PIC Assembly Code Examples

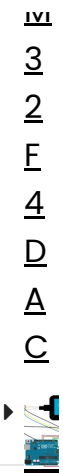
Roland Pelayo January 15, 2017 15 mins read

Here are some PIC assembly codes I have compiled over the years. If you'd like some explanation over ho these codes work, check out my tutorials page.

Beginner:

Blink One LED:

Toggle the state of RB0.



```
list p=16f84a
include
COUNT1 EQU 08h
COUNT2 EQU 09h

org 0x00
goto start

start bsf STATUS, RP0 ;bank 1
movlw 0xFE
movwf TRISB ;set all PORTB input except for RB0
bcf STATUS, RP0 ;bank 0

main bsf PORTB, 0 ;make RB0 high
call delay ;delay subroutine
bcf PORTB, 0 ;make RB0 low
goto main

delay
loop1 decfsz COUNT1,1 ;decrement COUNT1 variable until zero
goto loop1
decfsz COUNT2,1 ;decrement COUNT2, if not zero, go back to loop1
goto loop1
return
end
```

Blink All LED:

All pins of PORTB simultaneously turns on and then turns off on a loop.

```
list p=16f84a
include

COUNT1 EQU 08h
COUNT2 EQU 09h

org 0x00
start

start bsf STATUS, RP0 ;bank 1
movlw 0x00
movwf TRISB ;make all PORTB output
bsf STATUS, RP1 ;bank 0

main movlw 0xFF
movwf PORTB ;make all PORTB pins high
call delay ;delay subroutine
movlw 0x00
movwf PORTB ;make all PORTB pins low
call delay
goto main

delay
loop1 decfsz COUNT1,1 ;decrement COUNT1 variable until zero
goto loop1
decfsz COUNT2,1 ;decrement COUNT2, if not zero, go back to loop1
goto loop1
return
end
```

Using a Switch:

A switch is connected normally high at RA0. If the switch is pressed, RB0 is pulled low, otherwise, RB0 stays high.

```
list p=16f84a
include

COUNT1 EQU 08h
COUNT2 EQU 09h

org 0x00
start

start bsf STATUS,RP0 ;bank 1
movlw 0x01
movwf TRISA ;make all PORTA output except RA0
movlw 0x00
movwf TRISB ;make all PORTB output
bcf STATUS,RP0 ;bank 0

main btfss PORTA,0 ;check RA0 if high, if it is, skip the next line
goto sub1 ;program goes here if RA0 is low
bcf PORTB,0 ;program goes here if RA0 is high; make RBO low
call delay ;delay subroutine
goto main

sub1 bsf PORTB,0 ;make RB0 high
call delay
goto main

delay
loop1 decfsz COUNT1,1 ;decrement COUNT1 variable until zero
goto loop1
decfsz COUNT2,1 ;decrement COUNT2, if not zero, go back to loop1
goto loop1
return
end
```

Count Button Press (w/ Seven Segment Display):

Display on a common cathode seven segment display (with decoder connected to PORTB) the number of button presses on RA0. If the number of presses exceeds nine, the counter goes back to zero.

```
list p=16f84a
include

org 0x00
goto start

start bsf STATUS, 5 ;goto bank 1
movlw 0x00 ;clear TRISB - set all PORTB as output
movwf TRISB
movlw 0x1F ;set all PORTA as input
movwf TRISA
bcf STATUS, 5 ;goto bank 0
clrf PORTB ;initialize PORTB to 0

main btfsc PORTA,0 ;check if RA.0 button is pressed
call delay ;for debounce
call delay
call delay
btfsc PORTA,0 ;check again
goto main ;if not pressed, go back to main
incf PORTB,1 ;if pressed increment PORTB
movlw 0x09 ;for checking if PORTB = 9: put 0x09 to W
subwf PORTB,0 ;subtract 0x09 from W register
btfss STATUS, 2 ;check if zero was the result of the last operation
goto main ;if not zero (PORTB is not equal to 0x09), go back to main
clrf PORTB ;if zero (PORTB is equal to 0x09), set PORTB to zero
goto main ;go back and check the button again

delay
loop1 decfsz COUNT1,1 ;decrement COUNT1 variable until zero
goto loop1
decfsz COUNT2,1 ;decrement COUNT2, if not zero, go back to loop1
goto loop1
return
end
```

