Central Processor Unit (CPU)

I'm not going to bore you with the operation of the CPU at this stage, however it is important to state that the CPU is manufactured with in RISC technology an important factor when deciding which microprocessor to use.

RISC *Reduced Instruction Set Computer*, gives the PIC16F887 two great advantages:

- The CPU can recognizes only 35 simple instructions (In order to program some other microcontrollers it is necessary to know more than 200 instructions by heart).
- The execution time is the same for all instructions except two and lasts 4 clock cycles (oscillator frequency is stabilized by a quartz crystal). The Jump and Branch instructions execution time is 2 clock cycles. It means that if the microcontroller's operating speed is 20MHz, execution time of each instruc tion will be 200nS, i.e. the program will be executed at the speed of 5 million instructions per second!

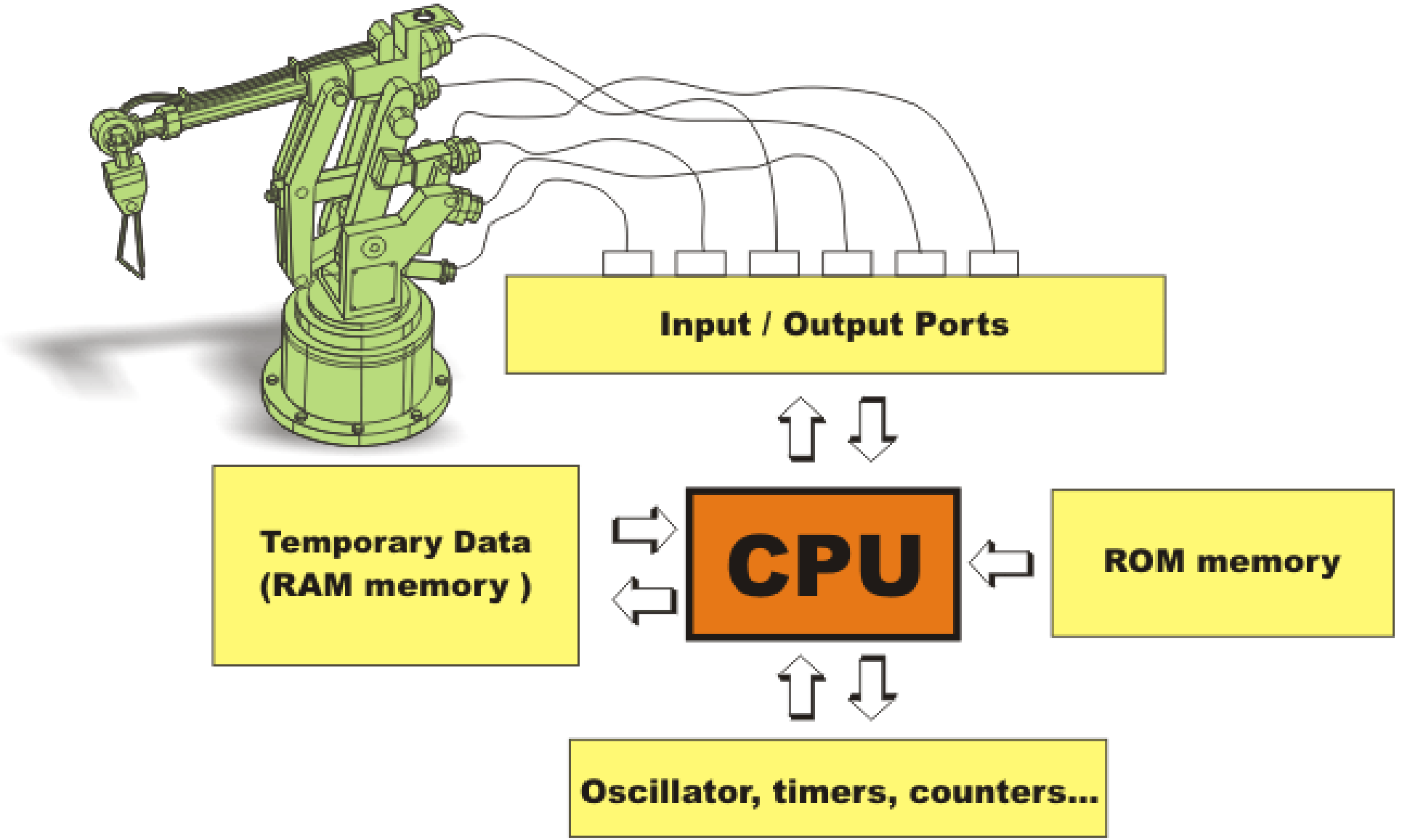


Fig. 1-4 CPU Memory

Memory

This microcontroller has three types of memory- ROM, RAM and EEPROM. All of them will be separately discussed since each has specific functions, features and organization.

