E·XFL



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "<u>Embedded -</u> <u>Microcontrollers</u>"

Details

Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	4MHz
Connectivity	-
Peripherals	Brown-out Detect/Reset, POR, WDT
Number of I/O	13
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SSOP (0.209", 5.30mm Width)
Supplier Device Package	20-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lc622at-04i-ss

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, KEELOQ, MPLAB, PIC, PICmicro, PICSTART, PRO MATE and PowerSmart are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, microID, MXDEV, MXLAB, PICMASTER, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Accuron, Application Maestro, dsPIC, dsPICDEM, dsPICDEM.net, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICC, PICkit, PICDEM, PICDEM.net, PowerCal, PowerInfo, PowerMate, PowerTool, rfLAB, rfPIC, Select Mode, SmartSensor, SmartShunt, SmartTel and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2003, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELoQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.



FIGURE 3-1: BLOCK DIAGRAM



3.1 Clocking Scheme/Instruction Cycle

The clock input (OSC1/CLKIN pin) is internally divided by four to generate four non-overlapping quadrature clocks namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-2.

3.2 Instruction Flow/Pipelining

An "Instruction Cycle" consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO) then two cycles are required to complete the instruction (Example 3-1).

A fetch cycle begins with the program counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the "Instruction Register (IR)" in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).



FIGURE 3-2: CLOCK/INSTRUCTION CYCLE

EXAMPLE 3-1: INSTRUCTION PIPELINE FLOW



4.2 Data Memory Organization

The data memory (Figure 4-4, Figure 4-5, Figure 4-6 and Figure 4-7) is partitioned into two banks, which contain the General Purpose Registers and the Special Function Registers. Bank 0 is selected when the RP0 bit is cleared. Bank 1 is selected when the RP0 bit (STATUS <5>) is set. The Special Function Registers are located in the first 32 locations of each bank. Register locations 20-7Fh (Bank0) on the PIC16C620A/CR620A/621A and 20-7Fh (Bank0) and A0-BFh (Bank1) on the PIC16C622 and PIC16C622A are General Purpose Registers implemented as static RAM. Some Special Purpose Registers are mapped in Bank 1.

Addresses F0h-FFh of bank1 are implemented as common ram and mapped back to addresses 70h-7Fh in bank0 on the PIC16C620A/621A/622A/CR620A.

4.2.1 GENERAL PURPOSE REGISTER FILE

The register file is organized as 80 x 8 in the PIC16C620/621, 96 x 8 in the PIC16C620A/621A/CR620A and 128 x 8 in the PIC16C622(A). Each is accessed either directly or indirectly through the File Select Register FSR (Section 4.4).

4.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and Peripheral functions for controlling the desired operation of the device (Table 4-1). These registers are static RAM. The Special Function Registers can be classified into two sets (core and peripheral). The Special Function Registers associated with the "core" functions are described in this section. Those related to the operation of the peripheral features are described in the section of that peripheral feature.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset	Value on all other RESETS ⁽¹⁾
Bank 0											
00h	INDF	Addressin register)	ig this locat	on uses co	ntents of FS	SR to addre	ess data me	mory (not a	a physical	XXXX XXXX	XXXX XXXX
01h	TMR0	Timer0 Mo	odule's Reg	ister						xxxx xxxx	uuuu uuuu
02h	PCL	Program (Counter's (F	PC) Least S	Significant B	yte				0000 0000	0000 0000
03h	STATUS	IRP ⁽²⁾	RP1 ⁽²⁾	RP0	TO	PD	Z	DC	С	0001 1xxx	000q quuu
04h	FSR	Indirect da	ata memory	address po	ointer					xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	—	RA4	RA3	RA2	RA1	RA0	x 0000	u 0000
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
07h-09h	Unimplemented									_	_
0Ah	PCLATH	—	—	—	Write buffe	er for upper	5 bits of pr	ogram coui	nter	0 0000	0 0000
0Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	CMIF	—	—	—	—	—	—	-0	-0
0Dh-1Eh	Unimplemented									_	_
1Fh	CMCON	C2OUT	C10UT	—	—	CIS	CM2	CM1	CM0	00 0000	00 0000
Bank 1											
80h	INDF	Addressin register)	g this locat	ion uses co	ntents of FS	SR to addre	ess data me	mory (not a	a physical	xxxx xxxx	xxxx xxxx
81h	OPTION	RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Program (Counter's (F	PC) Least S	ignificant B	yte				0000 0000	0000 0000
83h	STATUS	IRP ⁽²⁾	RP1 ⁽²⁾	RP0	TO	PD	Z	DC	С	0001 1xxx	000q quuu
84h	FSR	Indirect da	ata memory	address po	ointer					xxxx xxxx	uuuu uuuu
85h	TRISA	-	-	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1 1111	1 1111
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
87h-89h	Unimplemented									_	_
8Ah	PCLATH	-	-	—	Write buffe	er for upper	5 bits of pr	ogram coui	nter	0 0000	0 0000
8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
8Ch	PIE1	—	CMIE	—	—	—	—	—	—	-0	-0
8Dh	Unimplemented									_	_
8Eh	PCON	_	_	_	_	—	—	POR	BOR	0x	uq
8Fh-9Eh	Unimplemented						-		-	_	_
9Fh	VRCON	VREN	VROE	VRR	_	VR3	VR2	VR1	VR0	000- 0000	000- 0000

