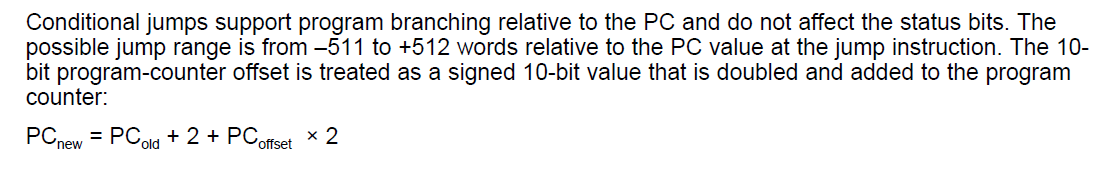
**ECE 382 Embedded Systems I**

****

Word Operations (W) = 0

Byte Operations (B) = 1

**Table 1. Source Addressing Modes (As)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address Mode** | **\*As** | **Registers** | **Syntax** | **Operation** |
| **Register** | **00** | **R0-R2, R4-R15** | **R*n*** | **Register Contents.** |
| **0** | **00** | **R3** | **#0** | **0 Constant** |
| **Symbolic** | **01** | **R0** | ***addr*** | **(PC+next word) points to operand. (*x*(PC))** |
| **Indexed** | **01** | **R1, R4-R15** | ***x*(R*n*)** | **(R*n*+*x*) points to operand. *x* is next code word.** |
| **Absolute** | **01** | **R2** | **&*addr*** | **Next code word is the absolute address. (*x*(SR))** |
| **+1** | **01** | **R3** | **#1** | **+1 Constant** |
| **Indirect** | **10** | **R0-R1,R4-R15** | **@R*n*** | **R*n* points to operand.** |
| **+4** | **10** | **R2** | **#4** | **+4 Constant** |
| **+2** | **10** | **R3** | **#2** | **+2 Constant** |
| **Immediate** | **11** | **R0** | **#*N*** | **Next word is the constant *N*. (@PC+)** |
| **Indirect auto-inc** | **11** | **R1,R4-R15** | **@R*n*+** | **R*n* points to operand, R*n* is incremented (1 or 2).** |
| **+8** | **11** | **R2** | **#8** | **+8 Constant** |
| **-1** | **11** | **R3** | **#-1** | **-1 Constant** |

**\*Bits 4 and 5 in Single (Table 3) and Double (Table 5) Operand Instructions**

**Table 2. Destination Addressing Modes (Ad)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address Mode** | **\*Ad** | **Registers** | **Syntax** | **Operation** |
| **Register** | **0** | **R0-R2, R4-R15** | **R*n*** | **Register Contents.** |
| **0** | **0** | **R3** | **#0** | **Bit bucket** |
| **Symbolic** | **1** | **R0** | ***addr*** | **(PC+next word) points to operand. (*x*(PC))** |
| **Indexed** | **1** | **R1, R4-R15** | ***x*(R*n*)** | **(R*n*+*x*) points to operand. *x* is next code word.** |
| **Absolute** | **1** | **R2** | **&*addr*** | **Next code word is the absolute address. (*x*(SR))** |

**\*Bit 7 in (Table 5) Operand Instructions**

**Table 3. Single Operand Instructions**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **9-bit Opcode** | | | | | | | | | **b/w** | **As** | | **D/S Register** | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Mnemonic** | **Opcode** | | | | | | | | | **V** | **N** | **Z** | **C** | **Operation** | **Description** |
| **RRC** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **•** | **•** | **•** | **•** | **C→MSB→…LSB→C** | **Roll dst right through C** |
| **SWPB** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **–** | **–** | **–** | **–** | **Swap bytes** | **Swap bytes** |
| **RRA** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **•** | **•** | **•** | **MSB→MSB→…LSB→C** | **Roll destination right** |
| **SXT** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **0** | **•** | **•** | **z** | **bit 7→bit 8…bit 15** | **Sign extend destination** |
| **PUSH** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **0** | **0** | **–** | **–** | **–** | **–** | **SP-2→SP, src→@SP** | **Push source on stack** |
| **CALL** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **0** | **1** | **–** | **–** | **–** | **–** | **SP-2→SP, PC+2→@SP, dst→PC** | **Subroutine call** |
| **RETI** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **•** | **•** | **•** | **•** | **@SP+→SR, @SP+→PC** | **Return from interrupt** |

**Table 4. Jump Instructions**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **6-bit Opcode** | | | | | | **10-bit, 2’s complement PC Offset** | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Mnemonic** | **Opcode** | | | | | | **V** | **N** | **Z** | **C** | **Description** |
| **JNZ/JNE** | **0** | **0** | **1** | **0** | **0** | **0** | **–** | **–** | **0** | **–** | **Jump if not equal** |
| **JZ/JEQ** | **0** | **0** | **1** | **0** | **0** | **1** | **–** | **–** | **1** | **–** | **Jump if equal** |
| **JNC/JLO** | **0** | **0** | **1** | **0** | **1** | **0** | **–** | **–** | **–** | **0** | **Jump if carry flag equal to zero** |
| **JC/JHS** | **0** | **0** | **1** | **0** | **1** | **1** | **–** | **–** | **–** | **1** | **Jump if carry flag equal to one** |
| **JN** | **0** | **0** | **1** | **1** | **0** | **0** | **–** | **1** | **–** | **–** | **Jump if negative (N = 1)** |
| **JGE** | **0** | **0** | **1** | **1** | **0** | **1** | **•** | **•** | **–** | **–** | **Jump if greater than or equal (N = V)** |
| **JL** | **0** | **0** | **1** | **1** | **1** | **0** | **•** | **•** | **–** | **–** | **Jump if lower (N ≠ V)** |
| **JMP** | **0** | **0** | **1** | **1** | **1** | **1** | **–** | **–** | **–** | **–** | **Unconditional jump** |

