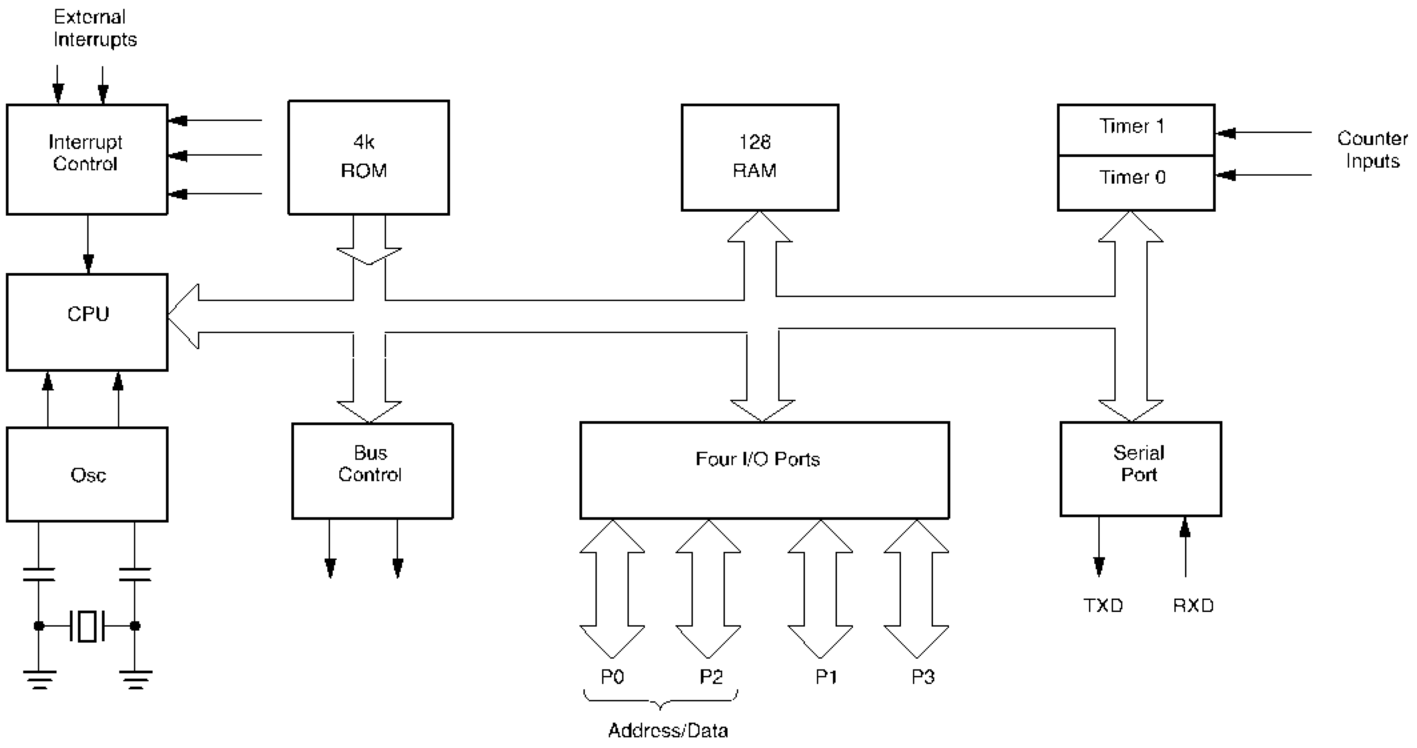
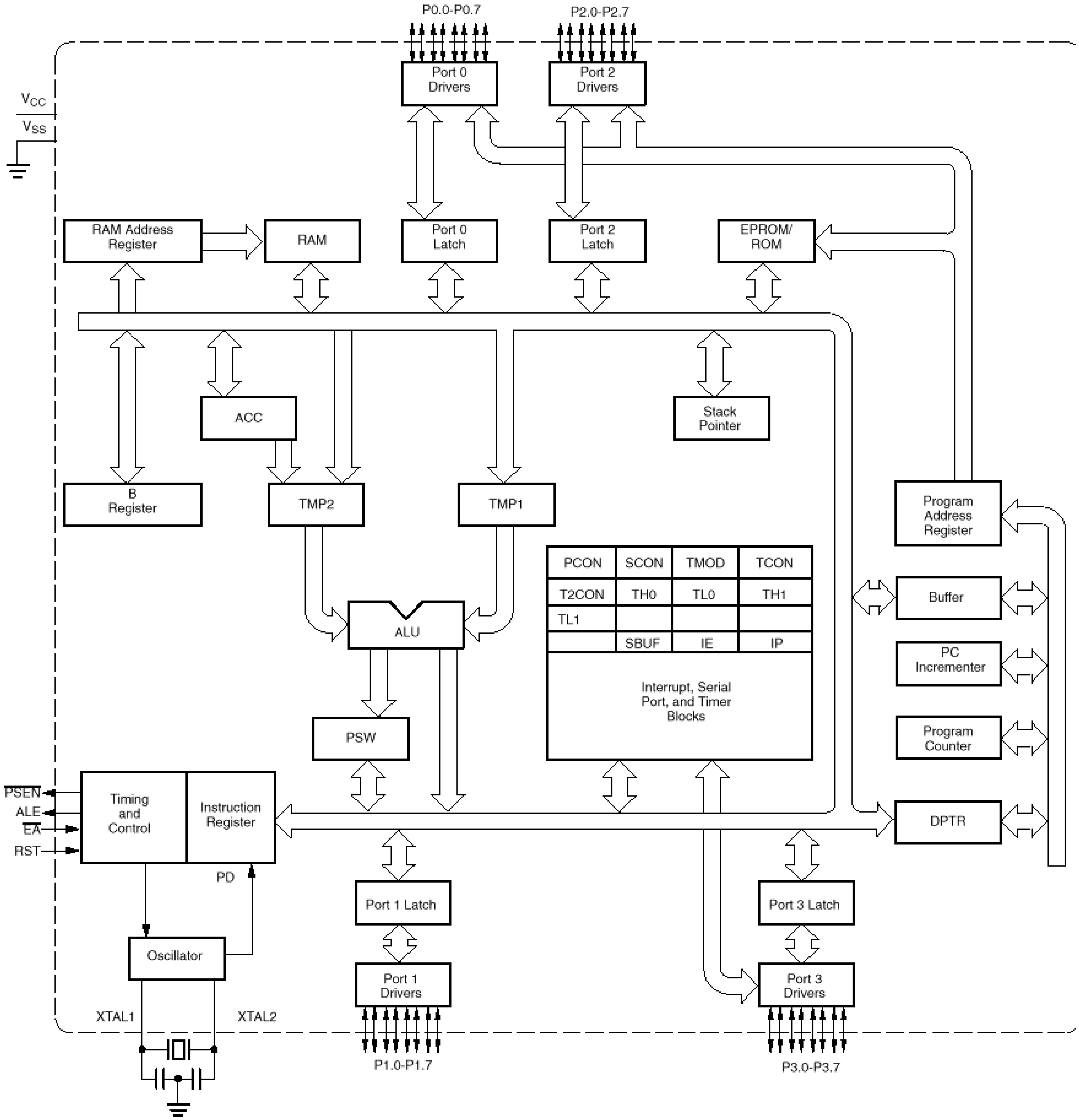
80C51 Block Diagram

