**89c51 Single-Chip Microprocessor**

[a quick reference] 101-06-20

**CONTENTS:**

**[A] General Descriptions**

**[B] Pin-out Assignment**

**[C] Built-in Memory Space Allotment**

**[D] Timing sequence in an Instruction Cycle**

**[E] IO ports**

**[F] Built-in Timer Operations**

**[G] Built-in SIO Operations**

**[H] Interrupt Mechanism**

**[I] Instruction Set**

**[J] External Memory Access Cycle**

**[A] General Descriptions**

89c51 is a member in the renowned 8051 single-chip microprocessor family, featuring internal 4KB flash as code memory and 256B RAM as data memory, as well as built-in timers (x3) and serial/parallel IOs.

(A.1) Of the built-in 256B RAM, some addresses are designated as special purpose registers in support of general data operations or operation controls of built-in IO devices (refer to sec.[C]).

(A.2) The interrupt mechanism handles 6 (5 for 80c51) interrupts, of which two are external and three are internal, with priority policy (refer to sec.[F]).

(A.3) Offering 4 parallel ports (4 top) for 8-bit IO, and 1 serial port for bit-sequence IO which could be readily adapted for RS232 communication.

(A.4) Providing with 3 (2 for 89c51) built-in timers, of which the time-out event, by proper operation control, may occur as flag-setting for s/w polling or as interrupt.

(A.5) If the built-in memory is not enough, either the code or data space, for a certain applications, external memory operational mode can be deployed, under which a maximum of 64KB would be available respectively for the code and data space (refer to sec.[G]).

**[B] Pin-out Assignment**

Fig.1 below shows the 40-pin DIP-package of a typical 8051 microcontroller, 89c51 and many other 51-variants as well.

|  |
| --- |
| VCC    40    XTAL1(18) (39) P0.0  PORT 0  XTAL2(19) [ADDR0-7 |  RESET(9) (32) P0.7 DATA0-7]   1. P1.0   EA(31) PORT 1  PSEN(29)  (8) P1.7  RX(10) (21) P2.0  TX(11) PORT2  INT0(12) [ADDR8-15]  PORT 3 INT1(13) (28) P2.7  T0(14)  T1(15)  WR(16)  RD(17)  VSS  Fig.1 89c51 DIP pin outs |

|  |  |  |
| --- | --- | --- |
| pin# | function name | description |
| 18/19 | XTAL1/2 | oscillatory crystal input/output |
| 9 | RESET | system startup |
| 31 | EA [VPP] | external access;  pulled LOW for operating in external memory access mode, otherwise working with built-in memories alone |
| 30 | ALE [PROG] | Address Latch Enable;  during the external memory operating mode, serving as ADDR/DATA indicator of P0.  ALE:=Hi 🡪 P0 acting as the address bus(ADDR0-7)  ALE:=Lo 🡪 P0acting as the data bus. |
| 29 | PSEN | Program Store Enable;  during the external memory operating mode, indicating the phase of code-fetch during an instruction cycle. |
| 10 | Rx [P3:0] | either as P3 or specific IO pins  bit-sequence input |
| 11 | Tx [P3:1] | bit-sequence output |
| 12 | INT0 [P3:2] | external interrupt |
| 13 | INT1 [P3:3] | external interrupt |
| 14 | T0 [P3:4] | external clock source for TMR0 |
| 15 | T1 [P3:5] | External clock source for TMR1 |
| 16 | WR [P3:6] | control of external data-memory write |
| 17 | RD [P3:7] | control of external data-memory read |
| 40 | Vcc | PWR |
| 20 | Vss | GND |
| 39-32 | P0 [P0:0-7] | either as port 0 or  as ADDR0-7|DATA0-7 when EA is pulled low |
| 1-8 | P1 [P1:0-7] | port 1 |
| 21-28 | P2 [P2:0-7] | either as port 2 or as ADDR8-15 when EA is pulled low |

[notes]

\*\* When EA is pulled low, the chip works inexternal memory operation mode with a maximum of 64KB respectively for the code and data space; in such a case, port 0 and port 2 no longer act as IO ports, and ALE, PSEN, WR and RD becomes active in a specific memory access cycle as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EA | ALE | PSEN | WR | RD | **types of memory cycle** |
| 0 | 1 | 1 | 1 | 1 | P0 conveying ADDR0-7  P2 conveying ADDR8-15 |
| 0 | 1 | 1 | 1 | P0 switching from ADDRL to DATA0-7 |
| 0 | 1 | 0 | 1 | data memory read |
| 0 | 1 | 1 | 0 | data memory write |
| 0 | 0 | 1 | 1 | code memory read (FETCH) |
| other combinations | | | | malfunction |
| 1 | **X** | | | | built-in memory operations |

**[C] Built-in Memory Space Allotment**

(c.1) Code memory space: 4KB large for 89C51, address starting from 0000H to

0FFFH.

(c.2) Data memory space: 256B divided into two 128B halves, the allotment of which is as follows.

|  |
| --- |
| 00H 4 banks of  gen.-purpose 32 bytes  registers  1FH [R0-R7]\*4  20H BIP-MAP 16 bytes  2FH [128 bits space]  30H  User’s Space 80 bytes  7FH  80H  SFR 128 bytes  FFH  Fig.2 Built-in 256B SRAM |

|  |
| --- |
| 00H addr[R0]: 00H  R0-R7 . . .  [bank0]  07H addr[R7]: 07H  08H addr[R0]: 08H  R0-R7 . . .  [bank1]  0FH addr[R7]: 0FH  10H addr[R0]: 10H  R0-R7 . . .  [bank2]  17H addr[R7]: 17H  18H addr[R0]: 18H  R0-R7 . . .  [bank3]  1FH addr[R7]: 1FH  Fig.3 32-byte of R0-R7 banks  [only one bank of four is accessible by 51-instruction at any instant, to be determined by RS0-RS1 field of PSW register in SFR] |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit7 bit0   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 07h | 06h | 05h | 04h | 03h | 02h | 01h | 00h | | 0Fh | 0Eh |  |  |  |  | 09h | 08h | | (BIT MAP)  **. . .**  (128 bits from 00h to 7Fh) | | | | | | | | | 77h | 76h |  |  |  |  | 71h | 70h | | 7Fh | 7Eh | 7Dh | 7Ch | 7Bh | 7Ah | 79h | 78h |   21H  **. . .**  2FH  Fig.4 bit-address assignment in BIT MAP |

(c.3) Allotment in SFR

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **addr** | **registers** | **functions** | | 80H | **P0\*** | port 0 | | 81H | **SP** | stack pointer | | 82H | **DPL** | DPTRL | | 83H | **DPH** | DPTRH | |  |  |  | | 87H | **PCON\*** | power control register | | 88H | **TCON\*** | timer control register | | 89H | **TMOD** | timer mode register | | 8AH | **TL0** | T-length register0L | | 8BH | **TL1** | T-length register1L | | 8CH | **TH0** | T-length register0H | | 8DH | **TH1** | T-length register1H | |  |  |  | | 90H | **P1\*** | port 1 | |  |  |  | | 98H | **SCON\*** | SIO control register | | 99H | **SBUF** | SIO buffer register | |  |  |  | | A0H | **P2\*** | port 2 | |  |  |  | | A8H | **IE\*** | interrupt enable register | |  |  |  | | B0H | **P3\*** | port 3 | |  |  |  | | B8H | **IP\*** | interrupt priority register | |  |  |  | | D0H | **PSW\*** | Processor Status Word register | |  |  |  | | E0H | **ACC\*** | accumulator | |  |  |  | | F0H | **B\*** | B register |   Fig.5 Special Function Registers |

(c.4) usage of SFRs

[**ACC**]: for accumulating the result of arithmetic/logic operations; thus the

name– accumulator

[**PSW**]: bookkeeping of CPU status out of all instruction execution

bit7 bit0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CY | AC | F0 | RS1 | RS0 | OV | F1 | P |

carry

auxiliary carry

user defined

bank selector(R0-7/bank))

1. bank0
2. bank1
3. bank2
4. bank3

overflow

user defined

1 for odd-parity in ACC

[**IE**]: masking of interrupts, “1” as unmasked (enabled), “0” otherwise

(refer to section [G] for more details)

bit7 bit0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| EA |  |  | ES | ET1 | EX1 | ET0 | EX0 |

enable all

SIO Tx/Rxinterrupt

TMR1 interrupt

external interrupt INT1

TMR0 interrupt

external interrupt INT0

[**IP**]: settings of interrupt priorities under the following policy

\*\* “1”setting implying higher priority than “0”setting

\*\* priorities of the same setting resolved by default:

Less significant bit position possessing higher priority, i.e.,

PX0>PT0>PX1>PT1>PS

(refer to section [G] for more details)

bit7 bit0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | PS | PT1 | PX1 | PT0 | PX0 |

SIO Tx/Rxinterrupt

TMR1 interrupt

external interrupt INT1

TMR0 interrupt

external interrupt INT0

Entry points of interrupt handlers (hardwired)

|  |  |
| --- | --- |
| interrupt | handler entry points |
| external interrupt INT0 | 0003H |
| TMR0 interrupt | 000BH |
| external interrupt INT1 | 0013H |
| TMR1 | 001BH |
| SIO Tx/Rx interrupt | 0023H |

[**TCON**] lower half of which is used for external interrupts settings

(refer to section [G] for more details)

bit3 bit0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | IE1 | IT1 | IE0 | IT0 |

external interrupt INT1 status

external interrupt INT1 triggering mode

external interrupt INT0 status

external interrupt INT0 triggering mode

IT0: “0” for level-triggering, “1” for edge-triggering

IE0: “1” for edge-triggered INT0 occurrence, un-serviced yet,

“0” for INT0-handler processing under way

[**TMOD**] settings for the operating modes for the two built-in timers/counters, by which the two timer/counter units could be programmed and work independenly.

(refer to section [E] for more details)

TMR/CNTR1 TMR/CNTR0

bit7 bit0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| GATE | C/ T | M1 | M0 | GATE | C/ T | M1 | M0 |

external

triggering of

TMR1

clock-source

selection

operation modes

external triggering of TMR0

clock-source selection

operation modes

[**TCON**] TMR/CNTR timing/counting initiation and termination

(refer to section [E] for more details)

bit7 bit0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TF1 | TR1 | TF0 | TR0 |  |  |  |  |

CNTR1

count done

count start

CNTR0

[**SCON**] settings for SIO operation in 4 modes by which serial communication in bit-sequence is accomplished; data rate of SIO is determined by mode setting as well as MSB in PCON register;

|  |
| --- |
| FEATURE: SM2 supports the operation of multi-51 system in master-slaves architecture in which linking among CPUs are achieved via SIO communications |

(refer to section [F] for more details)

bit7 bit0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |

SIO mode

selection

multi-51

operations

Rx input enable

the 9th data-bit in an outbound 11-bit frame

the 9th data-bit in an inbound 11-bit frame

outbound transmission-complete indicator

inbound transmission-complete indicator

[**PCON**] 51CPUpower control register

|  |
| --- |
| In consideration of power consumption, 89C51 supports the following operation modes when required.   1. stop-clocking: external oscillatory clocking rate can be can be down   to 0Hz, while contents of built-in RAM and SFRs remain intact;   1. IDLE mode:   \*\* by setting PCON:IDL bit to “1”, 89c51 goes sleeping while all built-in modules(TMRs and SIO) remain active, data in RAM and SFRs stay intact;  \*\* 89c51 can be waked up from IDLE mode by either any interrupts or by RESET;  \*\* when invoked by an interrupt, the associated handler will be executed and then the execution resumes from the instruction immediately following the IDLE-initiating instruction;  \*\* when invoked by RESET, the execution resumes from the instruction immediately following the one initiating IDLE mode, for two machine cycles, after which the normal H/W reset process starts.   1. Power Down mode:   \*\* by setting PCON:PD bit to “1”, PD mode is entered; during the period data in built-in RAM and SFRs are retained at as low as 2V;  \*\* exiting from PD can only be done by RESET or external interrupts, INT0 or INT1;  \*\* prior to the assertion of RESET or INT0/1, external clocking source and VCC must be restored to the standard value and stabilized. |

bit7 bit0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SMOD |  |  |  | GF1 | GF0 | PD | IDL |

half/full-rate

in SIO comm.

user defined

1:= 51CPU is in IDLE/PowerDown state

0:= 51CPU at normal state

1-setting entering Power Down mode

0-setting clear PD mode

1-setting entering IDLE mode

0-setting clear IDLE mode

**[D] Timing sequence in an Instruction Cycle**

The instruction cycle, I-cycle, refers to the time duration required for an instruction to be executed from its beginning to the ending, which consists of three phases from the perspective of CPU code execution:

\*\* read the 1st byte machine code

FETCH according to PC; PC++

\*\* read the 2nd byte of machine code

Next I-Cycle DECODE according to PC; PC++

\*\* read the 3rd byte of machine code

according to PC; PC++

EXECUTE

\*\*

The code length of a 51-instruction may vary from one to three byte(s), and the required execution time varies from 1, 2, or 4 machine cycle(s). Each machine cycle consists of 12 clock cycles from the external oscillatory clocking source, which is a 12MHz osc.-crystal in the 89c51-based microprocessor circuit board. Thus the instruction may take 1, 2 or 4 microsecond(s) to complete in our lab-work.

Note that each machine cycle is also staged up in six phases, S1 to S6, each being of two clocks long. FETCH occurs regularly at both S1 and S4, DECODE at S2.

(D.1) one-machine-cycle instruction (1-μs):

I-Cyclej I-Cyclej+1

M1 M1

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |

CLK

**1B-inst.**

[F] [D] [E]

**2B-inst.** [F1] [D] [F2] [E]

\*\* [F]: fetch stage; [D]: decode stage; [E]: execution stage

\*\* [F1]: first fetch for the two-byte codes

\*\* [F2]: second fetch for the two-byte codes

(D.2) two-machine-cycle instruction (2-μs):

I-Cyclej

M1 M2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |

CLK

**1B-inst.**

[F] [D] [E]

**2B-inst.** [F1] [D] [F2] [E]

**3B-inst.** [F1] [D] [F2] [F3] [E]

\*\* [F]: fetch stage; [D]: decode stage; [E]: execution stage

\*\* [F1]: first fetch for the two/three-byte codes

\*\* [F2]: second fetch for the two-/three-byte codes

\*\* [F3]: third fetch for the three-byte codes

(D.3) four-machine-cycle instruction (4-μs):

MUL and DIV only

I-Cyclej

M1 M2 M3 M4

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

CLK

**1B-inst.**

[F] [E]

[D]

\*\* why are these two taking so loooong :-) ??

**[E] IO ports**

internal memory access mode: 4 IO ports available, P0, P1, P2, and P3;

external memory access mode: P0 as D0-7 and A0-7

P2 as A8-15

P1 and P3 as IO ports

**[PORT0]**

ADDR/DATA control [0: int. mem-access mode]

latch-RD

int.-pullup

P0.x

int. Dx D Q

int. WR

pin-RD

**[PORT2]** ADDR control

latch-RD

int,-pullup

P2.x

int Dx D Q

int WR

pin-RD

**[PORT1]**

latch-RD

int,-pullup

P2.x

int Dx D Q

int WR

pin-RD

**[F] Built-in Timer Operations**

8051 supports two built-in counters (which could be used as timers when fed with period counting signals; 89c51 offers one extra counter, not discussed in this quick reference though. Fig.E.1 depicts functional block diagram of the two counters, TMR0 and TMR1.

|  |
| --- |
| ÷12  TMR0  T0 clk time-out TF0  T-intrrpt  start LD  TR0  INT0 GATE TH0 TL0  TMOD TCON    TMR1  T1 clk time-out TF1  T-intrrpt  start LD  TR1  INT1 GATE TH1 TL1 |

|  |
| --- |
| XTAL ÷12  TF0  T0 TMR0  TMR0  intrrpt.  INT0    GATE TR0 C/ T  counting mode0 : 13-bit counter  mode1: 16-bit counter  mode2: 8-bit counter with auto-reload & GO  mode3: TMR0 🡪 normal 8-bit counter loaded with TL0  TMR1 🡪 8-bit counter loaded with TH0  clocked by AND[TR1, XTAL/12]  **timer start-up procedure:**  [1] TMOD register setting  [2] TL0 [TH0] register setting; TL1[TH1] register setting  [3] TCON register setting; start the timer by TCON:TR0|TR1  [4] IE register and IP register setting if time-up interrupt is desired  **run-time timer status monitoring and restart after time-up:**  [1] TCON:TF0|TF1 polling  [2] resetting TL0 [TH0] register and/or TL1[TH1] register  [3] restart the timer by TCON:TR0|TR1 in not in mode2 operation  [4] handling [2] and [3] in TMR0/TMR1 interrupt service routines if time-up interrupt is in effect.  **Fig. E.1** |

**[G] Built-in SIO Operations**

89c51 supports serial-IO for communication in bit-sequence with a built-in SIO, which allows 4 modes of operations of programmable bit rate (or baud rate) and bit frame. Fig.F.1 depicts functional block diagram of SIO and the context of operations. Multi-51 system operating in master-slave structure is supported by one-to-many interconnecting of master:Tx and slaves:Rx, as well as many-to-one interconnecting of slave:Tx and master:Rx, as illustrated in Fig.F.2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| XTAL ÷12  TMR1 SIO Rx  T0 clk clk Tx  Start load  INT0 TR0/GATE    TH1 TL1 SCON  TCON PCON:SMOD[7]  TMOD  TBUF  RBUF  SIO operational modes:   |  |  |  |  | | --- | --- | --- | --- | | modes | function | clk source | notes | | 0 | extended-IO | XTAL/12 | Rx: serial-out of TBUF  Tx: sync. clk out | | Rx: serial-in of  external data  Tx: sync. clk out | | 1 | UART 10-bit | TMR1 |  | | 2 | UART 11-bit | XTAL/32|64 |  | | 3 | UART 11-bit | TMR1 |  |   10-bit frame  (1) (2) (3)  11-bit frame  (1) (2) (4) (3)  (1) START bit (2) D0-D7 (3) STOP bit (4) TB8 or RB8 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| baud rate computation   |  |  |  |  |  | | --- | --- | --- | --- | --- | | SMOD | mode0 | mod1 | mode2 | mode3 | | 0 |  | TMR1/(2x16) | XTAL/(2x2x16) | TMR1/(2x16) | | 1 |  | TMR1/16 | XTAL/(2x16) | TMR1/16 |     **SIO start-up procedure**  [1] SCON register setting  [2] PCON:SMOD[7] setting  Fig.F.1 |

|  |
| --- |
| **multi-51 system operations**  master  Tx Rx  Tx Rx Tx Rx  slave#0 slave#(n-1)  Tx Rx  slave#j  [1] system start-up  1.) all SIOs take mode2/3 operations at the same baud rate, with REN:=1; also a specific code byte should be agreed upon by both sides as the end-of-comm. message (last byte of comm.);  2.) each slave SIO is assigned an ID and SCON:SB2 is set as 1, which make communications between the master and a designated slave feasible;  [2] master-slave serial comm. procedure  It’s the master initiating the procedure, which then evolves as follows.  1.) the master sends out a slave ID via Tx (a broadcasting in fact), with TB8:=1 (indicating to slave units that this is a ID-broadcast);  2.) with SM2:=1, the receiving of RT8:=1 would result in SIO-interrupt in all slaves (if exploited) and thus the SIO-handler should be so designed as to respond or to reject the ‘poll’ from the master;  3.) only the slave whose ID matches the polling-ID would respond by clearing SM2 and be ready for further data from the master;  4.) for those slaves which are not ‘polled’, SM2 remains as 1;  5.) once sending out the ID with TB8:=1, the master starts sending out data in bytes with TB8:=0  6.) with SM2:=1, slave SIOs simply ignore (without raising RI or generating an SIO-interrupt to 51 CPU) the data broadcasting from the master, because of RT8:=0 in all those data bytes;  7.) the designated slave, while SM2 has been cleared to 0, thus become the receiver of the data from the master;  8.) on receiving the end-of-comm. mark, the designated slave reset SM2 such that SM2:=1 again, waiting for the next ‘polling’ from the master.  \* If the slave needs to send data up to the master, how can it be done given the procedure aforementioned?  Fig.F.2 |

**[G] Interrupt Mechanism**

During the operations of a typical computer system, say control or monitoring, all sorts of events may occur and have to be resolved so that the system may continue. Event-detection generally undertakes two approaches: **polling** and **interrupt**. Event-detection by polling would requires the setting up of check-points in the code execution, and how frequently the check-points should be deployed immediately becomes a critical issue. (why so?) Obviously, polling would not be appropriate for detecting events of which the occurrence is extremely rare, highly unpredictable, and yet the consequence may be severe if not detected in time; fire alarm for instance. This is where interrupt mechanism gets in.

Issues concerning interrupt mechanism supported by all microprocessors may include, yet not limited to, the followings.

1. types of interrupt [internal, external, causes, etc.]
2. number of interrupts [from a few to hundreds]
3. priority policy [hardwired or programmable]
4. interrupt masking [run-time enable/disable of interrupt request(s) or non-maskable]
5. level or edge triggered interrupts
6. handler entry points [hardwired or programmable via interrupt-vector table]

The phenomenon of how the CPU control switching from the interrupted process to the handler, and then back to the process, is depicted in Fig.G.1.

|  |
| --- |
| main interrupt occurs  procedure (0)  (2)  handler   1. (3)   routine work in the system (4)   1. interrupt occurs at a certain time instance while the system is running 2. at S5 of the last machine cycle of an I-cycle, the interrupt is received and acknowledged 3. the interrupt-cycle begins as follows   2.1] save current program-counter value PC\* and CPU flags on top of the stack  2.2] replace PC with handler entry-point, either by hardwired or retrieved form interrupt-vector table   1. as the 1st I-cycle after the interrupt begins, the handler is invoked and starts execution until the last instruction RETI is executed, by which   CPU flags and PC are restored with values popped off the STACK   1. the I-cycle after RETI resumes with the instruction immediately following the interrupted instruction in the main procedure   Fig.G.1 |

**[I] Instruction Set**

The syntax of a 51-instruction line may be in one of the following formats,

\*\*) mnemonics,

\*\*) mnemonics + OPND,

\*\*) mnemonics + OPND1 +OPND2,

\*\*) mnemonics + OPND1 + OPND2 +OPND3,

where OPND may be a 51-register,

an internal/external data memory location,

a label in the code lines,

or a subroutine name;

OPND1 serves the same as OPND, except for being the label or subroutine name;

OPND2 may be a 51-register, an internal/external memory location, an immediate value, or a label in the code lines;

OPND3 can only be a label in the code lines.

The code length of a 51-instruction may vary from 1 to 3 bytes and the code formats

are shown below.

\*\*) OPCODE

\*\*) OPCODE + #data8

\*\*) OPCODE + direct8

\*\*) OPCODE + addr8|11|16

\*\*) OPCODE + direct8 + #data8

\*\*) OPCODE + direct18 + direct28

\*\*) OPCODE + #data8 + addr8

\*\*) OPCODE + direct8 + addr8

By the operations performed, the entire 8051 instruction set can be categorized into functional groups listed below:

(H.1) Arithmetic Operations

|  |  |
| --- | --- |
| A, Rn  ADD A, direct8  ADDC A, @Ri  SUBB A, #data8 | MUL A, B  DIV A, B  DA A |

(H.2) Accumulator Operations(H.3) Increment/Decrement

|  |  |
| --- | --- |
| CLR  CPL  RL  RLC A  RR  RRC  SWAP | A  INC Rn  DEC direct8  @Ri  INC DPTR |

(H.4) Logic Operations(H.5) Data Initialization

|  |  |
| --- | --- |
| A, Rn  A, direct8  ANL A, @Ri  ORL A, #data8  XRL direct8, A  direct8, #data8 | A, #data8  Rn, #data8  MOV direct8, #data8  @Ri, #data8  DPTR, #data16 |

(H.6) Data Transfer

|  |  |
| --- | --- |
| A, Rn  A, direct8  MOV A, @Ri  Rn, direct8  @Ri, direct8  MOV direct18, direct28 | Rn, A  direct8, A  MOV @Ri, A  direct8, Rn  direct8, @Ri |

(H.7) Carry Operations (H.8) Bit Operations

|  |  |
| --- | --- |
| ANL C, bit8  ORL C, bit8  MOV C, bit8  bit8, C  CLR  SETB C  CPL | CLR  SETB bit8  CPL |

(H.9) Unconditional Jumps (H.10) Conditional Jumps

|  |  |
| --- | --- |
| AJMP addr11  LJMP addr16  SJMP rel8  JMP @A+DPTR | A, direct8,rel8  CJNE A, #data8, rel8  Rn, #data8, rel8  @Ri, #data8, rel8    JZ  JNZ rel8  JC  JNC  JB  JNB bit8, rel8  JBC  DJNZ Rn, rel8  direct8, rel8 |

(H.11) Data Exchange (H.12) External Memory Access

|  |  |
| --- | --- |
| A, Rn  XCH A, direct8  A, @Ri  XCHD A, @Ri | A, @Ri  MOVX A, @DPTR  @Ri, A  @DPTR, A |

(H.13) Stack Operations (H.14) Subroutine Calls

|  |  |
| --- | --- |
| PUSH direct8  POP direct8 | ACALL addr11  LCALL addr16  RET  RETI |

**[J] External Memory Access**

[I.1] one-machine-cycle instruction (1-μs):

I-Cyclej I-Cyclej+1

M1 M1

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |

CLK

**1B-inst.**

[F] [D] [E]

**2B-inst.** [F1] [D] [F2] [E]

\*\* [F]: fetch stage; [D]: decode stage; [E]: execution stage

\*\* [F1]: first fetch for the two-byte codes

\*\* [F2]: second fetch for the two-byte codes

timing sequence

I-Cyclej I-Cyclej+1

M1 M1

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |

CLK

ALE

/PSEN

P0 Cj ADDR CODEj+1 ADDR CODEj+1 ADDR CODEJ+2 ADDR

P2 Aj ADDRj+1 ADDRj+1 ADDRj+2 ADDRj+2

F1 X F1 X

[I.2] two-machine-cycle instruction (2-μs):

I-Cyclej

M1 M2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |

CLK

**1B-inst.**

[F] [D] [E]

**2B-inst.** [F1] [D] [F2] [E]

**3B-inst.** [F1] [D] [F2] [F3] [E]

\*\* [F]: fetch stage; [D]: decode stage; [E]: execution stage

\*\* [F1]: first fetch for the two/three-byte codes

\*\* [F2]: second fetch for the two-/three-byte codes

\*\* [F3]: third fetch for the three-byte codes

Timing sequence [1-byte instruction]

I-Cyclej

M1 M2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |

ALE

PSEN

P0 CODEj ADDR CODEj+1 ADDR CODEj+1 ADDR CODEj+1 ADDR Cj+1

P2 ADDRj ADDRj+1 ADDRj+1 ADDRj+1 ADDRJ+1

F1 X X X F1

Timing sequence [2-byte instruction]

I-Cyclej

M1 M2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |

ALE

PSEN

P0 CODEj ADDR CODEj+1 ADDR CODEj+1 ADDR CODEj+1 ADDR Cj+2

P2 ADDRj ADDRj+1 ADDRj+1 ADDRj+1 ADDRJ+2

F1 F2 X X F1

Timing sequence [3-byte instruction]

I-Cyclej

M1 M2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |

ALE

PSEN

P0 CODEj ADDR CODEj+1 ADDR CODEj2 ADDR CODEj+2 ADDR Cj+3

P2 ADDRj ADDRj+1 ADDRj+2 ADDRj+2 ADDRJ+3

F1 F2 F3 X F1

[I.3] two-machine-cycle instruction with external **data-write**

Timing sequence [1-byte instruction]

I-Cyclej

M1 M2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |

ALE

PSEN

WR

P0 CODEj ADDR CODEj+1 ADDR data from 51CPU ADDR Cj+1

P2 ADDRj ADDRj+1 ADDRDW ADDRJ+1

F1 X X X F1

[I.3] two-machine-cycle instruction with external **data-read**

Timing sequence [1-byte instruction]

I-Cyclej

M1 M2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |

ALE

PSEN

RD

P0 CODEj ADDR CODEj+1 ADDR data from RAM ADDR Cj+1

P2 ADDRj ADDRj+1 ADDRDR ADDRJ+1

F1 X X X F1