The following three pages contain the 80386 instruction set. You do not need to submit these sheets when you finish.

Remember that:

- Most instructions can have at most one memory operand.
- Brackets [ ] around a register name, immediate, or combination of the two indicates an effective address. That address is in the data segment unless otherwise specified.
  - Example: MOV AX, [10H] → contents of DS:10H moved to AX
- Parentheses around a logical address mean "the contents of memory at this address".
  - Example: (DS:10H) → the contents of memory at logical address DS:10H

Category	Instruction	Example	Meaning
Data transfer	Move	`MOV AX, BX`	`AX = BX`
	Move & sign-extend	`MOVSX EAX, DL`	`EAX = DL, sign-extended to 32 bits`
	Move and zero-extend	`MOVZX EAX, DL`	`EAX = DL, zero-extended to 32 bits`
	Exchange	`XCHG AX, BX`	`Swap contents of AX, BX`
	Load effective address	`LEA AX, [BX+SI+10H]`	`AX = BX + SI + 10H`

	Load full pointer	`LDS AX, [10H]`
		
		`LSS EBX, [100H]`
			`AX = (DS:10H)`
			`DS = (DS:12H)`
			
			`EBX = (DS:100H)`
			`SS = (DS:104H)`

Arithmetic	Add	`ADD AX, BX`	`AX = AX + BX`
	Add with carry	`ADC AX, BX`	`AX = AX + BX + CF`
	Increment	`INC [DI]`	`(DS:DI) = (DS:DI) + 1`
	Subtract	`SUB AX, [10H]`	`AX = AX – (DS:10H)`
	Subtract with borrow	`SBB AX, [10H]`	`AX = AX – (DS:10H) – CF`
	Decrement	`DEC CX`	`CX = CX – 1`
	Negate (2's complement)	`NEG CX`	`CX = -CX`

	Unsigned multiply (all operands are non-negative, regardless of MSB value)	`MUL BH`
		`MUL CX`
		`MUL DWORD PTR [10H]`
			`AX = BH * AL`
			`(DX,AX) = CX * AX`
			`(EDX,EAX) = (DS:10H) * EAX`

	Signed multiply (all operands are signed integers in 2's complement form)	`IMUL BH`
		`IMUL CX`
		`IMUL DWORD PTR[10H]`
			`AX = BH * AL`
			`(DX,AX) = CX * AX`
			`(EDX,EAX) = (DS:10H) * EAX`

	Unsigned divide	`DIV BH`
		
		`DIV CX`
		
		`DIV EBX`
			`AL = AX / BH (quotient)`
			`AH = AX % BH (remainder)`
			
			`AX = EAX / CX (quotient)`
			`DX = EAX % CX (remainder)`
			
			`EAX = (EDX,EAX) / EBX (Q)`
			`EDX = (EDX,EAX) % EBX (R)`

Category	Instruction	Example	Meaning
Logical	Logical AND	`AND AX, BX`	`AX = AX & BX`
	Logical inclusive OR	`OR AX, BX`	`AX = AX | BX`
	Logical exclusive OR	`XOR AX, BX`	`AX = AX ^ BX`
	Logical NOT (1's complement)	`NOT AX`	`AX = ~AX`

Shift/rotate (NOTE: for all instructions except RCL/RCR, CF = last bit shifted out)	Shift left	`SHL AX, 7`
		
		`SAL AX, CX`
			`AX = AX << 7`
			
			`AX = AX << CX`

	Logical shift right (treat value as unsigned, shift in 0s)	`SHR AX, 7`	`AX = AX >> 7` `(upper 7 bits = 0)`
	Arithmetic shift right (treat value as signed; maintain sign)	`SAR AX, 7`	`AX = AX >> 7` `(upper 7 bits = MSB of original value)`
	Rotate left	`ROL AX, 7`	`AX = AX rotated left by 7` `(lower 7 bits of AX = upper 7 bits of original value)`
	Rotate right	`ROR AX, 7`	`AX=AX rotated right by 7` `(upper 7 bits of AX = lower 7 bits of original value)`
	Rotate left through carry	`RCL AX, 7`	`(CF,AX) rotated left by 7` `(Treat CF & AX as 17-bit value with CF as MSB)`
	Rotate right through carry	`RCR AX, 7`	`(AX,CX) rotated right by 7` `(Treat CF & AX as 17-b8t value with CF as LSB)`
Bit test/ scan	Bit test	`BT AX, 7`	`CF = Value of bit 7 of AX`
	Bit test and reset	`BTR AX, 7`	`CF = Value of bit 7 of AX` `Bit 7 of AX = 0`
	Bit test and set	`BTS AX, 7`	`CF = Value of bit 7 of AX` `Bit 7 of AX = 1`
	Bit test and complement	`BTC AX, 7`	`CF = Value of bit 7 of AX` `Bit 7 of AX is flipped`
	Bit scan forward	`BSF DX, AX`	`DX = index of first non-zero bit of AX, starting with bit 0` `ZF = 0 if AX = 0, 1 otherwise`
	Bit scan reverse	`BSR DX, AX`	`DX = index of first non-zero bit of AX, starting with MSB` `ZF = 0 if AX = 0, 1 otherwise`