TABLE 4-1: SPECIAL REGISTERS FOR THE PIC16C62X

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown,

 ${\rm q}$ = value depends on condition, shaded = unimplemented

Note 1: Other (non Power-up) Resets include MCLR Reset, Brown-out Reset and Watchdog Timer Reset during normal operation.

2: IRP & RP1 bits are reserved; always maintain these bits clear.

4.4 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses data pointed to by the File Select Register (FSR). Reading INDF itself indirectly will produce 00h. Writing to the INDF register indirectly results in a no-operation (although STATUS bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 4-9. However, IRP is not used in the PIC16C62X.

A simple program to clear RAM location 20h-7Fh using indirect addressing is shown in Example 4-1.

EXAN	IPLE 4-	1: INI	DIRECT ADDRESSING
	movlw	0x20	;initialize pointer
	movwf	FSR	;to RAM
NEXT	clrf	INDF	clear INDF register;
	incf	FSR	;inc pointer
	btfss	FSR,7	;all done?
	goto	NEXT	;no clear next
			;yes continue
CONTI	NUE:		

FIGURE 4-9: DIRECT/INDIRECT ADDRESSING PIC16C62X











6.2 Using Timer0 with External Clock

When an external clock input is used for Timer0, it must meet certain requirements. The external clock requirement is due to internal phase clock (Tosc) synchronization. Also, there is a delay in the actual incrementing of Timer0 after synchronization.

6.2.1 EXTERNAL CLOCK SYNCHRONIZATION

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 (Figure 6-5). Therefore, it is necessary for T0CKI to be high for at least 2Tosc (and a small RC delay of 20 ns) and low for at least 2Tosc (and a small RC delay of 20 ns). Refer to the electrical specification of the desired device. When a prescaler is used, the external clock input is divided by the asynchronous ripple-counter type prescaler, so that the prescaler output is symmetrical. For the external clock to meet the sampling requirement, the ripple-counter must be taken into account. Therefore, it is necessary for TOCKI to have a period of at least 4Tosc (and a small RC delay of 40 ns) divided by the prescaler value. The only requirement on TOCKI high and low time is that they do not violate the minimum pulse width requirement of 10 ns. Refer to parameters 40, 41 and 42 in the electrical specification of the desired device.

6.2.2 TIMER0 INCREMENT DELAY

Since the prescaler output is synchronized with the internal clocks, there is a small delay from the time the external clock edge occurs to the time the TMR0 is actually incremented. Figure 6-5 shows the delay from the external clock edge to the timer incrementing.





7.4 Comparator Response Time

Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output has a valid level. If the internal reference is changed, the maximum delay of the internal voltage reference must be considered when using the comparator outputs. Otherwise the maximum delay of the comparators should be used (Table 12-2).

7.5 Comparator Outputs

The comparator outputs are read through the CMCON register. These bits are read only. The comparator outputs may also be directly output to the RA3 and RA4 I/O pins. When the CM<2:0> = 110, multiplexors in the output path of the RA3 and RA4 pins will switch and the output of each pin will be the unsynchronized output of the comparator. The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications. Figure 7-3 shows the comparator output block diagram.

The TRISA bits will still function as an output enable/ disable for the RA3 and RA4 pins while in this mode.

- Note 1: When reading the PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert an analog input according to the Schmitt Trigger input specification.
 - 2: Analog levels on any pin that is defined as a digital input may cause the input buffer to consume more current than is specified.