ROM Memory

ROM memory is used to permanently save the program being executed. This is why it is often called “program memory”. The PIC16F887 has 8Kb of ROM (in total of 8192 locations). Since this ROM is made with FLASH technology, its contents can be changed by providing a special programming voltage (13V).

Anyway, there is no need to explain it in detail because it is automatically performed by means of a special program on the PC and a simple electronic device called the Programmer.

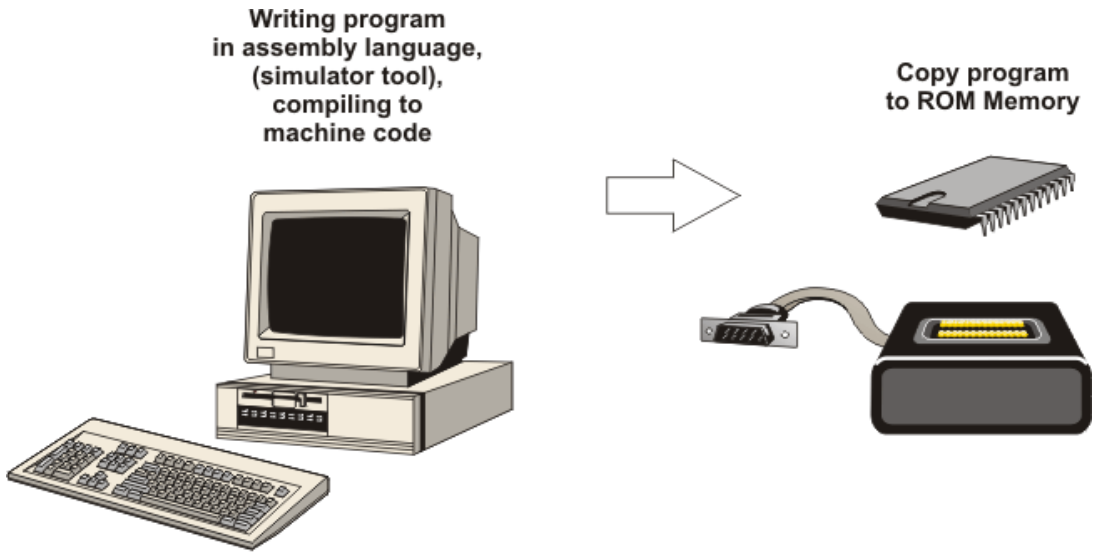


Fig. 1-5 ROM Memory Consept

EEPROM Memory

Similar to program memory, the contents of EEPROM is permanently saved, even the power goes off. However, unlike ROM, the contents of the EEPROM can be changed during operation of the microcontroller. That is why this memory (256 locations) is a perfect one for permanently saving results created and used during the operation.

RAM Memory

This is the third and the most complex part of microcontroller memory. In this case, it consists of two parts: general-purpose registers and special-function registers (SFR).

Even though both groups of registers are cleared when power goes off and even though they are manufactured in the same way and act in the similar way, their functions do not have many things in common.

General-Purpose Registers

General-Purpose registers are used for storing temporary data and results created during operation. For example, if the program performs a counting (for example, counting products on the assembly line), it is necessary to have a register which stands for what we in everyday life call “sum”. Since the microcontroller is not creative at all, it is necessary to specify the address of some general purpose register and assign it a new function. A simple program to increment the value of this register by 1, after each product passes through a sensor, should be created.

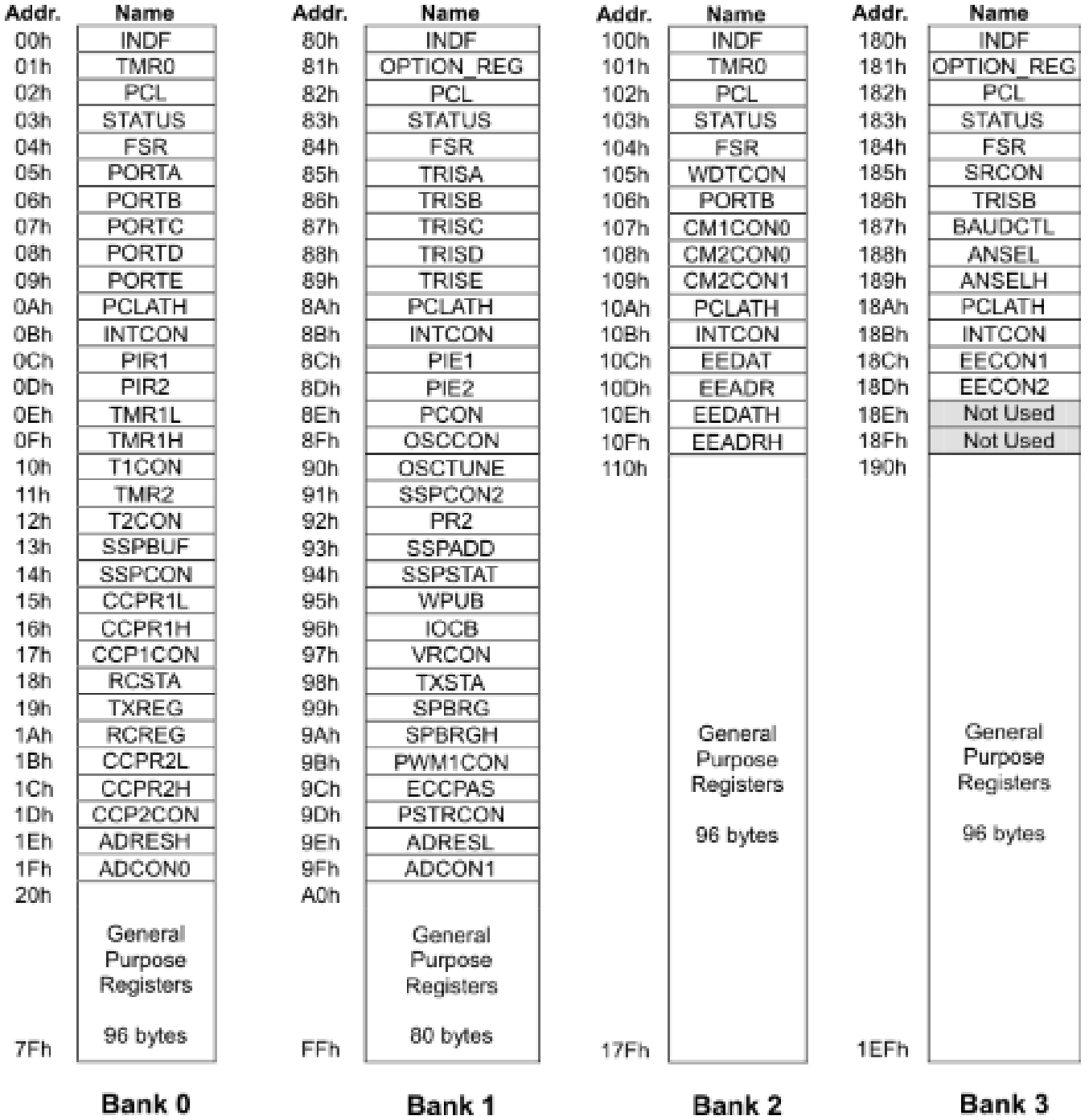
Therefore, the microcontroller can execute that program because it now knows what and where the sum which must be incremented is. Similarly to this simple example, each program variable must be preassigned some of general-purpose register.

SFR Registers

Special-Function registers are also RAM memory locations, but unlike general-purpose registers, their purpose is predetermined during manufacturing process and cannot be changed. Since their bits are physically connected to particular circuits on the chip (A/D converter, serial communication module, etc.), any change of their contents directly affects the operation of the microcontroller or some of its circuits. For example, by changing the TRISA register, the function of each port A pin can be changed in a way it acts as input or output. Another feature of these memory locations is that they have their names (registers and their bits), which considerably facilitates program writing. Since high-level programming language can use the list of all registers with their exact addresses, it is enough to specify the register’s name in order to read or change its contents.

RAM Memory Banks

The data memory is partitioned into four banks. Prior to accessing some register during program writing (in order to read or change its contents), it is necessary to select the bank which contains that register. Two bits of the STATUS register are used for bank selecting, which will be discussed later. In order to facilitate operation, the most commonly used SFRs have the same address in all banks which enables them to be easily accessed.


Addr.
Name
00h INDF
01h TMR0
02h PCL
03h STATUS
04h FSR
05h PORTA
06h PORTB
07h PORTC
08h PORTD
09h PORTE
0Ah PCLATH
0Bh INTCON
0Ch PIR1
0Dh PIR2
0Eh TMR1L
0Fh TMR1H
10h T1CON
11h TMR2
12h T2CON
13h SSPBUF
14h SSPCON
15h CCPR1L
16h CCPR1H
17h CCP1CON
18h RCSTA
19h TXREG
1Ah RCREG
1Bh CCPR2L
1Ch CCPR2H
1Dh CCP2CON
1Eh ADRESH
1Fh ADCON0
20h
General Purpose Registers
96 bytes
7Fh
Bank 0
Addr.
Name
80h INDF
81h OPTION_REG
82h PCL
83h STATUS
84h FSR
85h TRISA
86h TRISB
87h TRISC
88h TRISD
89h TRISE
8Ah PCLATH
8Bh INTCON
8Ch PIE1
8Dh PIE2
8Eh PCON
8Fh OSCCON
90h OSCTUNE
91h SSPCON2
92h PR2
93h SSPADD
94h SSPSTAT
95h WPUB
96h IOCB
97h VRCON
98h TXSTA
99h SPBRG
9Ah SPBRGH
9Bh PWM1CON
9Ch ECCPAS
9Dh PSTRCON
9Eh ADRESL
9Fh ADCON1
A0h
General Purpose Registers
80 bytes
FFh
Bank 1
Addr.
Name
100h INDF
101h TMR0
102h PCL
103h STATUS
104h FSR
105h WDTCON
106h PORTB
107h CM1CON0
108h CM2CON0
109h CM2CON1
10Ah PCLATH
10Bh INTCON
10Ch EEDAT
10Dh EEADR
10Eh EEDATH
10Fh EEADRH
110h
General Purpose Registers
96 bytes
17Fh
Bank 2
Addr.
Name
180h INDF
181h OPTION_REG
182h PCL
183h STATUS
184h FSR
185h SRCON
186h TRISB
187h BAUDCTL
188h ANSEL
189h ANSELH
18Ah PCLATH
18Bh INTCON
18Ch EECON1
18Dh EECON2
18Eh Not Used
18Fh Not Used
190h
General Purpose Registers
96 bytes
1EFh
Bank 3

SFRs bank 0

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
00h	INDF	Indirect register							
01h	TMR0	Timer T0 Register							
02h	PCL	Least Significant Byte of Program Counter							
03h	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C
04h	FSR	Indirect Data Memory Address Pointer							
05h	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
09h	PORTE	-	-	-	-	RE3	RE2	RE1	RE0
0Ah	PCLATH	-	-	-	Upper 5 bits of Program Counter				
0Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0Ch	PIR1	-	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
0Dh	PIR2	OSFIF	C2IF	C1IF	EEIF	BCLIF	ULPWUIF	-	CCP2IF
0Eh	TMR1L	Least Significant Byte of the 16-bit Timer TMR0							
0Fh	TMR1H	Most Significant Byte of the 16-bit Timer TMR0							
10h	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
11h	TMR2	Timer T2 Register							
12h	T2CON	-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register							
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
15h	CCPR1L	Capture/ComparePWM Register 1 Low Byte (LSB)							
16h	CCPR1H	Capture/ComparePWM Register 1 High Byte (LSB)							
17h	CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
19h	TXREG	EUSART Transmit Data Register							
1Ah	RCREG	EUSART Receive Data Register							
1Bh	CCPR2L	Capture/Compare PWM Register 1 Low Byte (LSB)							
1Ch	CCPR2H	Capture/Compare PWM Register 1 High Byte (LSB)							
1Dh	CCP2CON	-	-	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
1Eh	ADRESH	A/D Result Register High Byte							
1Fh	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON

SFRs bank 1

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
80h	INDF	Indirect Register							
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
82h	PCL	Least Significant Byte of Program Counter							
83h	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C
84h	FSR	Indirect Data Memory Address Pointer							
85h	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
87h	TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
88h	TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0
89h	TRISE	-	-	-	-	TRISE3	TRISE2	TRISE1	TRISE0
8Ah	PCLATH	-	-	-	Upper 5 bits of the Ptogram Counter				
8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
8Ch	PIE1	-	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
8Dh	PIE2	OSFIE	C2IE	C1IE	EEIE	BCLIE	ULPWUIE	-	CCP2IE
8Eh	PCON	-	-	ULPWUE	SBOREN	-	-	POR	BOR
8Fh	OSCCON	-	IRCF2	IRCF1	IRCF0	OSTS	HTS	LTS	SCS
90h	OSCTUNE	-	-	-	TUN4	TUN3	TUN2	TUN1	TUN0
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
92h	PR2	Timer T2 Period Register							
93h	SSPADD	Synchronous Serial Port (I^2 C mode) Address Register							
93h	SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0
94h	SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF
95h	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
96h	IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
97h	VRCON	VREN	VROE	VRR	VRSS	VR3	VR2	VR1	VR0
98h	TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
99h	SPBRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
9Ah	SPBRGH	BRG15	BRG14	BRG13	BRG12	BRG11	BRG10	BRG9	BRG8
9Bh	PWM1CON	PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
9Ch	ECCPAS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0
9Dh	PSTRCON	-	-	-	STRSYNC	STRD	STRC	STRB	STRA
9Eh	ADRESL	A/D Result Register Low Byte							
9Fh	ADCON1	ADFM	-	VCFG1	VCFG0	-	-	-	-

SFRs bank 2

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
100h	INDF	Indirect register							
101h	TMR0	Timer T0 Register							
102h	PCL	Least Significant Byte of the Program Counter							
103h	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C
104h	FSR	Indirect Data Memory Address Pointer							
105h	WDTCON	-	-	-	WDTPS3	WDTPS2	WDTPS1	WDTPS0	SWDTEN
106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
107h	CM1CON0	C1ON	C1OUT	C1OE	C1POL	-	C1R	C1CH1	C1CH0
108h	CM2CON0	C2ON	C2OUT	C2OE	C2POL	-	C2R	C2CH1	C2CH0
109h	CM2CON1	MC1OUT	MC2OUT	C1RSEL	C2RSEL	-	-	T1GSS	C2SYNC
10Ah	PCLATH	-	-	-	Upper 5 bits of the Program Counter				
10Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
10Ch	EEDAT	EEDAT7	EEDAT6	EEDAT5	EED AT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0
10Dh	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0
10Eh	EEDATH	-	-	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0
10Fh	EEADRH	-	-	-	EEADRH4	EEADRH3	EEADRH2	EEADRH1	EEADRH0

SFRs bank 3

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
180h	INDF	Indirect Register							
181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
182h	PCL	Least Significan Byte of the Program Counter							
183h	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C
184h	FSR	Indirect Data Memory Address Pointer							
185h	SRCON	SR1	SR0	C1SEN	C2REN	PULSS	PULSR	-	FVREN
186h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
187h	BAUDCTL	ABDOVF	RCIDL	-	SCKP	BRG16	-	WUE	ABDEN
188h	ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
189h	ANSELH	-	-	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
19Ah	PCLATH	-	-	-	Upper 5 bits of the Program Counter				
19Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
19Ch	EECON1	EEPGD	-	-	-	WRERR	WREN	WR	RD
19Dh	EECON2	EEPROM Control Register 2							

STACK

A part of the RAM used for the stack consists of eight 13-bit registers. Before the microcontroller starts to execute a subroutine (`CALL` instruction) or when an interrupt occurs, the address of first next instruction being currently executed is pushed onto the stack, i.e. onto one of its registers. In that way, upon subroutine or interrupt execution, the microcontroller knows from where to continue regular program execution. This address is cleared upon return to the main program because there is no need to save it any longer, and one location of the stack is automatically available for further use.

It is important to understand that data is always circularly pushed onto the stack. It means that after the stack has been pushed eight times, the ninth push overwrites the value that was stored with the first push. The tenth push overwrites the second push and so on. Data overwritten in this way is not recoverable. In addition, the programmer cannot access these registers for write or read and there is no Status bit to indicate stack overflow or stack underflow conditions. For that reason, one should take special care of it during program writing.

Interrupt System

The first thing that the microcontroller does when an interrupt request arrives is to execute the current instruction and then stop regular program execution. Immediately after that, the current program memory address is automatically pushed onto the stack and the default address (predefined by the manufacturer) is written to the program counter. That location from where the program continues execution is called the interrupt vector. For the PIC16F887 microcontroller, this address is 0004h. As seen in Fig. 1-7 below, the location containing interrupt vector is passed over during regular program execution.

Part of the program being activated when an interrupt request arrives is called the interrupt routine. Its first instruction is located at the interrupt vector. How long this subroutine will be and what it will be like depends on the skills of the programmer as well as the interrupt source itself. Some microcontrollers have more interrupt vectors (every interrupt request has its vector), but in this case there is only one. Consequently, the first part of the interrupt routine consists in interrupt source recognition.

Finally, when the interrupt source is recognized and interrupt routine is executed, the microcontroller reaches the `RETFIE` instruction, pops the address from the stack and continues program execution from where it left off.

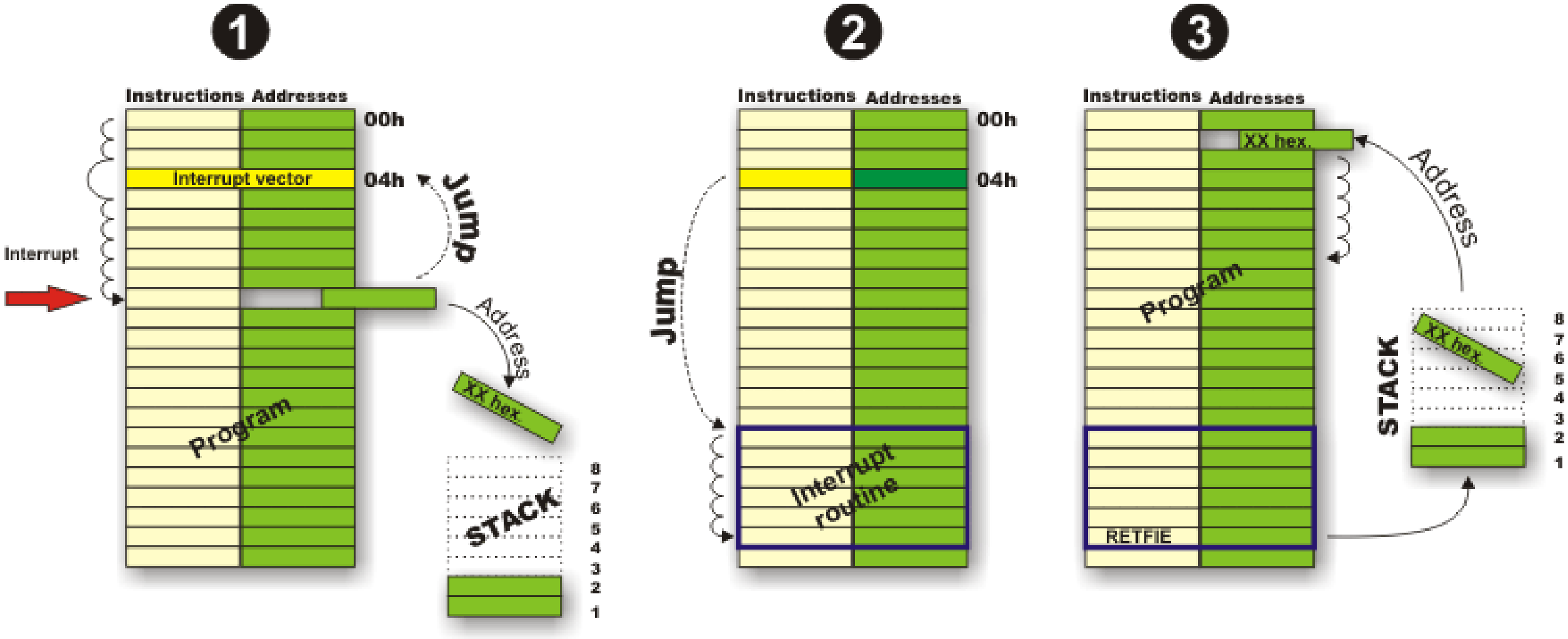


Fig.1-7 Interrupt System

PIC 16F877 Block Diagram