Category	Instruction	Example	Meaning
Flag control	Clear carry flag	`CLC`	`CF = 0`
	Set carry flag	`STC`	`CF = 1`
	Complement carry flag	`CMC`	`CF = ~CF`
	Clear interrupt flag	`CLI`	`IF = 0`
	Set interrupt flag	`STI`	`IF = 1`
	Load AH with contents of flags register	`LAHF`	`AH = FLAGS`
	Store contents of AH in flags register	`SAHF`	`FLAGS = AH` `(Updates SF,ZF,AF,PF,CF)`
Conditional tests	Compare	`CMP AX, BX`	`Subtract AX – BX` `Updates flags`
	Byte set on condition	`SETcc AH`	`AH = FF if condition true` `AH = 0 if condition false`
Jumps and loops	Unconditional jump	`JMP label`	`Jump to label`
	Conditional jump	`Jcc label`	`Jump to label if condition true`
	Loop	`LOOP label`	`Decrement CX; jump to label if CX != 0`
	Loop if equal/zero	`LOOPE label` `LOOPZ label`	`Decrement CX; jump to label if (CX != 0) && (ZF == 1)`
	Loop if not equal/zero	`LOOPNE label` `LOOPNZ label`	`Decrement CX; jump to label if (CX != 0) && (ZF == 0)`
Subroutine-related instructions	Call subroutine	`CALL label`	`Jump to label; save address of instruction after CALL`
	Return from subroutine	`RET label`	`Return from subroutine (jump to saved address from CALL)`

	Push	`PUSH AX`
		
		`PUSH EAX`
			`SP = SP – 2`
			`(SS:SP) = AX`
			
			`SP = SP – 4`
			`(SS:SP) = EAX`

	Pop	`POP AX`
		
		`POP EAX`
			`AX = (SS:SP)`
			`SP = SP + 2`
			
			`EAX = (SS:SP)`
			`SP = SP + 4`

	Push flags	`PUSHF`	`Store flags on stack`
	Pop flags	`POPF`	`Remove flags from stack`
	Push all registers	`PUSHA`	`Store all general purpose registers on stack`
	Pop all registers	`POPA`	`Remove general purpose registers from stack`

The following three pages contain tables listing the PIC 16F684 instruction set, memory map, and block diagram.

Remember that, in the table below:

- f = a register file address
- W = the working register
- d = destination select: “F” for a file register, “W” for the working register
- b = bit position within an 8-bit file register
- k = literal field, constant data or label
- PC = the program counter
- C = the carry bit in the status register
- Z = the zero bit in the status register

TABLE 13-2: PIC16F684 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED 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	0xxx	xxxx	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	C	1, 2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	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-ORIENTED FILE 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 AND CONTROL 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	$\overline{TO}$, $\overline{PD}$	
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	$\overline{TO}$, $\overline{PD}$	
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	

FIGURE 2-2: DATA MEMORY MAP OF THE PIC16F684

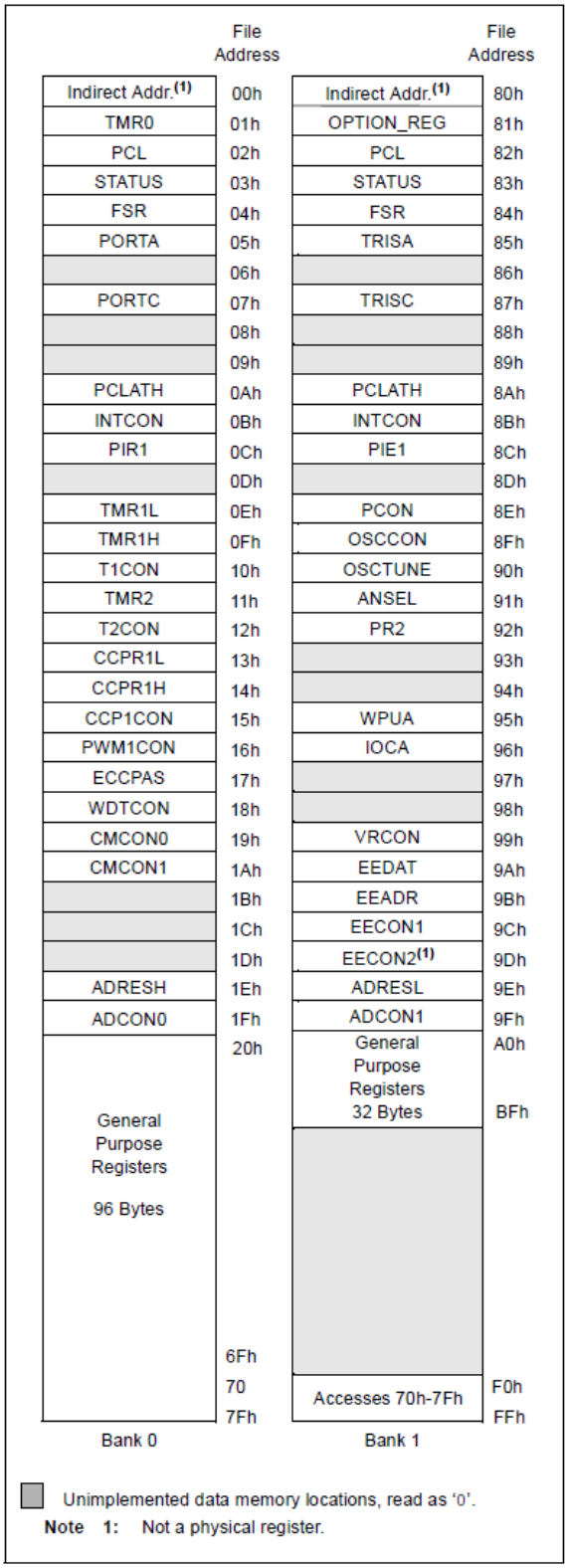
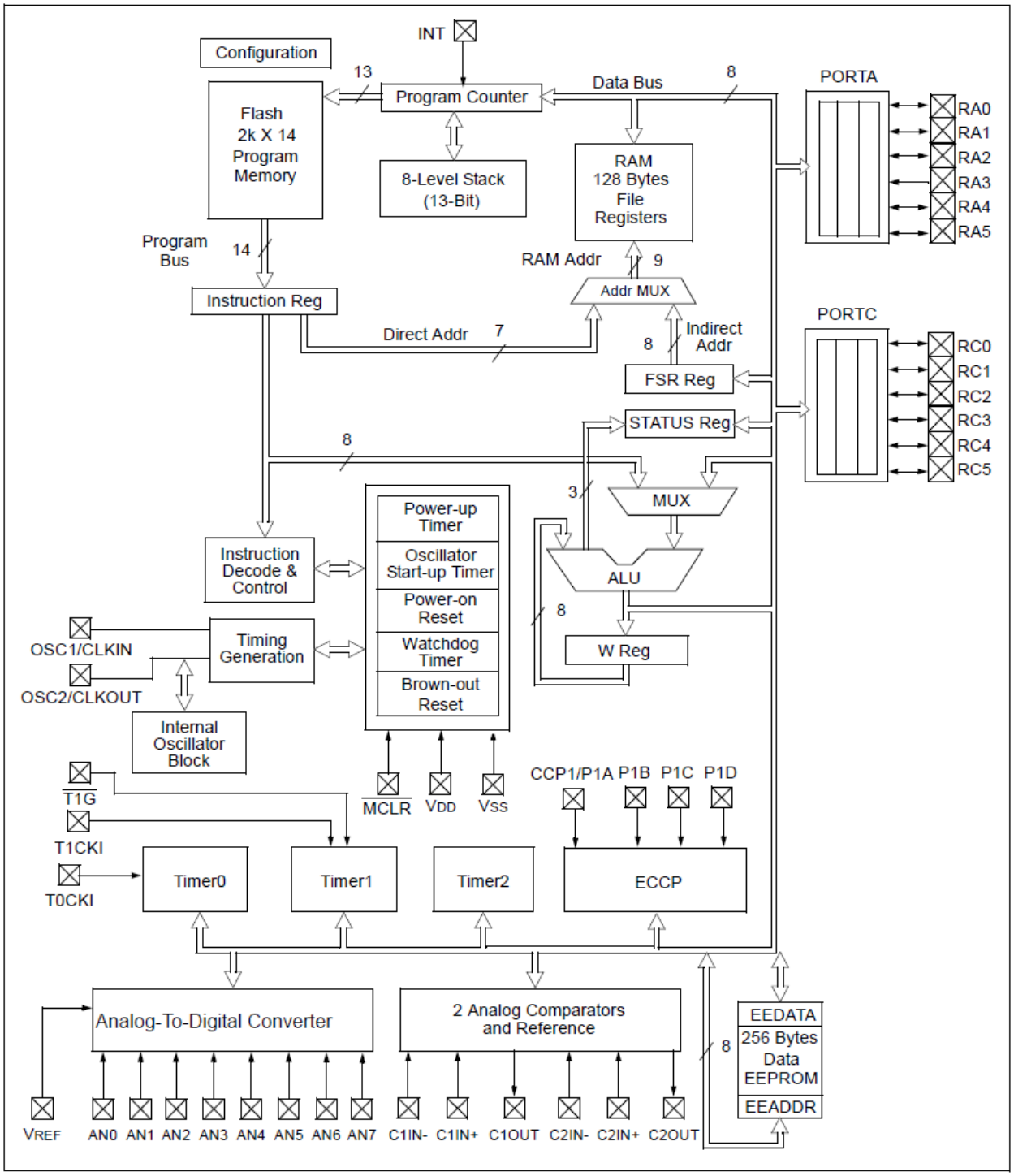


FIGURE 1-1: PIC16F684 BLOCK DIAGRAM



Source for all figures: *"PIC 16F684 Data Sheet", Microchip Technology, Inc. http://ww1.microchip.com/downloads/en/DeviceDoc/41202F-print.pdf*