FIGURE 7-3: COMPARATOR OUTPUT BLOCK DIAGRAM





FIGURE 9-9: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2



FIGURE 9-10: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD)



TABLE 10-2: PIC16C62X INSTRUCTION S

Mnemonic,		Description	Cycles		14-Bit	Opcode	9	Status	Notes
Operands				MSb			LSb	Affected	
BYTE-ORIE	NTED I	FILE REGISTER OPERATIONS							
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0000	0011	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	С	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	С	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENT	ED FIL	E REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL A	ND COI	NTROL OPERATIONS	-					-	-
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	TO,PD	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into Standby mode	1	00	0000	0110	0011	TO,PD	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., MOVF PORTB, 1), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

DECFSZ	Decrement f, Skip if 0							
Syntax:	[label] DECFSZ f,d							
Operands:	$0 \le f \le 127$ $d \in [0,1]$							
Operation:	(f) - 1 \rightarrow (dest); skip if result = 0							
Status Affected:	None							
Encoding:	00 1011 dfff ffff							
Description:	I ne contents of register 't' are decremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 0, the next instruc- tion, which is already fetched, is discarded. A NOP is executed instead making it a two-cycle instruction.							
Words:	1							
Cycles:	1(2)							
Example	HERE DECFSZ CNT, 1 GOTO LOOP CONTINUE • • •							
	After Instruction CNT = CNT - 1 if CNT = 0, PC = address CONTINUE if CNT ≠ 0, PC = address HERE+1							
GOTO	Unconditional Branch							
Syntax:	[<i>label</i>] GOTO k							
Operands:	$0 \leq k \leq 2047$							
Operation:	k → PC<10:0> PCLATH<4:3> → PC<12:11>							
Status Affected:	None							
Encoding:	10 1kkk kkkk kkkk							
Description:	GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two- cycle instruction.							
Words:	1							
Cycles:	2							
Example	GOTO THERE							
	After Instruction PC = Address THERE							

INCF	Increment f						
Syntax:	[<i>label</i>] INCF f,d						
Operands:	$\begin{array}{l} 0 \leq f \leq 127 \\ d \in [0,1] \end{array}$						
Operation:	(f) + 1 \rightarrow (dest)						
Status Affected:	Z						
Encoding:	00 1010 dfff ffff						
Description:	I he contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.						
Words:	1						
Cycles:	1						
Example	INCF CNT, 1						
	Before Instruction CNT = 0xFF Z = 0 After Instruction CNT = 0x00 Z = 1						

INCFSZ	Increment f, Skip if 0	IORWF	Inclusive OR W with f			
Syntax:	[label] INCFSZ f,d	Syntax:	[<i>label</i>] IORWF f,d			
Operands:	$0 \le f \le 127$ $d \in [0,1]$	Operands:	$\begin{array}{l} 0 \leq f \leq 127 \\ d \in [0,1] \end{array}$			
Operation:	(f) + 1 \rightarrow (dest), skip if result = 0	Operation:	(W) .OR. (f) \rightarrow (dest)			
Status Affected:	None	Status Affected:	Z			
Encoding:	00 1111 dfff ffff	Encoding:	00 0100 dfff ffff			
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.	Description:	Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.			
	If the result is 0, the next instruc-	Words:	1			
	discarded. A NOP is executed	Cycles:	1			
	instead making it a two-cycle	Example	IORWF RESULT, 0			
	Instruction.		Before Instruction			
vvoras:	1		$\begin{array}{rcl} RESULI &= & 0x13 \\ W &= & 0x91 \end{array}$			
Cycles: Example	1(2) HERE INCFSZ CNT, 1 GOTO LOOP CONTINUE • •		After Instruction $\begin{array}{rcl} RESULT &= & 0x13 \\ W & = & 0x93 \\ Z & = & 1 \end{array}$			
	Before Instruction	MOVLW	Move Literal to W			
	PC = address HERE After Instruction	Syntax:	[<i>label</i>] MOVLW k			
	CNT = CNT + 1	Operands:	$0 \le k \le 255$			
	if $CNT = 0$, PC = address CONTINUE	Operation:	$k \rightarrow (W)$			
	if $CNT \neq 0$,	Status Affected:	None			
	PC = address HERE +1	Encoding:	11 00xx kkkk kkkk			
IORLW	Inclusive OR Literal with W	Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.			
Syntax:	[<i>label</i>] IORLW k	Words:	1			
Operands:	$0 \le k \le 255$	Cycles:	1			
Operation:	(W) .OR. $k \rightarrow$ (W)	Example	MOVLW 0x5A			
Status Affected:	Z	·	After Instruction			
Encoding:	11 1000 kkkk kkkk		W = 0x5A			
Description:	The contents of the W register is OR'ed with the eight bit literal 'k'. The result is placed in the W register.					
Words:	1					
Cycles:	1					
Example	IORLW 0x35					
	Before Instruction W = 0x9A					
	After Instruction					