Intermediate:

Timer Interrupt:

Strobe lights from RB7 to RB0 and back using timer overflow interrupt.

```
list p=16f84a
include

cblock 0x0c
COUNT1
COUNT2
endc

org 0x00
goto start
org 0x04
goto isr

start bsf STATUS, RP0 ;switch to Bank 1
movlw 0x00
movwf TRISA ;PORTA all output
movlw 0x00
movwf TRISB ;PORTB all output
movlw 07h
movwf OPTION_REG ;Prescaler set to maximum
bcf STATUS, RP0 ;switch to Bank 0
bsf INTCON,GIE ;Enable all interrupts
bsf INTCON,T0IE ;Enable timer overflow interrupt
movlw 0x80
movwf PORTB
goto main

;---------------interrupt handler
isr
bcf INTCON,GIE ;Disable interrupt inside handler
rrf PORTB ;Complement PORTA
call delay ;Delay subroutine
bcf INTCON,T0IF ;Clear TMR overflow bit
bsf INTCON,GIE ;Re-enable interrupts
retfie ;Return from interrupt

;---------------main routine
main movf PORTA,1 ;Dummy commands
clrf PORTA
goto main

;---------------delay subroutine
delay
loop1 decfsz COUNT1,1
goto loop1
```

```
goto loop1
return
end
```

RBO Interrupt:

Turn off blinking when RB0 is pressed.

```
list p=16f84a
include

cblock 0x0c
COUNT1
COUNT2
endc

org 0x00
goto start
org 0x04
goto isr

start bsf STATUS, RP0 ;bank 1
movlw 0x00
movwf TRISA ;make all PORTA output
movlw 0x01
movwf TRISB ;make all PORTB output except RB0
bcf STATUS, RP0 ;bank 0
movlw b'10010000'
movwf INTCON ;enable GIE, INTE
goto main

;-----------interrupt handler
isr
bcf INTCON, GIE ;disable all interrupt inside handler
bcf PORTA,0 ;make RA0 low
call delay ;multiple calls to delay for longer pause
call delay
call delay
bcf INTCON,INTF ;clear the interrupt flag
bsf INTCON, GIE ;enable interrupt
goto main

;-----------main routine
main call delay
bsf PORTA,0 ;make RA0 high
call delay
bcf PORTA,0 ;make RA0 low
call delay
goto main

;-----------delay subroutine
delay
loop1 decfsz COUNT1,1
```

```
decfsz count2,1
goto loop1
return
end
```

RB Change Interrupt:

Similar to the code above except interrupt is triggered when changes to RB4, RB5, RB6 or RB7 is made.

```
list p=16f84a
include

cblock 0x0c
COUNT1
COUNT2
endc

org 0x00
goto start
org 0x04
goto isr

start bsf STATUS, RP0 ;bank 1
movlw 0x00
movwf TRISA ;make all PORTA output
movlw 0x01
movwf TRISB ;make all PORTB output except RB0
bcf STATUS, RP0 ;bank 0
movlw b'10001000'
movwf INTCON ;enable GIE, RBIE
goto main

;-----------interrupt handler
isr
bcf INTCON, GIE ;disable all interrupt inside handler
bcf PORTA,0 ;make RA0 low
call delay ;multiple calls to delay for longer pause
call delay
call delay
bcf INTCON, RBIF ;clear the interrupt flag
bsf INTCON, GIE ;enable interrupt
goto main

;-----------main routine
main call delay
bsf PORTA,0 ;make RA0 high
call delay
bcf PORTA,0 ;make RA0 low
call delay
goto main

;-----------delay subroutine
delay
loop1 decfsz COUNT1,1
```

```
decfsz count2,1
goto loop1
return
end
```