**Table 5. Double Operand Instructions**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **4-bit Opcode** | | | | **Source Register** | | | | **Ad** | **b/w** | **As** | | **Destination Register** | | | |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Mnemonic** | **Opcode** | | | | **V** | **N** | **Z** | **C** | **Operation** | **Description** |
| **MOV** | **0** | **1** | **0** | **0** | **–** | **–** | **–** | **–** | **src→dst** | **Move source to destination** |
| **ADD** | **0** | **1** | **0** | **1** | **•** | **•** | **•** | **•** | **src+dst→dst** | **Add source to destination** |
| **ADDC** | **0** | **1** | **1** | **0** | **•** | **•** | **•** | **•** | **src+dst+C→dst** | **Add src and C to dst** |
| **SUBC** | **0** | **1** | **1** | **1** | **•** | **•** | **•** | **•** | **dst+.not.src+C→dst** | **Subtract src and NOT C from dst** |
| **SUB** | **1** | **0** | **0** | **0** | **•** | **•** | **•** | **•** | **dst+.not.src+1→dst** | **Subtract source from destination** |
| **CMP** | **1** | **0** | **0** | **1** | **•** | **•** | **•** | **•** | **dst-src** | **Compare source to destination** |
| **DADD** | **1** | **0** | **1** | **0** | **•** | **•** | **•** | **•** | **src+dst+C→dst(dec)** | **Decimal add src and C to dst** |
| **BIT** | **1** | **0** | **1** | **1** | **0** | **•** | **•** | **z** | **src.and.dst** | **Test bits in destination** |
| **BIC** | **1** | **1** | **0** | **0** | **–** | **–** | **–** | **–** | **.not.src.and.dst→dst** | **Clear bits in destination** |
| **BIS** | **1** | **1** | **0** | **1** | **–** | **–** | **–** | **–** | **src.or.dst→dst** | **Set bits in destination** |
| **XOR** | **1** | **1** | **1** | **0** | **•** | **•** | **•** | **z** | **src.xor.dst→dst** | **XOR source with destination** |
| **AND** | **1** | **1** | **1** | **1** | **0** | **•** | **•** | **z** | **src.and.dst→dst** | **AND source with destination** |

**Status Register: • = bit affected, – = bit not affected, 0 = cleared, 1 = set, z = same as Z**

**Table 6. Source Operands Using Status (R2) and Constant Generator (R3) Registers**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Register** | **As** | **Addr Mode** | **Syntax** | **Constant** | **Remarks** |
| **R2** | **00** | **Register** | **–** | **–** | **Register mode** |
| **R2** | **01** | ***x*(R2)** | ***addr*** | **(0)** | **Absolute address mode, next word contains address** |
| **R2** | **10** | **@R2** | **#4** | **0x0004** | **+4** |
| **R2** | **11** | **@R2+** | **#8** | **0x0008** | **+8** |
| **R3** | **00** | **R3** | **#0** | **0x0000** | **0** |
| **R3** | **01** | ***x*(R3)** | **#1** | **0x0001** | **+1, No extension word** |
| **R3** | **10** | **@R3** | **#2** | **0x0002** | **+2** |
| **R3** | **11** | **@R3+** | **#-1** | **0xFFFF** | **-1** |

**Mnemonic Description Operation V N Z C**

ADC(.B) dst Add C to destination dst + C → dst \* \* \* \*

ADD(.B) src,dst Add source to destination src + dst → dst \* \* \* \*



ADDC(.B) src,dst Add source and C to destination src + dst + C → dst \* \* \* \*

AND(.B) src,dst AND source and destination src .and. dst → dst 0 \* \* \*

BIC(.B) src,dst Clear bits in destination .not.src .and. dst → dst - - - -

BIS(.B) src,dst Set bits in destination src .or. dst → dst - - - -

BIT(.B) src,dst Test bits in destination src .and. dst 0 \* \* \*

BR dst Branch to destination dst → PC - - - -

CALL dst Call destination PC+2 → stack, dst → PC - - - -

CLR(.B) dst Clear destination 0 → dst - - - -

CLRC Clear C 0 → C - - - 0

CLRN Clear N 0 → N - 0 - -

CLRZ Clear Z 0 → Z - - 0 -

CMP(.B) src,dst Compare source and destination dst - src \* \* \* \*

DADC(.B) dst Add C decimally to destination dst + C → dst (decimally) \* \* \* \*

DADD(.B) src,dst Add source and C decimally to dst src + dst + C → dst (decimally) \* \* \* \*

DEC(.B) dst Decrement destination dst - 1 → dst \* \* \* \*

DECD(.B) dst Double-decrement destination dst - 2 → dst \* \* \* \*

DINT Disable interrupts 0 → GIE - - - -

EINT Enable interrupts 1 → GIE - - - -

INC(.B) dst Increment destination dst +1 → dst \* \* \* \*

INCD(.B) dst Double-increment destination dst+2 → dst \* \* \* \*

INV(.B) dst Invert destination .not.dst → dst \* \* \* \*

JC/JHS label Jump if C set/Jump if higher or same - - - -

JEQ/JZ label Jump if equal/Jump if Z set - - - -

JGE label Jump if greater or equal - - - -

JL label Jump if less - - - -

JMP label Jump PC + 2 × offset → PC - - - -

JN label Jump if N set - - - -

JNC/JLO label Jump if C not set/Jump if lower - - - -

JNE/JNZ label Jump if not equal/Jump if Z not set - - - -

MOV(.B) src,dst Move source to destination src → dst - - - -

NOP No operation - - - -

POP(.B) dst Pop item from stack to destination @SP → dst, SP+2 → SP - - - -

PUSH(.B) src Push source onto stack SP - 2 → SP, src → @SP - - - -

RET Return from subroutine @SP → PC, SP + 2 → SP - - - -

RETI Return from interrupt \* \* \* \*

RLA(.B) dst Rotate left arithmetically \* \* \* \*

RLC(.B) dst Rotate left through C \* \* \* \*

RRA(.B) dst Rotate right arithmetically 0 \* \* \*

RRC(.B) dst Rotate right through C \* \* \* \*

SBC(.B) dst Subtract not(C) from destination dst + 0FFFFh + C → dst \* \* \* \*

SETC Set C 1 → C - - - 1

SETN Set N 1 → N - 1 - -

SETZ Set Z 1 → Z - - 1 -

SUB(.B) src,dst Subtract source from destination dst + .not.src + 1 → dst \* \* \* \*

SUBC(.B) src,dst Subtract source and not(C) from dst dst + .not.src + C → dst \* \* \* \*

SWPB dst Swap bytes - - - -

SXT dst Extend sign 0 \* \* \*

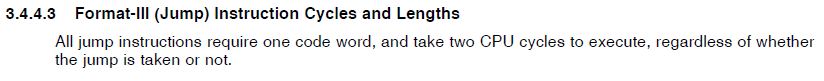
TST(.B) dst Test destination dst + 0FFFFh + 1 0 \* \* 1

