

**Features**

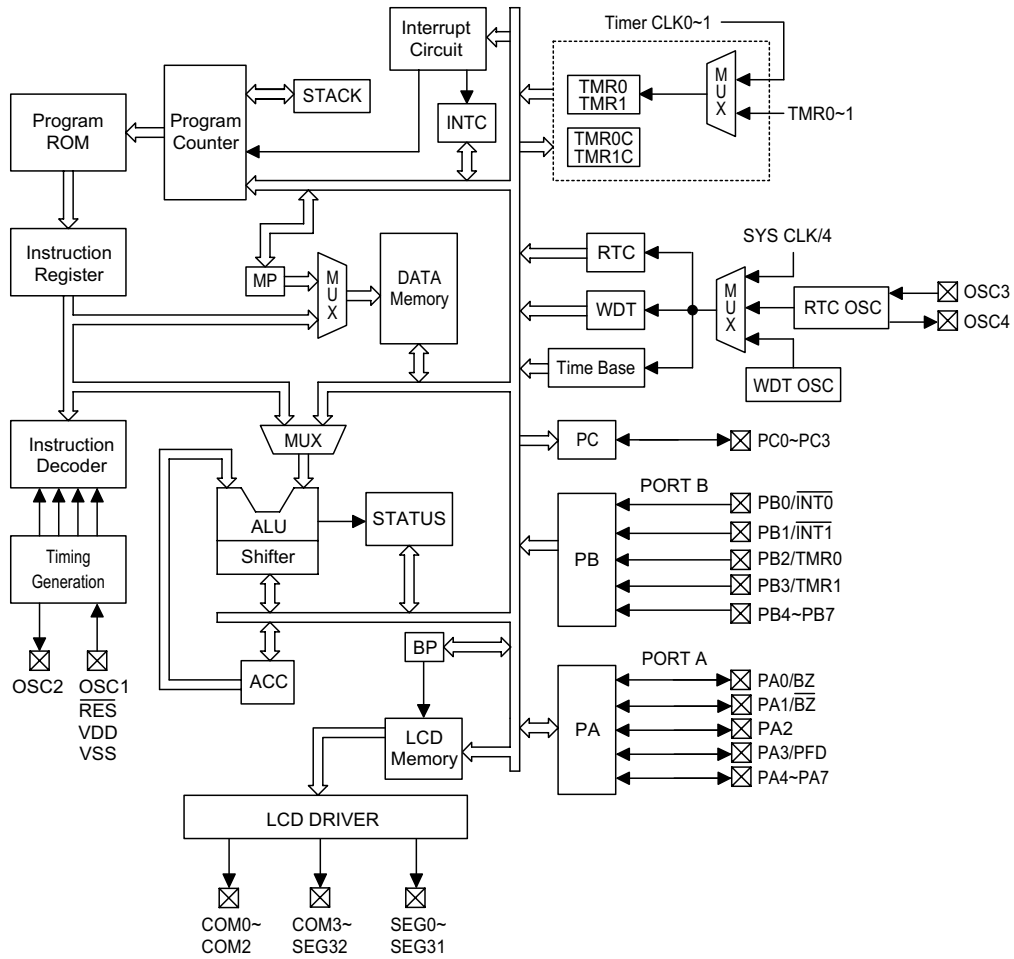
- Operating voltage: 2.2V~5.2V
- 8 input lines
- 12 bidirectional I/O lines
- Two external interrupt input
- Two 8-bit programmable timer/event counter with PFD (programmable frequency divider) function
- LCD driver with  $33 \times 3$  or  $32 \times 4$  segments
- $4K \times 15$  program memory ROM
- $160 \times 8$  data memory RAM
- Real Time Clock (RTC)
- 8-bit prescaler for RTC
- Watchdog timer
- Buzzer output
- On-chip crystal and RC oscillator
- Halt function and wake-up feature reduce power consumption
- 6-level subroutine nesting
- Bit manipulation instruction
- 15-bit table read instruction
- Up to  $1\mu s$  instruction cycle with 4MHz system clock
- 63 powerful instructions
- All instructions in 1 or 2 machine cycles
- 80/100-pin QFP package