EEPROM Write Interrupt:

Triggers an interrupt when the data to EEPROM has been written.

```
list p=16f84a
include

org 0x00
goto start
org 0x04
goto isr

start bsf STATUS, RP0 ; Bank 1
clrf TRISA
bsf EECON1, WREN ; Enable Write
; this is a required sequence according to the datasheet
movlw 55h
movwf EECON2 ; Write 55h
movlw AAh ;
movwf EECON2 ; Write AAh
bsf EECON1,WR ; Set WR bit
; end of required sequence
bcf STATUS, RP0 ; Bank 0
bcf PORTA,0 ;clear RA0
movlw 0FFh ;write 0FFh to EEDATA
movwf EEDATA
movlw b'11000000' ;enable EEPROM write interrupt
movwf INTCON
goto main

;-----------interrupt handler
isr
bcf INTCON, GIE
bcf INTCON, EEIE
bsf PORTA,0
bsf INTCON, EEIE
bsf INTCON, GIE
goto start
end
```

PWM:

Generate hardware PWM at RB3 using PIC16F628A.

```
list p=16f628A
include

cblock 0x20
COUNT1
COUNT2
endc

org 0x00
goto init

init movlw .50
movwf COUNT1
movwf COUNT2
;-------SET PWM FREQUENCY
bsf STATUS, RP0 ;Bank 1
movlw D'128' ;Set PR2 to 128 decimal; PWM period = 2064uS => PWM frequency = 484Hz
movwf PR2
bcf STATUS, RP1 ;Bank 0
clrf CCPR1L ;Set PWM starting duty cycle
comf CCPR1L
movlw B'00001100' ;Set PWM mode, bits 5 and 4 are the two LSBs of the 10-bit duty cycle
register
movwf CCP1CON ;SET PWM PIN TO OUTPUT MODE;;;
bsf STATUS, RP0 ;Bank 1
bcf TRISB, 3 ;Set RB3 as output for PWM
bcf STATUS, RP0 ;Bank 0
movlw B'00000010' ;Set Timer2 prescale value to 16 so the PWM period = 2064uS => PWM
frequency = 484Hz
movwf T2CON
clrf TMR2 ;clear Timer2 module
bsf T2CON, TMR2ON ;enable Timer2 module

main call delay
goto main

delay
loop1 decfsz COUNT1,1
goto loop1
decfsz COUNT2,1
goto loop1
return
end
```

USART Communication:

Send "!" character to serial port.

```
list p = 16f628A
INCLUDE

cblock 0x20
char0
COUNT1
COUNT2
endc

org 0x00
goto init

init clrf PORTB
clrf PORTA
bsf STATUS, RP0 ;bank 1
clrf TRISB ;all PORTB pins are output
movlw 01h
movwf TRISA ;make all PORTA pins output except RA0
movlw 07h
movwf CMCON ;disable comparator modules
;---CONFIGURE SPBRG FOR DESIRED BAUD RATE
movlw D'25' ;baud rate = 9600bps
movwf SPBRG ;at 4MHZ
;---CONFIGURE TXSTA
movlw B'00100100'
movwf TXSTA
;Configures TXSTA as 8 bit transmission, transmit enabled, async mode, high speed baud
rate
bcf STATUS, RP0 ;bank 0
movlw B'10000000'
movwf RCSTA ;enable serial port receive
movlw 0x21
movwf char0 ;put ! (ascii code 21) character to char0 register

main btfsc PORTA, 0 ;check if button at RA0 is pressed
goto main ;if not, wait
;else transmit a byte
movf char0, W
movwf TXREG ;place the ! character to TXREG
goto wthere
call delay
goto main

wthere btfss TXSTA, TRMT ;check if TRMT is empty
goto wthere ;if not, check again
```

```
return

delay
loop1 decfsz COUNT1,1
goto loop1
decfsz COUNT2,1
goto loop1
return
end
```