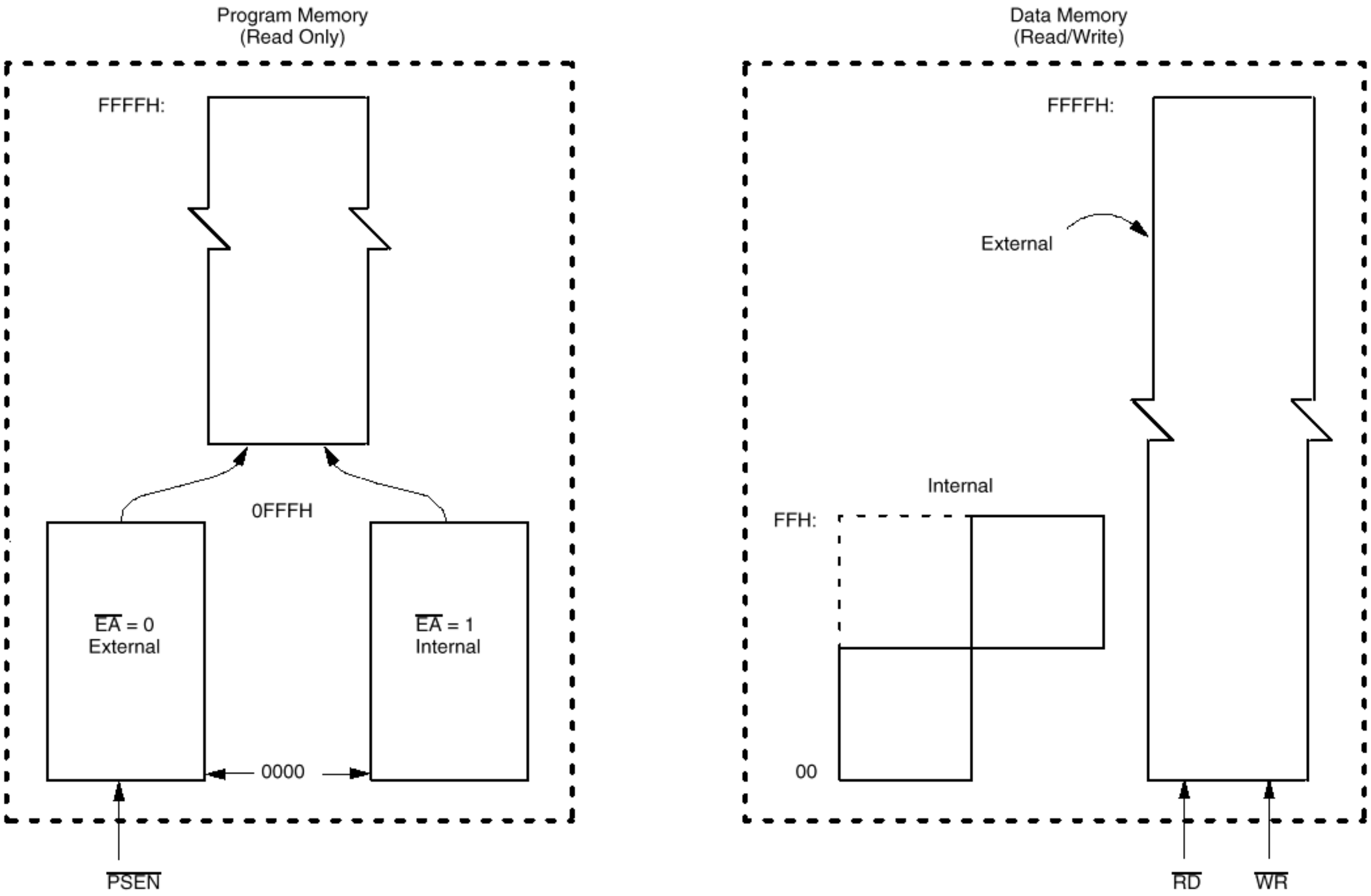




PCON	SCON	TMOD	TCON
T2CON	TH0	TL0	TH1
TL1			
	SBUF	IE	IP

Interrupt, Serial Port, and Timer Blocks

80C51 Memory



8051 Memory

- The data width is 8 bits
- Registers are 8 bits
- Addresses are 8 bits
  - i.e. addresses for only 256 bytes!
  - PC is 16 bits (up to 64K program memory)
  - DPTR is 16 bits (for external data - up to 64K)
- C types

char -	8 bits	<-- use this if at all possible!
short -	16 bits	
int -	16 bits	
long -	32 bits	
float -	32 bits	

- C standard **`signed/unsigned`**

Accessing External Memory

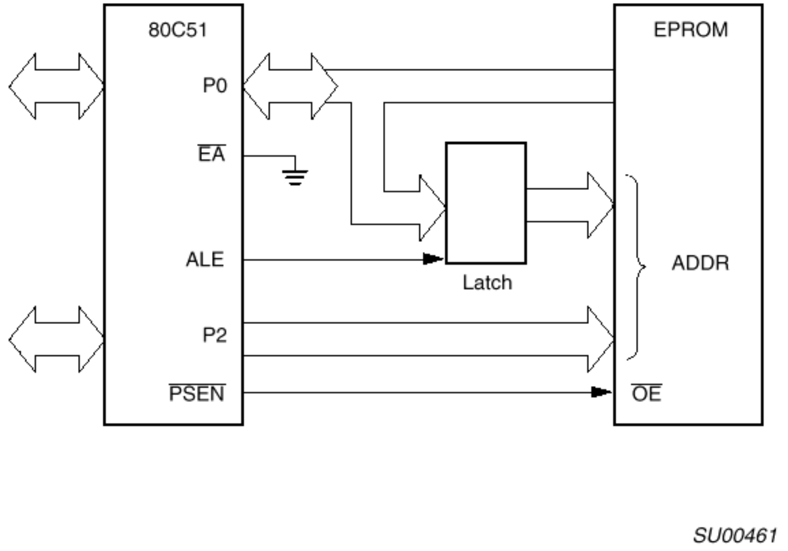


Figure 4. Executing from External Program Memory

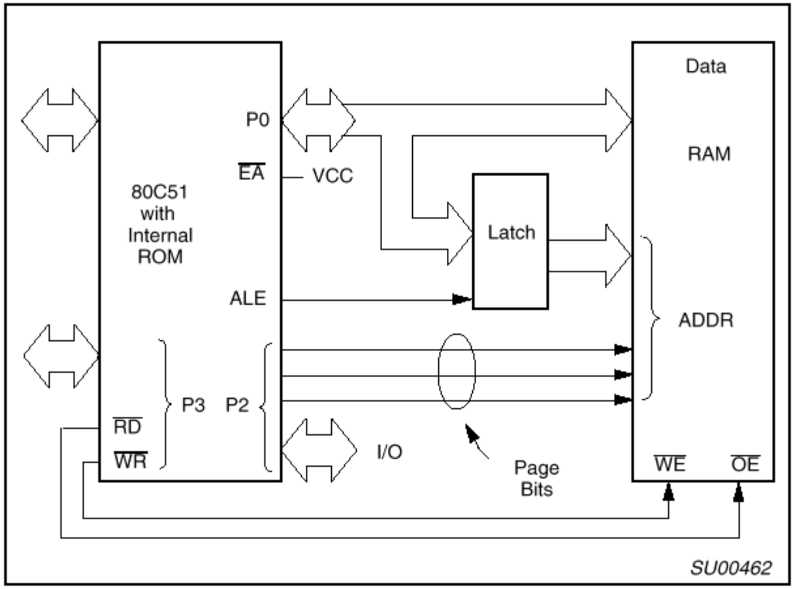


Figure 5. Accessing External Data Memory

Program Memory

- Program and Data memory are separate
- Can be internal and/or external
  - 20K internal flash for the Atmel controller
- Read-only
  - Instructions
  - Constant data

```
char code table[5] =        {‘1’,‘2’,‘3’,‘4’,‘5’} ;
```

  - Compiler uses instructions for moving “immediate” data

External Data Memory

- External Data - **`xdata`**
  - Resides off-chip
  - Accessed using the DPTR and MOVX instruction
  - We will not use **`xdata`**
  - **We will use the SMALL memory model**
    - all data is on-chip
    - limited to only ~128 bytes of data!

Internal Data Memory

- Internal data memory contains all the processor state
  - Lower 128 bytes: registers, general data
  - Upper 128 bytes:
    - indirectly addressed: 128 bytes, used for the stack (small!)
    - directly addressed: 128 bytes for "special" functions

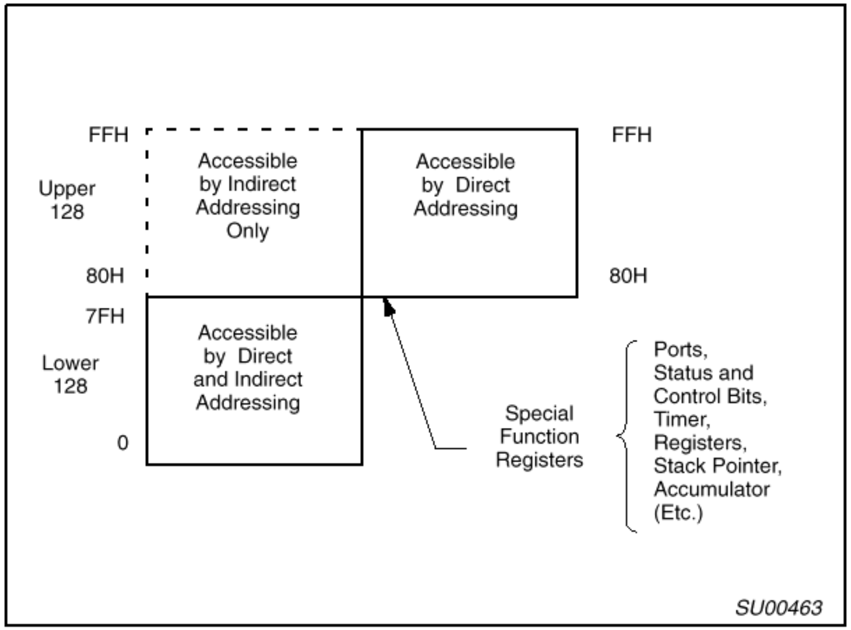

FFH
Upper 128
Accessible by Indirect Addressing Only
80H
7FH
Lower 128
Accessible by Direct and Indirect Addressing
0
Accessible by Direct Addressing
FFH
80H
Special Function Registers
Ports, Status and Control Bits, Timer, Registers, Stack Pointer, Accumulator (Etc.)
SU00463

Figure 6. Internal Data Memory

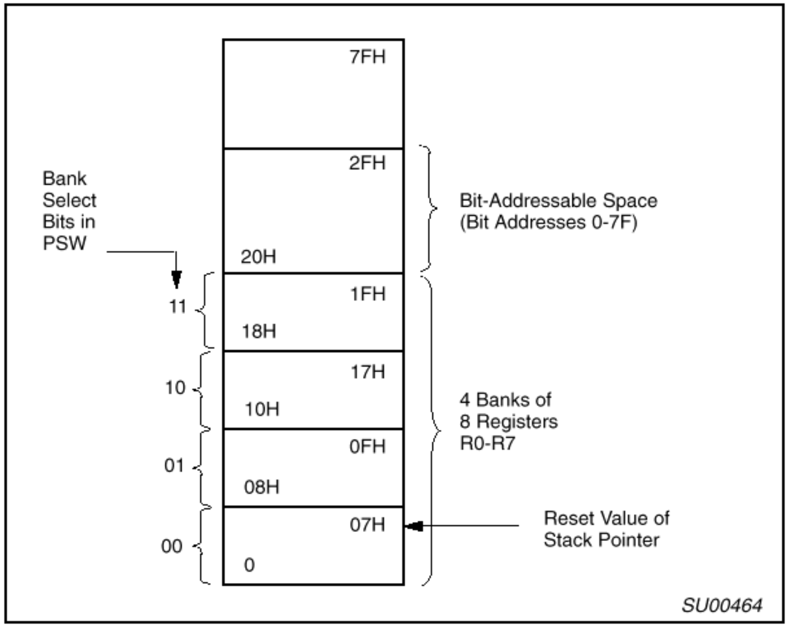

7FH
2FH
Bit-Addressable Space (Bit Addresses 0-7F)
Bank Select Bits in PSW
20H
1FH
11
18H
17H
10
10H
0FH
4 Banks of 8 Registers R0-R7
01
08H
07H
Reset Value of Stack Pointer
00
0
SU00464

Figure 7. Lower 128 Bytes of Internal RAM

Lower 128 bytes

- Register banks, bit addressable data, general data
  - you can address any register!
  - let the C compiler deal with details (for now)

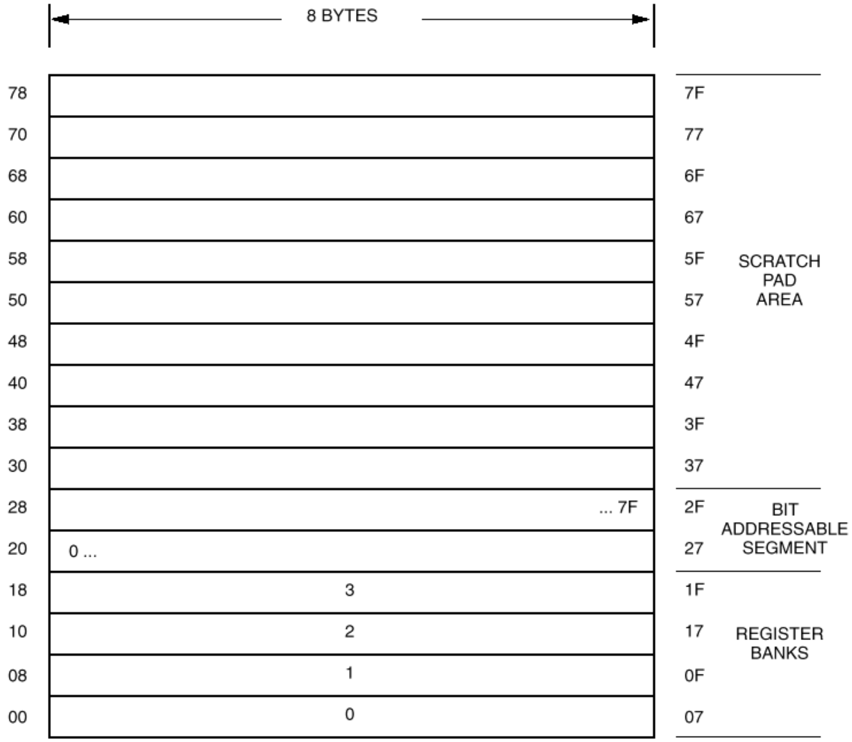


Figure 3. 128 Bytes of RAM Direct and Indirect Addressable

Data Memory Specifiers

- “data” - first 128 bytes, directly addressed
  - the default
- “idata” - all 256 bytes, indirectly addressed (slower)
- “bdata” - bit-addressable memory
  - 16 bytes from addresses 0x20 to 0x2F
  - 128 bit variables max

```
bit flag1, flag2;
flag1 = (a == b);
```

  - can access as bytes or bits

```
char bdata flags;
sbit flag0 = flags ^ 0; /* use sbit to “overlay” */
sbit flag7 = flags ^ 7; /* ^ specifies bit */
flags = 0;    /* Clear all flags */
flag7 = 1;     /* Set one flag */
```

Upper 128 bytes: SFR area

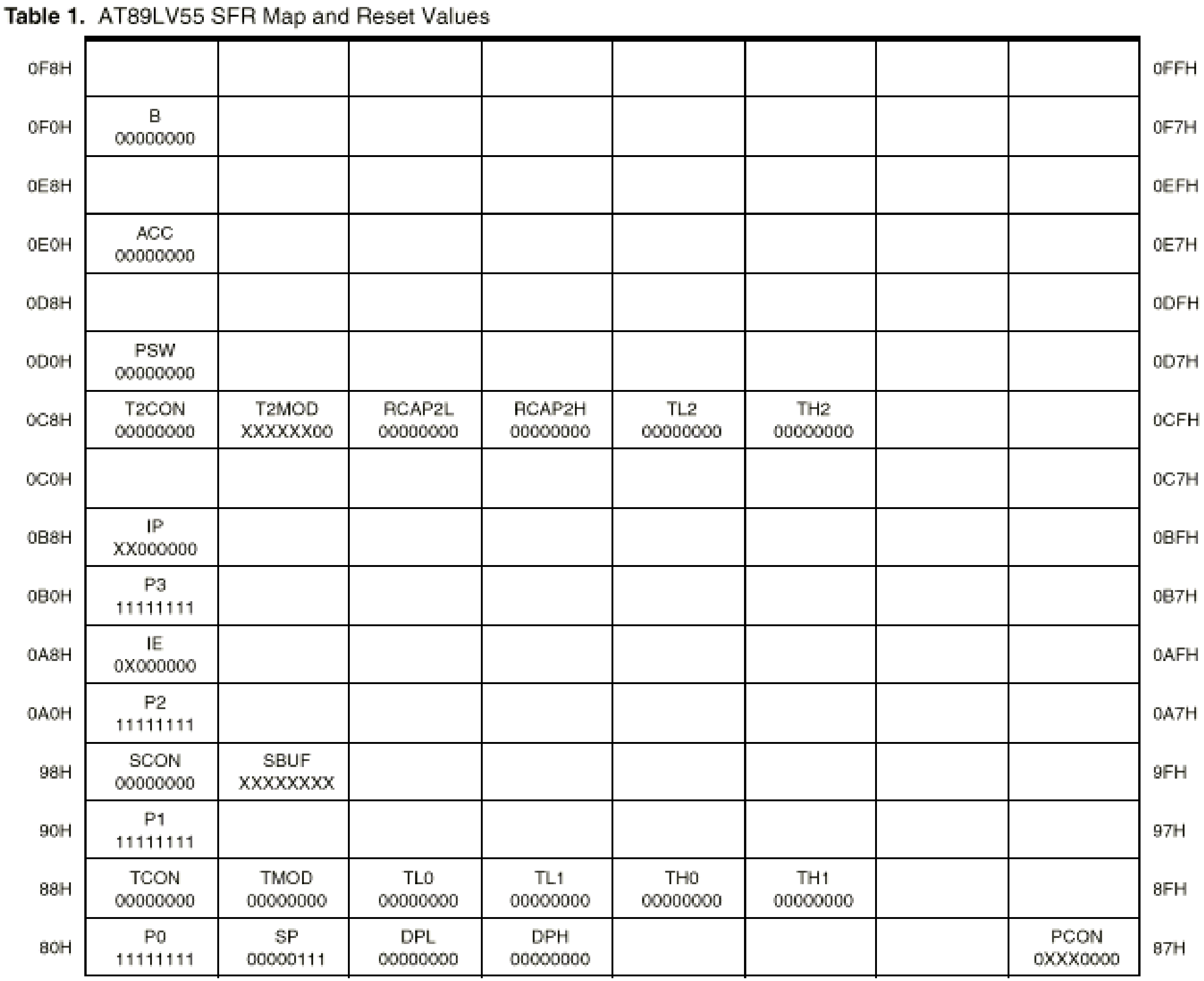


Table 1. AT89LV55 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000								0AFH
0A0H	P2 11111111								0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H	P0 11111111	SP 00000111	DPL 00000000	DPH 00000000				PCON 0XXX0000	87H

Table 1. 80C51 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION								RESET VALUE
			MSB							LSB	
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	–	ES	ET1	EX1	ET0	EX0	0x000000B
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–	–	–	PS	PT1	PX1	PT0	PX0	xx000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	–	–	–	–	–	–	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{\text{RD}}$	$\overline{\text{WR}}$	T1	T0	$\overline{\text{INT1}}$	$\overline{\text{INT0}}$	TxD	Rxd	FFH
PCON[1]	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxxxxxxB
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SBUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial controller	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
TH0	Timer high 0	8CH									00H
TH1	Timer high 1	8DH									00H
TL0	Timer low 0	8AH									00H
TL1	Timer low 1	8BH									00H
TMOD	Timer mode	89H	GATE	C/$\overline{\text{T}}$	M1	M0	GATE	C/$\overline{\text{T}}$	M1	M0	00H

Accessing SFRs

- The interesting SFRs are bit-addressable
  - addresses 0x80, 0x88, 0x90, . . . , 0xF8
- SFRs can be addressed by bit, char or int

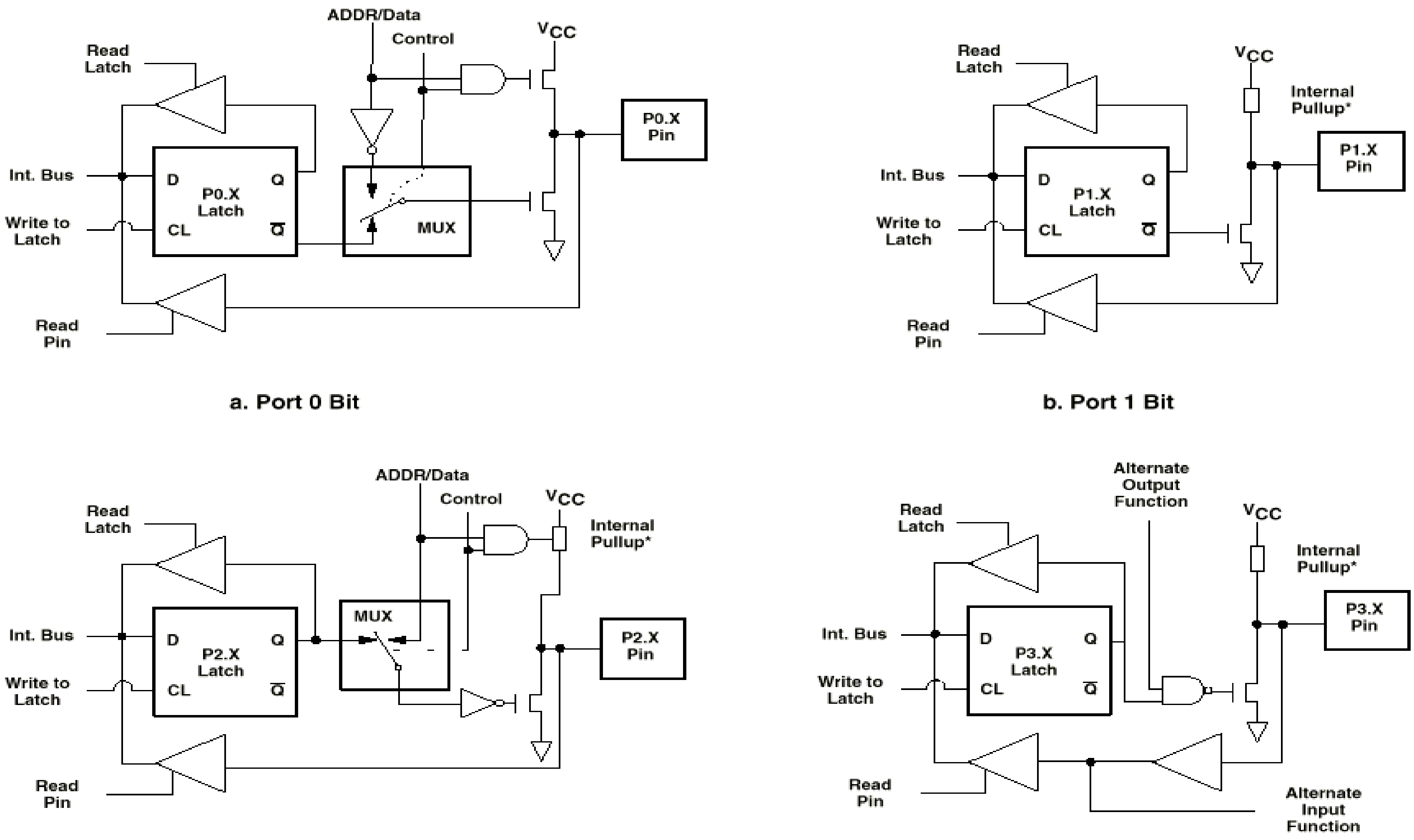
```
sbit EA = 0xAF; /* one of the interrupt enables
sfr Port0 = 0x80;  /* Port 0 */
sfr16 Timer2 = 0xCC;  /* Timer 2 */
sbit LED0 = Port1 ^ 2;  /* Define a port bit */

EA = 1;     /* Enable interrupts */
Port0 = 0xff;  /* Set all bits in Port 0 to 1
if (Timer2 > 100) . . .
LED0 = 1;   /* Turn on one bit in Port 2 */
```

Ports

- Port 0 - external memory access
  - low address byte/data
- Port 2 - external memory access
  - high address byte
- Port 1 - general purpose I/O
  - pins 0, 1 for timer/counter 2
- Port 3 - Special features
  - 0 - RxD: serial input
  - 1 - TxD: serial output
  - 2 - INT0: external interrupt
  - 3 - INT1: external interrupt
  - 4 - T0: timer/counter 0 external input
  - 5 - T1: timer/counter 1 external input
  - 6 - WR: external data memory write strobe
  - 7 - RD: external data memory read strobe

Ports



a. Port 0 Bit

b. Port 1 Bit

c. Port 2 Bit

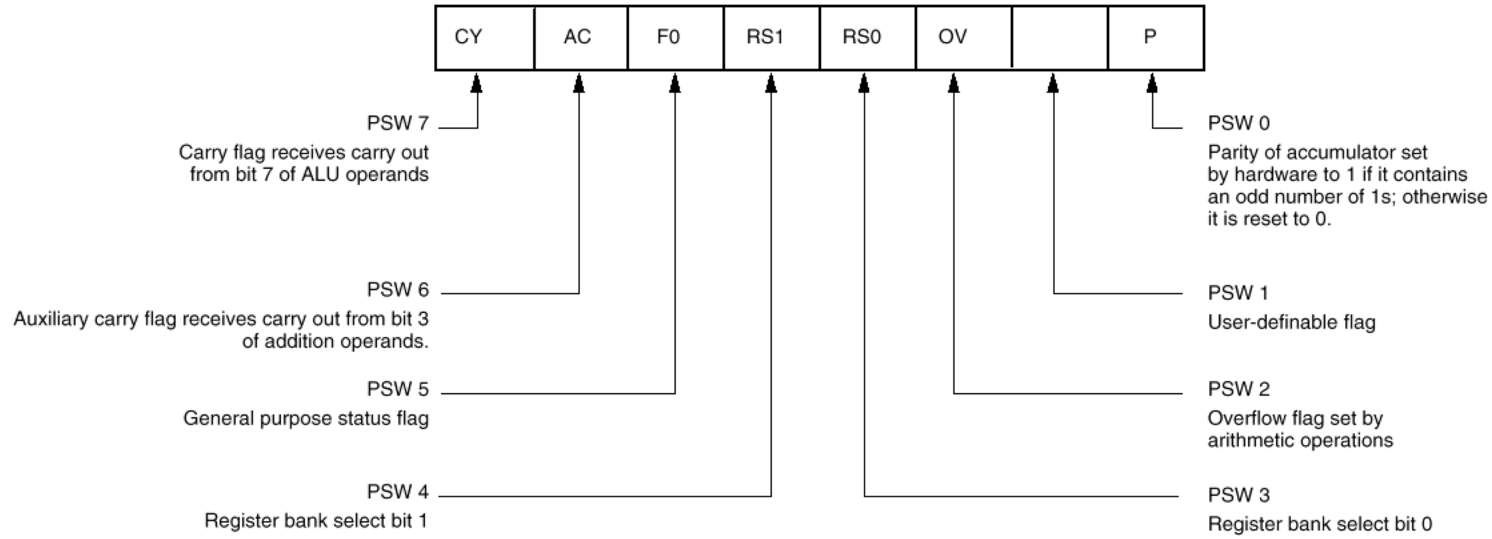
d. Port 3 Bit

Ports

- Port 0 - true bi-directional
- Port 1-3 - have internal pullups that will source current
- Output pins:
  - Just write 0/1 to the bit/byte
- Input pins:
  - Output latch must have a 1 (reset state)
    - Turns off the pulldown
    - pullup must be pulled down by external driver
  - Just read the bit/byte

Program Status Word

- Register set select
- Status bits



CY	AC	F0	RS1	RS0	OV		P

PSW 7
Carry flag receives carry out from bit 7 of ALU operands

PSW 6
Auxiliary carry flag receives carry out from bit 3 of addition operands.

PSW 5
General purpose status flag

PSW 4
Register bank select bit 1

PSW 0
Parity of accumulator set by hardware to 1 if it contains an odd number of 1s; otherwise it is reset to 0.

PSW 1
User-definable flag

PSW 2
Overflow flag set by arithmetic operations

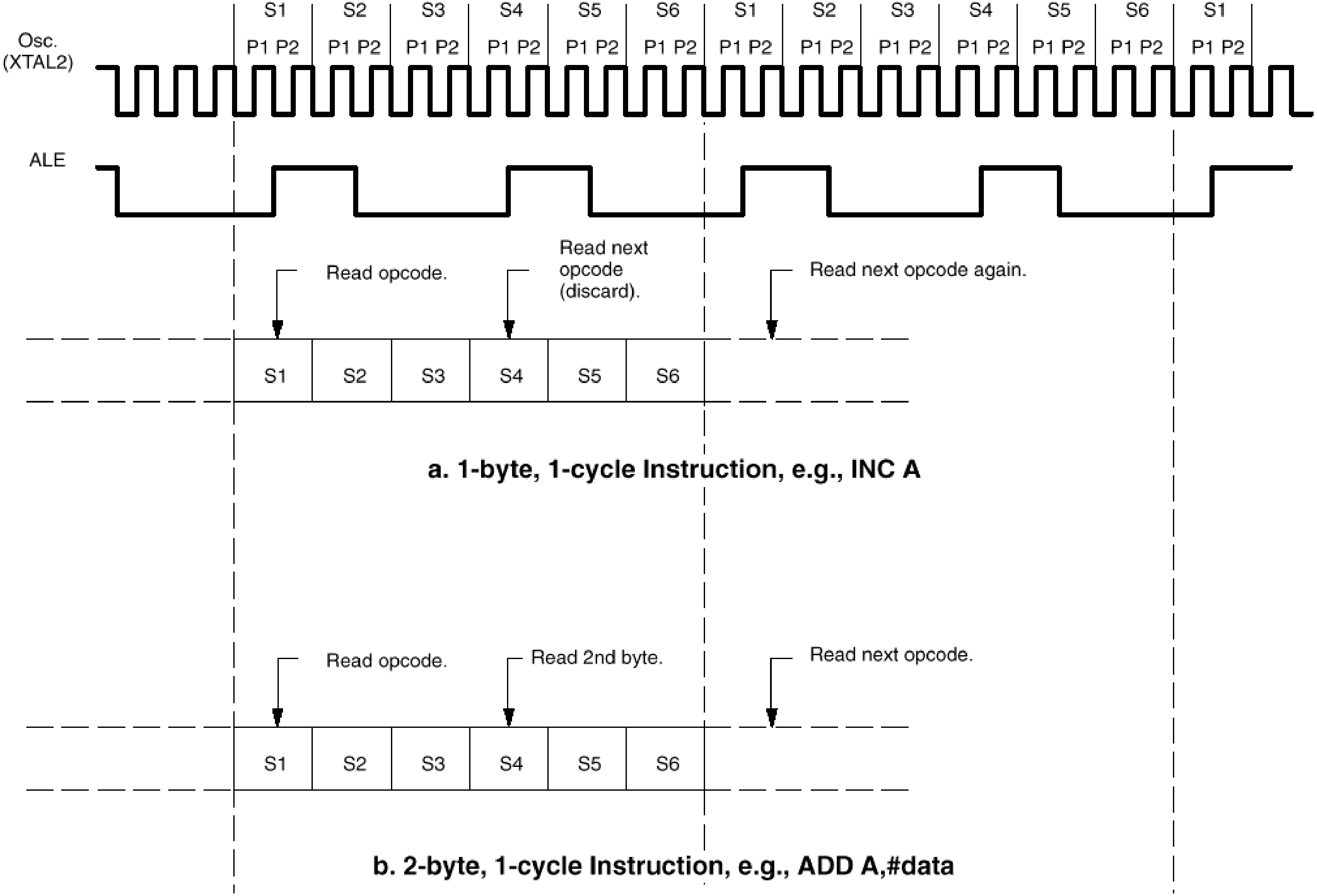
PSW 3
Register bank select bit 0

Figure 10. PSW (Program Status Word) Register in 80C51 Devices

Instruction Timing

- One "machine cycle" = 6 states (S1 - S6)
- One state = 2 clock cycles
  - One "machine cycle" = 12 clock cycles
- Instructions take 1 - 4 cycles
  - e.g. 1 cycle instructions: ADD, MOV, SETB, NOP
  - e.g. 2 cycle instructions: JMP, JZ
  - 4 cycle instructions: MUL, DIV

Instruction Timing


S1 S2 S3 S4 S5 S6 S1 S2 S3 S4 S5 S6 S1
P1 P2 P1 P2 P1 P2 P1 P2 P1 P2 P1 P2 P1 P2 P1 P2 P1 P2 P1 P2 P1 P2 P1 P2 P1 P2
Osc.
(XTAL2)
ALE
Read opcode.
Read next opcode (discard).
Read next opcode again.
S1 S2 S3 S4 S5 S6
Read opcode.
Read 2nd byte.
Read next opcode.
S1 S2 S3 S4 S5 S6

a. 1-byte, 1-cycle Instruction, e.g., INC A

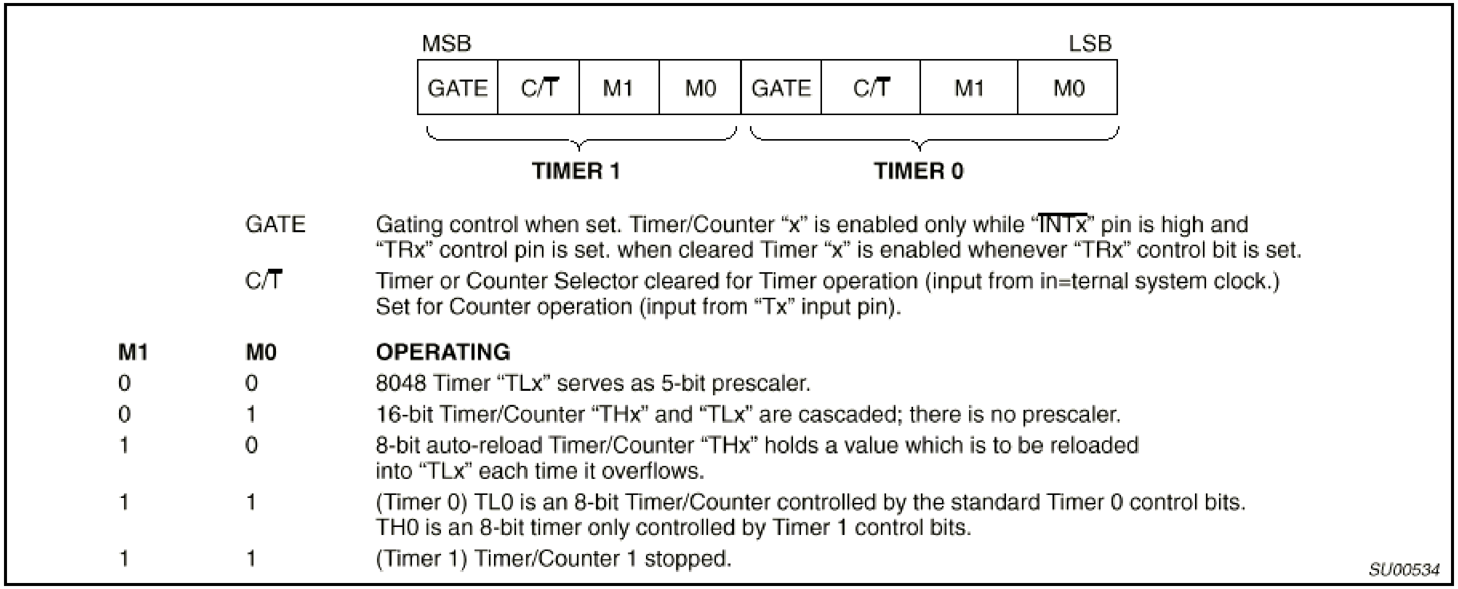
b. 2-byte, 1-cycle Instruction, e.g., ADD A,#data

Timers

- Base 8051 has 2 timers
  - we have 3 in the Atmel 89C55
- Timer mode
  - Increments every machine cycle (12 clock cycles)
- Counter mode
  - Increments when T0/T1 go from 1 - 0 (external signal)
- Access timer value directly
- Timer can cause an interrupt
- Timer 1 can be used to provide programmable baud rate for serial communications
- Timer/Counter operation
  - Mode control register (TMOD)
  - Control register (TCON)

Mode Control Register (TMOD)

- Modes 0-3
- GATE - allows external pin to enable timer (e.g. external pulse)
  - 0: INT pin not used
  - 1: counter enabled by INT pin (port 3.2, 3.3)
- C/T - indicates timer or counter mode



MSB LSB

GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0

TIMER 1 TIMER 0

GATE Gating control when set. Timer/Counter “x” is enabled only while “$\overline{\text{INTx}}$” pin is high and “TRx” control pin is set. when cleared Timer “x” is enabled whenever “TRx” control bit is set.

$C/\overline{T}$ Timer or Counter Selector cleared for Timer operation (input from in=ternal system clock.) Set for Counter operation (input from “Tx” input pin).

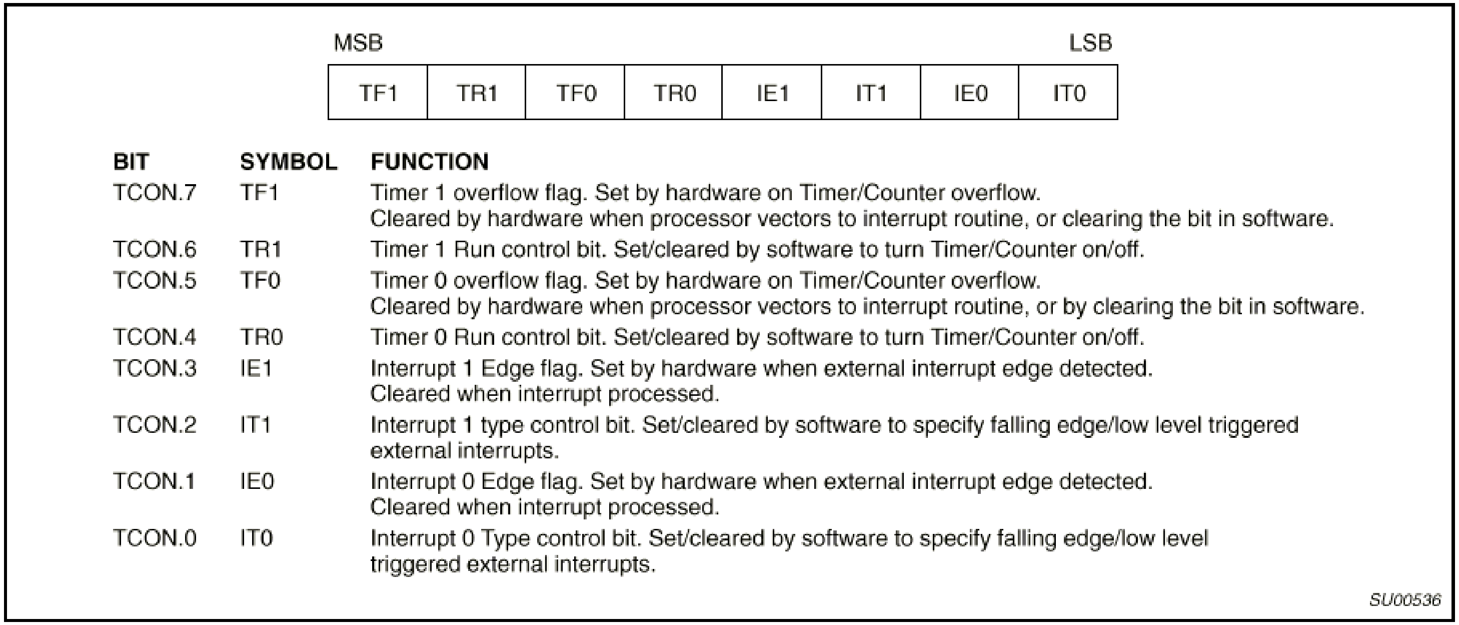
M1	M0	OPERATING
0	0	8048 Timer “TLx” serves as 5-bit prescaler.
0	1	16-bit Timer/Counter “THx” and “TLx” are cascaded; there is no prescaler.
1	0	8-bit auto-reload Timer/Counter “THx” holds a value which is to be reloaded into “TLx” each time it overflows.
1	1	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits.
1	1	(Timer 1) Timer/Counter 1 stopped.

SU00534

Figure 6. Timer/Counter Mode Control (TMOD) Register

Timer/Counter Control Register (TCON)

- TR - enable timer/counter
- TF - overflow flag: can cause interrupt
- IE/IT - external interrupts and type control
  - not related to the timer/counter



MSB LSB

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

BIT	SYMBOL	FUNCTION
TCON.7	TF1	Timer 1 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine, or clearing the bit in software.
TCON.6	TR1	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TCON.5	TF0	Timer 0 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine, or by clearing the bit in software.
TCON.4	TR0	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TCON.3	IE1	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
TCON.2	IT1	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
TCON.1	IE0	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
TCON.0	IT0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

SU00536

Figure 8. Timer/Counter Control (TCON) Register

Timer/Counter Mode 0

- Mode 1 same as Mode 0, but uses all 16 bits

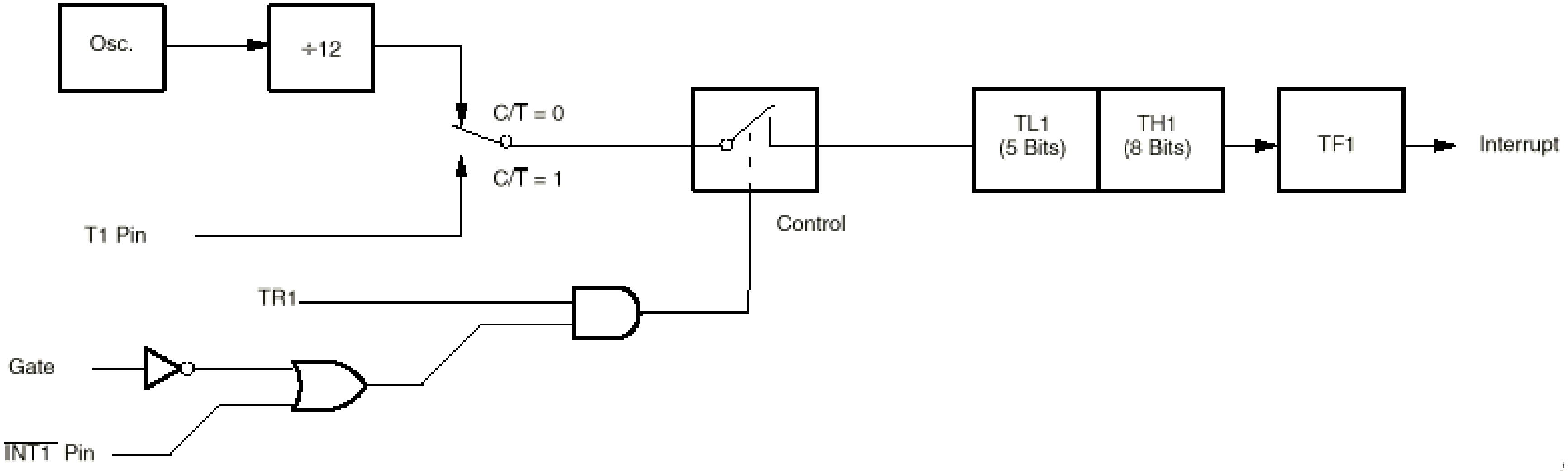


Figure 7. Timer/Counter Mode 0: 13-Bit Counter

Timer/Counter Mode 2

- 8-bit counter, auto-reload on overflow

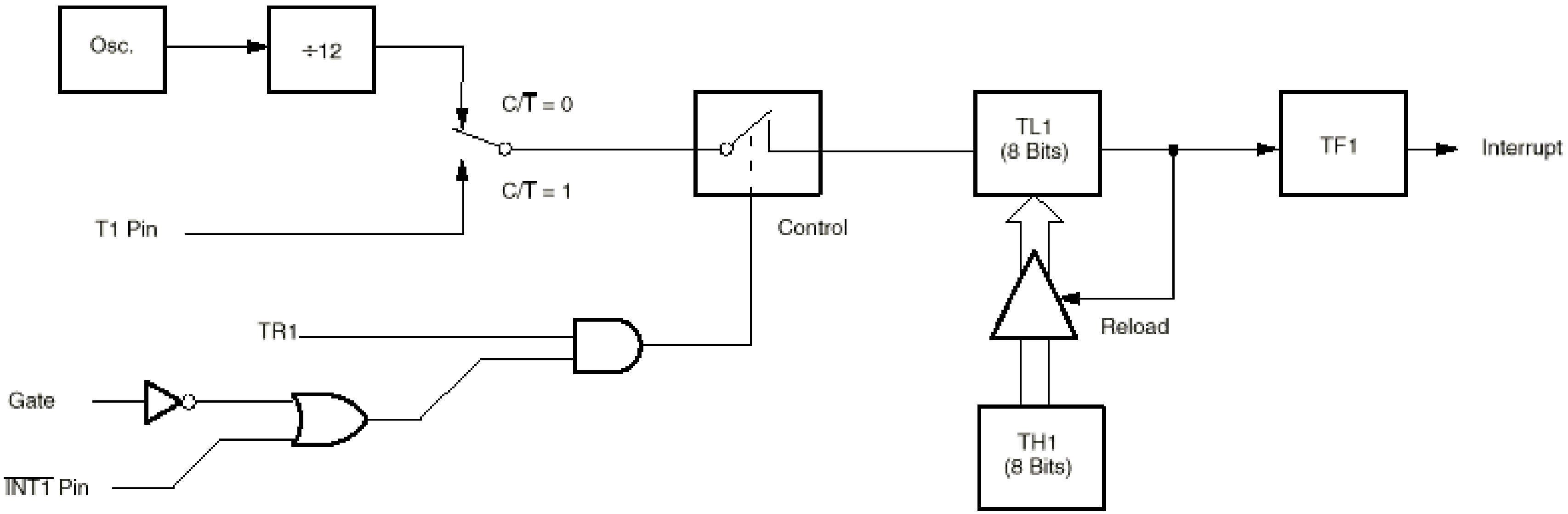


Figure 9. Timer/Counter Mode 2: 8-Bit Auto-Load

Timer/Counter Mode 3

- Applies to Timer/Counter 0
- Gives an extra timer

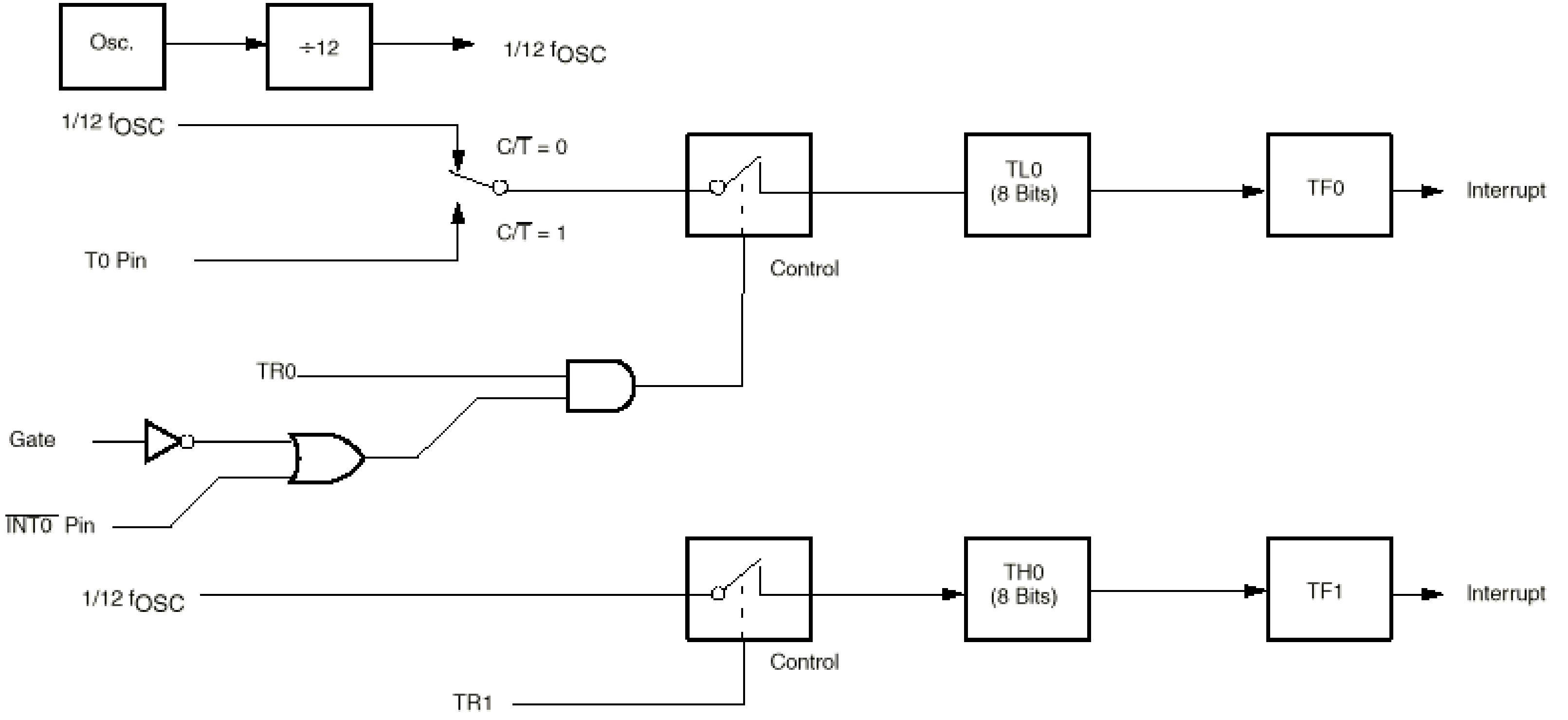


Figure 10. Timer/Counter 0 Mode 3: Two 8-Bit Counters

Interrupts

- Allow parallel tasking
  - Interrupt routine runs in “background”
- Allow fast, low-overhead interaction with environment
  - Don’t have to poll
  - Immediate reaction
- An automatic function call
  - Easy to program
- 8051 Interrupts
  - Serial port - wake up when data arrives/data has left
  - Timer 0 overflow
  - Timer 1 overflow
  - External interrupt 0
  - External interrupt 1

Interrupt Vector

- For each interrupt, which interrupt function to call
- In low program addresses

```
   0x00 - Reset PC address
0: 0x03 - External interrupt 0
1: 0x0B - Timer 0
2: 0x13 - External interrupt 1
3: 0x1B - Timer 1
4: 0x23 - Serial line interrupt
```

- Hardware generates an LCALL to address in interrupt vector
- Pushes PC (but nothing else) onto the stack
- RETI instruction to return from interrupt

Writing Interrupts in C

- The C compiler takes care of everything
  - Pushing/popping the right registers (PSW, ACC, etc.)
  - Generating the RTI instruction
  - No arguments/no return values

```
unsigned int count;
unsigned char second;

void timer0 (void) interrupt 1 using 2 {
    if (++count == 4000) {
        second++;
        count = 0;
    }
}
```

  - **Timer mode 2**
  - **Reload value = 6**

Timer Interrupts

- Wakeup after N clock cycles, i.e. at a specified time
- Wakeup every N clock cycles (auto reload)
  - Allows simple task scheduling
  - Clients queue function calls for time i
  - Interrupt routine calls functions at the right time
- Wakeup after N events have occurred on an input

Design Problem 1 - frequency counter

- Measure the frequency of an external signal
- Display as a number using the 7-segment display
  - e.g. number represents exponent of 2 or 10

Example Timer Setup

- What does this setup do?

```
TMOD = 0x62; // 01100010;
TCON = 0x50; // 01010000;
TH1 = 246;
TH0 = 6;

IE = 0x8A; // 10001010;
```

Using the timers

```
void counterInterrupt ( void ) interrupt 3 using 1 {
     timeLow = TL0;
     TL0 = 0;
     timeHigh = count;
     count = 0;
     if (timeHigh == 0 && timeLow < 10) *ledaddress = 0x6f;
     else if (timeHigh == 0 && timeLow < 100) *ledaddress = 0x6b;
     else if (timeHigh < 4) *ledaddress = 0x02;
     else if (timeHigh < 40) *ledaddress = 0x04;
     else if (timeHigh < 400) *ledaddress = 0x08;
     else if (timeHigh < 4000) *ledaddress = 0x10;
     else if (timeHigh < 40000) *ledaddress = 0x20;
     else *ledaddress = 0xf0;  // default
}


void timerInterrupt ( void ) interrupt 1 using 1 {
     count++;
}
```

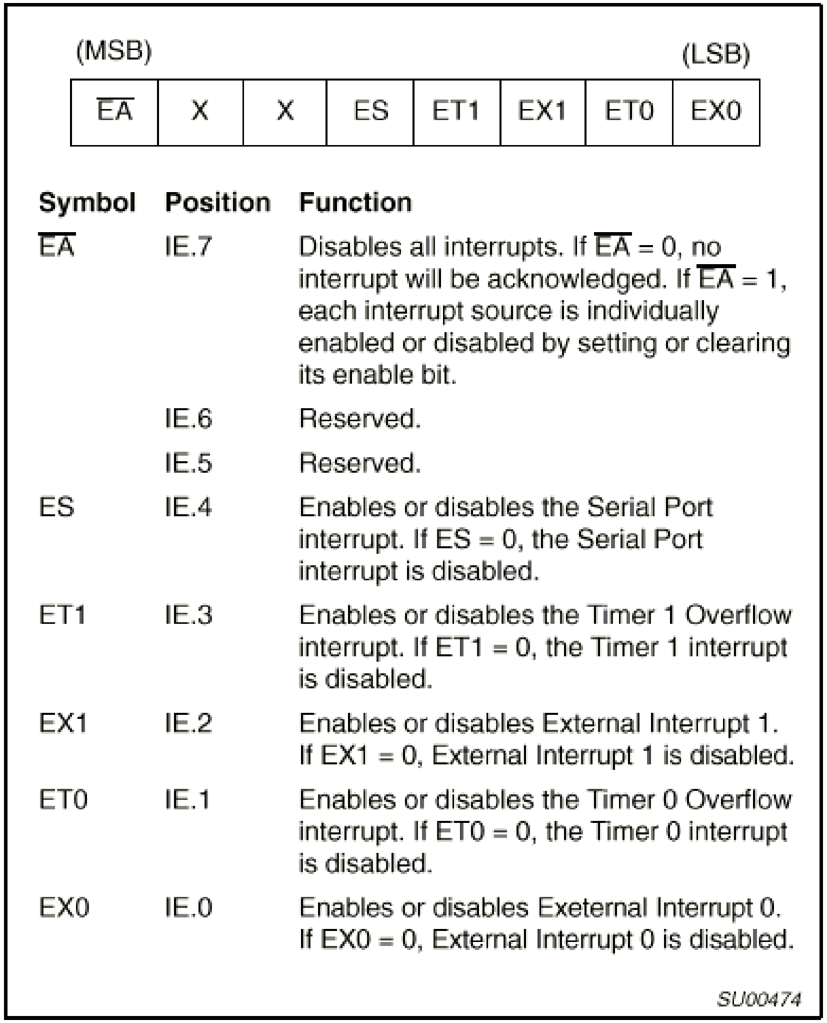
Design Problem 2 - Measure the pulse width

- Problem: send several bits of data with one wire
  - Serial data
    - precise, but complicated protocol
  - Pulse width
    - precise enough for many sensors
    - simple measurement

Design Problem 3 - Accelerometer Interface

- Accelerometer
  - Two signals, one for each dimension
  - Acceleration coded as the duty cycle
    - pulse-width/cycle-length
    - cycle time = 1ms - 10ms (controlled by resistor)
      - 1ms gives faster sampling
      - 10ms gives more accurate data

Controlling Interrupts: Enables and Priority

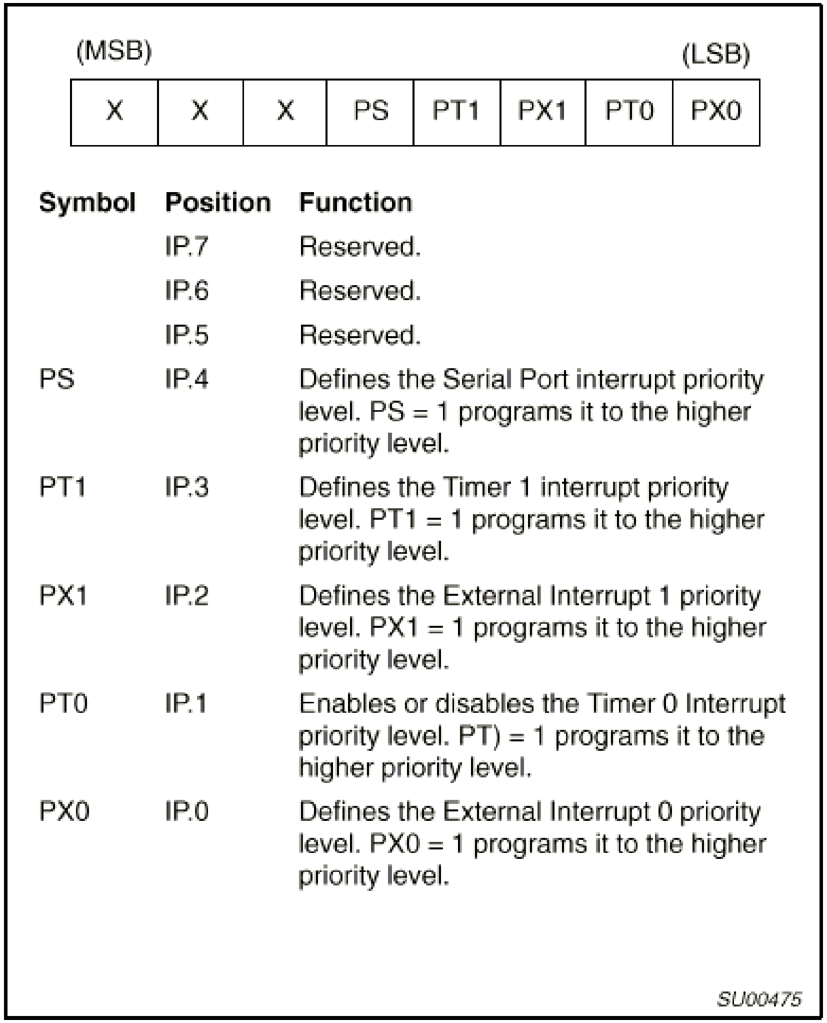


(MSB)							(LSB)
$\overline{\text{EA}}$	X	X	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
$\overline{\text{EA}}$	IE.7	Disables all interrupts. If $\overline{\text{EA}}$ = 0, no interrupt will be acknowledged. If $\overline{\text{EA}}$ = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
	IE.6	Reserved.
	IE.5	Reserved.
ES	IE.4	Enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	Enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	Enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	Enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	Enables or disables Exeternal Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

SU00474

Figure 17. Interrupt Enable (IE) Register



(MSB)							(LSB)
X	X	X	PS	PT1	PX1	PT0	PX0

Symbol	Position	Function
	IP.7	Reserved.
	IP.6	Reserved.
	IP.5	Reserved.
PS	IP.4	Defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	Defines the Timer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PX1	IP.2	Defines the External Interrupt 1 priority level. PX1 = 1 programs it to the higher priority level.
PT0	IP.1	Enables or disables the Timer 0 Interrupt priority level. PT) = 1 programs it to the higher priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

SU00475

Figure 18. Interrupt Priority (IP) Register

Interrupt Controls

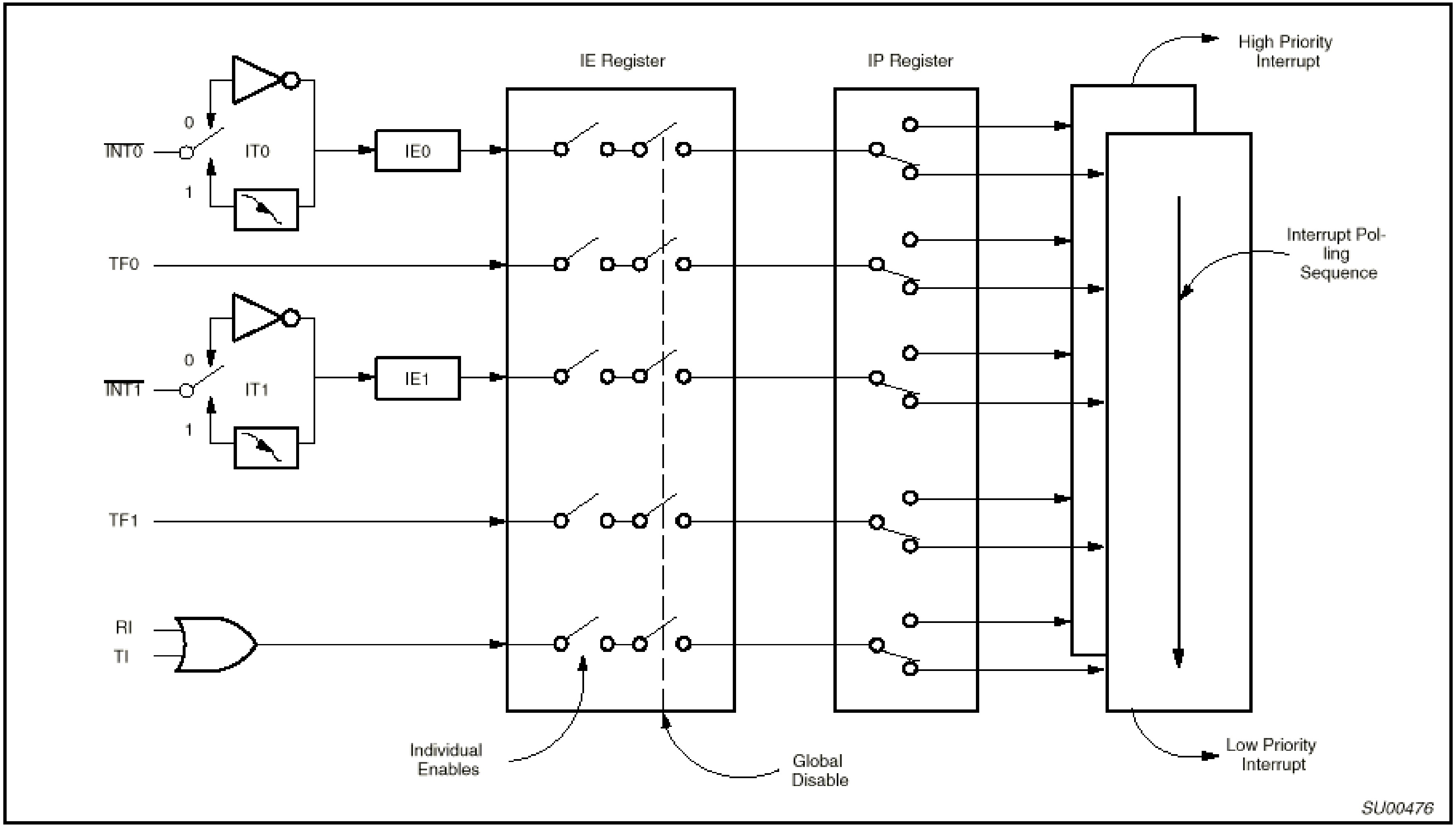


Figure 19. Interrupt Control System

Interrupt Priorities

- Two levels of priority
  - Set an interrupt priority using the interrupt priority register
  - A high-priority interrupt can interrupt an low-priority interrupt routine
  - In no other case is an interrupt allowed
  - An interrupt routine can always disable interrupts explicitly
    - But you don't want to do this
- Priority chain within priority levels
  - Choose a winner if two interrupts happen simultaneously
  - Order shown on previous page

Re-entrant Functions

- A function can be called simultaneously be different processes
- Recursive functions must be re-entrant
- Functions called by interrupt code and non-interrupt code must be re-entrant
- Keil C functions by default are *not* re-entrant
  - Does not use the stack for everything
  - Use the reentrant specifier to make a function re-entrant

```
int calc (char i, int b) reentrant {
  int x;
  x = table[i];
  return (x * b);
}
```

External Interrupts

- Can interrupt using the INT0 or INT1 pins (port 3: pin 2,3)
  - Interrupt on level or falling edge of signal (TCON specifies which)
  - Pin is sampled once every 12 clock cycles
    - for interrupt on edge, signal must be high 12 cycles, low 12 cycles
  - Response time takes at least 3 instuctions cycles
    - 1 to sample
    - 2 for call to interrupt routine
    - more if a long instruction is in progress (up to 6 more)