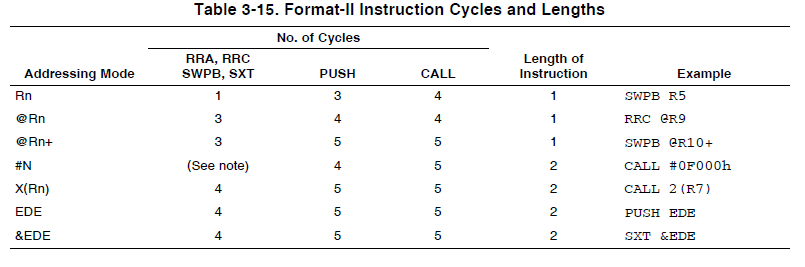
XOR(.B) src,dst Exclusive OR source and destination src .xor. dst → dst \* \* \* \*

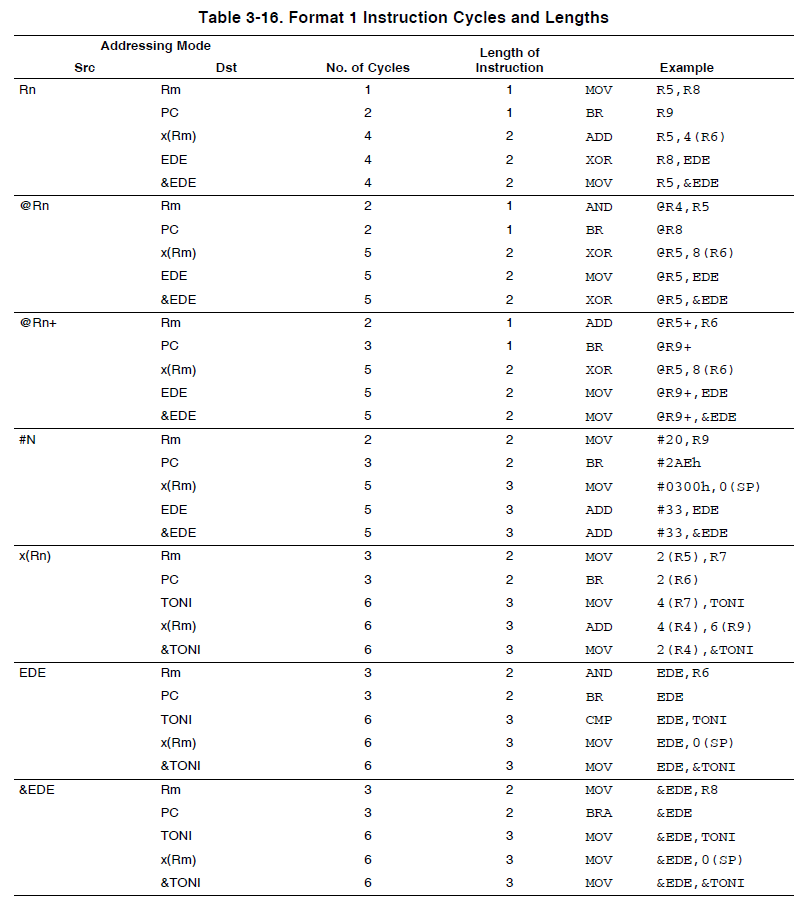
Legend: \* = Status bit cleared or set on results - = Status bit not affected

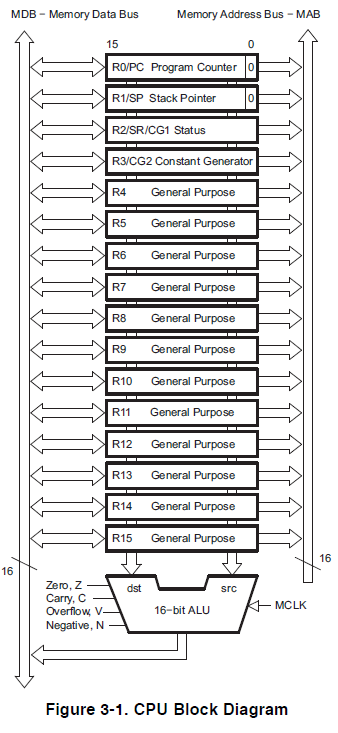
0 = Status bit always cleared 1 = Status bit always set

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 | 0 | 0 | 1 | 0 | 0 | Opcode | | | W=0/B=1 | As | | Dest Reg | | | |
| 0 | 0 | 1 | Condition | | | PC Offset (10-Bit) | | | | | | | | | |
| Opcode | | | | Source Reg | | | | Ad | W=0/B=1 | As | | Dest Reg | | | |