W = Z =

0xBF 1

RLF	Rotate	Left f th	oug	h Car	ry		
Syntax:	[label]	RLF	f,d			I	
Operands:	0 ≤ f ≤ 1 d ∈ [0,1	27]					
Operation:	See des	scription	belo	w			
Status Affected:	С						
Encoding:	00	1101	d	fff	ffff]	
Description:	rotated the Carr is place 1, the re register	The contents of register 't' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is stored back in register 'f'.					
Words:	1						
Cycles:	1						
Example	RLF	REG1,	0				
Before Instruction							
		REG1	=	111	0 0110		
	After In	C	=	0			
	7 1101 111	REG1	=	111	0 0110		
		W	=	110	0 1100		
		С	=	1			

RRF	Rotate Right f through Carry								
Syntax:	[label]	RRF f	,d						
Operands:	$\begin{array}{l} 0\leq f\leq 12\\ d\in \left[0,1\right] \end{array}$	27							
Operation:	See description below								
Status Affected:	С								
Encoding:	00	1100	df	ff	ffff				
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.								
			Regis	ter f]->				
Words:	1								
Cycles:	1								
Example	RRF		REG 0	61,					
	Before In	structior	ı						
		REG1	=	1110	0110				
	After Inst	ruction	=	U					
		REG1	=	1110	0110				
		W	=	0111	0011				
		C	=	0					

SLEEP

Syntax:	[label]	SLEEF	D					
Operands:	None							
Operation:	$\begin{array}{l} 00h \rightarrow WDT, \\ 0 \rightarrow WDT \text{ prescaler,} \\ 1 \rightarrow \overline{TO}, \\ 0 \rightarrow PD \end{array}$							
Status Affected:	TO, PD							
Encoding:	00	0000	0110	0011				
Description:	The power-down STATUS bit, PD is cleared. Time-out STATUS bit, TO is set. Watch- dog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped. See Section 9.8 for more datails							
Words:	1							
Cycles:	1							
Example:	SLEEP							

11.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI C compilers for Microchip's PIC17CXXX and PIC18CXXX family of microcontrollers. These compilers provide powerful integration capabilities, superior code optimization and ease of use not found with other compilers.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

11.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can link relocatable objects from pre-compiled libraries, using directives from a linker script.

The MPLIB object librarian manages the creation and modification of library files of pre-compiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

11.5 MPLAB C30 C Compiler

The MPLAB C30 C compiler is a full-featured, ANSI compliant, optimizing compiler that translates standard ANSI C programs into dsPIC30F assembly language source. The compiler also supports many command-line options and language extensions to take full advantage of the dsPIC30F device hardware capabilities, and afford fine control of the compiler code generator.

MPLAB C30 is distributed with a complete ANSI C standard library. All library functions have been validated and conform to the ANSI C library standard. The library includes functions for string manipulation, dynamic memory allocation, data conversion, time-keeping, and math functions (trigonometric, exponential and hyperbolic). The compiler provides symbolic information for high level source debugging with the MPLAB IDE.

11.6 MPLAB ASM30 Assembler, Linker, and Librarian

MPLAB ASM30 assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 compiler uses the assembler to produce it's object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire dsPIC30F instruction set
- · Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- · MPLAB IDE compatibility

11.7 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC hosted environment by simulating the PICmicro series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user defined key press, to any pin. The execution can be performed in Single-Step, Execute Until Break, or Trace mode.

The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C17 and MPLAB C18 C Compilers, as well as the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent, economical software development tool.

11.8 MPLAB SIM30 Software Simulator

The MPLAB SIM30 software simulator allows code development in a PC hosted environment by simulating the dsPIC30F series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user defined key press, to any of the pins.

The MPLAB SIM30 simulator fully supports symbolic debugging using the MPLAB C30 C Compiler and MPLAB ASM30 assembler. The simulator runs in either a Command Line mode for automated tasks, or from MPLAB IDE. This high speed simulator is designed to debug, analyze and optimize time intensive DSP routines.

11.9 MPLAB ICE 2000 High Performance Universal In-Circuit Emulator

The MPLAB ICE 2000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PICmicro microcontrollers. Software control of the MPLAB ICE 2000 in-circuit emulator is advanced by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PICmicro microcontrollers.

