement complete
        goto    INT_END         ; else end interrupt so other routines may continue
processing

        bsf     IRLCLK          ; start bit
        call    DELAY           ;
        bcf     IRLCLK          ;
        call    DELAY           ;

LIR7    bsf     IRLCLK          ; bit 7
        btfss   IRLDAT          ;
        goto    LIRLO7          ;
        goto    LIRHI7          ;
LIRLO7  bcf     IRLDATA,0x07    ;
        goto    LIR6            ;
LIRHI7  bsf     IRLDATA,0x07    ;
        goto    LIR6            ;

LIR6    call    DELAY           ; bit 6
        bcf     IRLCLK          ;
        call    DELAY           ;
        bsf     IRLCLK          ;
```

```
        btfss   IRLDAT          ;
        goto    LIRLO6          ;
        goto    LIRHI6          ;
LIRLO6  bcf     IRLDATA,0x06    ;
        goto    LIR5            ;
LIRHI6  bsf     IRLDATA,0x06    ;
        goto    LIR5            ;

LIR5    call    DELAY           ; bit 5
        bcf     IRLCLK          ;
        call    DELAY           ;
        bsf     IRLCLK          ;
        btfss   IRLDAT          ;
        goto    LIRLO5          ;
        goto    LIRHI5          ;
LIRLO5  bcf     IRLDATA,0x05    ;
        goto    LIR4            ;
LIRHI5  bsf     IRLDATA,0x05    ;
        goto    LIR4            ;

LIR4    call    DELAY           ; bit 4
        bcf     IRLCLK          ;
        call    DELAY           ;
        bsf     IRLCLK          ;
        btfss   IRLDAT          ;
        goto    LIRLO4          ;
        goto    LIRHI4          ;
LIRLO4  bcf     IRLDATA,0x04    ;
        goto    LIR3            ;
LIRHI4  bsf     IRLDATA,0x04    ;
        goto    LIR3            ;

LIR3    call    DELAY           ; bit 3
        bcf     IRLCLK          ;
        call    DELAY           ;
        bsf     IRLCLK          ;
        btfss   IRLDAT          ;
        goto    LIRLO3          ;
        goto    LIRHI3          ;
LIRLO3  bcf     IRLDATA,0x03    ;
        goto    LIR2            ;
LIRHI3  bsf     IRLDATA,0x03    ;
        goto    LIR2            ;

LIR2    call    DELAY           ; bit 2
        bcf     IRLCLK          ;
        call    DELAY           ;
        bsf     IRLCLK          ;
        btfss   IRLDAT          ;
        goto    LIRLO2          ;
        goto    LIRHI2          ;
LIRLO2  bcf     IRLDATA,0x02    ;
        goto    LIR1            ;
LIRHI2  bsf     IRLDATA,0x02    ;
        goto    LIR1            ;

LIR1    call    DELAY           ; bit 1
        bcf     IRLCLK          ;
        call    DELAY           ;
        bsf     IRLCLK          ;
        btfss   IRLDAT          ;
        goto    LIRLO1          ;
        goto    LIRHI1          ;
LIRLO1  bcf     IRLDATA,0x01    ;
        goto    LIR0            ;
LIRHI1  bsf     IRLDATA,0x01    ;
        goto    LIR0            ;

LIR0    call    DELAY           ; bit 0
        bcf     IRLCLK          ;
        call    DELAY           ;
```

```
        bsf     IRLCLK          ;
        btfss   IRLDAT          ;
        goto    LIRLO0          ;
        goto    LIRHI0          ;
LIRLO0  bcf     IRLDATA,0x00    ;
        bsf     IRR             ;
        bcf     IRF             ;
        goto    IR_END          ;
LIRHI0  bsf     IRLDATA,0x00    ;
        bsf     IRR             ;
        bcf     IRF             ;
        goto    IR_END          ;



IR_END  bcf     IRREAD          ; reset flag to indicate a measurement has not been started
        goto    INT_END         ;


NEXT3   btfss   PIR1,ADIF       ; Check if A 2 D interrupt
        goto    NEXT3           ;
        bcf     PIR1,ADIF       ; clear flag

        goto    INT_END         ;
;-------------------------------------------------------------------------------------
---------

INT_END movf    W_TEMP,0x00     ; restore w reg contents
        retfie                  ;
```