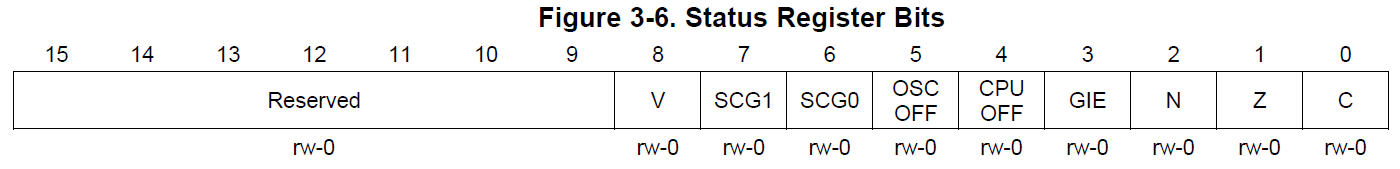






****

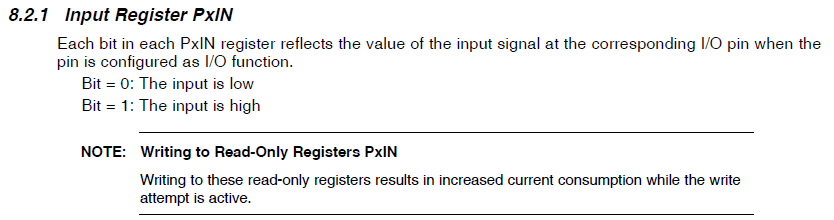
03FFh

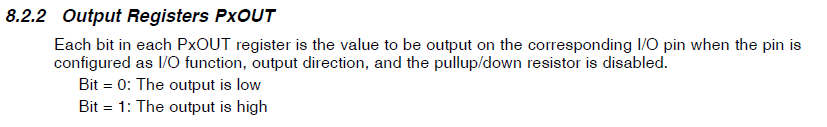
****

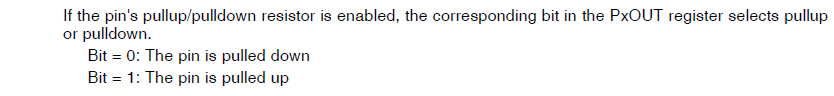
****

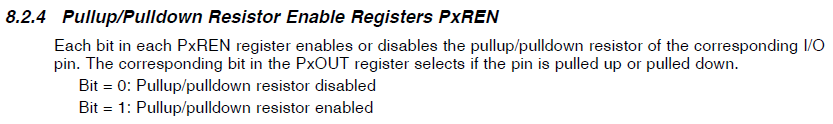
LED1 is P1.0