The MPLAB ICE 2000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft[®] Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

11.10 MPLAB ICE 4000 High Performance Universal In-Circuit Emulator

The MPLAB ICE 4000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for highend PICmicro microcontrollers. Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICD 4000 is a premium emulator system, providing the features of MPLAB ICE 2000, but with increased emulation memory and high speed performance for dsPIC30F and PIC18XXXX devices. Its advanced emulator features include complex triggering and timing, up to 2 Mb of emulation memory, and the ability to view variables in real-time.

The MPLAB ICE 4000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

11.11 MPLAB ICD 2 In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD 2, is a powerful, low cost, run-time development tool, connecting to the host PC via an RS-232 or high speed USB interface. This tool is based on the FLASH PICmicro MCUs and can be used to develop for these and other PICmicro microcontrollers. The MPLAB ICD 2 utilizes the in-circuit debugging capability built into the FLASH devices. This feature, along with Microchip's In-Circuit Serial Programming[™] (ICSP[™]) protocol, offers cost effective in-circuit FLASH debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by setting breakpoints, single-stepping and watching variables, CPU status and peripheral registers. Running at full speed enables testing hardware and applications in real-time. MPLAB ICD 2 also serves as a development programmer for selected PICmicro devices.

11.12 PRO MATE II Universal Device Programmer

The PRO MATE II is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features an LCD display for instructions and error messages and a modular detachable socket assembly to support various package types. In Stand-Alone mode, the PRO MATE II device programmer can read, verify, and program PICmicro devices without a PC connection. It can also set code protection in this mode.

11.13 PICSTART Plus Development Programmer

The PICSTART Plus development programmer is an easy-to-use, low cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient. The PICSTART Plus development programmer supports most PICmicro devices up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus development programmer is CE compliant.

12.2 DC Characteristics: PIC16C62XA-04 (Commercial, Industrial, Extended) PIC16C62XA-20 (Commercial, Industrial, Extended) PIC16LC62XA-04 (Commercial, Industrial, Extended (CONT.)

PIC16C	62XA	Stand Opera	dard O ating te	perati empera	n g Con iture -4 -4	ditions (unless otherwise stated) $10^{\circ}C \leq TA \leq +85^{\circ}C$ for industrial and $0^{\circ}C \leq TA \leq +70^{\circ}C$ for commercial and $10^{\circ}C \leq TA \leq +125^{\circ}C$ for extended	
PIC16LC62XA				dard O ating te	perati empera	ng Con ature -4 -4	$\begin{array}{llllllllllllllllllllllllllllllllllll$
Param. No.	Sym	Characteristic	Min	Тур†	Max	Units	Conditions
D022	ΔİWDT	WDT Current ⁽⁵⁾	—	6.0	10 12	μA μA	VDD = 4.0V (125°C)
D022A	Δ IBOR	Brown-out Reset Current ⁽⁵⁾	—	75	125	μA	BOD enabled, VDD = 5.0V
D023		Comparator Current for each Comparator ⁽⁵⁾	_	30	60	μA	VDD = 4.0V
D023A	ΔIVREF	VREF Current ⁽³⁾	_	80	135	μA	VDD = 4.0V
D022	ΔI WDT	WDT Current ⁽⁵⁾	—	6.0	10	μΑ	VDD=4.0V
DOODA	41	Descent Descet Operation (5)		75	12	μA	$\frac{(125^{\circ}C)}{200} = 5.0$
D022A		Brown-out Reset Current ^(e)		75	125	μΑ	BOD enabled, $VDD = 5.0V$
D023	AICOMP	Comparator Current for each		30	60	μΑ	VDD - 4.0V
D023A	Δ IVREF	VREF Current ⁽⁵⁾	_	80	135	μA	VDD = 4.0V
1A	Fosc	LP Oscillator Operating Frequency	0	—	200	kHz	All temperatures
		RC Oscillator Operating Frequency	0	—	4	MHz	All temperatures
		XT Oscillator Operating Frequency	0		4	MHz	All temperatures
		HS Oscillator Operating Frequency	U	—	20	MHZ	All temperatures
1A	Fosc	LP Oscillator Operating Frequency	0	—	200	kHz	All temperatures
		RC Oscillator Operating Frequency	0	—	4	MHz	All temperatures
		HS Oscillator Operating Frequency	0	_	4 20	MHZ MHZ	All temperatures All temperatures

These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in Active Operation mode are:

OSC1 = external square wave, from rail to rail; all I/O pins tri-stated, pulled to VDD,

 $\overline{\text{MCLR}}$ = VDD; WDT enabled/disabled as specified.

3: The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD or VSS.

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula: Ir = VDD/2REXT (mA) with REXT in kΩ.

5: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD or IPD measurement.

6: Commercial temperature range only.













NOTES: