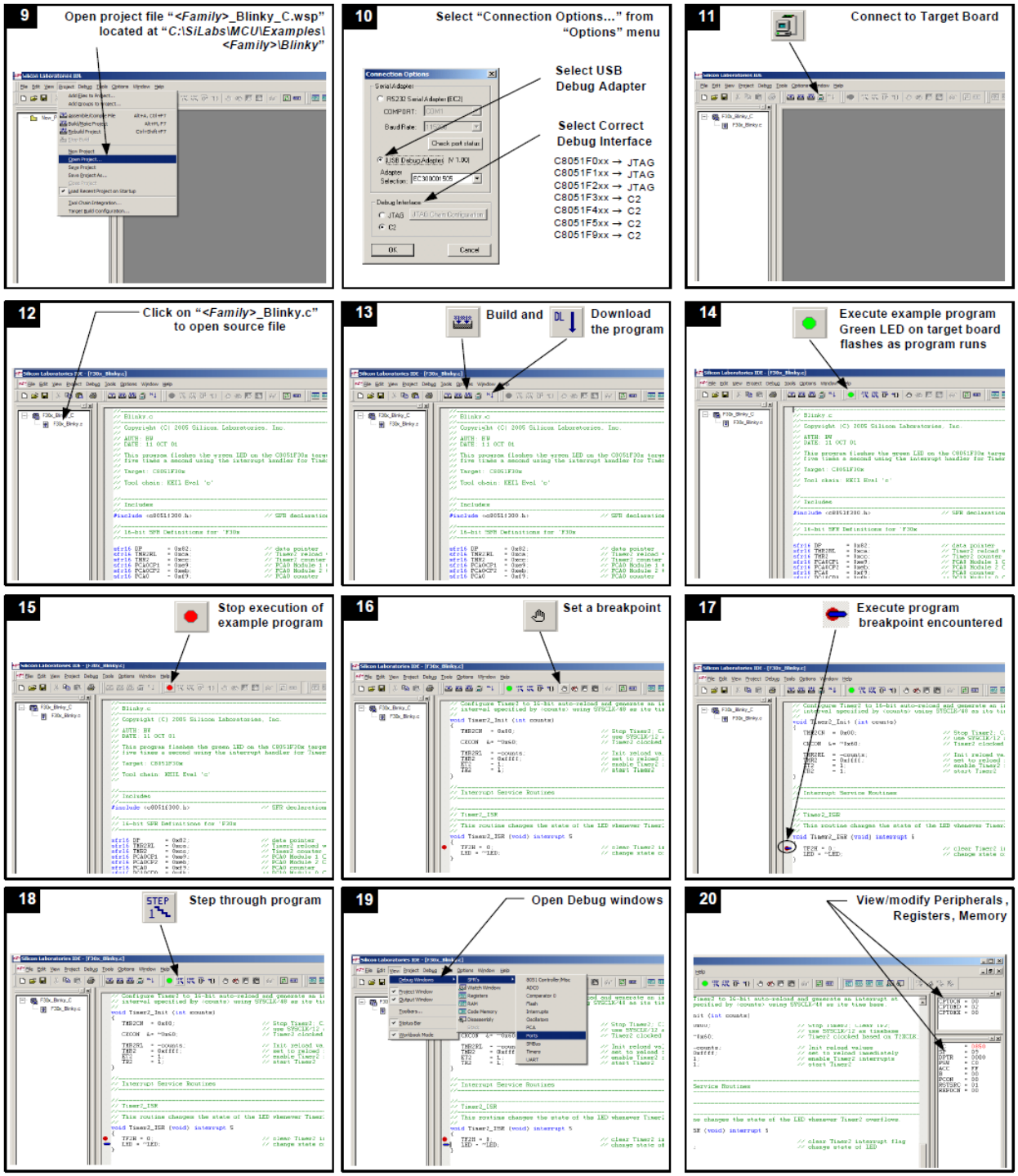
8051 Notes

My board "Family" is **C8051F34x-DK**

Trouble installing and/or using the development kit, please use the following support resources:

- C8051Fxxx Development Kit User's Guide (click "Browse Documentation")
- Application Note "AN104: Integrating Keil 8051 Tools Into the Silicon Labs IDE"
- Latest versions of Application Notes can be found at http://www.silabs.com/products/microcontroller/applications.asp
- MCU Knowledgebase (available at www.silabs.com → SUPPORT)
- Applications Engineer online information request form (available at www.silabs.com → SUPPORT).



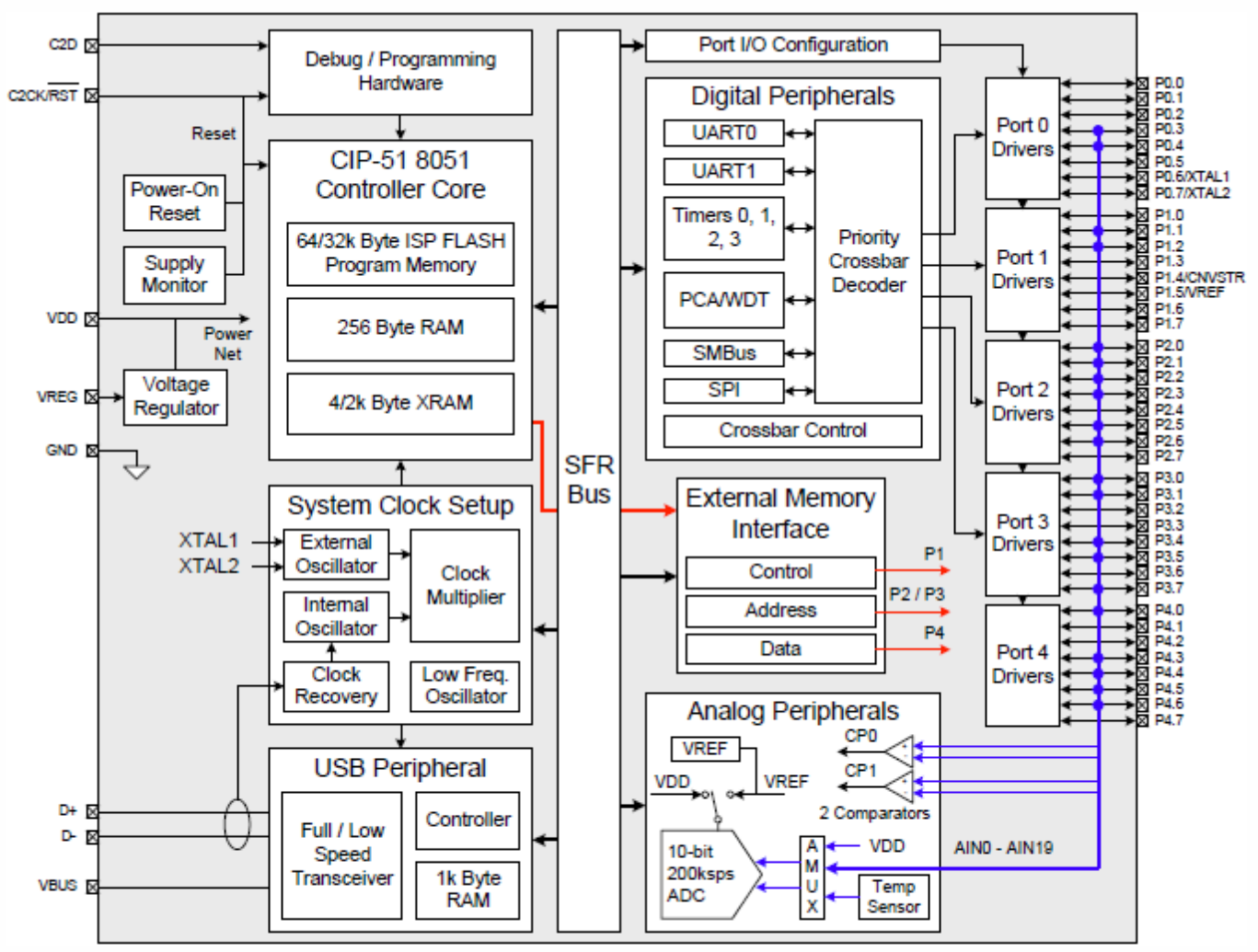


Figure 1.1. C8051F340/1/4/5 Block Diagram

8051 (8751 is the same as 8051 except it replaces ROM with UVEPROM)

- 4K bytes ROM
- 128 bytes RAM
- Two 16-bit counters/timers
- 5 interrupt sources (2 external)
- 1 duplex serial port
- 1 bit level Boolean processor

Table 1.1. Product Selection Guide

Ordering Part Number	MIPS (Peak)	Flash Memory (Bytes)	RAM	Calibrated Internal Oscillator	Low Frequency Oscillator	USB with 1k Endpoint RAM	Supply Voltage Regulator	SMBus/I2C	Enhanced SPI	UARTs	Timers (16-bit)	Programmable Counter Array	Digital Port I/Os	External Memory Interface (EMIF)	10-bit 200 ksps ADC	Temperature Sensor	Voltage Reference	Analog Comparators	Package
C8051F340-GQ	48	64k	4352	✓	✓	✓	✓	✓	✓	2	4	✓	40	✓	✓	✓	✓	2	TQFP48

10-Bit ADC (ADC0, C8051F340/1/2/3/4/5/6/7/A/B Only)

The ADC0 subsystem for the C8051F34x devices consists of two analog multiplexers (referred to collectively as AMUX0), and a 200 ksps, 10-bit successive-approximation-register ADC with integrated track-and-hold and programmable window detector. The AMUX0, data conversion modes, and window detector are all configured under software control via the Special Function Registers shown in Figure 5.1. ADC0 operates in both Single-ended and Differential modes, and may be configured to measure voltages at port pins, the Temperature Sensor output, or VDD with respect to a port pin, VREF, or GND. The connection options for AMUX0 are detailed in SFR Definition 5.1 and SFR Definition 5.2. The ADC0 subsystem is enabled only when the AD0EN bit in the ADC0 Control register (ADC0CN) is set to logic 1. The ADC0 subsystem is in low power shutdown when this bit is logic 0.

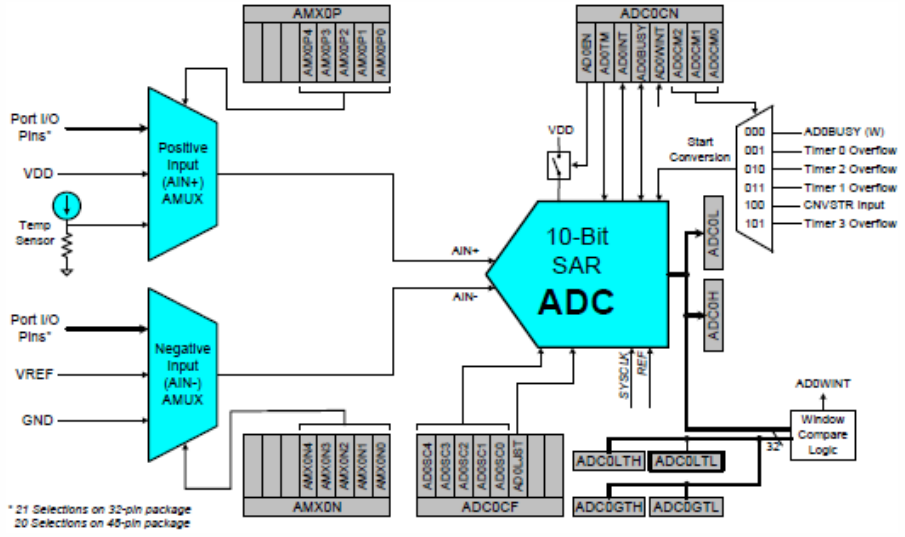


Figure 5.1. ADC0 Functional Block Diagram

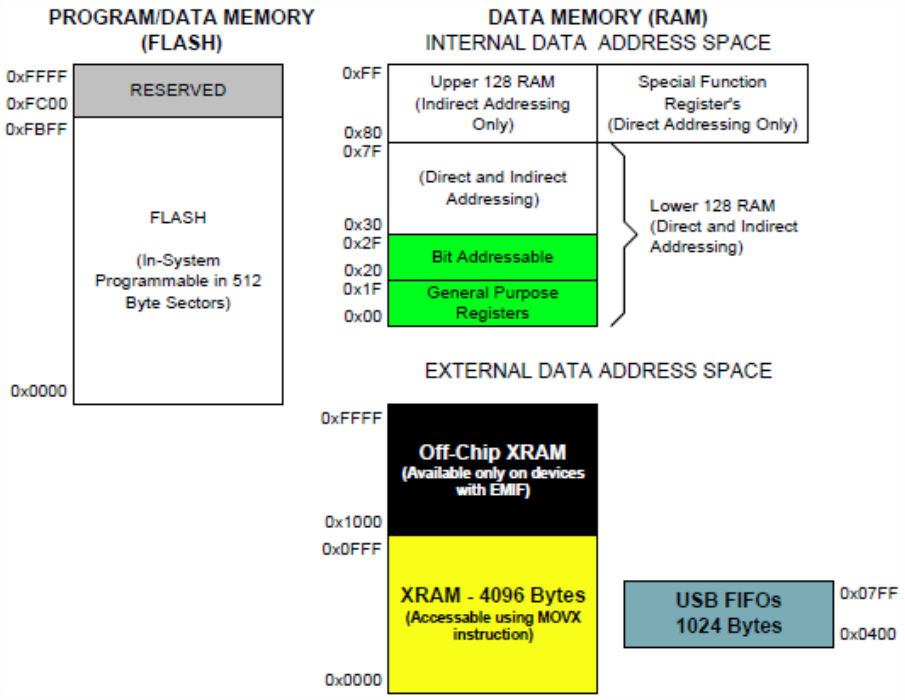


Figure 9.2. On-Chip Memory Map for 64 kB Devices

Special Function Registers

The direct-access data memory locations from 0x80 to 0xFF constitute the special function registers (SFRs). The SFRs provide control and data exchange with the CIP-51's resources and peripherals. The CIP-51 duplicates the SFRs found in a typical 8051 implementation as well as implementing additional SFRs used to configure and access the sub-systems unique to the MCU.

Table 9.2. Special Function Register (SFR) Memory Map

F8	SPI0CN	PCA0L	PCA0H	PCA0CPL0	PCA0CPH0	PCA0CPL4	PCA0CPH4	VDM0CN
F0	B	P0MDIN	P1MDIN	P2MDIN	P3MDIN	P4MDIN	EIP1	EIP2
E8	ADC0CN	PCA0CPL1	PCA0CPH1	PCA0CPL2	PCA0CPH2	PCA0CPL3	PCA0CPH3	RSTSRC
E0	ACC	XBR0	XBR1	XBR2	IT01CF	SMOD1	EIE1	EIE2
D8	PCA0CN	PCA0MD	PCA0CPM0	PCA0CPM1	PCA0CPM2	PCA0CPM3	PCA0CPM4	P3SKIP
D0	PSW	REF0CN	SCON1	SBUF1	P0SKIP	P1SKIP	P2SKIP	USB0XCN
C8	TMR2CN	REG0CN	TMR2RLL	TMR2RLH	TMR2L	TMR2H	-	-
C0	SMB0CN	SMB0CF	SMB0DAT	ADC0GTL	ADC0GTH	ADC0LTL	ADC0LTH	P4
B8	IP	CLKMUL	AMX0N	AMX0P	ADC0CF	ADC0L	ADC0H	-
B0	P3	OSCXCN	OSCICN	OSCICL	SBRLL1	SBRLH1	FLSCL	FLKEY
A8	IE	CLKSEL	EMI0CN	-	SBCON1	-	P4MDOUT	PFE0CN
A0	P2	SPI0CFG	SPI0CKR	SPI0DAT	P0MDOUT	P1MDOUT	P2MDOUT	P3MDOUT
98	SCON0	SBUF0	CPT1CN	CPT0CN	CPT1MD	CPT0MD	CPT1MX	CPT0MX
90	P1	TMR3CN	TMR3RLL	TMR3RLH	TMR3L	TMR3H	USB0ADR	USB0DAT
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	PSCTL
80	P0	SP	DPL	DPH	EMI0TC	EMI0CF	OSCLCN	PCON
	0(8) (bit addressable)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)

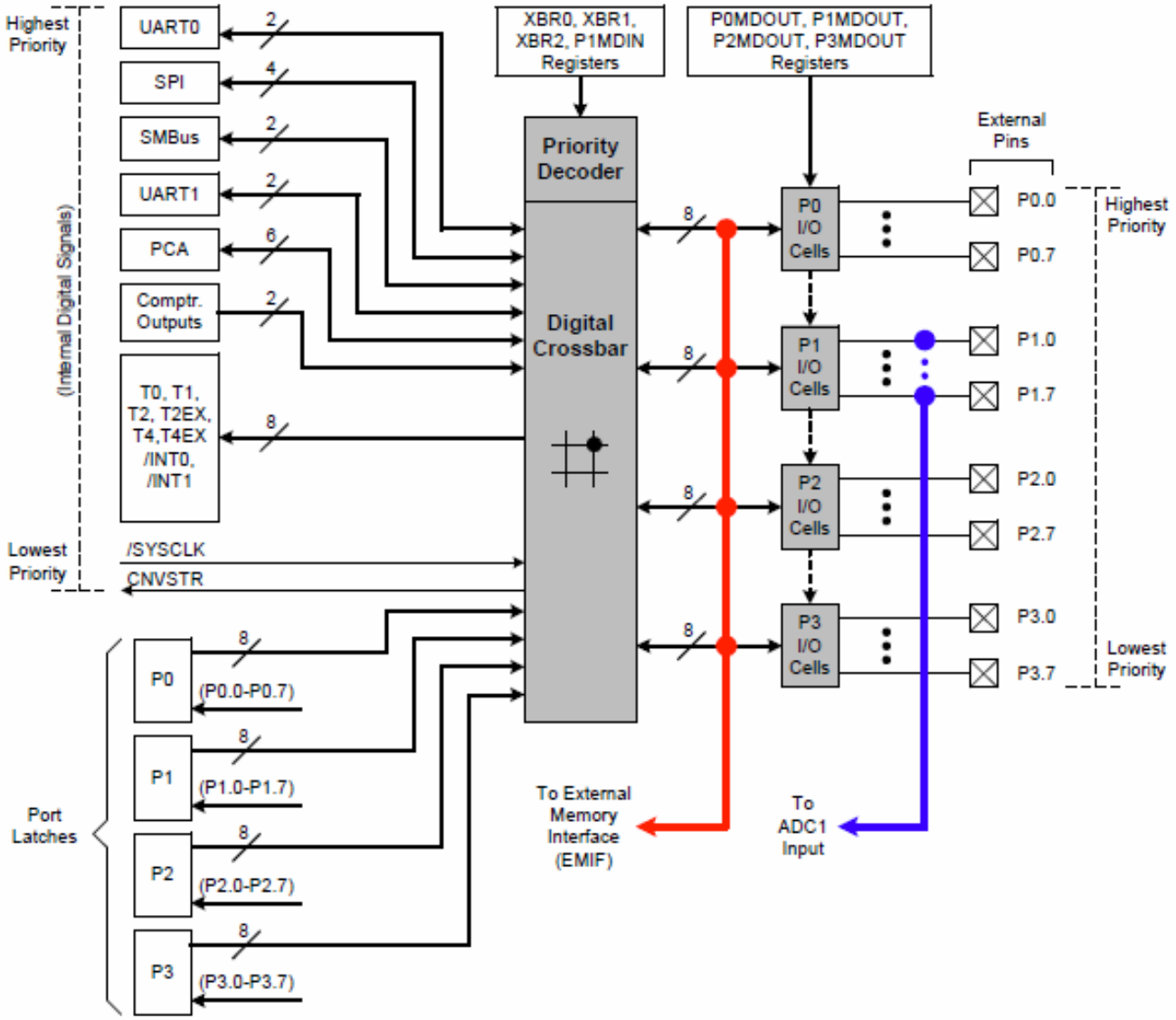


Figure 2.6 Block Diagram of Lower Port I/O (P0 to P3)

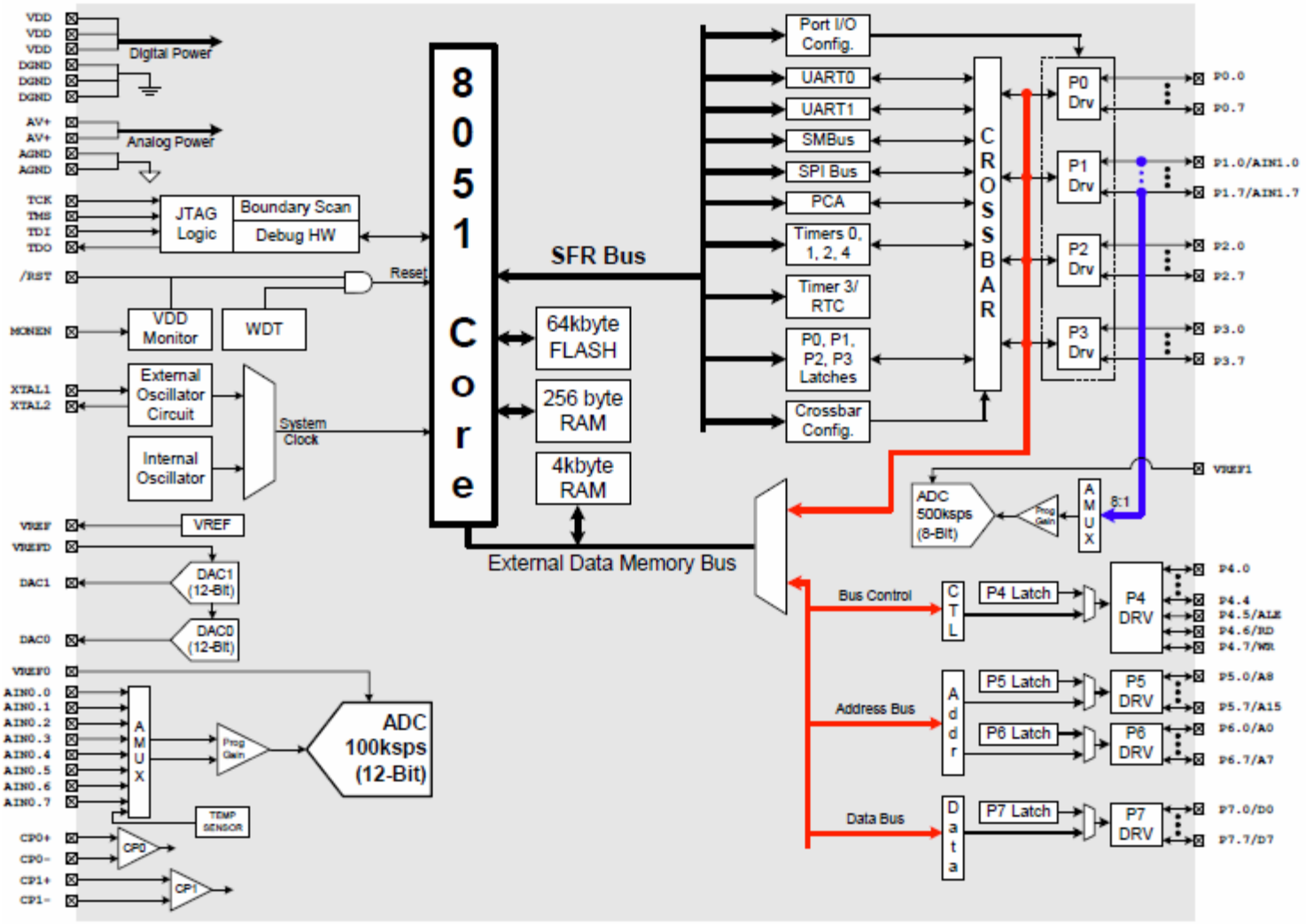


Figure 2.3 Block Diagram of C8051F020

Specs

Analog Peripherals

-**10-Bit ADC (C8051F340/1/2/3/4/5/6/7/A/B only)**

- Up to 200 ksps
- Built-in analog multiplexer with single-ended and differential mode
- VREF from external pin, internal reference, or V_{DD}
- Built-in temperature sensor
- External conversion start input option

-**Two comparators**

-**Internal voltage reference(C8051F340/1/2/3/4/5/6/7/A/B only)**

-**Brown-out detector and POR Circuitry**

USB Function Controller

-USB specification 2.0 compliant

-Full speed (12 Mbps) or low speed (1.5 Mbps) operation

-Integrated clock recovery; no external crystal required for full speed or low speed

-Supports eight flexible endpoints

-1 kB USB buffer memory

-Integrated transceiver; no external resistors required

On-Chip Debug

-On-chip debug circuitry facilitates full speed, non-intrusive in-system debug (No emulator required)

-Provides breakpoints, single stepping, inspect/modify memory and registers

-Superior performance to emulation systems using ICE-chips, target pods, and sockets

Voltage Supply Input: 2.7 to 5.25 V

-Voltages from 3.6 to 5.25 V supported using On-Chip Voltage Regulator

HIgh Speed 8051 µC Core

-Pipelined instruction architecture; executes 70% of Instructions in 1 or 2 system clocks

-48 MIPS and 25 MIPS versions available.

-Expanded interrupt handler

Memory

-4352 or 2304 Bytes RAM

-64 or 32 kB Flash; In-system programmable in 512-byte sectors

Digital Peripherals

-40/25 Port I/O; All 5 V tolerant with high sink current

-Hardware enhanced SPI™, SMBus™, and one or two enhanced UART serial ports

-Four general purpose 16-bit counter/timers

-16-bit programmable counter array (PCA) with five capture/compare modules

-External Memory Interface (EMIF)

Clock Sources

-Internal Oscillator: ±0.25% accuracy with clock recovery enabled. Supports all USB and UART modes

-External Oscillator: Crystal, RC, C, or clock (1 or 2 Pin modes)

-Low Frequency (80 kHz) Internal Oscillator

-Can switch between clock sources on-the-fly

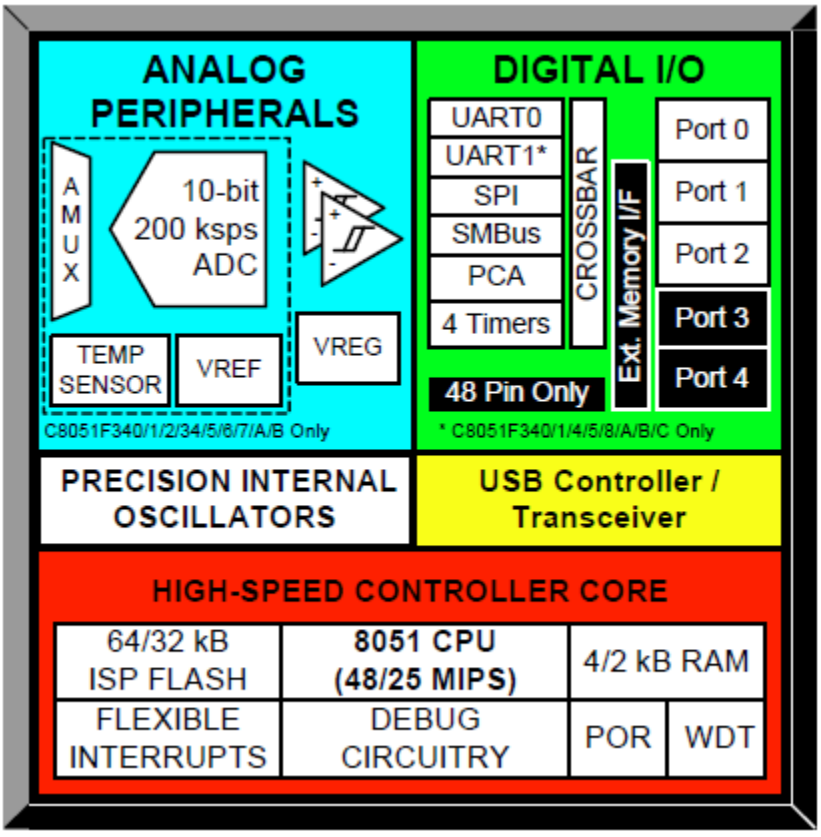
Packages

-48-pin TQFP (C8051F340/1/4/5/8/C)

-32-pin LQFP (C8051F342/3/6/7/9/A/B/D)

-5x5 mm 32-pin QFN (C8051F342/3/6/7/9/A/B)

Temperature Range: –40 to +85 °C



Stack Pointer: 8-bits wide, max size 256 bytes. Grows upward, SP incremented before data are stored per a **push** or **call**.

Data Pointer: 16-bit held in two 8-bit parts. Holds 16-bit address for certain instructions. Can be used as single 16 bit reg or two 8 bit regs.

Port Latches (P0, P1, P2, P3): 32 IO pins organized into four 8-bit ports. B-bit port outputs are latched.

Serial Data Buffer: two regs which share a common address. Data written to SBUF goes to a transmit buffer and is held for serial transmission. Data read from the SBUF comes from the serial data receive buffer.

Timer Registers: TH0 and TL0 are high and low bytes of the 16-bit counter/timer 0. Likewise for counters 1 and 2.

Control Registers: registers for control and status of the interrupt system, timer/counter system, and serial port. They are:

IP: interrupt priority
IE: interrupt enabled
TMOD: timer mode
TCON: timer control
SCON: serial port control
PCON: power control

Open Collector / Open Drain

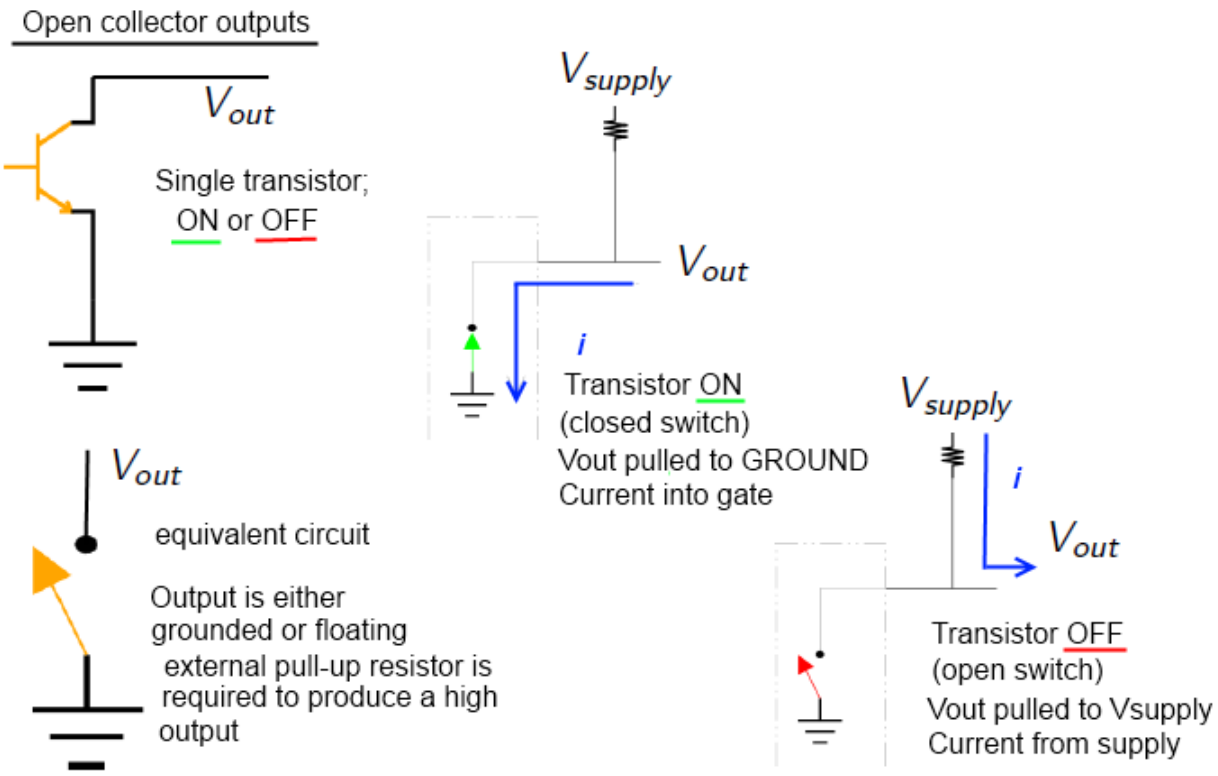
Open-collector/open-drain is a circuit technique which allows multiple devices to communicate bi-directionally on a single wire.

Open-collector/open-drain devices sink (flow) current in their low voltage active (logic 0) state, or are high impedance (no current flows) in their high voltage non-active (logic 1) state. These devices usually operate with an external pull-up resistor that holds the signal line high until a device on the wire sinks enough current to pull the line low. Many devices can be attached to the signal wire. If all devices attached to the wire are in their non-active state, the pull-up will hold the wire at a high voltage. If one or more devices are in the active state, the signal wire voltage will be low.

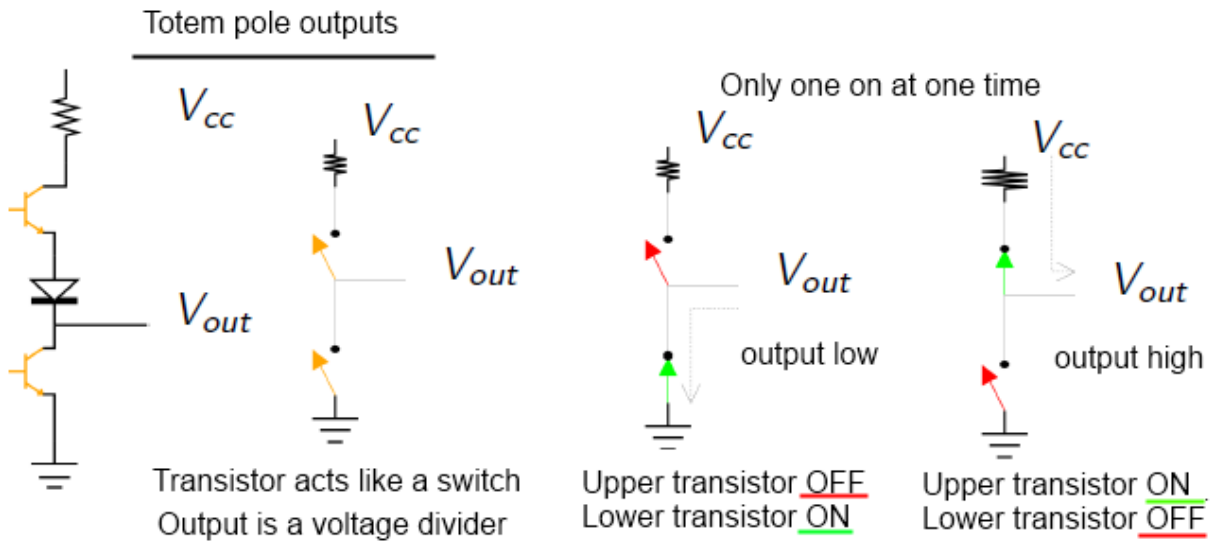
An open-collector/open-drain signal wire can also be bi-directional. Bi-directional means that a device can both output and input a signal on the wire at the same time. In addition to controlling the state of its pin that is connected to the signal wire (active, or non-active), a device can also sense the voltage level of the signal wire. Although the output of a open-collector/open-drain device may be in the non-active (high) state, the wire attached to the device may be in the active (low) state, due to activity of another device attached to the wire.

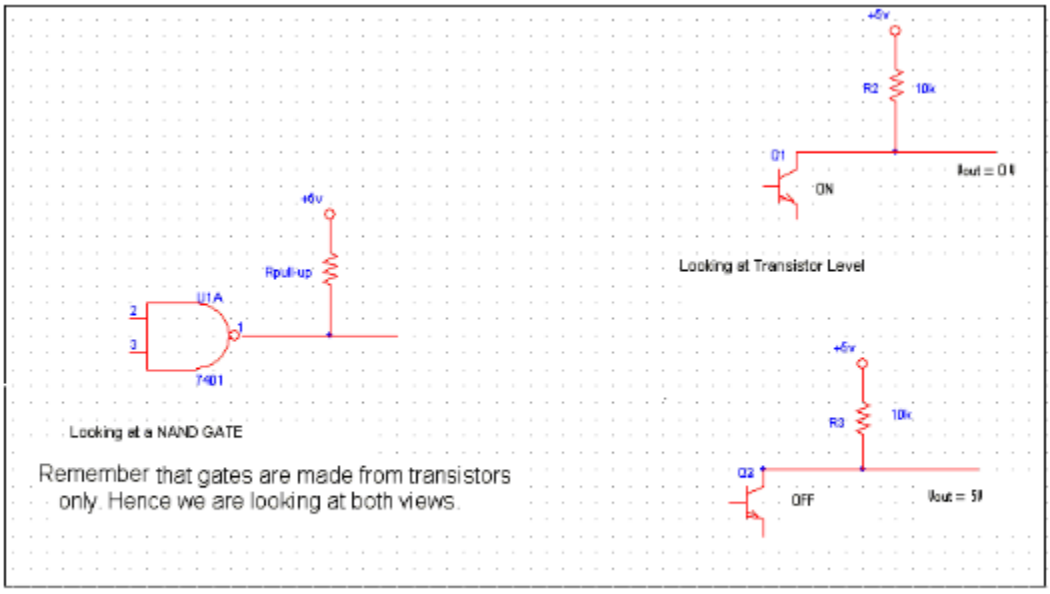
The bi-directional nature of an open-collector/open-drain device is what makes this circuit so important in interconnecting many devices on a common line. The I2C Bus and SMBus uses this technique for connecting up to 127 devices.

Open-drain refers to the drain terminal of a MOS FET transistor. Open-collector is the same concept on a bipolar device



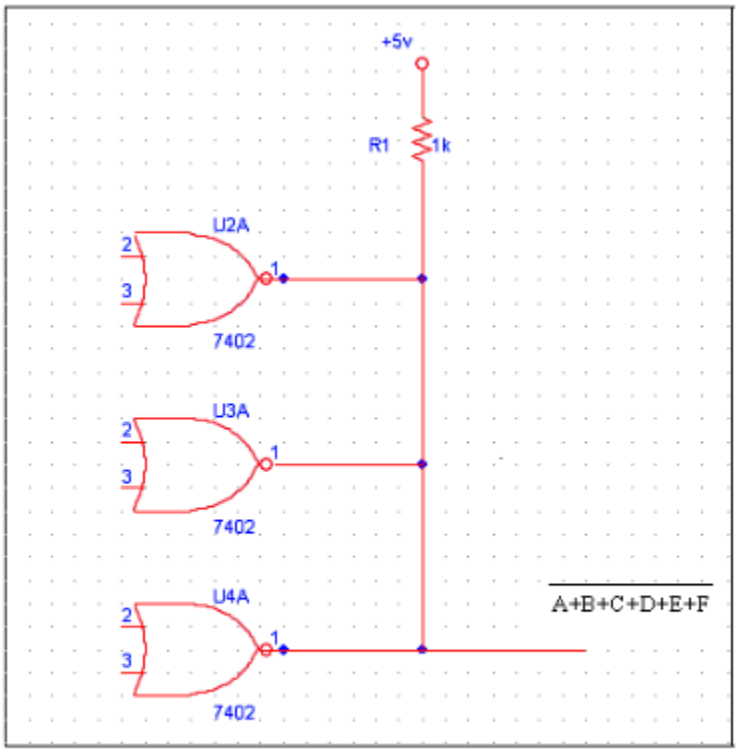
Open Collector ("OC") outputs are low powered, solid state switches. Although the term "Open Collector" stipulates the use of bipolar transistors (NPN-type or PNP-type) as a switch, nowadays Field Effect Transistors (FET or MOSFET) are used. Unlike electro-mechanical switches (e.g. pushbuttons or dry contact relays) these OC switches are very fast, use little power, are inexpensive, do not bounce and do not wear. However, OC's are also more limited in terms of voltage and current rating as well as being polarized (i.e. they have a "plus" and "minus" terminal and thus DC only switching capability). They are less tolerant to overload abuse than electromechanical devices. Usually these switches have higher resistance and voltage drop.





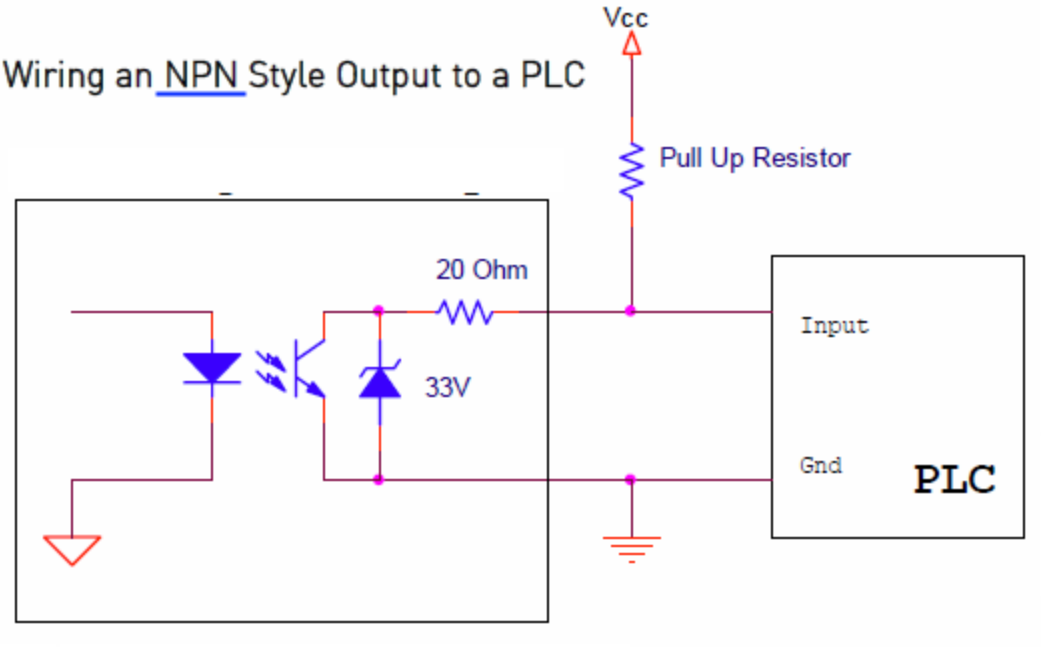
[+] Enlarge Image

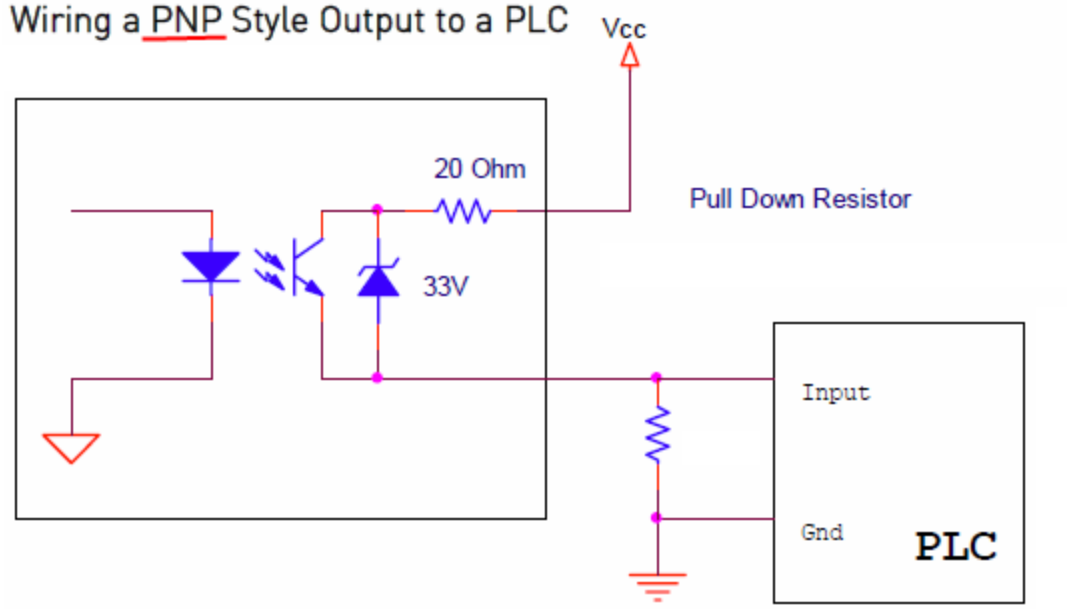
Figure 4. Using a Pull-up Resistor with an Open-Collector Output



[+] Enlarge Image

Figure 7. Wired-OR Connection





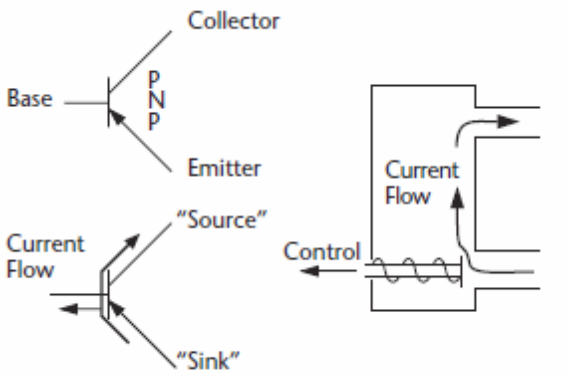


Figure 1-7: Operation of a bipolar PNP transistor.

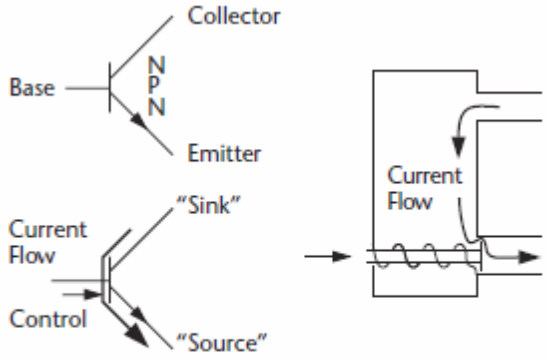


Figure 1-8: Operation of a bipolar NPN transistor.

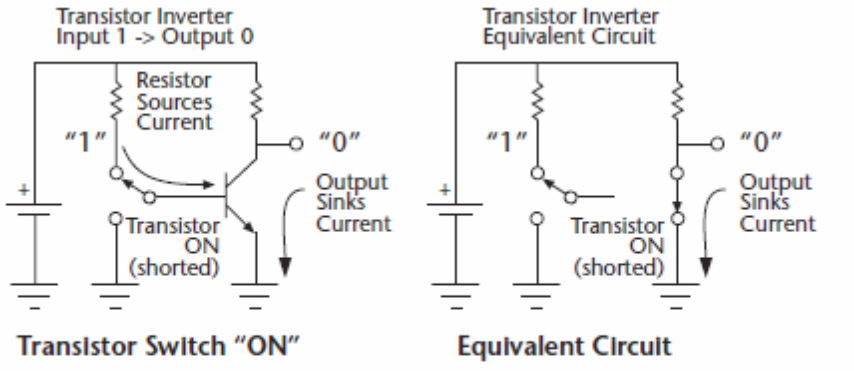


Figure 1-10: The transistor inverter; input = 1 and transistor ON. The transistor ON configuration is at left and the equivalent circuit is at right.

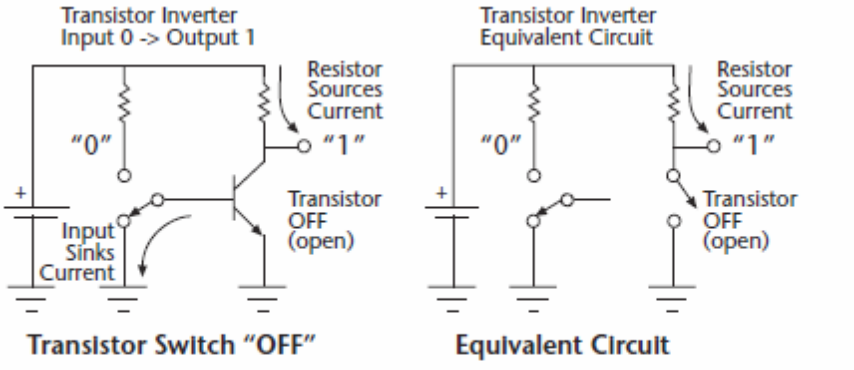


Figure 1-11: The transistor inverter; input = 0 and transistor OFF. The transistor OFF configuration is at left and the equivalent circuit is at right.

FET

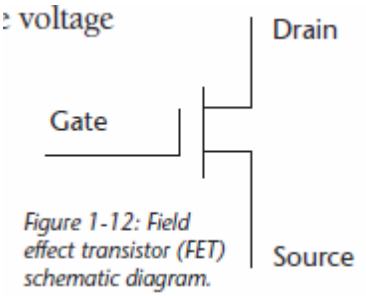


Figure 1-12: Field effect transistor (FET) schematic diagram.

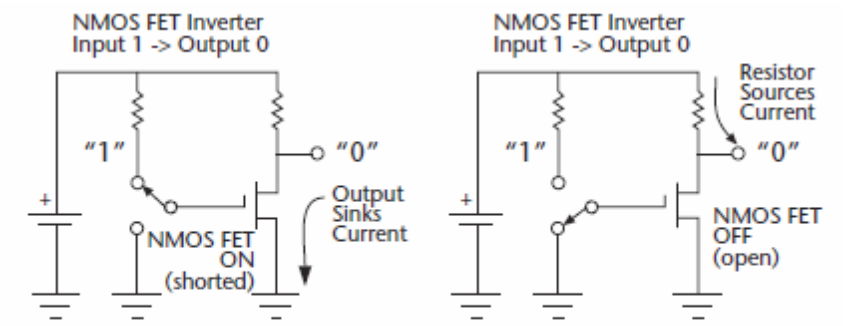


Figure 1-14: NMOS inverter circuit.

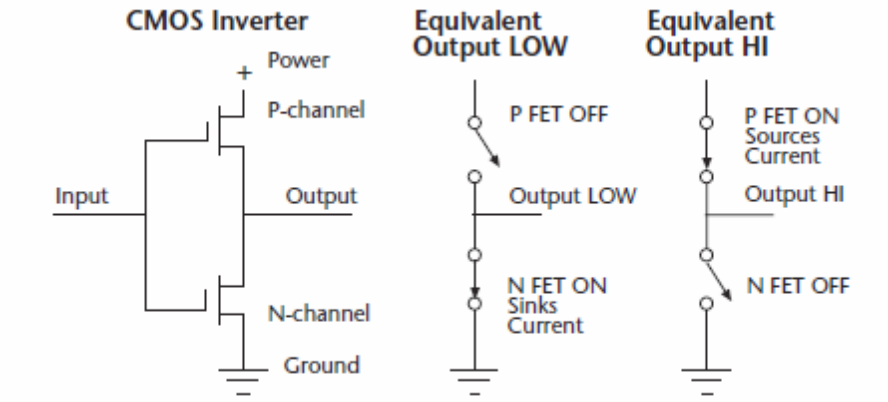


Figure 1-15: CMOS inverter circuit and equivalent output.

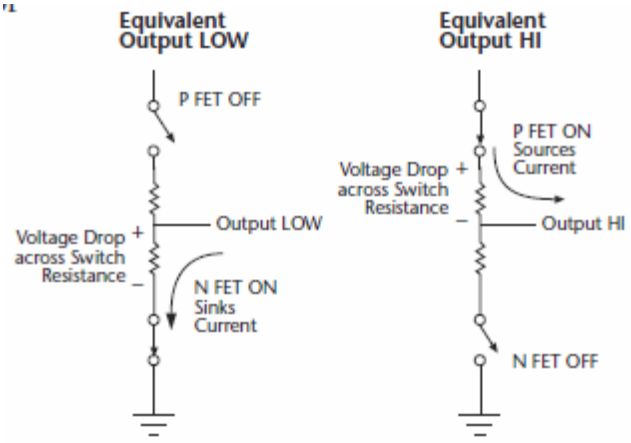


Figure 1-16: Logic output voltage is current dependent.

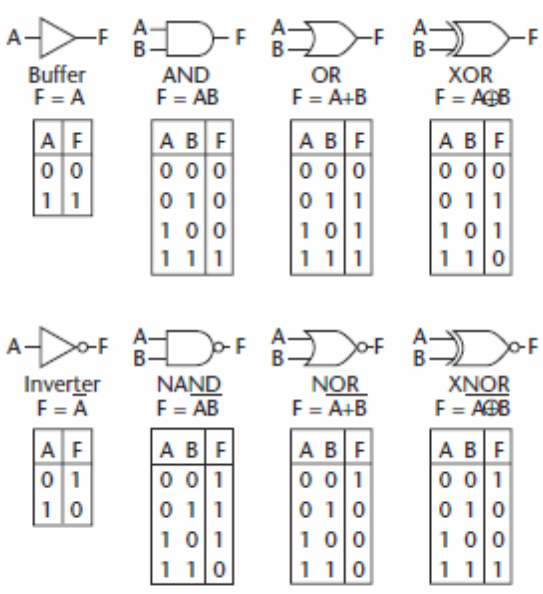


Figure 1-18: Logic symbols, symbolic notation, and truth tables.

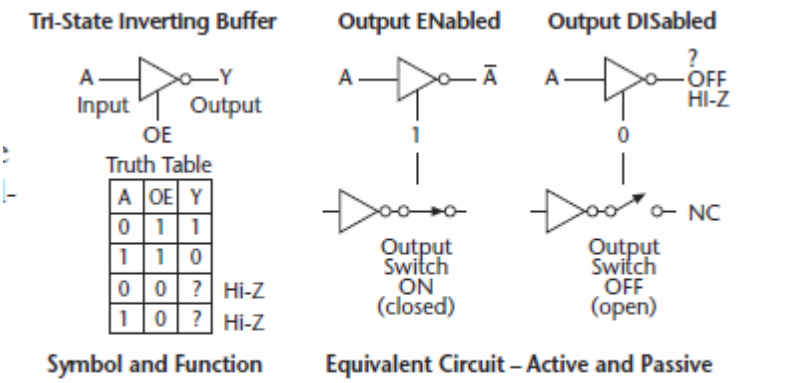


Figure 1-19: Active and passive states of a tri-state buffer.

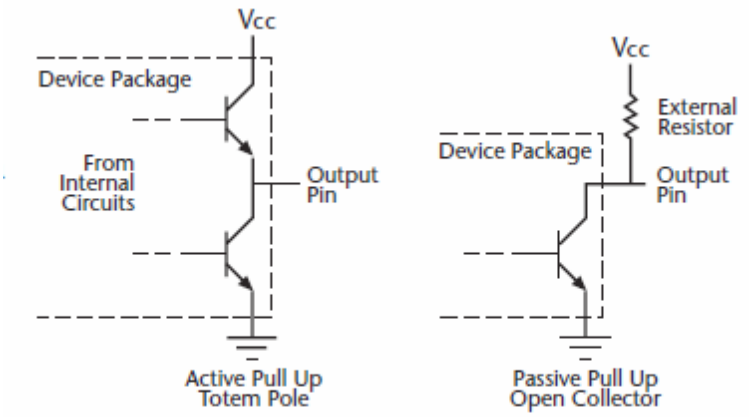


Figure 3-11: TTL outputs, totem pole and open collector.

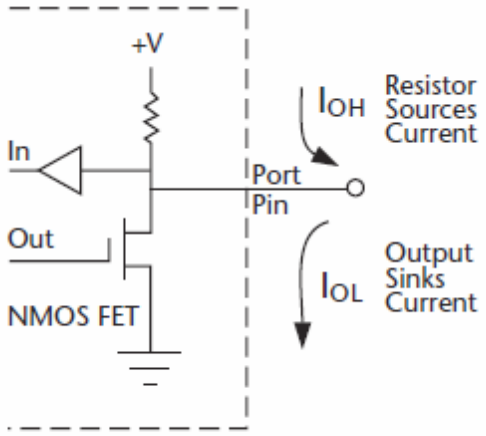


Figure 8-1: Simplified I/O port circuit.

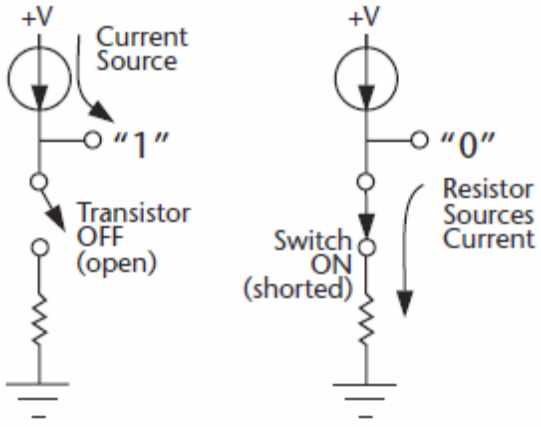


Figure 8-2: Quasi-bi-directional pin.

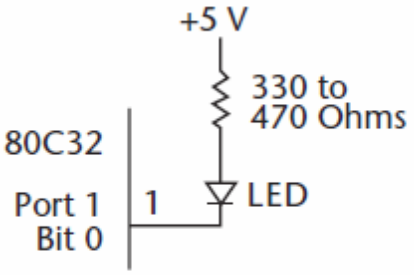


Figure 8-3: Driving a LED directly from a port pin.

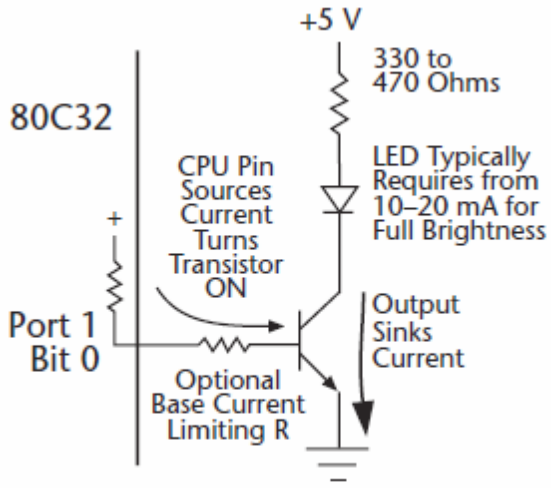


Figure 8-4: NPN transistor for greater load current.

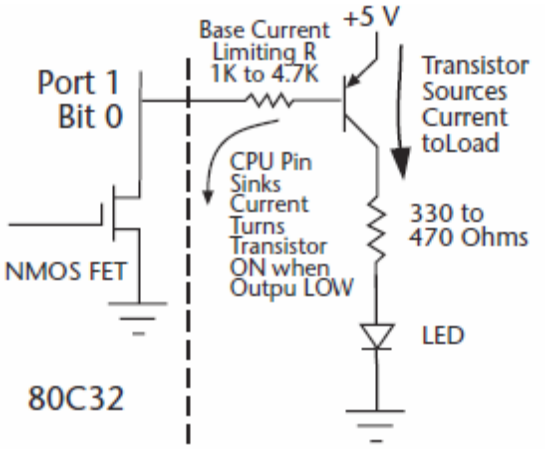


Figure 8-5: PNP transistor output driver.

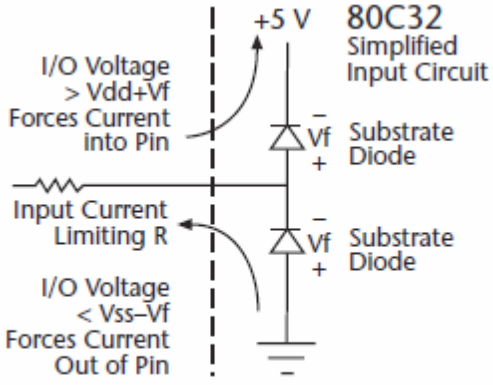


Figure 8-6: I/O pin voltage limits.

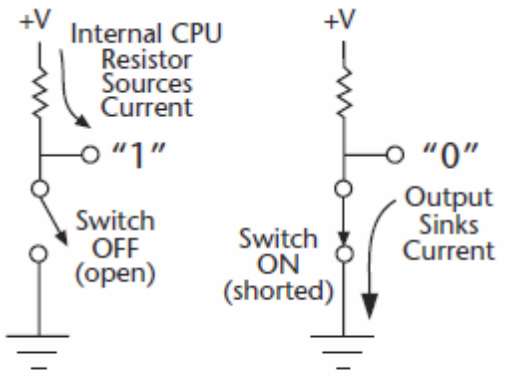


Figure 8-7: Simple switch used as input.

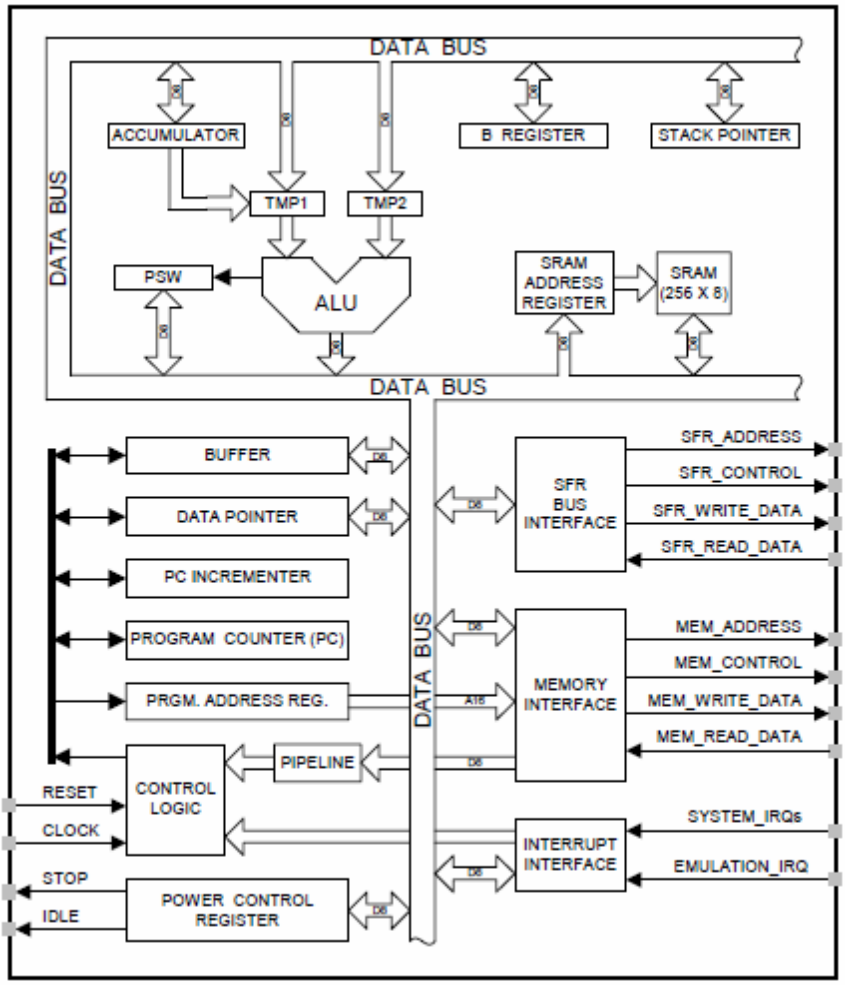


Figure 2.1 Block Diagram of CIP-51

Alternate Functions: all the pins of port 3 have an alternate function, enable alt function by writing a 1 to the corresponding bit in the port latch.

SPI: full duplex. A serial port interrupt is issued in order to read or write serial port data.Reciever frame double buffered, keep receiving while servicing. Tx and Rx buffers are accessed at the same location in the SFR space; the SBUF register. Writing loads Tx, reading offloads Rx. Four modes of operation.

- **Mode 0**: half duplex synchronous, Tx and Rx but not simultaneous. TXD and RXD for synchronous operation.
- **Mode 1**: full duplex, async. Data in and out through RXD and TXD. Frame=Start Bit + 8 data bits + Stop Bit. Interface for data is SBUF.
- **Mode 2**: Similar to mode 1 with two exceptions; 9 data bits (ninth bit obtained from TB8 in SCON), baud rate determined by SMOD [in PCON reg] is 1/32 or 1/64 of oscillator frequency.
- **Mode 3**: same as mode 2 except baud rate is variable.

PCON: of the 8 bits in the PCON power control register, only bit 7, SMOD, is implemented in the standard 8051.

Interrupts: replaces polling and buffering; 5 sources of interrupts

Two from external pins **INT0** and **INT1** **Two** from the **Timer/Counter TF0** and **TF1** **One** from the **serial** port (**TI** or **SI**)

The interrupt sources are associated with bit locations in registers. Each (or all) can be enabled by setting or clearing the appropriate bit in the IE register. If enabled, an interrupt will cause a call to one of the predefined locations in RAM. RETI = return from interrupt, pops last address before jump from stack.

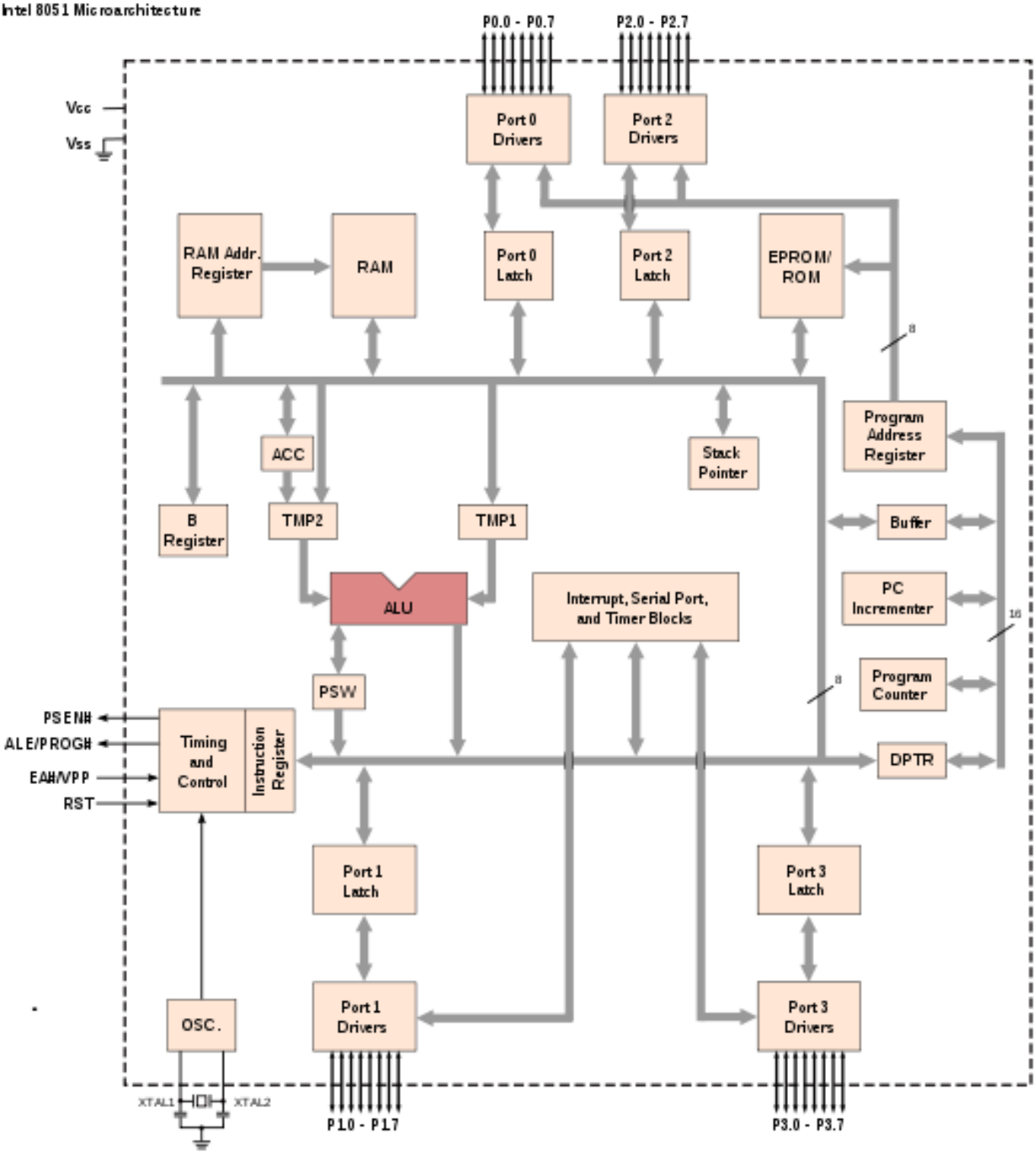
Maskable: an interrupt which can be blocked by software. Interrupts which occur while masked are called **pending**.

Priority scheme is necessary for interrupts which occur simultaneously. The 8051 has a two-tier priority scheme; high and low. Set bits in the IP register to assign one of two priority levels to each interrupt. The second tier of priority is used to resolve simultaneous interrupts within the same interrupt level.

8051 shortest response time: 3 machine cycles, 9 machine cycles is the worst case.

Interrupts can be either level activated or transition (edge) activated. The 8051 has the following predefined vector addresses: **RESET 00H**, **EXTI0 03H**, **TIMER0 0BH**, **EXTI1 13H**, **TIMER1 1BH**, **SINT 23H**.

Intel 8051 Microarchitecture



Instructions & Addressing

The 8051 has five addressing modes and five groups of instructions.

Direct Addressing: instructions using direct addressing are two bytes long; an 8-bit op code followed by an 8-bit address. The address is either a byte location or a specific bit in a bit addressable byte.

Indirect Addressing: instructions specify a register which contains the address of the operand. Must instructions which use indirect addressing are only 1 byte long making them efficient in execution time and space.

8 bit addresses use registers R0-R7.

16 bit addresses use the DPTR register.

Register Instructions

Register instructions use the contents of one of the registers, R0 – R7, as the operand. Always 1 byte long. A few are register specific.

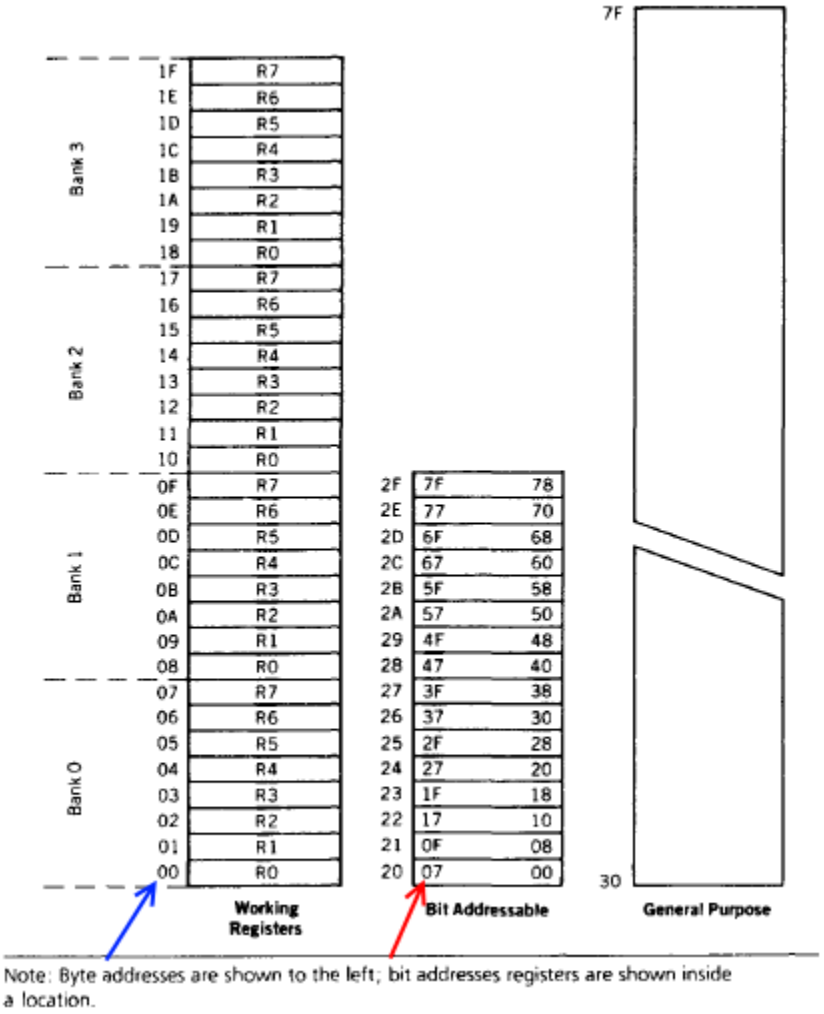


Table 2.2 Predefined bit addresses for 8051.

Symbol	Bit position	Bit address	Meaning
CY	PSW.7	D7H	Carry Flag
AC	PSW.6	D6H	Auxiliary Carry Flag
F0	PSW.5	D5H	Flag 0
RS1	PSW.4	D4H	Register Bank Select Bit 1
RS0	PSW.3	D3H	Register Bank Select Bit 0
OV	PSW.2	D2H	Overflow Flag
P	PSW.0	D0H	Parity Flag
TF1	TCON.7	8FH	Timer 1 Overflow Flag
TR1	TCON.6	8EH	Timer 1 Run Control Bit
TF0	TCON.5	8DH	Timer 0 Overflow Flag
TR0	TCON.4	8CH	Timer 0 Run Control Bit
IE1	TCON.3	8BH	Interrupt 1 Edge Flag
IT1	TCON.2	8AH	Interrupt 1 Type Control Bit
IE0	TCON.1	89H	Interrupt 0 Edge Flag
IT0	TCON.0	88H	Interrupt 0 Type Control Bit
SM0	SCON.7	9FH	Serial Mode Control Bit 0
SM1	SCON.6	9EH	Serial Mode Control Bit 1
SM2	SCON.5	9DH	Serial Mode Control Bit 2
REN	SCON.4	9CH	Receiver Enable
TB8	SCON.3	9BH	Transmit Bit 8
RB8	SCON.2	9AH	Receive Bit 8
TI	SCON.1	99H	Transmit Interrupt Flag
RI	SCON.0	98H	Receive Interrupt Flag
EA	IE.7	AFH	Enable All Interrupts
ES	IE.4	ACH	Enable Serial Port Interrupt
ET1	IE.3	ABH	Enable Timer 1 Interrupt
EX1	IE.2	AAH	Enable External Interrupt 1
ET0	IE.1	A9H	Enable Timer 0 Interrupt
EX0	IE.0	A8H	Enable External Interrupt 0
RD	P3.7	B7H	Read Data for External Memory
WR	P3.6	B6H	Write Data for External Memory
T1	P3.5	B5H	Time/Counter 1 External Flag
T0	P3.4	B4H	Timer/Counter 0 External Flag
INT1	P3.3	B3H	Interrupt 1 Input Pin
INT0	P3.2	B2H	Interrupt 0 Input Pin
TXD	P3.1	B1H	Serial Port Transmit Pin
RXD	P3.0	B0H	Serial Port Receive Pin
PS	IP.4	BCH	Priority of Serial Port Interrupt
PT1	IP.3	BBH	Priority of Timer 1 Interrupt
PX1	IP.2	BAH	Priority of External Interrupt 1
PT0	IP.1	B9H	Priority of Timer 0
PX0	IP.0	B8H	Priority of External Interrupt 0

Table 2.3 Predefined data addresses for 8051.

Symbol	Hexadecimal address	Meaning
ACC	E0	Accumulator
B	F0	Multiplication Register
DPH	83	Data Pointer (high byte)
DPL	82	Data Pointer (low byte)
IE	A8	Interrupt Enable
IP	B8	Interrupt Priority
P0	80	Port 0
P1	90	Port 1
P2	A0	Port 2
P3	B0	Port 3
PSW	D0	Program Status Word
SBUF	99	Serial Port Buffer
SCON	98	Serial Port Controller
SP	81	Stack Pointer
TCON	88	Timer Control
TH0	8C	Timer 0 (high byte)
TH1	8D	Timer 1 (high byte)
TL0	8A	Timer 0 (low byte)
TL1	8B	Timer 1 (low byte)
TMOD	89	Timer Mode

Immediate Operand: has a numeric constant, or it's symbolic name, following the opcode.

Index Addressing: two uses; reading data tables from program memory space, implementing jump tables. In each case a 16-bit register holds the base address and the accumulator holds an 8-bit displacement or index. The address of the jump is the sum of the 16-bit base and the 8-bit displacement. Because unsigned, the result is always a forward reference. The base register is either DPTR or PC.

Operand Modifiers: **@** before an operand means indirect addressing is being used, **#** before operand means it is an immediate operand (constant).

Instruction Groups: arithmetic, logic, data transfer, Boolean, branching.

Instruction Sets:

Mnemonic		Description	Byte	Cycle
Arithmetic Operations				
ADD	A,Rn	Add register to accumulator	1	1
ADD	A,direct	Add direct byte to accumulator	2	1
ADD	A, @Ri	Add indirect RAM to accumulator	1	1
ADD	A,#data	Add immediate data to accumulator	2	1
ADDC	A,Rn	Add register to accumulator with carry flag	1	1
ADDC	A,direct	Add direct byte to A with carry flag	2	1
ADDC	A, @Ri	Add indirect RAM to A with carry flag	1	1
ADDC	A, #data	Add immediate data to A with carry flag	2	1
SUBB	A,Rn	Subtract register from A with borrow	1	1
SUBB	A,direct	Subtract direct byte from A with borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A with borrow	1	1
SUBB	A,#data	Subtract immediate data from A with borrow	2	1
INC	A	Increment accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
INC	DPTR	Increment data pointer	1	2
MUL	AB	Multiply A and B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal adjust accumulator	1	1

Logic Operations

ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to accumulator	1	1
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive OR register to accumulator	1	1
XRL	A direct	Exclusive OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive OR indirect RAM to accumulator	1	1
XRL	A,#data	Exclusive OR immediate data to accumulator	2	1
XRL	direct,A	Exclusive OR accumulator to direct byte	2	1
XRL	direct,#data	Exclusive OR immediate data to direct byte	3	2
CLR	A	Clear accumulator	1	1
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate accumulator left through carry	1	1
RR	A	Rotate accumulator right	1	1
RRC	A	Rotate accumulator right through carry	1	1
SWAP	A	Swap nibbles within the accumulator	1	1

Data Transfer

MOV	A,Rn	Move register to accumulator	1	1
MOV	A,direct *)	Move direct byte to accumulator	2	1
MOV	A,@Ri	Move indirect RAM to accumulator	1	1
MOV	A,#data	Move immediate data to accumulator	2	1
MOV	Rn,A	Move accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct byte	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct,#data	Move immediate data to direct byte	3	2
MOV	@Ri,A	Move accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Ri, #data	Move immediate data to indirect RAM	2	1
MOV	DPTR, #data16	Load data pointer with a 16-bit constant	3	2
MOVC	A,@A + DPTR	Move code byte relative to DPTR to accumulator	1	2
MOVC	A,@A + PC	Move code byte relative to PC to accumulator	1	2
MOVX	A,@Ri	Move external RAM (8-bit addr.) to A	1	2
MOVX	A,@DPTR	Move external RAM (16-bit addr.) to A	1	2
MOVX	@Ri,A	Move A to external RAM (8-bit addr.)	1	2
MOVX	@DPTR,A	Move A to external RAM (16-bit addr.)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with accumulator	1	1
XCH	A,direct	Exchange direct byte with accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with accumulator	1	1
XCHD	A,@Ri	Exchange low-order nibble indir. RAM with A	1	1

Boolean Variable Manipulation

CLR	C	Clear carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set carry flag	1	1
SETB	bit	Set direct bit	2	1
CPL	C	Complement carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to carry flag	2	2
ANL	C,/bit	AND complement of direct bit to carry	2	2
ORL	C,bit	OR direct bit to carry flag	2	2
ORL	C,/bit	OR complement of direct bit to carry	2	2
MOV	C,bit	Move direct bit to carry flag	2	1
MOV	bit,C	Move carry flag to direct bit	2	2

Program and Machine Control

ACALL	addr11	Absolute subroutine call	2	2
LCALL	addr16	Long subroutine call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute jump	2	2
LJMP	addr16	Long iump	3	2
SJMP	rel	Short jump (relative addr.)	2	2
JMP	@A + DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if accumulator is zero	2	2
JNZ	rel	Jump if accumulator is not zero	2	2
JC	rel	Jump if carry flag is set	2	2
JNC	rel	Jump if carry flag is not set	2	2
JB	bit,rel	Jump if direct bit is set	3	2
JNB	bit,rel	Jump if direct bit is not set	3	2
JBC	bit,rel	Jump if direct bit is set and clear bit	3	2
CJNE	A,direct,rel	Compare direct byte to A and jump if not equal	3	2
CJNE	A,#data,rel	Compare immediate to A and jump if not equal	3	2
CJNE	Rn,#data rel	Compare immed. to reg. and jump if not equal	3	2
CJNE	@Ri,#data,rel	Compare immed. to ind. and jump if not equal	3	2
DJNZ	Rn,rel	Decrement register and jump if not zero	2	2
DJNZ	direct,rel	Decrement direct byte and jump if not zero	3	2
NOP		No operation	1	1

Assembler directives are used to define symbols, reserve memory space, store values in program memory and switch between different memory spaces. There are also directives that set the location counter for the active segment and identify the end of the source file.

EQU	Define symbol
DATA	Define internal memory symbol
IDATA	Define indirectly addressed internal memory symbol
XDATA	Define external memory symbol
BIT	Define internal bit memory symbol
CODE	Define program memory symbol
DS	Reserve bytes of data memory
DBIT	Reserve bits of bit memory
DB	Store byte values in program memory
DW	Store word values in program memory
ORG	Set segment location counter
END	End of assembly language source file
CSEG	Select program memory space
DSEG	Select internal memory data space
XSEG	Select external memory data space
ISEG	Select indirectly addressed internal
BSEG	Select bit addressable memory space memory space
USING	Select register bank
IF	Begin conditional assembly block
ELSE	Alternative conditional assembly block
ENDIF	End conditional assembly block

[refer to “8051 Cross Assembler User's Manual” chapter 5 in 8051 directory for definitions.]

code	program memory (up to 64 Kbytes)
data	directly addressable data memory allowing fastest access to variables in the first 128 bytes of the on-chip RAM
idata	indirectly addressable data memory across the full 256 bytes of the on-chip RAM
bdata	bit-addressable data memory allowing bit access of character and integer variables stored inside the (16 bytes) on-chip bit data space
xdata	external data memory up to 64 Kbytes
pdata	paged external data memory allowing access 256 bytes at a time through port 0

Figure 4.1 Six different memory areas in an 8051 system.

The selection of memory types for the different variables directly affects the efficiency of your final program. As a rule, use the memory area of type **data** to store variables that require fast and frequent access. Variables of time-critical interrupt routines should also be put inside the directly addressable internal memory. The **idata** type of internal memory is slower than the **data** memory as it is accessed through indirect addressing. The **idata** memory area is best for arrays or structures that are small in size. All variables that are not time-critical and arrays and structures that are large in size should go inside the **xdata** memory area.

There may be variables whose memory types are not declared explicitly by memory type specifiers, or you may not want to go through the trouble of specifying the memory types of every variable because they are all in the same memory area. Possibly, your program is small enough so that everything would fit into the on-chip memory. In those cases, the following *memory model* definitions allow you to choose either an overall memory area for the entire program (or function) or a default memory type for those variables that are not declared explicitly.

small	Variables and local data are defined to reside in internal data memory, the same as if they were defined by the **data** specifier.
compact	Variables and local data are defined to reside in external data memory, the same as if they were defined by the **pdata** specifier.
large	Variables and local data are defined to reside in external data memory, the same as if they were defined by the **xdata** specifier.

The selection of a memory model can be done either at compile-time (in the Options pull-down menu of Franklin's ProView, for example) or by using the C directive #pragma inside the source code. The default memory model is *small*.

Data Type	*Bits*	*Bytes*	*Value Range*
signed char	8	1	–128 to +127
unsigned char	8	1	0 to 255
enum	16	2	–32768 to +32767
signed short	16	2	–32768 to +32767
unsigned short	16	2	0 to 65535
signed int	16	2	–32768 to +32767
unsigned int	16	2	0 to 65535
signed long	32	4	–2147483648 to 2147483647
unsigned long	32	4	0 to 4294967295
float	32	4	±1.175494E-38 to ±3.402823E+38
bit	1		0 to 1
sbit	1		0 to 1 (SPEC. FUNC. REGS)
sfr	8	1	0 to 255
sfr16	16	2	0 to 65535

There is no explicit compare instruction, it has been absorbed into a special branching instruction.

Data transfer instructions are compatible with all addressing modes.

The Boolean operators are associated with the 8051 single-bit Boolean processor.

Branching instructions include calls, returns, and various conditional and unconditional jumps.

Conditional jumps are relative to the first byte of the next instruction. Jumps are given as signed twos-compliment 8-bit numbers, the range is -128 to +127 bytes (forward and backwards).

Jumps:

- **short**, relative offset
- **long**, 16-bit address, access full memory
- **absolute**, 11-bit address, 8 lower bits, 3 upper bits combined with a 5-bit operation specific

CJNE combines the functions of separate compare and jump instructions.

NOP: 1-byte instruction which does nothing, takes one machine cycle. Use for padding delay loops, canbe used to determine pulse width.

Single-Bit Boolean Processor: all port lines as well as 128-bits of RAM and many bits in the SFR register. This is useful in many on-off kinds of control applications such as I/O, switches, lamps, relays, stepper-motor drives, etc.

Carry: The Boolean Accumulator

Bit 7 of the PSW is the carry bit, it is the equivalent of an accumulator for the Boolean processor. Carry bit is known by several names depending on function:

CY: when referring to a bit address.

C: used in register specific instructions which reference the carry bit.

CLR C: one byte register instruction, clears the carry bit.

CLR CY: 2-byte instruction that clears the carry bit by referencing its address in the SFR space.

Bits from SFR register, from internal RAM and from I/O ports, can be read into CY. AND and OR operations can be performed on the CY and result written back to a bit address. Extensive bit manipulation can be done without having to resort to extraneous code.