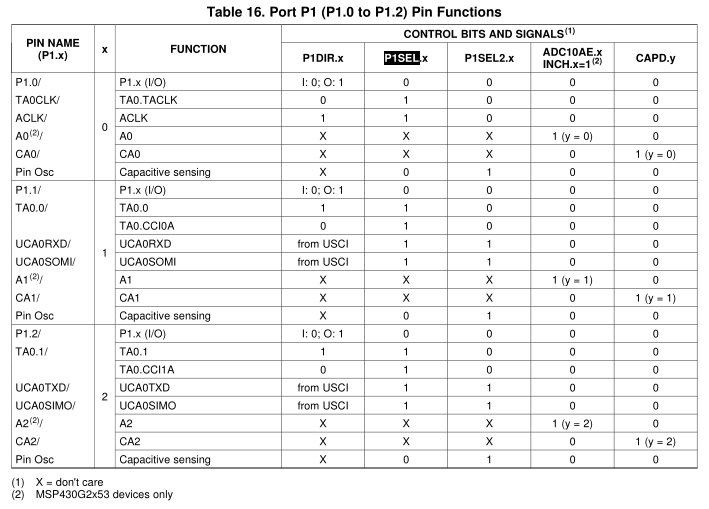
LED2 is P1.6

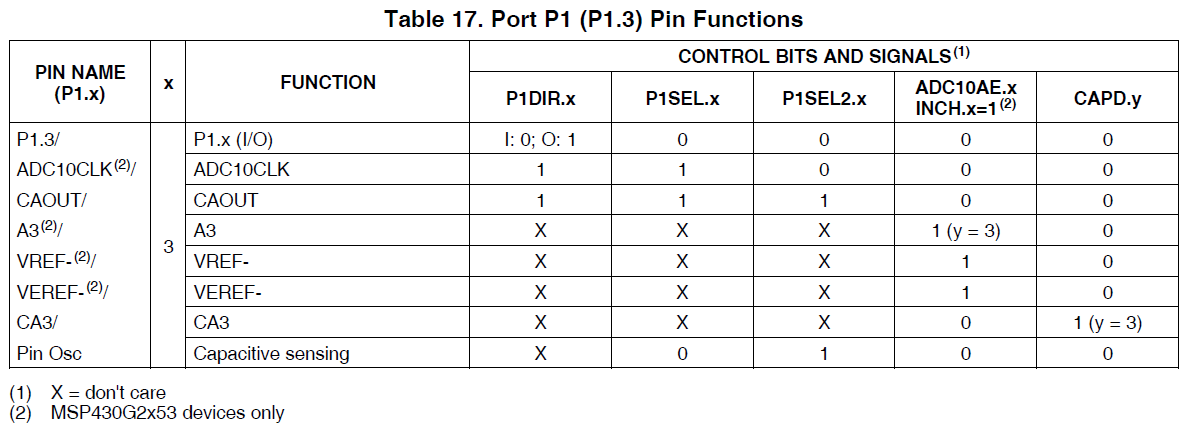


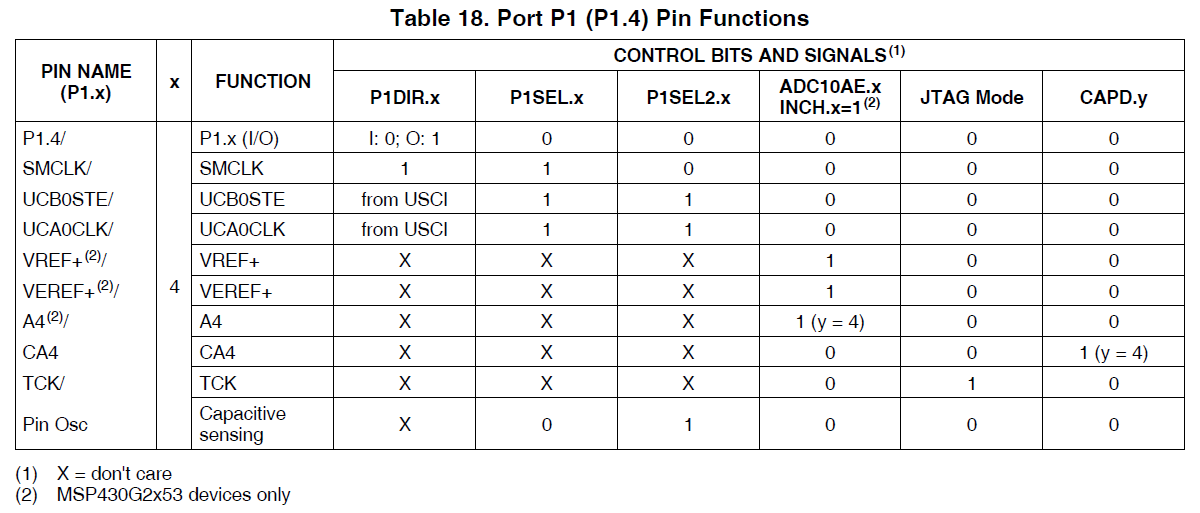


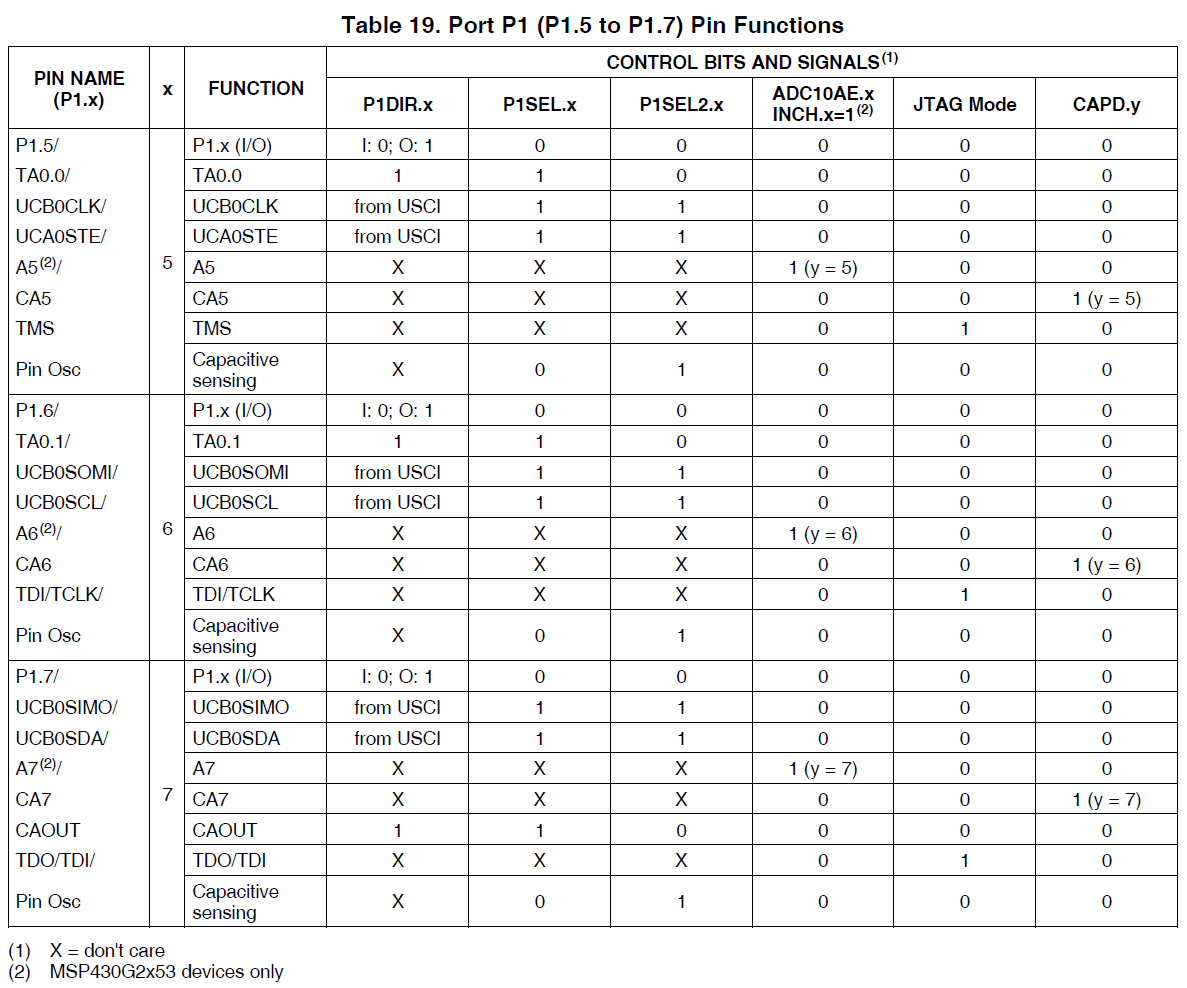


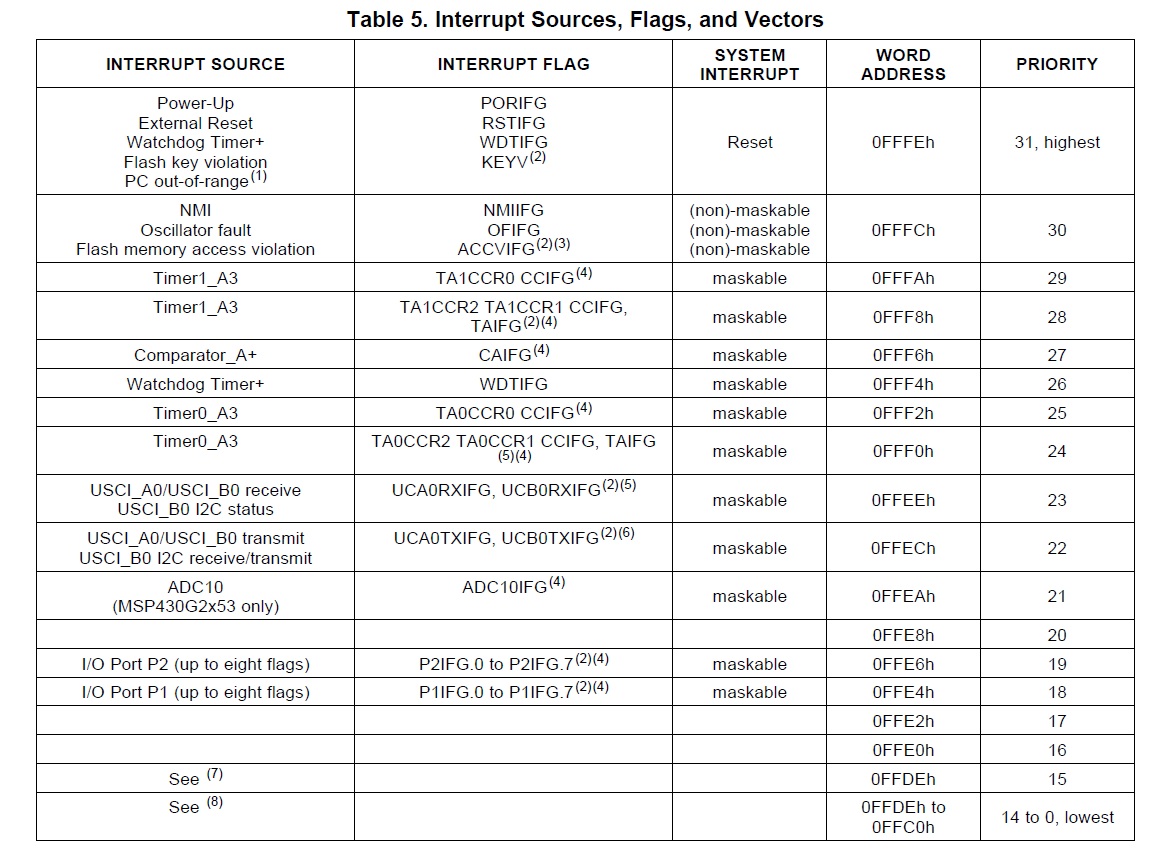


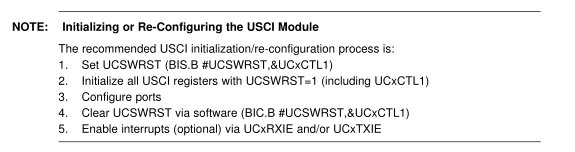
****

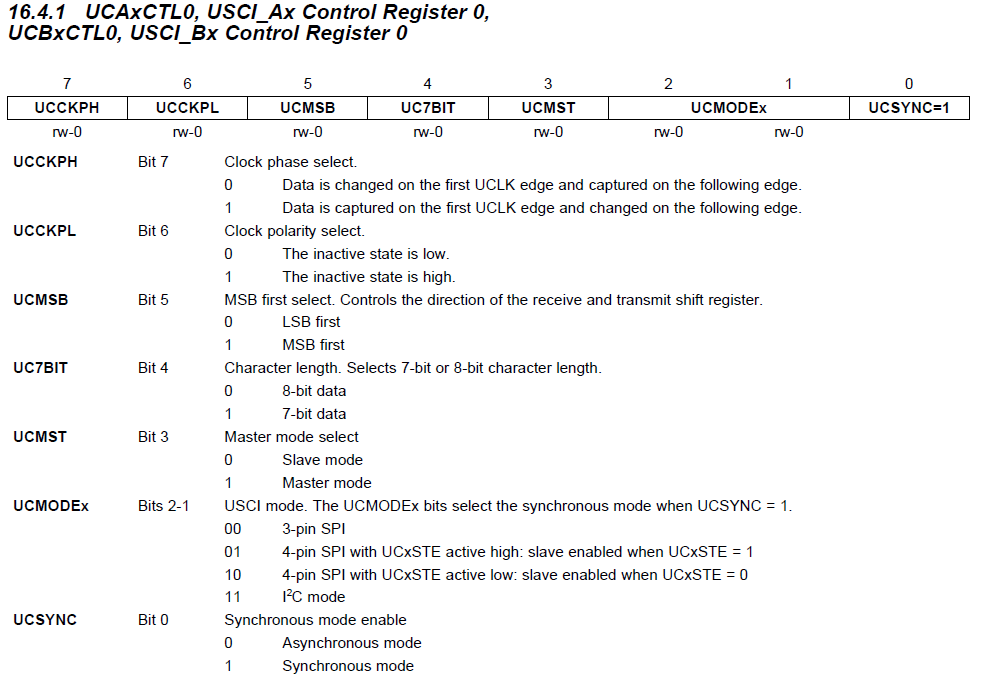


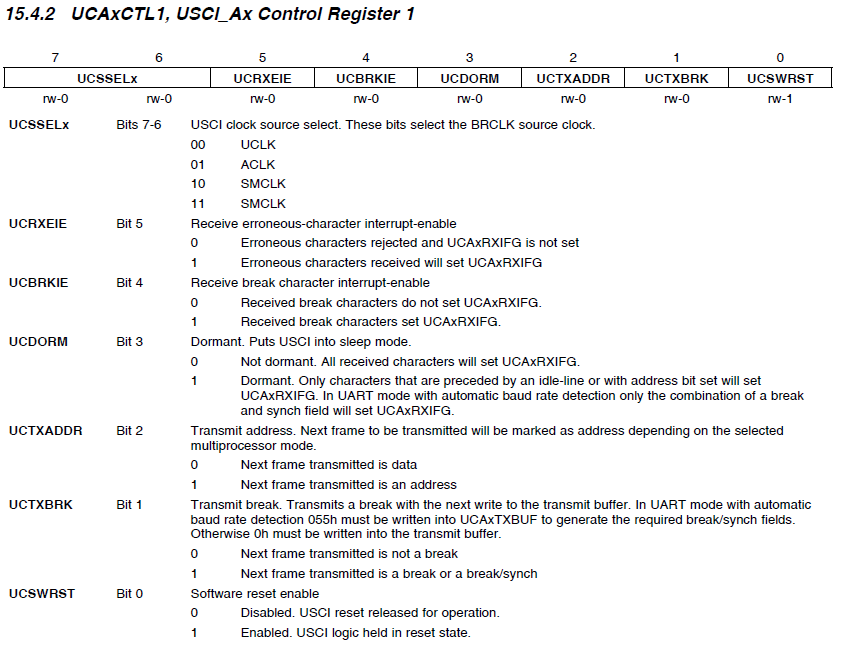


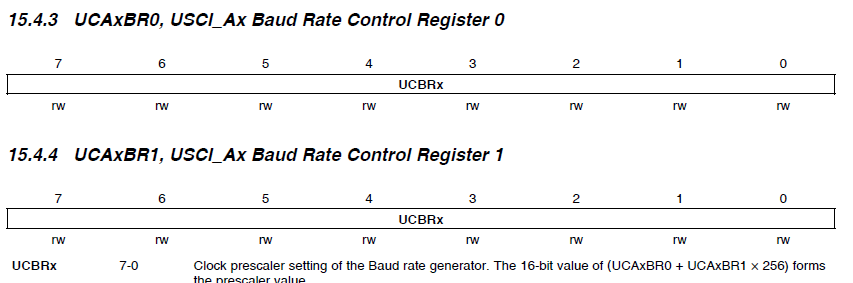


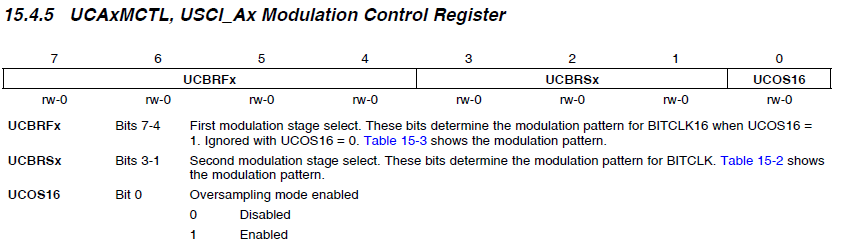
****

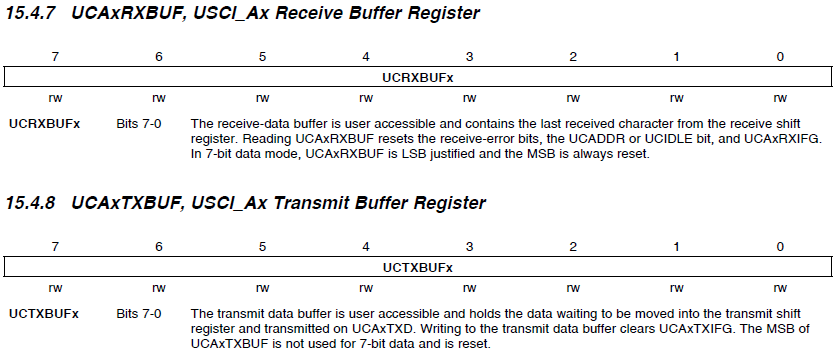
****

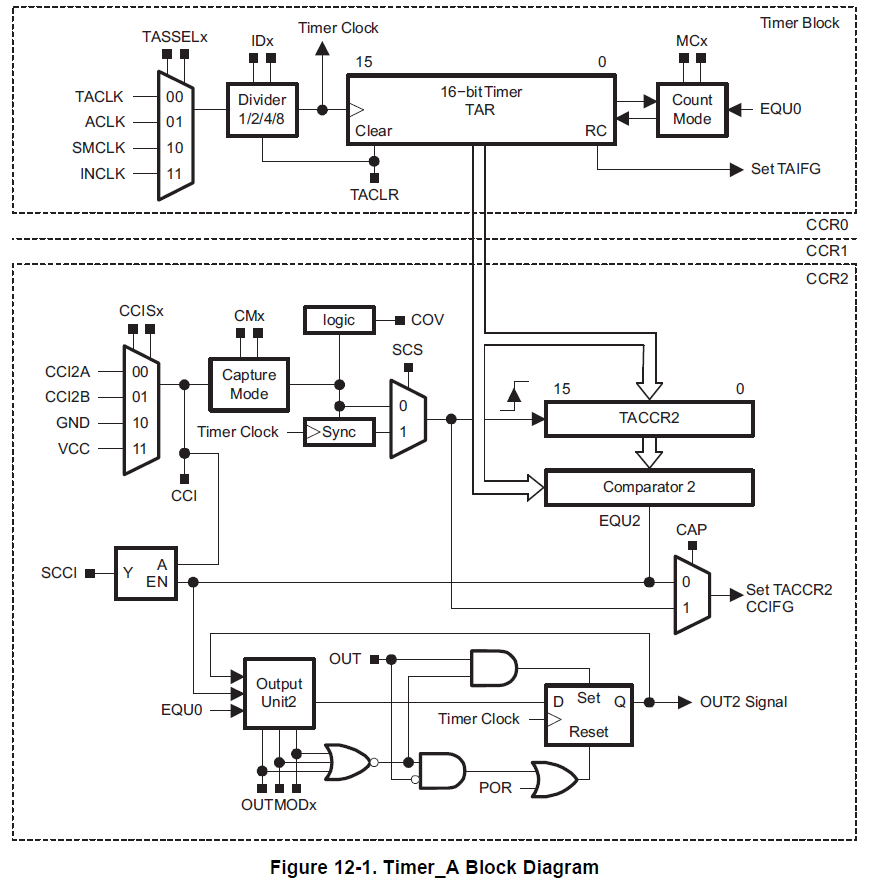
****

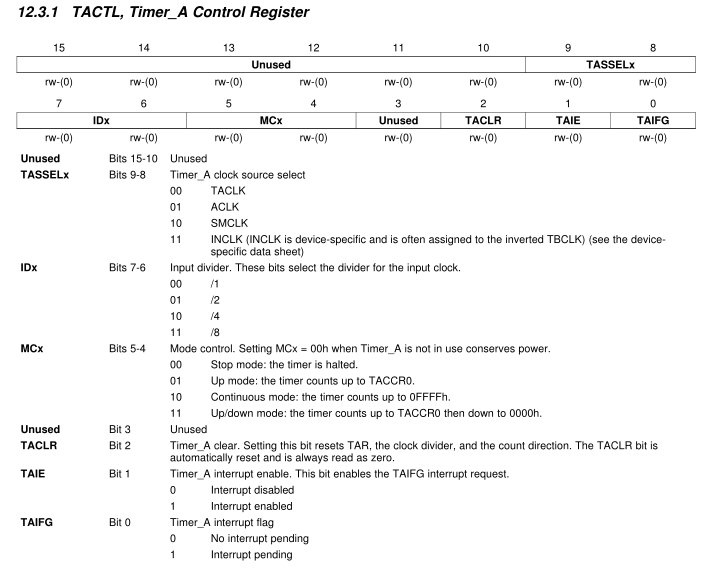


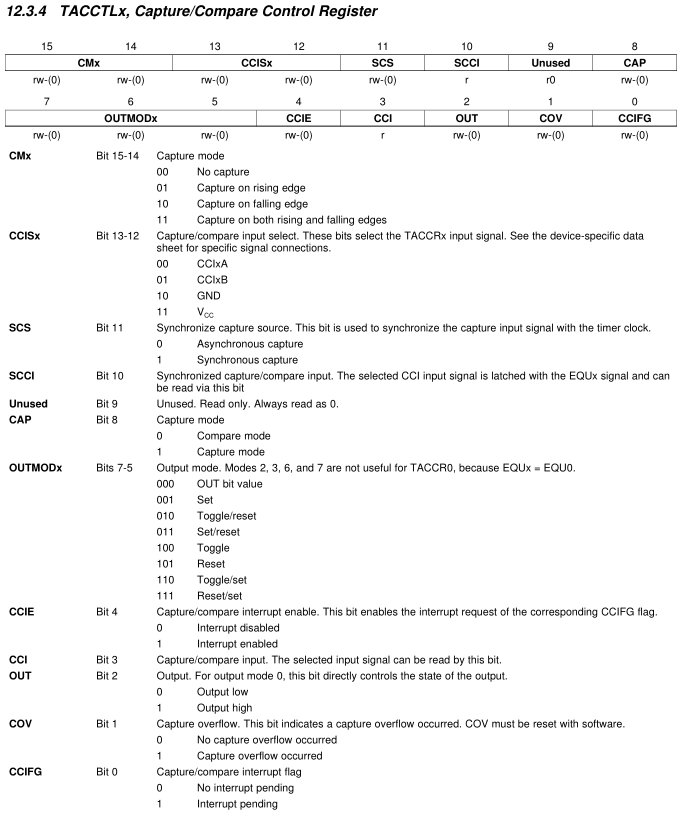


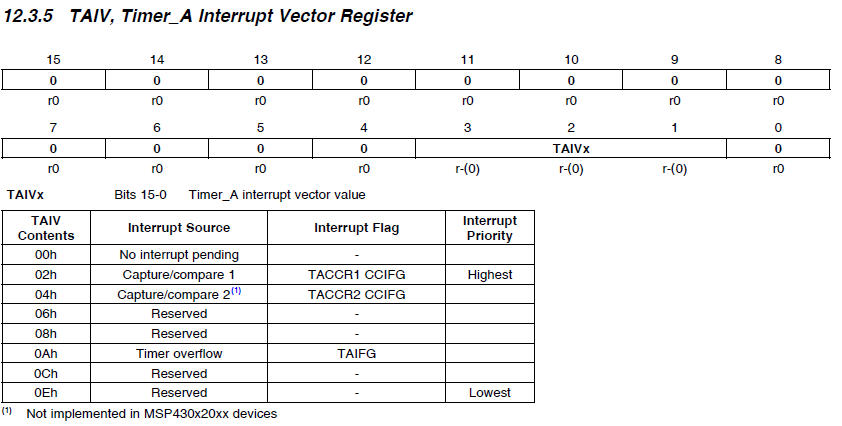


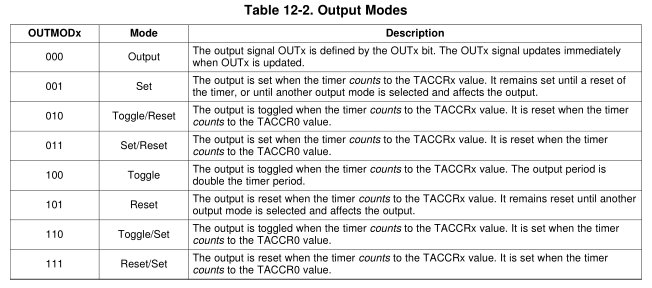