USART - Display a Rabbit:

```
list p = 16f628A
INCLUDE

cblock 0x20
charOpPa
charClPa
charBlash
charSlash
charUnSc
charEq
charSQ
charDQ
charPrd
spc
lf
COUNT1
COUNT2
endc

org 0x00
goto init

init clrf PORTB
clrf PORTA
bsf STATUS, RP0 ;bank 1
clrf TRISB ;all PORTB pins are output
movlw 01h
movwf TRISA ;make all PORTA pins output except RA0
movlw 07h
movwf CMCON ;disable comparator modules
;---CONFIGURE SPBRG FOR DESIRED BAUD RATE
movlw D'25' ;baud rate = 9600bps
movwf SPBRG ;at 4MHZ
;---CONFIGURE TXSTA
movlw B'00100100'
movwf TXSTA
;Configures TXSTA as 8 bit transmission, transmit enabled, async mode, high speed baud
rate
bcf STATUS, RP0 ;bank 0
movlw B'10000000'
movwf RCSTA ;enable serial port receive
movlw .40 ;(
movwf charOpPa
movlw .41 ;)
movwf charClPa
```

```
movwf charBlash
movlw .47 ;/
movwf charSlash
movlw .95 ;_
movwf charUnSc
movlw .61 ;=
movwf charEq
movlw .39 ;'
movwf charSQ
movlw .34
movwf charDQ
movlw .46 ;.
movwf charPrd
movlw .32
movwf spc
movlw .13
movwf lf

main btfsc PORTA, 0 ;check if button at RA0 is pressed
goto main ;if not, wait
;else transmit a byte
;------------line1
movf spc, W
movwf TXREG ;TO TXREG
call wthere
movf spc, W
movwf TXREG ;TO TXREG
call wthere
movf charOpPa, W
movwf TXREG ;TO TXREG
call wthere
movf charBlash, W
movwf TXREG ;TO TXREG
call wthere
movf charUnSc, W
movwf TXREG ;TO TXREG
call wthere
movf charUnSc, W
movwf TXREG ;TO TXREG
call wthere
movf charUnSc, W
movwf TXREG ;TO TXREG
call wthere
movf charSlash, W
movwf TXREG ;TO TXREG
```

