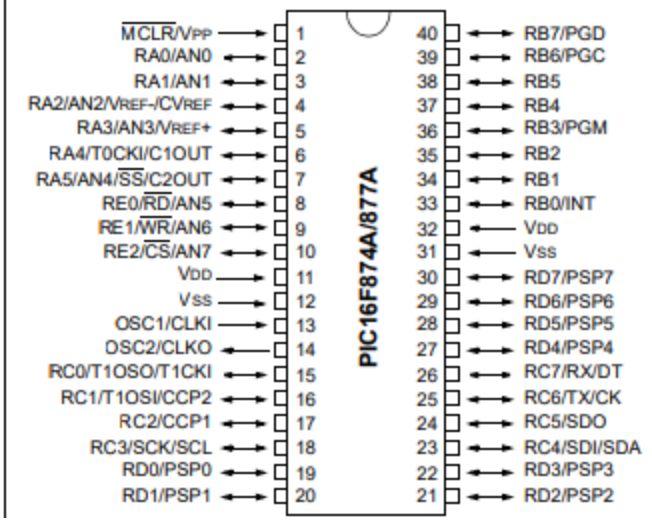
Pinout

Pin Diagrams (Continued)

40-Pin PDIP



Memory Organization:

3 Memory Blocks

Program Memory and Data Memory have separate busses

We have 8k words with 14 bits of flash program memory

Reset Vector = 0000h (the program will go here on reset)

Interrupt Vector = 0004h (the program will go here on interrupt)

FIGURE 2-1: PIC16F876A/877A PROGRAM MEMORY MAP AND STACK

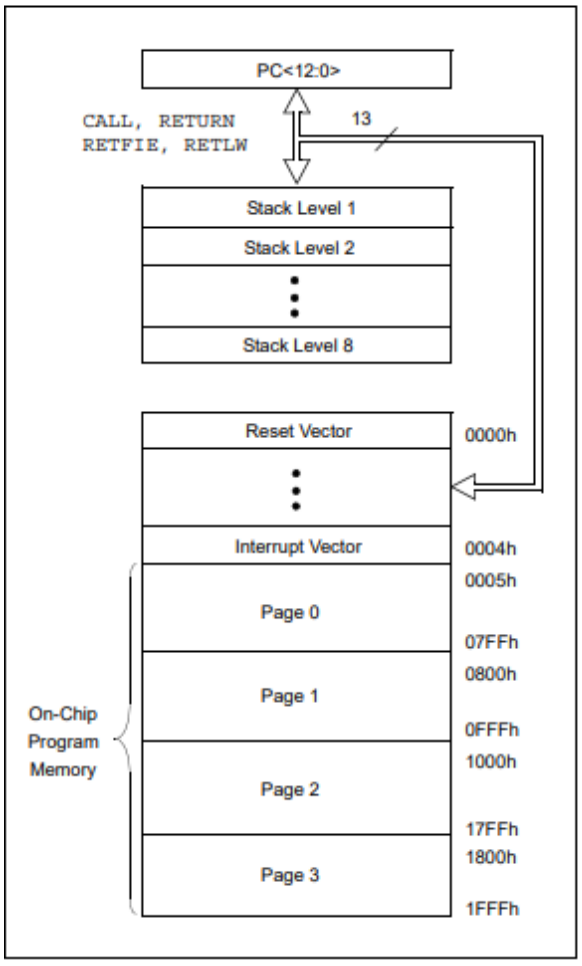
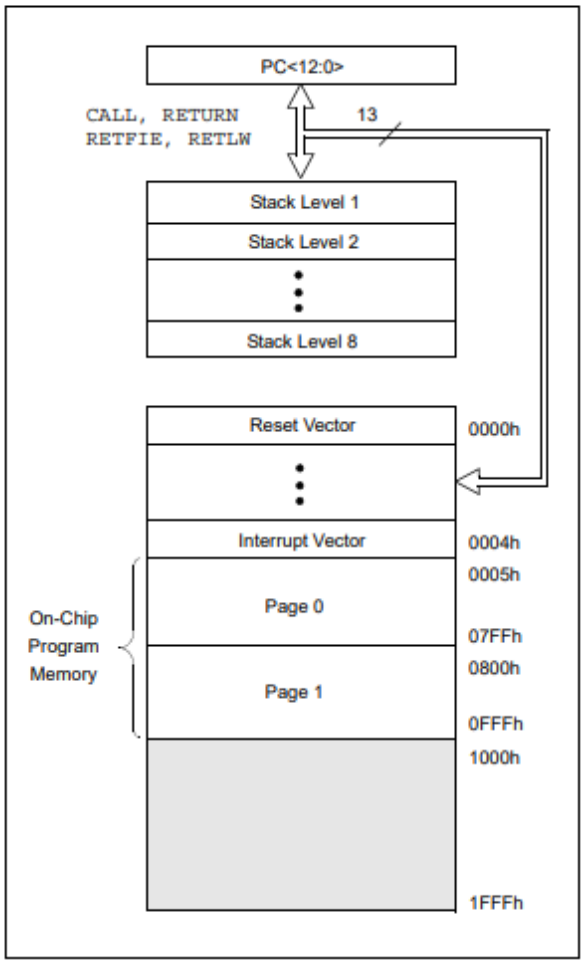


FIGURE 2-2: PIC16F873A/874A PROGRAM MEMORY MAP AND STACK



We can write variables from 1000h to 1FFFh

The 4 Banks: (see highlighted cells)

FIGURE 2-3: PIC16F876A/877A REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h	General Purpose Register 16 Bytes	117h	General Purpose Register 16 Bytes	197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
			EFh		16Fh		1EFh
accesses 70h-7Fh	7Fh	accesses 70h-7Fh	F0h FFh	accesses 70h-7Fh	170h 17Fh	accesses 70h - 7Fh	1F0h 1FFh

Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876A.

2: These registers are reserved; maintain these registers clear.

So we can write variables from:

- [20h to 7Fh) in BANK 0
- [A0h to EFh] in BANK 1
- [110h to 11Fh] and [120h to 16Fh] in BANK 2
- [190h to 19Fh] and [1A0h to 1EFh] in BANK 3

Use BANKSEL to select a bank

e.g. : BANKSEL PORTB

Use :

`

#include p16f877a.inc ; Include register definition file

`

To include a file that will make you use the common names rather than the addresses.

اذا ما عملنا ال

Include

اللي فوق، وقتها مضطرين كل مرة بدنا نروح عمحل أو نستخدم رجستر لازم نحط المكان تبعه مش اسمه، فهيك خلص بنّاديهم بالاسم طوالي

Special Functions Registers:

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral features section.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
01h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
02h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
03h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
04h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	43, 150
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	45, 150
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	47, 150
08h(4)	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	48, 150
09h(4)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	49, 150
0Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
0Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	26, 150
0Dh	PIR2	—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF	-0-0 0--0	28, 150
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	--00 0000	57, 150
11h	TMR2	Timer2 Module Register								0000 0000	62, 150
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	61, 150
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	79, 150
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	82, 82, 150
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	63, 150
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	63, 150
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	64, 150
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	112, 150
19h	TXREG	USART Transmit Data Register								0000 0000	118, 150
1Ah	RCREG	USART Receive Data Register								0000 0000	118, 150
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxx xxxx	63, 150
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxx xxxx	63, 150
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	64, 150
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	133, 150
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	127, 150

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 1											
80h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
81h	OPTION_REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
83h(3)	STATUS	IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	0001 1xxx	22, 150
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	43, 150
86h	TRISB	PORTB Data Direction Register								1111 1111	45, 150
87h	TRISC	PORTC Data Direction Register								1111 1111	47, 150
88h(4)	TRISD	PORTD Data Direction Register								1111 1111	48, 151
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	50, 151
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
8Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	25, 151
8Dh	PIE2	—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE	-0-0 0--0	27, 151
8Eh	PCON	—	—	—	—	—	—	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	---- --qq	29, 151
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	83, 151
92h	PR2	Timer2 Period Register								1111 1111	62, 151
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	79, 151
94h	SSPSTAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF	0000 0000	79, 151
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	111, 151
99h	SPBRG	Baud Rate Generator Register								0000 0000	113, 151
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	135, 151
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	141, 151
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	133, 151
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	128, 151