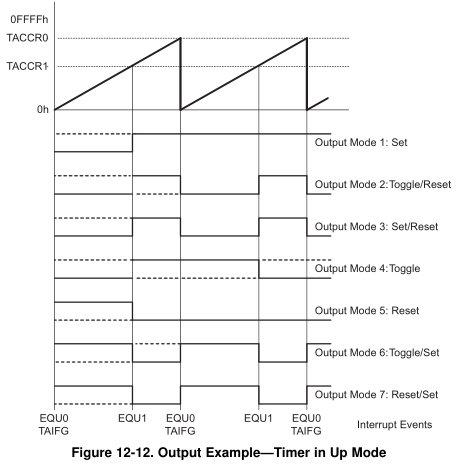
****

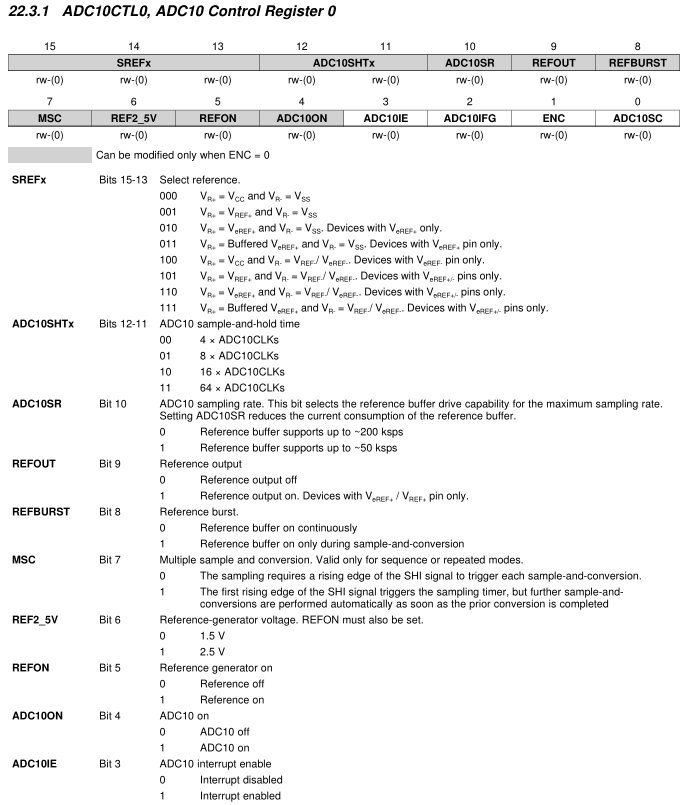
****

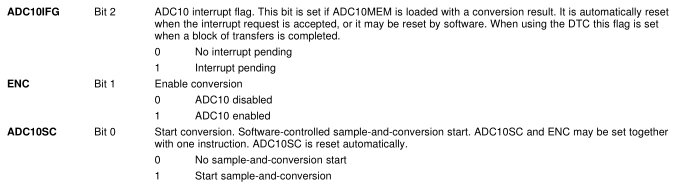
****

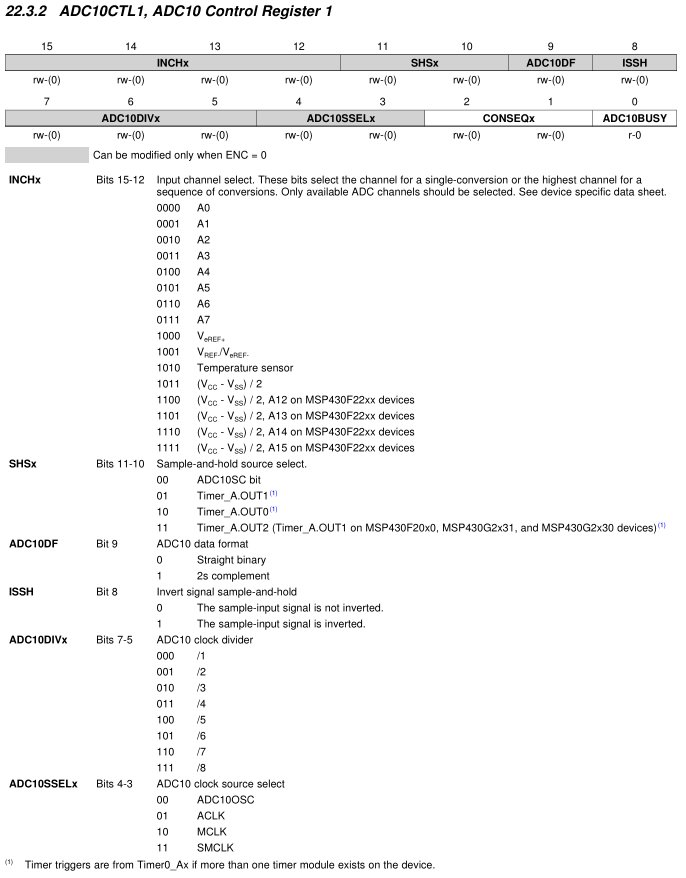


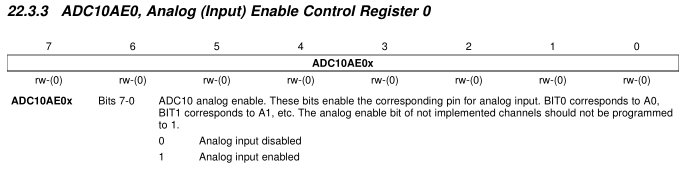
****

****

****

****

****

****