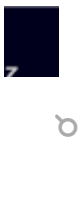
```
movf charCLPa, W
movwf TXREG
call wthere
movf lf, W ;line feed
movwf TXREG ;TO TXREG
call wthere
;-------------line2
movf spc, W
movwf TXREG ;TO TXREG
call wthere
movf spc, W
movwf TXREG ;TO TXREG
call wthere
movf charOpPa, W
movwf TXREG ;TO TXREG
call wthere
movf charEq, W
movwf TXREG ;TO TXREG
call wthere
movf charSQ, W
movwf TXREG ;TO TXREG
call wthere
movf charPrd, W
movwf TXREG ;TO TXREG
call wthere
movf charSQ, W
movwf TXREG ;TO TXREG
call wthere
movf charEq, W
movwf TXREG ;TO TXREG
call wthere
movf charClPa, W
movwf TXREG
call wthere
movlw .13 ;line feed
movwf TXREG ;TO TXREG
call wthere
;------------line3
movf spc, W
movwf TXREG ;TO TXREG
call wthere
movf spc, W
movwf TXREG ;TO TXREG
call wthere
movf charOpPa, W
```

```
call wthere
movf charDQ, W
movwf TXREG ;TO TXREG
call wthere
movf charClPa, W
movwf TXREG ;TO TXREG
call wthere
movf charUnSc, W
movwf TXREG ;TO TXREG
call wthere
movf charOpPa, W
movwf TXREG ;TO TXREG
call wthere
movf charDQ, W
movwf TXREG ;TO TXREG
call wthere
movf charClPa, W
movwf TXREG ;TO TXREG
call wthere
movlw .13 ;line feed
movwf TXREG ;TO TXREG
call wthere
call delay
goto last

wthere
loop22 bsf STATUS,RP0
btfss TXSTA, TRMT ;check if TRMT is empty
goto loop22 ;if not, check again
bcf STATUS, RP0
return

delay
loop1 decfsz COUNT1,1
goto loop1
decfsz COUNT2,1
goto loop1
return

last nop ;so that the display will stay
nop
sleep
end
```



```
list p=16f84a
#include

cblock 0x0c
TEMP
TEMP2
DADDR
COUNT1
COUNT2
CHAR1
CHAR2
CHAR3
CHAR4
CHAR5
CHAR_COUNT
endc

LCD_DATA equ PORTB
#define LCD_E PORTA,2
#define LCD_RW PORTA,1
#define LCD_RS PORTA,0

org 0x00
goto init

init movlw 0x05
movwf CHAR_COUNT
movlw 0x48 ;H
movwf CHAR1
movlw 0x45 ;E
movwf CHAR2
movlw 0x4C ;L
movwf CHAR3
movlw 0x4C ;L
movwf CHAR4
movlw 0x4F ;O
movwf CHAR5
bsf STATUS, RP0
clrf TRISB
clrf TRISA
bcf STATUS, RP1
call LCD_prep
call delay
movlw 0x80
call LCD_cmd
```

```
main goto LCD_sendchr

LCD_sendchr movf CHAR1,W
call delay
call send_it
movf CHAR2,W
call delay
call send_it
movf CHAR3,W
call delay
call send_it
movf CHAR4,W
call delay
call send_it
movf CHAR5,W
call delay
call send_it

send_it call LCD_putch
call delay
decfsz CHAR_COUNT,f
return
call LCD_prep
call delay
movlw 0x80
call LCD_cmd
call delay
goto end_here

LCD_poll bsf STATUS, RP0
clrf TRISB
comf TRISB
bcf STATUS, RP0
bcf LCD_RS
bsf LCD_RW
bsf LCD_E
movf LCD_DATA, TEMP2
nop
bcf LCD_E
btfsc TEMP2,7
goto LCD_poll
bsf STATUS, RP0
clrf TRISB
bcf STATUS, RP0
```

```
LCD_cmd movwf TEMP
call LCD_poll
bcf LCD_RS
bcf LCD_RW
bsf LCD_E
movf TEMP, W
movwf LCD_DATA
nop
bcf LCD_E
return
LCD_prep clrf PORTA
movlw 0x3C ;Configure to 2 x 40, 8-bit mode
call LCD_cmd
movlw 0x0F ;Turn on display and cursor
call LCD_cmd
movlw 0x14 ;Shift cursor right
call LCD_cmd
movlw 0x01 ;clear cursor and return to home position
call LCD_cmd
return
get_DDRAM bsf STATUS, RP0
clrf TRISB
comf TRISB
bcf STATUS, RP0
bcf LCD_RS
bsf LCD_RW
bsf LCD_E
movf LCD_DATA, W
nop
bcf LCD_E
movwf DADDR
return
LCD_putch movwf TEMP
call LCD_poll
movf TEMP,W
bsf LCD_RS
bcf LCD_RW
bsf LCD_E
movwf LCD_DATA
nop
bcf LCD_E
return
chk_0x53 movlw 0x53
subwf DADDR
btfsc STATUS,Z
```

```
movlw 0x94 ;Set DDRAM address to 0x14 (start of 3rd line)
call LCD_cmd
call LCD_poll
movf TEMP,W
bsf LCD_RS
bcf LCD_RW
bsf LCD_E
movwf LCD_DATA
nop
bcf LCD_E
return
chk_0x27 movlw 0x27
subwf DADDR
btfsc STATUS,Z
return
movlw 0xD4
call LCD_cmd
call LCD_poll
movf TEMP, W
bsf LCD_RS
bcf LCD_RW
bsf LCD_E
movwf LCD_DATA
nop
bcf LCD_E
return
delay
loop1 decfsz COUNT1,1
goto loop1
decfsz COUNT2,1
goto loop1
return

end_here nop
nop
end
```