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 2											
100h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
101h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
102h(3)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	30, 150
103h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
104h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	45, 150
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
10Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	39, 151
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	39, 151
10Eh	EEDATH	—	—	EEPROM Data Register High Byte						--xx xxxx	39, 151
10Fh	EEADRH	—	—	—	—(5)	EEPROM Address Register High Byte				---- xxxx	39, 151
Bank 3											
180h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
181h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150
182h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
183h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
184h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	45, 150
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
18Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	34, 151
18Dh	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	39, 151
18Eh	—	Reserved; maintain clear								0000 0000	—
18Fh	—	Reserved; maintain clear								0000 0000	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

Status Register Configuration:

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h-1FFh)
0 = Bank 0, 1 (00h-FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h-1FFh)
10 = Bank 2 (100h-17Fh)
01 = Bank 1 (80h-FFh)
00 = Bank 0 (00h-7Fh)
Each bank is 128 bytes.

bit 4 **$\overline{TO}$**: Time-out bit
1 = After power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)
(for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high, or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Option Register Configuration:

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: When using Low-Voltage ICSP Programming (LVP) and the pull-ups on PORTB are enabled, bit 3 in the TRISB register must be cleared to disable the pull-up on RB3 and ensure the proper operation of the device

Intcon Register: IMPORTANT: Use it when you want to use interrupts or timer 0:

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIE Register: Peripheral Interrupts Enable. Used when connecting Analogue to Digital (A/D) conversion or the use of Timer 2 IE:

REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit(1)
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

Note 1: PSPIE is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D converter interrupt
0 = Disables the A/D converter interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIR1: also used in A/D and TMR2:

2.2.2.5 PIR1 Register

The PIR1 register contains the individual flag bits for the peripheral interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt bits are clear prior to enabling an interrupt.

REGISTER 2-5: PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit(1)
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred
Note 1: PSPIF is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag bit
1 = The SSP interrupt condition has occurred and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
- SPI – A transmission/reception has taken place.
- I^2C Slave – A transmission/reception has taken place.
- I^2C Master
  - A transmission/reception has taken place.
  - The initiated Start condition was completed by the SSP module.
  - The initiated Stop condition was completed by the SSP module.
  - The initiated Restart condition was completed by the SSP module.
  - The initiated Acknowledge condition was completed by the SSP module.
  - A Start condition occurred while the SSP module was Idle (multi-master system).
  - A Stop condition occurred while the SSP module was Idle (multi-master system).

0 = No SSP interrupt condition has occurred

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Calling Subroutine:

EXAMPLE 2-1: CALL OF A SUBROUTINE IN PAGE 1 FROM PAGE 0

```
        ORG 0x500
        BCF PCLATH,4
        BSF PCLATH,3    ;Select page 1
                        ;(800h-FFFh)
        CALL SUB1_P1    ;Call subroutine in
        :               ;page 1 (800h-FFFh)
        :
        ORG 0x900       ;page 1 (800h-FFFh)
SUB1_P1
        :               ;called subroutine
                        ;page 1 (800h-FFFh)
        :
        RETURN          ;return to
                        ;Call subroutine
                        ;in page 0
                        ;(000h-7FFh)
```

Indirect Addressing:

EXAMPLE 2-2: INDIRECT ADDRESSING

```
        MOVLW 0x20    ;initialize pointer
        MOVWF FSR     ;to RAM
NEXT    CLRF  INDF    ;clear INDF register
        INCF  FSR,F   ;inc pointer
        BTFSS FSR,4   ;all done?
        GOTO  NEXT    ;no clear next
CONTINUE
        :             ;yes continue
```

There is an EEPROM Register with examples on page 35-37 in PDF but idk if we should add them. Then after That (38-40) there is Flash Memory Example

Initializing a Port Example:

BCF: Bit Clear F (F is any Register that is not the Accumulator)

Status has all the flags != BSF: Bit Set F. The second parameter is which bit to set / clear (in the first line it was bit RP0)

CLRF: clears the whole F Register

EXAMPLE 4-1: INITIALIZING PORTA

```
BCF     STATUS, RP0   ;
BCF     STATUS, RP1   ; Bank0
CLRF    PORTA         ; Initialize PORTA by
                      ; clearing output
                      ; data latches
BSF     STATUS, RP0   ; Select Bank 1
MOVLW   0x06          ; Configure all pins
MOVWF   ADCON1        ; as digital inputs
MOVLW   0xCF          ; Value used to
                      ; initialize data
                      ; direction
MOVWF   TRISA         ; Set RA<3:0> as inputs
                      ; RA<5:4> as outputs
                      ; TRISA<7:6>are always
                      ; read as '0'.
```

MOVLW: mov a Literal to W register (the accumulator)

MOVWF: mov from the accumulator to a register (see [PIC16f887 instruction set - YouTube])

When writing 1 to a bit in TRISA (any TRIS) then we set as input. 0 is output.

(TRIS: Tri State Buffer. We use it to set the direction of each pin in each port (input 1) (output 0).

PINS A-E ARE DISCUSSED IN PAGES 43-52 IN DATASHEET

Using Timer0

5.2 Using Timer0 with an External Clock

When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks. Therefore, it is necessary for T0CKI to be high for at least 2 TOSC (and a small RC delay of 20 ns) and low for at least 2 TOSC (and a small RC delay of 20 ns). Refer to the electrical specification of the desired device.

5.3 Prescaler

There is only one prescaler available which is mutually exclusively shared between the Timer0 module and the Watchdog Timer. A prescaler assignment for the Timer0 module means that there is no prescaler for the Watchdog Timer and vice versa. This prescaler is not readable or writable (see Figure 5-1).

The PSA and PS2:PS0 bits (OPTION_REG<3:0>) determine the prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF 1`, `MOVWF 1`, `BSF 1,x`....etc.) will clear the prescaler. When assigned to WDT, a `CLRWDT` instruction will clear the prescaler along with the Watchdog Timer. The prescaler is not readable or writable.

Note:	Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count, but will not change the prescaler assignment.

REGISTER 5-1: OPTION_REG REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**

bit 6 **INTEDG**

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note:	To avoid an unintended device Reset, the instruction sequence shown in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023) must be executed when changing the prescaler assignment from Timer0 to the WDT. This sequence must be followed even if the WDT is disabled.

TABLE 5-1: REGISTERS ASSOCIATED WITH TIMER0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
01h,101h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
81h,181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

Prescaler is every when to count 1. If it was 1:2 then we can count for double the time (every 8 clock cycles, we count 1 instead of every 4)

After that it goes to interrupt (INTCON REGISTER DEFINES IT)

To measure the time for Timer 1:

Timer Frequency = Clock Frequency / 4 (e.g., 4MHz Clock => 1MHz timer)

Tick delay (when does the timer increase by 1) = Prescalar/Timer Frequency

(e.g., 16/1MHz = 16us)

If we wanted a delay of “n” then we want to count for: Tick delay/n

(e.g., we want a delay of 1s then we need to count for 1s/16u = 62500)

So, we need to fill counter 1 with: 65535 – 62500 = 3035

Which is 0BDBh and this will be put in TMR1H:TMR1L

So to measure everything in one equation:

Value to be put in Timer = *6*5536-((100m*s* * 20Mhz)/(Prescalar*4))

For Timer 2:

Reg Value = 256-(Delay * Fosc)/(Prescalar*4))

Configuration BITS IMPORTANT:

REGISTER 14-1: CONFIGURATION WORD (ADDRESS 2007h)(1)

R/P-1	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
CP	—	DEBUG	WRT1	WRT0	CPD	LVP	BOREN	—	—	$\overline{\text{PWRTEN}}$	WDTEN	Fosc1	Fosc0
bit 13													bit0

bit 13 **CP:** Flash Program Memory Code Protection bit
1 = Code protection off
0 = All program memory code-protected

bit 12 **Unimplemented:** Read as '1'

bit 11 **DEBUG:** In-Circuit Debugger Mode bit
1 = In-Circuit Debugger disabled, RB6 and RB7 are general purpose I/O pins
0 = In-Circuit Debugger enabled, RB6 and RB7 are dedicated to the debugger

bit 10-9 **WRT1:WRT0** Flash Program Memory Write Enable bits
For PIC16F876A/877A:
11 = Write protection off; all program memory may be written to by EECON control
10 = 0000h to 00FFh write-protected; 0100h to 1FFFh may be written to by EECON control
01 = 0000h to 07FFh write-protected; 0800h to 1FFFh may be written to by EECON control
00 = 0000h to 0FFFh write-protected; 1000h to 1FFFh may be written to by EECON control
For PIC16F873A/874A:
11 = Write protection off; all program memory may be written to by EECON control
10 = 0000h to 00FFh write-protected; 0100h to 0FFFh may be written to by EECON control
01 = 0000h to 03FFh write-protected; 0400h to 0FFFh may be written to by EECON control
00 = 0000h to 07FFh write-protected; 0800h to 0FFFh may be written to by EECON control

bit 8 **CPD:** Data EEPROM Memory Code Protection bit
1 = Data EEPROM code protection off
0 = Data EEPROM code-protected

bit 7 **LVP**: Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit
1 = RB3/PGM pin has PGM function; low-voltage programming enabled
0 = RB3 is digital I/O, HV on $\overline{\text{MCLR}}$ must be used for programming

bit 6 **BOREN**: Brown-out Reset Enable bit
1 = BOR enabled
0 = BOR disabled

bit 5-4 **Unimplemented:** Read as '1'

bit 3 **$\overline{\text{PWRTEN}}$**: Power-up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled

bit 2 **WDTEN**: Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0 **Fosc1:Fosc0**: Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

Note 1: The erased (unprogrammed) value of the Configuration Word is 3FFFh.