**General Description**

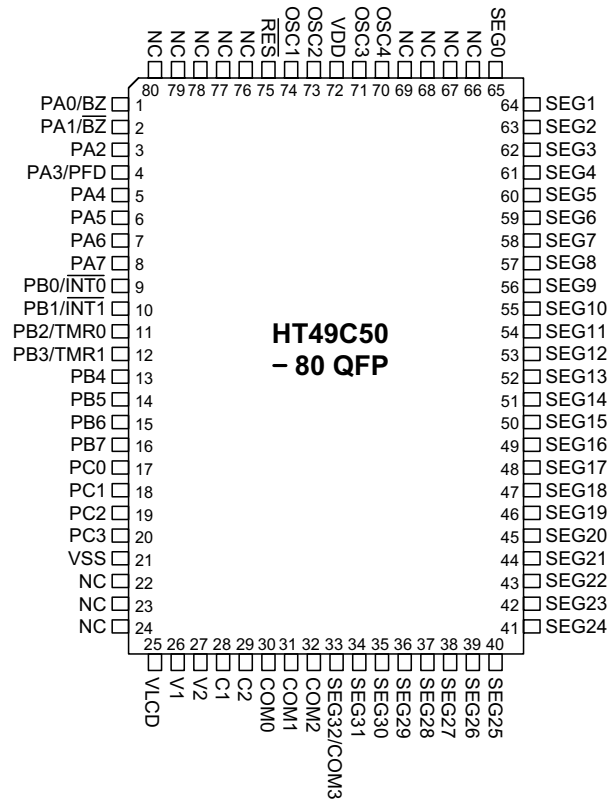
The HT49C50 is an 8-bit high performance single chip microcontroller. Its single cycle instruction and two-stage pipeline architecture make it suitable for high speed applications. The device is suited for use in multiple LCD low power applica-

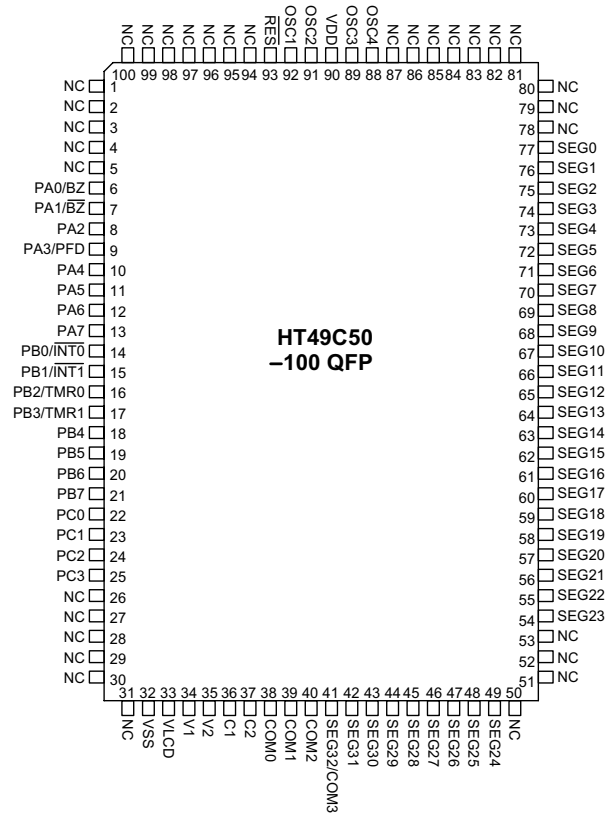
tions among which are calculators, clock timers, games, scales, leisure products, other hand held LCD products, and battery system in particular.