Dual Seven Segment w/ Lookup:

Uses persistence of vision (POV) to digits on a dual seven segment display.

```
list p=16f84a
include

cblock 0x0c
COUNT1
COUNT2
endc

org 0x00
goto start

start bsf STATUS,5
movlw 0x00
movwf TRISA
movlw 0x00
movwf TRISB
bcf STATUS,5
movlw .100
movwf COUNT1
movwf COUNT2

main movlw 0x02
movwf PORTA
movlw 0x00
call table
movwf PORTB
call delay
movlw 0x01
movwf PORTA
movlw 0x00
call table
movwf PORTB
call delay
call delay
call delay
call delay
movlw 0x02
movwf PORTA
movlw 0x00
call table
movwf PORTB
call delay
movlw 0x01
movwf PORTA
movlw 0x01
```

```
movwf PORTB
call delay
call delay
call delay
call delay
movlw 0x02
movwf PORTA
movlw 0x00
call table
movwf PORTB
call delay
movlw 0x01
movwf PORTA
movlw 0x02
call table
movwf PORTB
call delay
call delay
call delay
call delay
movlw 0x02
movwf PORTA
movlw 0x00
call table
movwf PORTB
call delay
movlw 0x01
movwf PORTA
movlw 0x03
call table
movwf PORTB
call delay
call delay
call delay
call delay
movlw 0x02
movwf PORTA
movlw 0x00
call table
movwf PORTB
call delay
movlw 0x01
movwf PORTA
movlw 0x04
call table
```

```
call delay
call delay
call delay
movlw 0x02
movwf PORTA
movlw 0x00
call table
movwf PORTB
call delay
movlw 0x01
movwf PORTA
movlw 0x05
call table
movwf PORTB
call delay
call delay
call delay
call delay
movlw 0x02
movwf PORTA
movlw 0x00
call table
movwf PORTB
call delay
movlw 0x01
movwf PORTA
movlw 0x06
call table
movwf PORTB
call delay
call delay
call delay
call delay
movlw 0x02
movwf PORTA
movlw 0x00
call table
movwf PORTB
call delay
movlw 0x01
movwf PORTA
movlw 0x07
call table
movwf PORTB
```

```
call delay
call delay
movlw 0x02
movwf PORTA
movlw 0x00
call table
movwf PORTB
call delay
movlw 0x01
movwf PORTA
movlw 0x08
call table
movwf PORTB
call delay
call delay
call delay
call delay
movlw 0x02
movwf PORTA
movlw 0x00
call table
movwf PORTB
call delay
movlw 0x01
movwf PORTA
movlw 0x09
call table
movwf PORTB
call delay
call delay
call delay
call delay
goto main

delay
loop1 decfsz COUNT1,1
goto loop1
decfsz COUNT2,1
goto loop1
return

table addwf PC
retlw b'00111111' ;digit 0
retlw b'00000110' ;digit 1
```

```
retlw b'01100110' ;digit 4
retlw b'01101101' ;digit 5
retlw b'01111101' ;digit 6
retlw b'00000111' ;digit 7
retlw b'01111111' ;digit 8
retlw b'01101111' ;digit 9
end
```


Roland Pelayo

Roland Pelayo started *TMM* in 2015. He is a firmware engineer who has over ten years of experience in developing electronic and microcontroller-based systems. Roland's designs include medical devices, security and automation, robots, emergency alert systems, and educational training modules. Have something that you like Roland to write about here? or do you need consultation for microcontroller firmware projects? just contact him via the contact page.