PWM and A/D and all those things:

CCP: Capture/Compare/PWM

Capture: measures signals

Compare: compares signals and time

PWM: pulse width modulation, needed for controlling intensity of a light or a motor. But it works on DC not AC.

There are 2 modules: ccp1,ccp2. (Pins RC2 -> CCP1 and RC1 -> CCP2)

Capture: capture an event (whether on Positive or Negative edge) and when that happens we 0 timer1 and see the signal.

Compare: I previously know the value i want to compare so i put it in Low and High of CCP1, and then i reset timer until we get same length of signals and on match you generate something. Match with timer signal (from this explanation it is like a timer and you are telling the timer to count up to..).

PWM, based on the signal, lets say i want from the signal to not turn on for the peak then go low, and on the -ve peak and go low

So vs there is 0 detection, and on match (lets say peak) and then you toggle it off. So i can control the dimming based on how much i wait before we reach the time we reach peak (say we here have 50Hz AC and this means we have a signal every 20ms so we know how much time is needed for a full signal. Half a signal is 10ms so we control the time from 0-10ms to see the dimming)

There is a prescaler for timer1, same how timer0 has a prescaler, so we can count for longer times.

If signal changes really fast we can capture every 4th edge or 16th edge ....etc.

The control register is CCP1CON at address 17h-1Dh

Capture and compare use timer1, and PWM uses timer2 which is 8 bits.

There is a PR2 register that is a prescaler

If PR2 is 255 and we work on 4MHz.

Period = (255+1) * 4 / (4 * 10^6) * 16

(Divide by 4M since this is the clk frequency, and 16 is the prescaler and 4 is from the equation itself (since 4 clocks for every instruction))

PWM Duty cycle resolution = is 10 bits.

If it was all 1s, the duty cycle is 10K * 1/4000000 * 16 = maximum.

16 prescaler And 4000000 is clk frequency.

Resolution is 10 bits for 20MHz crystal. For 8MHz it is 8 bits, for 4MHz it is less probably.

We adjust prescaler from register of timer2.

There is also a postscaler that we'd ان شاء الله take in lab.

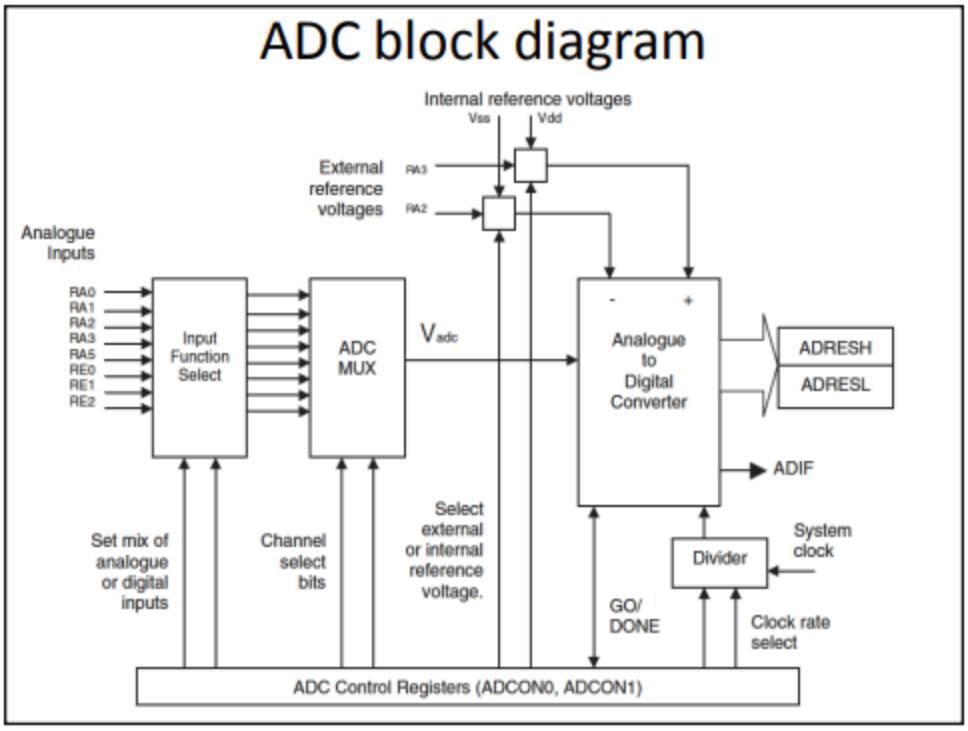
We also have a Comparator module that compares between 2 signals entering RA0 and RA3, and between RA and RA5

They enter an OpAmp.

Since we have 3 bits for configuring it we have 8 possibilites. They determine where the output also will go from and stuff like that.

PIC CCS compiler. We can program it using C but the compiler is not free

ADC BLOCK DIAGRAM



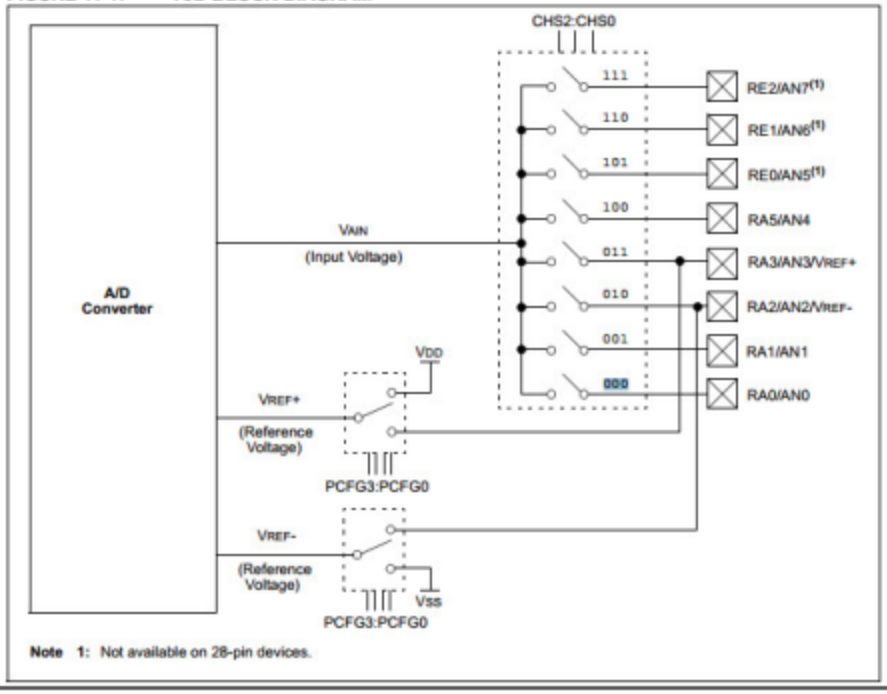
8-bit Conversion

- The 16F877 MCU has eight analogue inputs available, at RA0, RA1, RA2, RA3, RA5, RE0, RE1 and RE2.
- RA2 and RA3 may be used as reference voltage inputs, setting the minimum and maximum values for the measured voltage range.
- These inputs default to analogue operation, so the register ADCON1 has to be initialized explicitly to use these pins for digital input or output.
- The ADC converts an analogue input voltage (e.g. 0 – 2.56V) to 10-bit binary, but only the upper 8 bits of the result are used, giving a resolution of 10 mV per bit ((1/256) X 2.56 V).

ADC OPERATION

- The inputs are connected to a function selector block which sets up each pin for analogue or digital operation according to the 4-bit control code loaded into the A/D port configuration control bits, PCFG0–PCFG3 in ADCON1.
  - The code used, 0011, sets Port E as digital I/O, and Port A as analogue inputs with AN3 as the positive reference input.
- The analogue inputs are then fed to a multiplexer which allows one of the eight inputs to be selected at any one time.
  - This is controlled by the three analogue channel select bits, CHS0–CHS2 in ADCON0.
  - In the example, channel 0 is selected (000), RA0 input.
  - If more than one channel is to be sampled, these select bits need to be changed between ADC conversions.
- The conversion is triggered by setting the GO/DONE bit, which is later cleared automatically to indicate that the conversion is complete.

FIGURE 11-1: A/D BLOCK DIAGRAM



Note 1: Not available on 28-pin devices.

ADCON0 REGISTER (ADDRESS 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits (ADCON0 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-3 **CHS2:CHS0:** Analog Channel Select bits
000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)
110 = Channel 6 (AN6)
111 = Channel 7 (AN7)

Note: The PIC16F873A/876A devices only implement A/D channels 0 through 4; the unimplemented selections are reserved. Do not select any unimplemented channels with these devices.

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)
0 = A/D conversion not in progress

bit 1 **Unimplemented:** Read as '0'

bit 0 **ADON:** A/D On bit
1 = A/D converter module is powered up
0 = A/D converter module is shut-off and consumes no operating current

ADCON1 REGISTER (ADDRESS 9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

bit 6 **ADCS2:** A/D Conversion Clock Select bit (ADCON1 bits in shaded area and in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	FOSC/2
0	01	FOSC/8
0	10	FOSC/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	FOSC/4
1	01	FOSC/16
1	10	FOSC/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

A = Analog input D = Digital I/O
C/R = # of analog input channels/# of A/D voltage references

ADC control registers

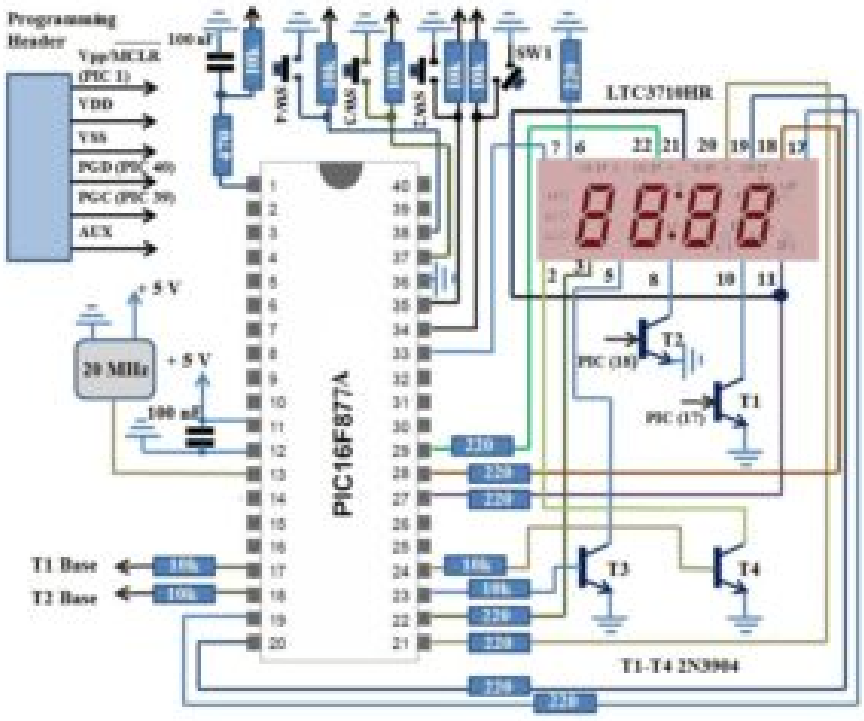
Register	Setting	Flags	Function
ADRESH	XXXX XXXX		ADC result high byte
ADRESL	XXXX XXXX		ADC result low byte
ADCON0	0100 0X01	ADCS1,0 GO/DONE, ADON	Conversion frequency select ADC start, ADC enable
ADCON1	0000 0011	ADFM, PCFG3-0	Result justify, ADC input mode control
INTCON	1100 0000	GIE,PEIE	Peripheral interrupt enable
PIE1	0100 0000	ADIE	ADC interrupt enable
PIR1	0100 0000	ADIF	ADC interrupt flag

(c)

	ADRESH	ADRESL
ADFM = 1 Right justified	0000 00RR	RRRR RRRR
ADFM = 0 Left justified	RRRR RRRR	RR00 0000

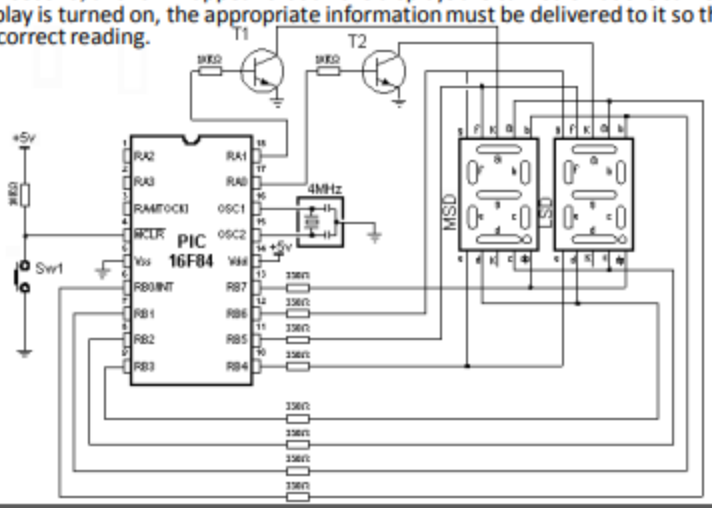
R = Result bits

Example of connecting a 7-segment clock:

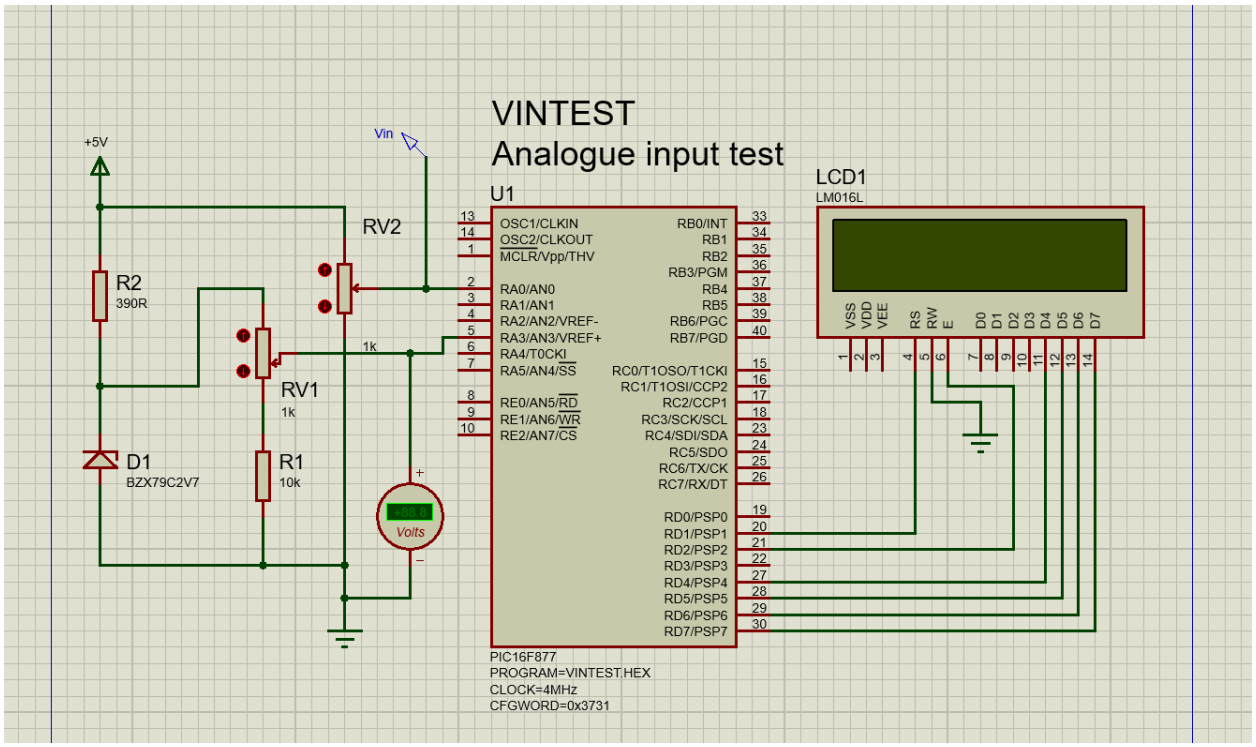


7 segment-display

- To produce a 4, 5 or 6 digit display, all the 7-segment displays are connected in parallel. The common line (the common-cathode line) is taken out separately and this line is taken low for a short period of time to turn on the display. Each display is turned on at a rate above 100 times per second, and it will appear that all the displays are turned on at the same time. As each display is turned on, the appropriate information must be delivered to it so that it will give the correct reading.



Testing VIN with LCD Panel: (see embedded systems pdf for pwm and ccp1con/ccp2con p91)



How To Configure ADC:

To do an A/D Conversion, follow these steps:

1. Configure the A/D module:
   - Configure analog pins/voltage reference and
   - digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D conversion clock (ADCON0)
   - Turn on A/D module (ADCON0)

2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set PEIE bit
   - Set GIE bit
3. Wait the required acquisition time
4. Start conversion:
   - Set GO/DONE bit (ADCON0)
5. Wait for A/D conversion to complete by either:
   - Polling for the GO/DONE bit to be cleared (interrupts disabled); OR Waiting for the A/D interrupt

6. Read A/D Result register pair (ADRESH:ADRESL), clear bit ADIF if required.
7. For the next conversion, go to step 1 or step 2 as required. The A/D conversion time per bit is defined as TAD.

Code:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

;	Project:		Interfacing PICs

;	Source File Name:	VINTEST.ASM

;	Devised by:		MPB

;	Date:			19-12-05

;	Status:			Final version

;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

;	Demonstrates simple analogue input

;	using an external reference voltage of 2.56V

;	The 8-bit result is converted to BCD for display

;	as a voltage using the standard LCD routines.

;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


	PROCESSOR 16F877A

;	Clock = XT 4MHz, standard fuse settings

	__CONFIG 0x3731


;	LABEL EQUATES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


	#INCLUDE "P16F877A.INC"	; standard labels


; GPR 70 - 75 allocated to included LCD display routine
```

```
count   EQU     30      ; Counter for ADC setup delay
ADbin   EQU     31      ; Binary input value
huns    EQU     32      ; Hundreds digit in decimal value
tens    EQU     33      ; Tens digit in decimal value
ones    EQU     34      ; Ones digit in decimal value

; PROGRAM BEGINS ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        ORG     0               ; Default start address
        NOP                     ; required for ICD mode

; Port & display setup.....................................

        BANKSEL         TRISC           ; Select bank 1
        CLRF    TRISD           ; Display port is output
        MOVLW           B'00000011'     ; Analogue input setup code
        MOVWF           ADCON1                  ; Left justify result,
                                ; Port A = analogue inputs

        BANKSEL PORTC                   ; Select bank 0
        CLRF    PORTD           ; Clear display outputs
        MOVLW           B'01000001'     ; Analogue input setup code
        MOVWF           ADCON0                  ; f/8, RA0, done, enable

        CALL    inid            ; Initialise the display

; MAIN LOOP ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

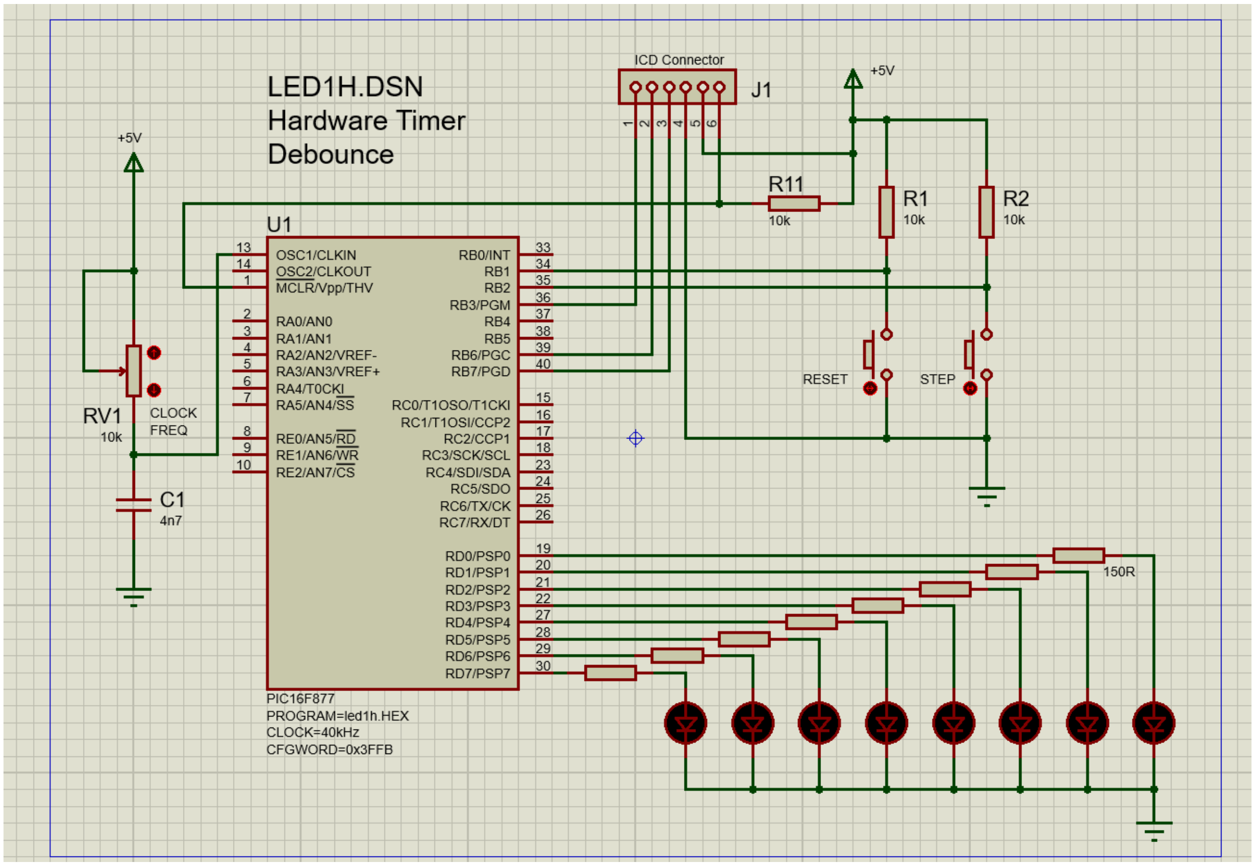
```
start   CALL    getADC          ; read input
        CALL    condec          ; convert to decimal
        CALL    putLCD          ; display input
        GOTO    start           ; jump to main loop

; SUBROUTINES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Read ADC input and store ...............................

getADC  BSF     ADCON0,GO       ; start ADC..
wait    BTFSC   ADCON0,GO       ; ..and wait for finish
        GOTO    wait
        MOVF    ADRESH,W        ; store result high byte
        RETURN

; Convert input to decimal ...............................

condec  MOVWF           ADbin           ; get ADC result
        CLRF    huns            ; zero hundreds digit
        CLRF    tens            ; zero tens digit
        CLRF    ones            ; zero ones digit

; Calclulate hundreds....................................

        BSF     STATUS,C        ; set carry for subtract
        MOVLW           D'100'          ; load 100
sub1    SUBWF   ADbin           ; and subtract from result
        INCF    huns            ; count number of loops
```

```
        BTFSC   STATUS,C        ; and check if done
        GOTO    sub1            ; no, carry on

        ADDWF ADbin             ; yes, add 100 back on
        DECF    huns            ; and correct loop count

; Calculate tens digit.....................................

        BSF     STATUS,C        ; repeat process for tens
        MOVLW           D'10'           ; load 10
sub2    SUBWF ADbin             ; and subtract from result
        INCF    tens            ; count number of loops
        BTFSC   STATUS,C        ; and check if done
        GOTO    sub2            ; no, carry on

        ADDWF ADbin             ; yes, add 100 back on
        DECF    tens            ; and correct loop count
        MOVF    ADbin,W                 ; load remainder
        MOVWF           ones            ; and store as ones digit

        RETURN                          ; done

; Output to display.....................................

putLCD  BCF     Select,RS       ; set display command mode
        MOVLW           080             ; code to home cursor
        CALL    send            ; output it to display
        BSF     Select,RS       ; and restore data mode
```

```
; Convert digits to ASCII and display.....................

        MOVLW   030             ; load ASCII offset
        ADDWF huns              ; convert hundreds to ASCII
        ADDWF tens              ; convert tens to ASCII
        ADDWF ones              ; convert ones to ASCII

        MOVF    huns,W          ; load hundreds code
        CALL    send            ; and send to display
        MOVLW   '.'             ; load point code
        CALL    send            ; and output
        MOVF    tens,W          ; load tens code
        CALL    send            ; and output
        MOVF    ones,W          ; load ones code
        CALL    send            ; and output
        MOVLW   ' '             ; load space code
        CALL    send            ; and output
        MOVLW   'V'             ; load volts code
        CALL    send            ; and output
        MOVLW   'o'             ; load volts code
        CALL    send            ; and output
        MOVLW   'l'             ; load volts code
        CALL    send            ; and output
        MOVLW   't'             ; load volts code
        CALL    send            ; and output
        MOVLW   's'             ; load volts code
        CALL    send            ; and output
```

```
        RETURN                          ; done

; INCLUDED ROUTINES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Include LCD driver routines
;
        #INCLUDE "LCDIS.INC"
;       Contains routines:
;       inid:   Initialises display
;       onems: 1 ms delay
;       xms:    X ms delay
;               Receives X in W
;       send:   Sends a character to display
;               Receives: Control code in W (Select,RS=0)
;                       ASCII character code in W (RS=1)

        END     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Leds with hardware timer debounce:



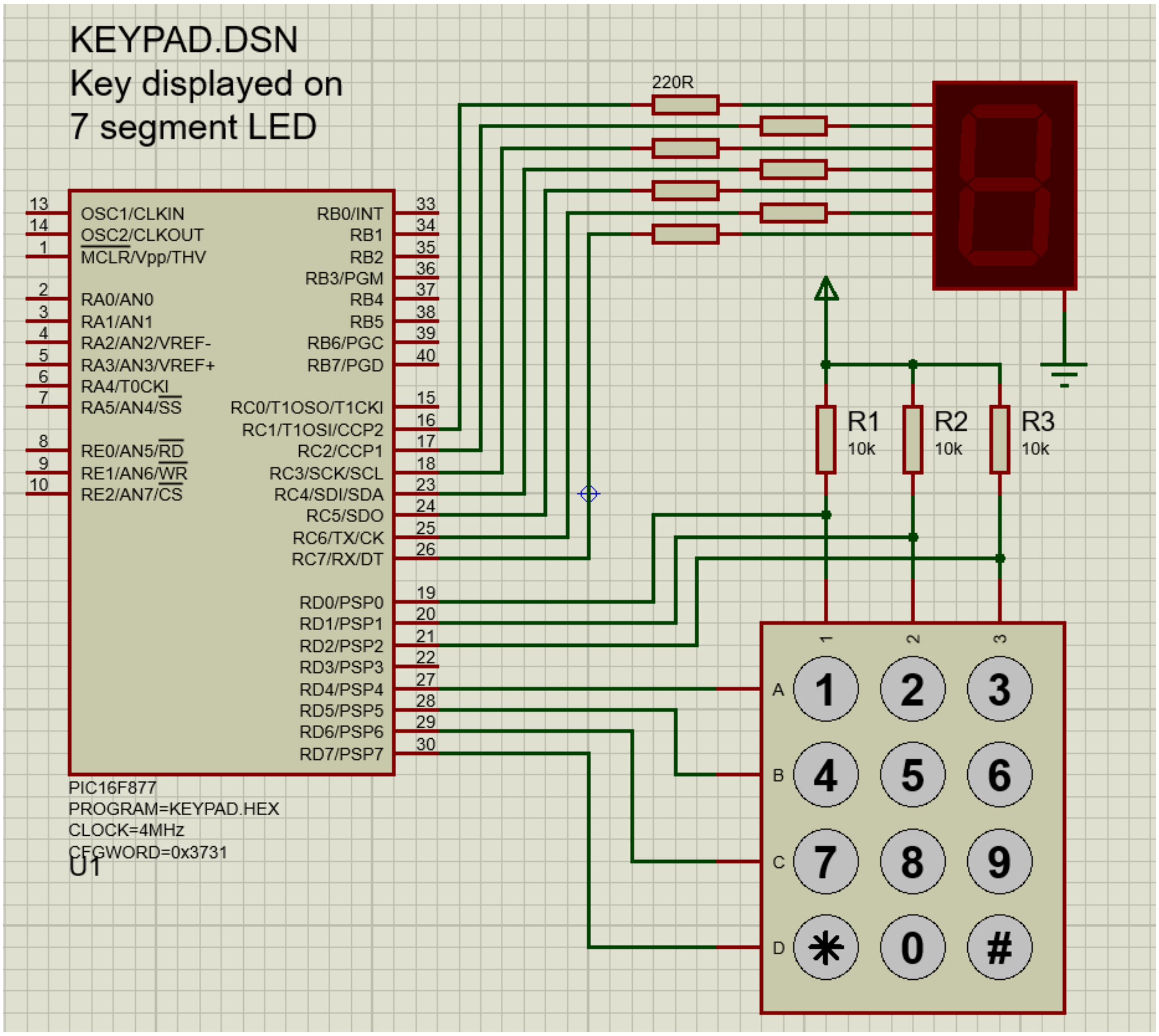
Code:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;	Source File:	LED1H.ASM
;		Author:		MPB
;		Date:		2-12-05
;
;		Output binary count incremented
;	and reset with push buttons.
;		Uses hardware timer to debounce input switch
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
        PROCESSOR 16F877        ; Define MCU type
        __CONFIG 0x3731                 ; Set config fuses

; Register Label Equates.....................................

PORTB   EQU     06              ; Port B Data Register
PORTD   EQU     08              ; Port D Data Register
TRISD   EQU     88              ; Port D Direction Register

TMR0    EQU     01              ; Hardware Timer Register
INTCON  EQU     0B              ; Interrupt Control Register
OPTREG  EQU     81              ; Option Register

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        ORG     000             ; Start of program memory
        NOP                     ; For ICD mode
        GOTO    init            ; Jump to main program

; Interrupt Service Routine .................................

        ORG     004
        BCF     INTCON,2        ; Reset TMR0 interrupt flag
        RETFIE                  ; Return from interrupt

; Initialise Port D (Port B defaults to inputs).............

init    NOP                     ; BANKSEL cannot be labelled
        BANKSEL         TRISD           ; Select bank 1
        MOVLW   b'00000000'     ; Port B Direction Code
```

```
        MOVWF TRISD             ; Load the DDR code into F86

; Initialise Timer0 .....................................

        MOVLW b'11011000'       ; TMR0 initialisation code
        MOVWF OPTREG            ; Int clock, no prescale
        BANKSEL         PORTD           ; Select bank 0
        MOVLW b'10100000'       ; INTCON init. code
        MOVWF INTCON            ; Enable TMR0 interrupt

; Start main loop .....................................

reset   CLRF    PORTD           ; Clear Port B Data

start   BTFSS       PORTB,1         ; Test reset button
        GOTO    reset           ; and reset Port B if pressed
    BTFSC       PORTB,2         ; Test step button
    GOTO        start           ; and repeat if not pressed

        CLRF    TMR0            ; Reset timer
wait    BTFSS   INTCON,2        ; Check for time out
        GOTO    wait            ; Wait if not
stepin  BTFSS   PORTB,2         ; Check step button
        GOTO    stepin          ; and wait until released
    INCF        PORTD           ; Increment output at Port B
    GOTO        start           ; Repeat main loop always

    END                 ; Terminate source code.....
```

KEYPAD:



Code:

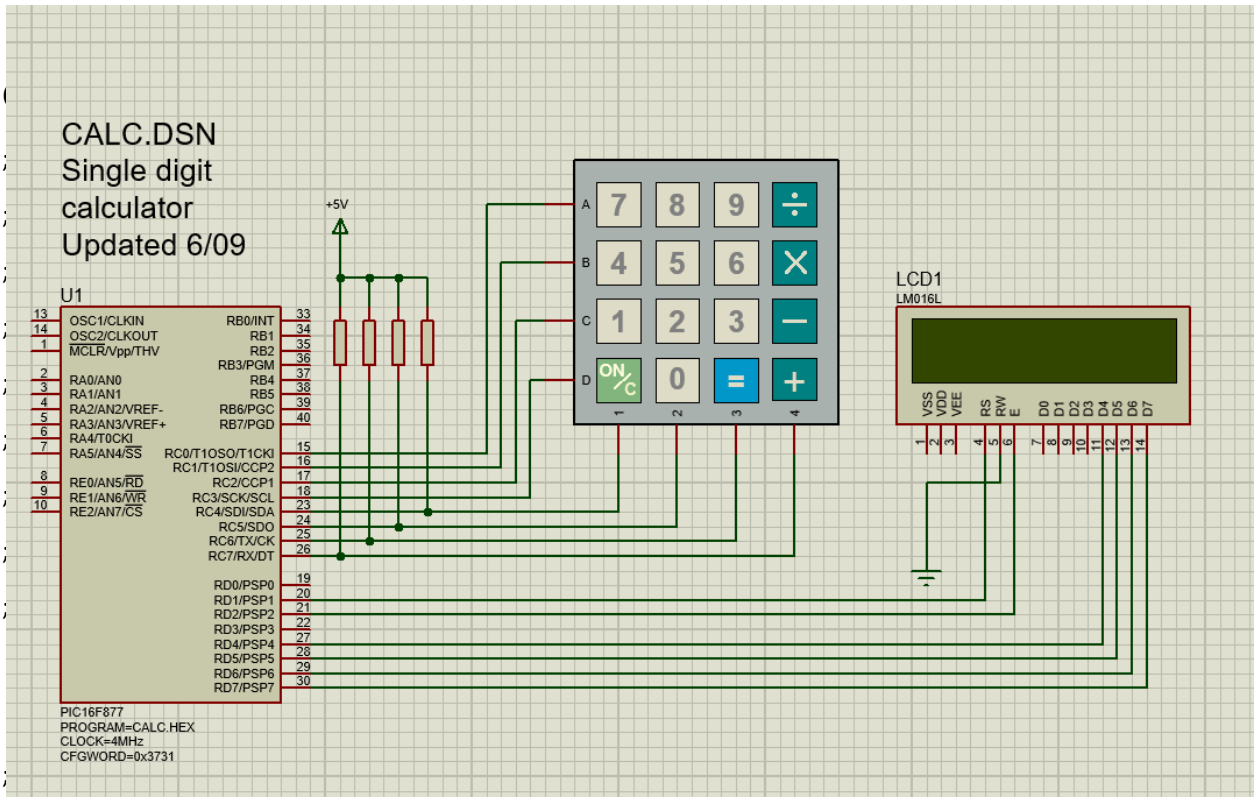
```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;	KEYPAD.ASM	MPB	Ver 1.0		28-8-05
;
;	Reads keypad and shows digit on display
;	Design file KEYPAD.DSN
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        PROCESSOR 16F877

PCL     EQU     002             ; Program Counter
PORTC   EQU     007             ; 7-Segment display
PORTD   EQU     008             ; 3x4 keypad

TRISC   EQU     087             ; Data direction
TRISD   EQU     088             ; registers

Key     EQU     020             ; Count of keys

; Initialise ports.......................................

        BANKSEL         TRISC           ; Display
        CLRW                    ; all outputs
        MOVWF TRISC             ;
        MOVLW B'00000111'       ; Keypad
        MOVWF TRISD             ; bidirectional

        BANKSEL         PORTC           ; Display off
        CLRF    PORTC           ; initially
        GOTO    main            ; jump to main

; Check a row of keys ....................................

row     INCF    Key             ; Count first key
        BTFSS   PORTD,0         ; Check key
        GOTO    found           ; and quit if on
```

```
        INCF    Key             ; and repeat
        BTFSS   PORTD,1         ; for second
        GOTO    found           ; key

        INCF    Key             ; and repeat
        BTFSS   PORTD,2         ; for third
        GOTO    found           ; key
        GOTO    next            ; go for next row

; Scan the keypad.....................................

scan    CLRF    Key             ; Zero key count
        BSF     3,0             ; Set Carry Flag
        BCF     PORTD,4         ; Select first row
newrow  GOTO    row             ; check row

next    BSF     PORTD,3         ; Set fill bit
        RLF     PORTD           ; Select next row
        BTFSC   3,0             ; 0 into carry flag?
        GOTO    newrow          ; if not, next row
        GOTO    scan            ; if so, start again

found   RETURN                  ; quit with key count

; Display code table.....................................

table   MOVF    Key,W           ; Get key count
        ADDWF   PCL             ; and calculate jump
        NOP                     ; into table
```

```
        RETLW   B'00001100'     ; Code for '1'
        RETLW   B'10110110'     ; Code for '2'
        RETLW   B'10011110'     ; Code for '3'
        RETLW   B'11001100'     ; Code for '4'
        RETLW   B'11011010'     ; Code for '5'
        RETLW   B'11111000'     ; Code for '6'
        RETLW   B'00001110'     ; Code for '7'
        RETLW   B'11111110'     ; Code for '8'
        RETLW   B'11001110'     ; Code for '9'
        RETLW   B'10010010'     ; Code for '*'
        RETLW   B'01111110'     ; Code for '0'
        RETLW   B'11101100'     ; Code for '#'

; Output display code.....................................

show    CALL    table           ; Get display code
        MOVWF PORTC             ; and show it
        RETURN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Read keypad & display....

main    MOVLW 0FF               ; Set all outputs
        MOVWF PORTD             ; to keypad high
        CALL    scan            ; Get key number
        CALL    show            ; and dsiplay it
        GOTO    main            ; and repeat
        END
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

CALCULATOR:




```
        __CONFIG 0x3731

;       LABEL EQUATES   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        INCLUDE "P16F877A.INC"
```

```
Char    EQU     30      ; Display character code
Num1    EQU     31      ; First number input
Num2    EQU     32      ; Second number input
Result  EQU     33      ; Calculated result
Oper    EQU     34      ; Operation code store
Temp    EQU     35      ; Temporary register for subtract
Kcount  EQU     36      ; Count of keys hit
Kcode   EQU     37      ; ASCII code for key
```

```
Msd     EQU     38          ; Most significant digit of result
Lsd     EQU     39          ; Least significant digit of result
Kval    EQU     40          ; Key numerical value

RS      EQU     1           ; Register select output bit
E       EQU     2           ; Display data strobe

; Program begins ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        ORG     0                   ; Default start address
        NOP                         ; required for ICD mode

        BANKSEL         TRISC               ; Select bank 1
        MOVLW   B'11110000'         ; Keypad direction code
        MOVWF   TRISC               ;
        CLRF    TRISD               ; Display port is output

        BANKSEL PORTC               ; Select bank 0
        MOVLW   0FF                 ;
        MOVWF   PORTC               ; Set keypad outputs high
        CLRF    PORTD               ; Clear display outputs
        GOTO    start               ; Jump to main program


; MAIN LOOP ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

start   CALL    inid                ; Initialise the display
        MOVLW   0x80                ; position to home cursor
        BCF     Select,RS           ; Select command mode
        CALL    send                ; and send code
```

```
        CLRW    Char        ; ASCII = 0
        CLRW    Kval        ; Key value = 0
        CLRW    DFlag       ; Digit flags = 0

scan    CALL    keyin       ; Scan keypad
        MOVF    Char,1      ; test character code
        BTFSS   STATUS,Z    ; key pressed?
        GOTO    keyon       ; yes - wait for release
        GOTO    scan        ; no - scan again

keyon   MOVF    Char,W      ; Copy..
        MOVWF   Kcode       ; ..ASCIIcode
        MOVLW   D'50'       ; delay for..
        CALL    xms         ; ..50ms debounce

wait    CALL    keyin       ; scan keypad again
        MOVF    Char,1      ; test character code
        BTFSS   STATUS,Z    ; key pressed?
        GOTO    wait        ; no - rescan
        CALL    disout      ; yes - show symbol

        INCF    Kcount      ; inc count..
        MOVF    Kcount,W    ; ..of keys pressed
        ADDWF   PCL         ; jump into table
        NOP
        GOTO    first       ; process first key
        GOTO    scan        ; get operation key
        GOTO    second      ; process second symbol
        GOTO    calc        ; calculate result
```

```
            GOTO    clear           ; clear display

first       MOVF    Kval,W          ; store..
            MOVWF   Num1            ; first num
            GOTO    scan            ; and get op key

second      MOVF    Kval,W          ; store..
            MOVWF   Num2            ; second number
            GOTO    scan            ; and get equals key



; SUBROUTINES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Include LCD driver routine

            INCLUDE"LCDIS.INC"
; Scan keypad .....................................

keyin       MOVLW   00F             ; deselect..
            MOVWF   PORTC           ; ..all rows
            BCF     PORTC,0         ; select row A
            CALL    onems           ; wait output stable

            BTFSC   PORTC,4         ; button 7?
            GOTO    b8              ; no
            MOVLW   '7'             ; yes
            MOVWF   Char            ; load key code
            MOVLW   07              ; and
            MOVWF   Kval            ; key value
            RETURN
```

```
b8      BTFSC   PORTC,5         ; button 8?
        GOTO    b9              ; no
        MOVLW   '8'             ; yes
        MOVWF   Char
        MOVLW   08
        MOVWF   Kval
        RETURN

b9      BTFSC   PORTC,6         ; button 9?
        GOTO    bd              ; no
        MOVLW   '9'             ; yes
        MOVWF   Char
        MOVLW   09
        MOVWF   Kval
        RETURN

bd      BTFSC   PORTC,7         ; button /?
        GOTO    rowb            ; no
        MOVLW   '/'             ; yes
        MOVWF   Char            ; store key code
        MOVWF   Oper            ; store operator symbol
        RETURN

rowb    BSF     PORTC,0         ; select row B
        BCF     PORTC,1
        CALL    onems

        BTFSC   PORTC,4         ; button 4?
        GOTO    b5              ; no
```

```
        MOVLW '4'           ; yes
        MOVWF Char
        MOVLW 04
        MOVWF Kval
        RETURN

b5      BTFSC  PORTC,5      ; button 5?
        GOTO   b6           ; no
        MOVLW '5'           ; yes
        MOVWF Char
        MOVLW 05
        MOVWF Kval
        RETURN

b6      BTFSC  PORTC,6      ; button 6?
        GOTO   bm           ; no
        MOVLW '6'           ; yes
        MOVWF Char
        MOVLW 06
        MOVWF Kval
        RETURN

bm      BTFSC  PORTC,7      ; button x?
        GOTO   rowc         ; no
        MOVLW 'x'           ; yes
        MOVWF Char
        MOVWF Oper
        RETURN

rowc    BSF    PORTC,1      ; select row C
```

```
        BCF     PORTC,2
        CALL    onems

        BTFSC   PORTC,4         ; button 1?
        GOTO    b2              ; no
        MOVLW   '1'             ; yes
        MOVWF   Char
        MOVLW   01
        MOVWF   Kval
        RETURN

b2      BTFSC   PORTC,5         ; button 2?
        GOTO    b3              ; no
        MOVLW   '2'             ; yes
        MOVWF   Char
        MOVLW   02
        MOVWF   Kval
        RETURN

b3      BTFSC   PORTC,6         ; button 3?
        GOTO    bs              ; no
        MOVLW   '3'             ; yes
        MOVWF   Char
        MOVLW   03
        MOVWF   Kval
        RETURN

bs      BTFSC   PORTC,7         ; button -?
        GOTO    rowd            ; no
        MOVLW   '-'             ; yes
```

```
        MOVWF Char
        MOVWF Oper
        RETURN

rowd    BSF     PORTC,2         ; select row D
        BCF     PORTC,3
        CALL    onems

        BTFSC   PORTC,4         ; button C?
        GOTO    b0              ; no
        MOVLW   'c'             ; yes
        MOVWF Char
        MOVWF Oper
        RETURN

b0      BTFSC   PORTC,5         ; button 0?
        GOTO    be              ; no
        MOVLW   '0'             ; yes
        MOVWF Char
        MOVLW 00
        MOVWF Kval
        RETURN

be      BTFSC   PORTC,6         ; button =?
        GOTO    bp              ; no
        MOVLW   '='             ; yes
        MOVWF Char
        RETURN

bp      BTFSC   PORTC,7         ; button +?
```

```
        GOTO    done            ; no
        MOVLW   '+'             ; yes
        MOVWF   Char
        MOVWF   Oper
        RETURN

done    BSF     PORTC,3         ; clear last row
        CLRF    Char            ; character code = 0
        RETURN


; Write display .....................................

disout  MOVF    Kcode,W         ; Load the code
        BSF     Select,RS       ; Select data mode
        CALL    send            ; and send code
        RETURN

; Process operations ................................

calc    MOVF    Oper,W          ; check for add
        MOVWF   Temp            ; load input op code
        MOVLW   '+'             ; load plus code
        SUBWF   Temp            ; compare
        BTFSC   STATUS,Z        ; and check if same
        GOTO    add             ; yes, jump to op

        MOVF    Oper,W          ; check for subtract
        MOVWF   Temp
        MOVLW   '-'
```

```
        SUBWF   Temp
        BTFSC   STATUS,Z
        GOTO    sub

        MOVF    Oper,W          ; check for multiply
        MOVWF Temp
        MOVLW 'x'
        SUBWF   Temp
        BTFSC   STATUS,Z
        GOTO    mul

        MOVF    Oper,W          ; check for divide
        MOVWF Temp
        MOVLW '/'
        SUBWF   Temp
        BTFSC   STATUS,Z
        GOTO    div
        GOTO    scan            ; rescan if key invalid


; Calclate results from 2 input numbers .....

add     MOVF    Num1,W          ; get first number
        ADDWF  Num2,W           ; add second
        MOVWF Result            ; and store result
        GOTO    outres          ; display result


sub     BSF     STATUS,C        ; Negative detect flag
        MOVF    Num2,W          ; get first number
```

```
        SUBWF   Num1,W          ; subtract second
        MOVWF   Result          ; and store result

        BTFSS   STATUS,C        ; answer negative?
        GOTO    minus           ; yes, minus result
        GOTO    outres          ; display result

minus   MOVLW   '-'             ; load minus sign
        BSF     Select,RS       ; Select data mode
        CALL    send            ; and send symbol

        COMF    Result          ; invert all bits
        INCF    Result          ; add 1
        GOTO    outres          ; display result


mul     MOVF    Num1,W          ; get first number
        CLRF    Result          ; total to Z
add1    ADDWF   Result          ; add to total
        DECFSZ  Num2            ; num2 times and
        GOTO    add1            ; repeat if not done
        GOTO    outres          ; done, display result


div     CLRF    Result          ; total to Z
        MOVF    Num2,W          ; get divisor
        BCF     STATUS,C        ; set C flag
sub1    INCF    Result          ; count loop start
        SUBWF   Num1            ; subtract
        BTFSS   STATUS,Z        ; exact answer?
```

```
        GOTO    neg             ; no
        GOTO    outres          ; yes, display answer
neg     BTFSC   STATUS,C        ; gone negative?
        GOTO    sub1            ; no - repeat
        DECF    Result          ; correct the result
        MOVF    Num2,W          ; get divisor
        ADDWF   Num1            ; calc remainder

        MOVF    Result,W; load result
        ADDLW   030             ; convert to ASCII
        BSF     Select,RS       ; Select data mode
        CALL    send            ; and send result

        MOVLW   'r'             ; indicate remainder
        CALL    send
        MOVF    Num1,W
        ADDLW   030             ; convert to ASCII
        CALL    send
        GOTO    scan


; Convert binary to BCD .....................................

outres  MOVF    Result,W; load result
        MOVWF   Lsd             ; into low digit store
        CLRF    Msd             ; high digit = 0
        BSF     STATUS,C        ; set C flag
        MOVLW   D'10'           ; load 10

again   SUBWF   Lsd             ; sub 10 from result
```

```
        INCF    Msd             ; inc high digit
        BTFSC   STATUS,C        ; check if negative
        GOTO    again           ; no, keep going
        ADDWF   Lsd             ; yes, add 10 back
        DECF    Msd             ; inc high digit


; display 2 digit BCD result .....................

        MOVF    Msd,W           ; load high digit result
        BTFSC   STATUS,Z        ; check if Z
        GOTO    lowd            ; yes, dont display Msd

        ADDLW   030             ; convert to ASCII
        BSF     Select,RS       ; Select data mode
        CALL    send            ; and send Msd

lowd    MOVF    Lsd,W           ; load low digit result
        ADDLW   030             ; convert to ASCII
        BSF     Select,RS       ; Select data mode
        CALL    send            ; and send Msd

        GOTO    scan            ; scan for clear key


; Restart ..........................................

clear   MOVLW   01              ; code to clear display
        BCF     Select,RS       ; Select data mode
        CALL    send            ; and send code
```

```
CLRF    Kcount          ; reset count of keys
GOTO    scan            ; and rescan keypad


END     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```