**Block Diagram**

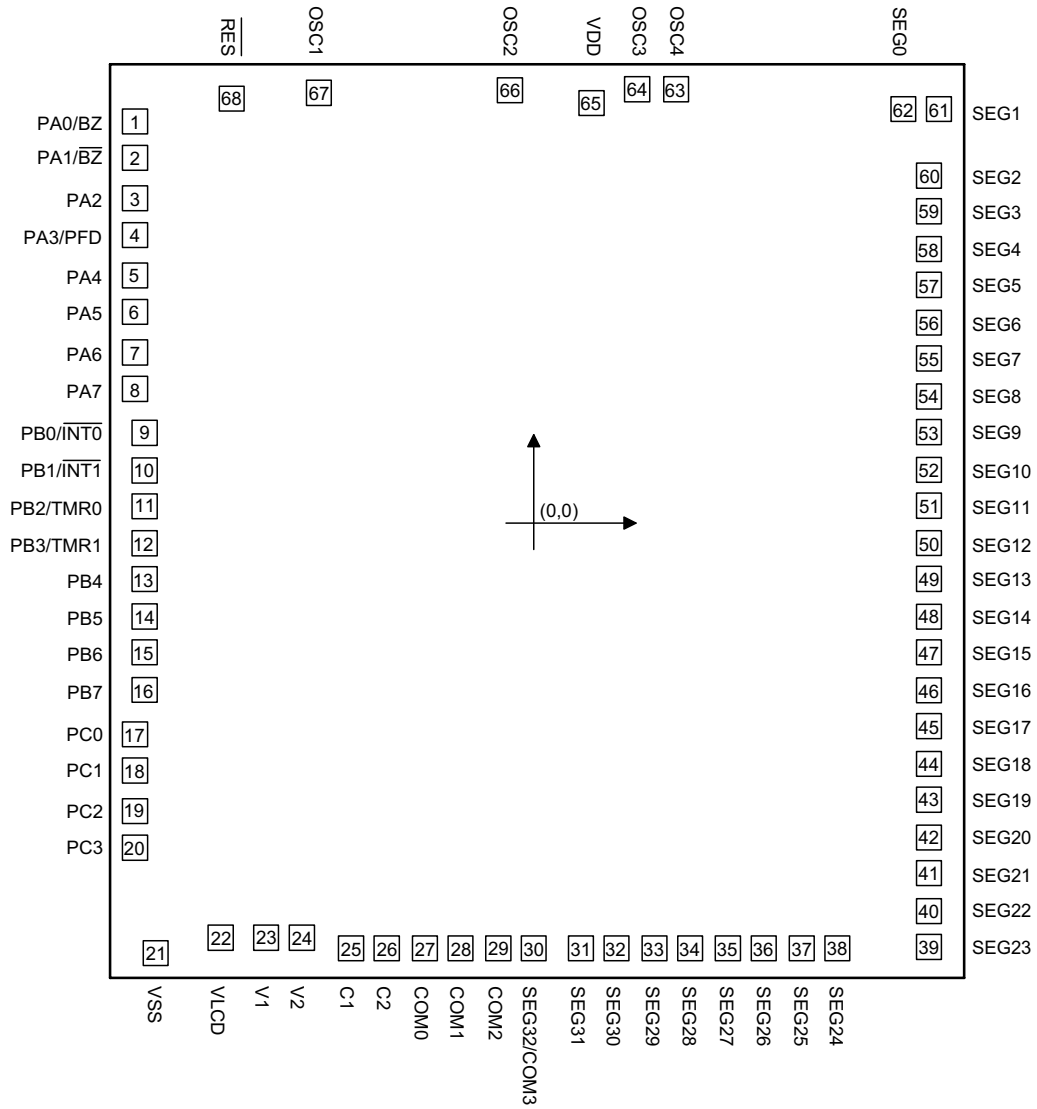


**Pin Assignment**





Pad Assignment



\* The IC substrate should be connected to VSS in the PCB layout artwork.

**Pin Description**

| Pin Name  | I/O    | Mask Option  | Description  |
|---|--------|--|--|
| PA0/BZ<br>PA1/BZ<br>PA2<br>PA3/PFD<br>PA4~PA7   | I/O    | Wake-up<br>Pull-high<br>or None<br>CMOS or<br>NMOS | PA0~PA7 constitute an 8-bit bidirectional input/output port with Schmitt trigger input capability. Each bit on port can be configured as a wake-up input by mask option. PA0~PA3 can be configured as a CMOS output or NMOS input/output with or without pull-high resistor by mask option. PA4~PA7 are always pull-high NMOS input/output. Of the eight bits, PA0~PA1 can be set as I/O pins or buzzer outputs by mask option. PA3 can be set as an I/O pin or as a PFD output also by mask option. |
| PB0/ $\overline{\text{INT0}}$<br>PB1/ $\overline{\text{INT1}}$<br>PB2/TMR0<br>PB3/TMR1<br>PB4~PB7 | I      | —  | PB0~PB7 constitute an 8-bit Schmitt trigger input port. Each bit on port are pull-high resistor. Of the eight bits, PB0 and PB1 can be set as input pins or as external interrupt control pins ( $\overline{\text{INT0}}$ ) and ( $\overline{\text{INT1}}$ ) respectively, by software application. PB2 and PB3 can be set as an input pin or as a timer/event counter input pin TMR0 and TMR1 also by software application.   |
| PC0~PC3   | I/O    | Pull-high<br>or None<br>CMOS or<br>NMOS            | PC0~PC3 constitute a 4-bit bidirectional input/output port with a schmitt trigger input capability. On the port, such can be configured as CMOS output or NMOS input/output with or without pull-high resistor by mask option.   |
| VSS   | I      | —  | Negative power supply, GND   |
| VLCD  | I      | —  | LCD power supply   |
| V1,V2,C1,C2   | I      | —  | Voltage pump   |
| SEG32/COM3<br>COM2~COM0   | O      | 1/3 or 1/4<br>Duty                                 | SEG32 can be set as a segment or as a common output driver for LCD panel by mask option. COM2~COM0 are outputs for LCD panel plate.  |
| SEG31~SEG0  | O      | —  | LCD driver outputs for LCD panel segments  |
| OSC4<br>OSC3  | O<br>I | —  | Real time clock oscillators  |
| VDD   | —      | —  | Positive power supply  |
| OSC2<br>OSC1  | O<br>I | Crystal or<br>RC                                   | OSC1 and OSC2 are connected to an RC network or a crystal (by mask option) for the internal system clock. In the case of RC operation, OSC2 is the output terminal for 1/4 system clock.   |
| $\overline{\text{RES}}$   | I      | —  | Schmitt trigger reset input, active low  |

### Absolute Maximum Ratings

Supply Voltage..... $V_{SS}$ -0.3V to 5.5V      Storage Temperature.....-50°C to 125°C  
 Input Voltage..... $V_{SS}$ -0.3V to  $V_{DD}$ +0.3V      Operating Temperature .....-25°C to 70°C

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

### D.C. Characteristics

 $T_a=25^\circ\text{C}$ 

| Symbol     | Parameter   | Test Conditions |  | Min. | Typ. | Max.    | Unit          |
|------------|---|-----------------|--|------|------|---------|---------------|
|            |   | $V_{DD}$        | Conditions   |      |      |         |               |
| $V_{DD}$   | Operating Voltage   | —               | —  | 2.2  | —    | 5.2     | V             |
| $I_{DD1}$  | Operating Current<br>(Crystal OSC)  | 3V              | No load,<br>$f_{SYS}=4\text{MHz}$  | —    | 1    | 2       | mA            |
|            |   | 5V              |  | —    | 2.5  | 5       | mA            |
| $I_{DD2}$  | Operating Current (RC OSC)  | 3V              | No load,<br>$f_{SYS}=2\text{MHz}$  | —    | 0.75 | 1.5     | mA            |
|            |   | 5V              |  | —    | 1.5  | 3       | mA            |
| $I_{STB1}$ | Standby Current<br>(RTC Enable, LCD ON)   | 3V              | No load,<br>system halt  | —    | —    | 5       | $\mu\text{A}$ |
|            |   | 5V              |  | —    | —    | 10      | $\mu\text{A}$ |
| $I_{STB2}$ | Standby Current<br>(RTC Disable, LCD OFF)   | 3V              | No load,<br>system halt  | —    | —    | 1       | $\mu\text{A}$ |
|            |   | 5V              |  | —    | —    | 2       | $\mu\text{A}$ |
| $V_{IL}$   | I/O Port Input Low Voltage  | 3V              | —  | 0    | —    | 0.9     | V             |
|            |   | 5V              | —  | 0    | —    | 1.5     | V             |
| $V_{IH}$   | I/O Port Input High Voltage   | 3V              | —  | 2.1  | —    | 3       | V             |
|            |   | 5V              | —  | 3.5  | —    | 5       | V             |
| $V_{IL1}$  | Input Low Voltage<br>( $\overline{\text{RES}}$ , $\overline{\text{INT0}}$ , $\overline{\text{INT1}}$ , TMR0, TMR1)  | 3V              | $\overline{\text{RES}}=0.5V_{DD}$<br>$\overline{\text{INT0}}/1=0.3V_{DD}$<br>$\text{TMR0}/1=0.3V_{DD}$ | 0    | —    | 1.5/0.9 | V             |
|            |   | 5V              |  | 0    | —    | 2.5/1.5 | V             |
| $V_{IH1}$  | Input High Voltage<br>( $\overline{\text{RES}}$ , $\overline{\text{INT0}}$ , $\overline{\text{INT1}}$ , TMR0, TMR1) | 3V              | 0.8 $V_{DD}$   | 2.4  | —    | 3       | V             |
|            |   | 5V              |  | 4.0  | —    | 5       | V             |
| $I_{OL}$   | I/O Ports Sink Current  | 3V              | $V_{DD}=3\text{V}$ ,<br>$V_{OL}=0.3\text{V}$   | 1.5  | 2.5  | —       | mA            |
|            |   | 5V              | $V_{DD}=5\text{V}$ ,<br>$V_{OL}=0.5\text{V}$   | 4    | 6    | —       | mA            |

| Symbol          | Parameter   | Test Conditions |   | Min. | Typ. | Max. | Unit |
|-----------------|---|-----------------|---|------|------|------|------|
|                 |   | V <sub>DD</sub> | Conditions                                    |      |      |      |      |
| I <sub>OH</sub> | I/O Ports Source Current                            | 3V              | V <sub>DD</sub> =3V,<br>V <sub>OH</sub> =2.7V | -1   | -1.5 | —    | mA   |
|                 |   | 5V              | V <sub>DD</sub> =5V,<br>V <sub>OH</sub> =4.5V | -2   | -3   | —    | mA   |
| R <sub>PH</sub> | Pull-high Resistance of<br>I/O Ports and INT0, INT1 | 3V              | —   | 40   | 60   | 80   | kΩ   |
|                 |   | 5V              | —   | 10   | 30   | 50   | kΩ   |

**A.C. Characteristics**

Ta=25°C

| Symbol              | Parameter                          | Test Conditions |                                  | Min. | Typ. | Max. | Unit             |
|---------------------|------------------------------------|-----------------|----------------------------------|------|------|------|------------------|
|                     |                                    | V <sub>DD</sub> | Conditions                       |      |      |      |                  |
| f <sub>SYS1</sub>   | System Clock (Crystal OSC)         | 3V              | V <sub>DD</sub> =3V              | 455  | —    | 4000 | kHz              |
|                     |                                    | 5V              | V <sub>DD</sub> =5V              | 455  | —    | 4000 | kHz              |
| f <sub>SYS2</sub>   | System Clock (RC OSC)              | 3V              | V <sub>DD</sub> =3V              | 400  | —    | 2000 | kHz              |
|                     |                                    | 5V              | V <sub>DD</sub> =5V              | 400  | —    | 3000 | kHz              |
| f <sub>TIMER</sub>  | Timer I/P Frequency<br>(TMR0/TMR1) | 3V              | V <sub>DD</sub> =3V              | 0    | —    | 4000 | kHz              |
|                     |                                    | 5V              | V <sub>DD</sub> =5V              | 0    | —    | 4000 | kHz              |
| t <sub>WDTOSC</sub> | Watchdog Oscillator                | 3V              | V <sub>DD</sub> =3V              | 45   | 90   | 180  | μs               |
|                     |                                    | 5V              | V <sub>DD</sub> =5V              | 35   | 65   | 130  | μs               |
| t <sub>RES</sub>    | External Reset Low<br>Pulse Width  | —               | —                                | 1    | —    | —    | μs               |
| t <sub>SST</sub>    | System Start-up<br>Timer Period    | —               | Power-up or<br>wake-up from halt | —    | 1024 | —    | t <sub>SYS</sub> |
| t <sub>INT</sub>    | Interrupt Pulse Width              | —               | —                                | 1    | —    | —    | μs               |

 Note: t<sub>SYS</sub>= 1/f<sub>SYS</sub>



## Functional Description

### Execution flow

The system clock is derived from either a crystal or an RC oscillator. It is internally divided into four non-overlapping clocks. One instruction cycle consists of four system clock cycles.

Instruction fetching and execution are pipelined in such a way that a fetch takes one instruction cycle while decoding and execution takes the next instruction cycle. The pipelining scheme causes each instruction to effectively execute in a cycle. If an instruction changes the value of the program counter, two cycles are required to complete the instruction.

### Program counter – PC

The program counter (PC) is of 12 bits wide and controls the sequence in which the instructions stored in the program ROM are executed. The contents of the PC can specify a maximum of 4096 addresses.

After accessing a program memory word to fetch an instruction code, the value of the PC is incremented by one. The PC then points to the memory word containing the next instruction code.

When executing a jump instruction, conditional skip execution, loading a PCL register, a subroutine call, an initial reset, an internal interrupt, an external interrupt, or returning from a subroutine, the PC manipulates the program

transfer by loading the address corresponding to each instruction.

The conditional skip is activated by instructions. Once the condition is met, the next instruction, fetched during the current instruction execution, is discarded and a dummy cycle replaces it to get a proper instruction; otherwise proceed with the next instruction.

The lower byte of the PC (PCL) is a readable and writeable register (06H). Moving data into the PCL performs a short jump. The destination is within 256 locations.

When a control transfer takes place, an additional dummy cycle is required.

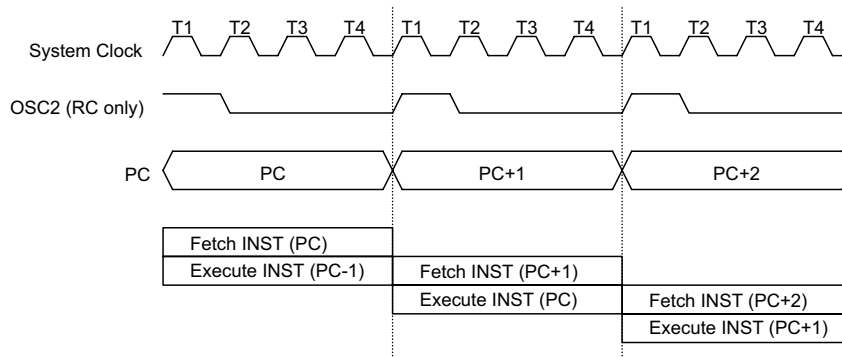
### Program memory – ROM

The program memory (ROM) is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into  $4096 \times 15$  bits which are addressed by the PC and table pointer.

Certain locations in the ROM are reserved for special usage:

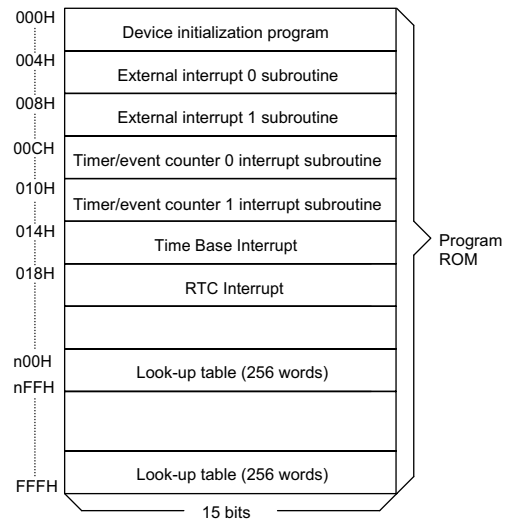
- Location 000H

Location 000H is reserved for program initialization. After chip reset, the program always begins execution at this location.



Execution flow

- **Location 004H**  
Location 004H is reserved for the external interrupt service program. If the INT0 input pin is activated, and the interrupt is enabled, and the stack is not full, the program begins execution at location 004H.
- **Location 008H**  
Location 008H is reserved for the external interrupt service program also. If the INT1 input pin is activated, and the interrupt is enabled, and the stack is not full, the program begins execution at location 008H.
- **Location 00CH**  
Location 00CH is reserved for the timer/event counter 0 interrupt service program. If a timer interrupt results from a timer/event counter 0 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.
- **Location 010H**  
Location 010H is reserved for the timer/event counter 1 interrupt service program. If a timer interrupt results from a timer/event



Program memory

counter 1 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 010H.

| Mode                           | Program Counter |     |    |    |    |    |    |    |    |    |    |    |
|--------------------------------|-----------------|-----|----|----|----|----|----|----|----|----|----|----|
|                                | *11             | *10 | *9 | *8 | *7 | *6 | *5 | *4 | *3 | *2 | *1 | *0 |
| Initial Reset                  | 0               | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| External Interrupt 0           | 0               | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  |
| External Interrupt 1           | 0               | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  |
| Timer/event counter 0 overflow | 0               | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 0  | 0  |
| Timer/event counter 1 overflow | 0               | 0   | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  |
| Time Base Interrupt            | 0               | 0   | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0  |
| RTC Interrupt                  | 0               | 0   | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 0  | 0  | 0  |
| Skip                           | <b>PC+2</b>     |     |    |    |    |    |    |    |    |    |    |    |
| Loading PCL                    | *11             | *10 | *9 | *8 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |
| Jump, Call Branch              | #11             | #10 | #9 | #8 | #7 | #6 | #5 | #4 | #3 | #2 | #1 | #0 |
| Return From Subroutine         | S11             | S10 | S9 | S8 | S7 | S6 | S5 | S4 | S3 | S2 | S1 | S0 |

Program counter

Notes: \*11~\*0: Program counter bits  
#11~#0: Instruction code bits

S11~S0: Stack register bits  
@7~@0: PCL bits

- Location 014H

Location 014H is reserved for the Time Base interrupt service program. If a Time Base interrupt occurs, and the interrupt is enabled, and the stack is not full, the program begins execution at location 014H.

- Location 018H

Location 018H is reserved for the real time clock interrupt service program. If a real time clock interrupt occurs, and the interrupt is enabled, and the stack is not full, the program begins execution at location 018H.

- Table location

Any location in the ROM can be used as a look-up table. The instructions "TABRDC [m]" (the current page, 1 page=256 words) and "TABRDL [m]" (the last page) transfer the contents of the lower-order byte to the specified data memory, and the contents of the higher-order byte to TBLH (Table Higher-order byte register) (08H). Only the destination of the lower-order byte in the table is well-defined; the other bits of the table word are all transferred to the lower portion of TBLH, and the remaining 1 bit is read as "0". The TBLH is read only, and the table pointer (TBLP) is a read/write register (07H), indicating the table location. Before accessing the table, the location should be placed in TBLP. All the table related instructions require 2 cycles to complete the operation. These areas may function as a normal ROM depending upon the user's requirements.

### Stack register – STACK

The stack register is a special part of the memory used to save the contents of the PC. The stack is organized into 6 levels and is neither part of the data nor part of the program, and is neither readable nor writeable. Its activated level is indexed by a stack pointer (SP) and is neither readable nor writeable. At a commencement of a subroutine call or an interrupt acknowledgment, the contents of the PC is pushed onto the stack. At the end of the subroutine or interrupt routine, signaled by a return instruction (RET or RETI), the contents of the PC is restored to its previous value from the stack. After chip reset, the SP will point to the top of the stack.

If the stack is full and a non-masked interrupt takes place, the interrupt request flag is recorded but the acknowledgment is still inhibited. Once the SP is decremented (by RET or RETI), the interrupt is serviced. This feature prevents stack overflow, allowing the programmer to use the structure easily. Likewise, if the stack is full, and a "CALL" is subsequently executed, a stack overflow occurs and the first entry is lost (only the most recent six return addresses are stored).

### Data memory – RAM

The data memory (RAM) is designed with 192×8 bits, and is divided into two functional groups, namely special function registers and general purpose data memory, most of which are readable/writeable, although some are read only.

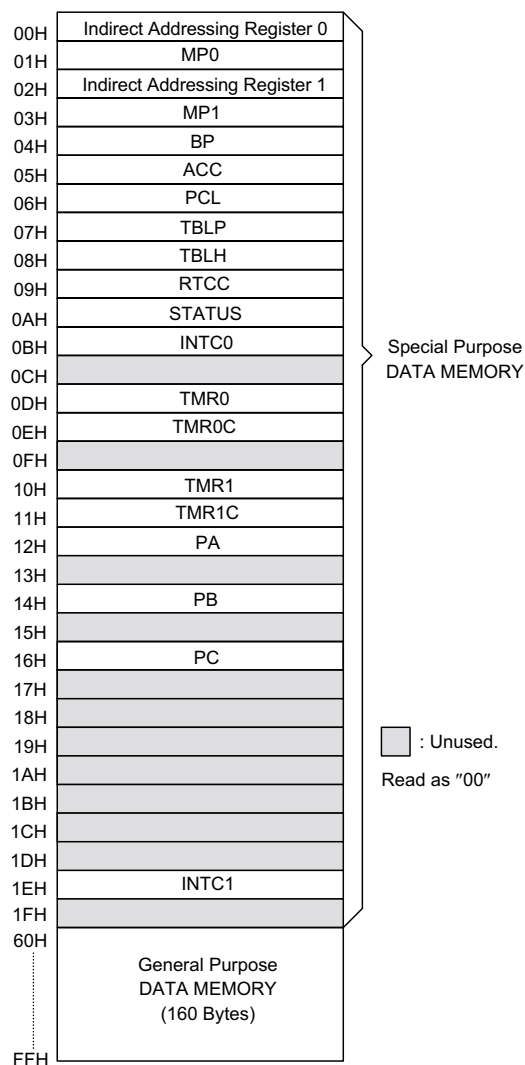
| Instruction(s) | Table Location |     |    |    |    |    |    |    |    |    |    |    |
|----------------|----------------|-----|----|----|----|----|----|----|----|----|----|----|
|                | *11            | *10 | *9 | *8 | *7 | *6 | *5 | *4 | *3 | *2 | *1 | *0 |
| TABRDC [m]     | P11            | P10 | P9 | P8 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |
| TABRDL [m]     | 1              | 1   | 1  | 1  | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |

Table location

Notes: \*11~\*0: Table location bits

P11~P8: Current program Counter bits

@7~@0: Table pointer bits



RAM mapping

Of the two types of functional groups, the special function registers consist of an Indirect addressing register 0 (00H), a Memory pointer register 0 (MP0;01H), an Indirect addressing register 1 (02H), a Memory pointer register 1 (MP1;03H), a Bank pointer (BP;04H), an Accumulator (ACC;05H), a Program counter lower-order byte register (PCL;06H), a Table pointer (TBLP;07H),

a Table higher-order byte register (TBLH;08H), a Real time clock control register (RTCC;09H), a Status register (STATUS;0AH), an Interrupt control register 0 (INTC0;0BH), a timer/event counter 0 (TMR0;0DH), a timer/event counter 0 control register (TMR0C;0EH), a timer/event counter 1 (TMR1;10H), a timer/event counter 1 control register (TMR1C;11H), I/O registers (PA;12H, PB;14H, PC;16H), and Interrupt control register 1 (INTC1;1EH). On the other hand, the general purpose data memory, addressed from 60H to FFH, is used for data and control information under instruction commands.

The areas in the RAM can directly handle arithmetic, logic, increment, decrement, and rotate operations. Except some dedicated bits, each bit in the RAM can be set and reset by "SET [m].i" and "CLR [m].i". They are also indirectly accessible through the Memory pointer register 0 (MP0;01H) or the Memory pointer register 1 (MP1;03H).

**Indirect addressing register**

Location 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation of [00H] and [02H] accesses the RAM pointed to by MP0 (01H) and MP1(03H) respectively. Reading location 00H or 02H indirectly returns the result 00H. While, writing it indirectly leads to no operation.

The function of data movement between two indirect addressing registers is not supported. The memory pointer registers, MP0 and MP1, are both 8-bit registers used to access the RAM by combining corresponding indirect addressing registers. MP0 can only be applied to data memory, while MP1 can be applied to data memory and LCD display memory.

**Accumulator – ACC**

The accumulator (ACC) is related to the ALU operations. It is also mapped to location 05H of the RAM and is capable of operating with immediate data. The data movement between two data memory locations must pass through the ACC.

### Arithmetic and logic unit – ALU

This circuit performs 8-bit arithmetic and logic operations and provides the following functions:

- Arithmetic operations (ADD, ADC, SUB, SBC, DAA)
- Logic operations (AND, OR, XOR, CPL)
- Rotation (RL, RR, RLC, RRC)
- Increment and Decrement (INC, DEC)
- Branch decision (SZ, SNZ, SIZ, SDZ etc.)

The ALU not only saves the results of a data operation but also changes the status register.

### Status register – STATUS

The status register (0AH) is of 8 bits wide and contains, a carry flag (C), an auxiliary carry flag (AC), a zero flag (Z), an overflow flag (OV), a power down flag (PD), and a watchdog time-out flag (TO). It also records the status information and controls the operation sequence.

Except the TO and PD flags, bits in the status register can be altered by instructions similar to other registers. Data written into the status register does not alter the TO or PD flags. Oper-

ations related to the status register, however, may yield different results from those intended. The TO and PD flags can only be changed by a watchdog timer overflow, chip power-up, or clearing the watchdog timer and executing the "HALT" instruction. The Z, OV, AC, and C flags reflect the status of the latest operations.

On entering the interrupt sequence or executing the subroutine call, the status register will not be automatically pushed onto the stack. If the contents of the status is important, and if the subroutine is likely to corrupt the status register, the programmer should take precautions and save it properly.

### Interrupts

The HT49C50 provides two external interrupts, two internal timer/event counter interrupts, an internal time base interrupt, and an internal real time clock interrupt. The interrupt control register 0 (INTC0;0BH) and interrupt control register 1 (INTC1;1EH) both contain the interrupt control bits that are used to set the enable/disable status and interrupt request flags.

| Labels | Bits | Function   |
|--------|------|--|
| C      | 0    | C is set if the operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction. |
| AC     | 1    | AC is set if the operation results in a carry out of the low nibbles in addition or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.  |
| Z      | 2    | Z is set if the result of an arithmetic or logic operation is zero; otherwise Z is cleared.  |
| OV     | 3    | OV is set if the operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.   |
| PD     | 4    | PD is cleared by either a system power-up or executing the "CLR WDT" instruction. PD is set by executing the "HALT" instruction.   |
| TO     | 5    | TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.  |
| —      | 6    | Undefined, read as "0"   |
| —      | 7    | Undefined, read as "0"   |

Status register

Once an interrupt subroutine is serviced, other interrupts are all blocked (by clearing the EMI bit). This scheme may prevent any further interrupt nesting. Other interrupt requests may take place during this interval, but only the interrupt request flag will be recorded. If a certain interrupt requires servicing within the service routine, the EMI bit and the corresponding bit of the INTC0 or of INTC1 may be set in order to allow interrupt nesting. Once the stack is

full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the SP is decremented. If immediate service is desired, the stack should be prevented from becoming full.

All these interrupts can support a wake-up function. As an interrupt is serviced, a control transfer occurs by pushing the contents of the PC onto the stack followed by a branch to a sub-

| Register       | Bit No. | Label | Function   |
|----------------|---------|-------|--|
| INTC0<br>(0BH) | 0       | EMI   | Control the master (global) interrupt<br>(1=enabled; 0=disabled)       |
|                | 1       | EEI0  | Control the external interrupt 0<br>(1=enabled; 0=disabled)            |
|                | 2       | EEI1  | Control the external interrupt 1<br>(1=enabled; 0=disabled)            |
|                | 3       | ET0I  | Control the timer/event counter 0 interrupt<br>(1=enabled; 0=disabled) |
|                | 4       | EIF0  | External interrupt 0 request flag<br>(1=active; 0=inactive)            |
|                | 5       | EIF1  | External interrupt 1 request flag<br>(1=active; 0=inactive)            |
|                | 6       | T0F   | Internal timer/event counter 0 request flag<br>(1=active; 0=inactive)  |
|                | 7       | —     | Unused bit, read as "0"  |
| INTC1<br>(1EH) | 0       | ET1I  | Control the timer/event counter 1 interrupt<br>(1=enabled; 0=disabled) |
|                | 1       | ETBI  | Control the time base interrupt<br>(1=enabled; 0=disabled)             |
|                | 2       | ERTI  | Control the real time clock interrupt<br>(1=enabled; 0=disabled)       |
|                | 3       | —     | Unused bit, read as "0"  |
|                | 4       | T1F   | Internal timer/event counter 1 request flag<br>(1=active; 0=inactive)  |
|                | 5       | TBF   | Time base request flag<br>(1=active; 0=inactive)                       |
|                | 6       | RTF   | Real time clock request flag<br>(1=active; 0=inactive)                 |
|                | 7       | —     | Unused bit, read as "0"  |

INTC register

routine at the specified location in the ROM. Only the contents of the PC is pushed onto the stack. If the contents of the register or of the status register (STATUS) is altered by the interrupt service program which corrupts the desired control sequence, the contents should be saved in advance.

External interrupts are triggered by a high to low transition of INT0 or INT1, and the related interrupt request flag (EIF0; bit 4 of INTC0, EIF1; bit 5 of INTC0) is set as well. After the interrupt is enabled, the stack is not full, and the external interrupt is active, a subroutine call to location 04H or 08H occurs. The interrupt request flag (EIF0 or EIF1) and EMI bits are all cleared to disable other interrupts.

The internal timer/event counter 0 interrupt is initialized by setting the timer/event counter 0 interrupt request flag (T0F; bit 6 of INTC0), which is normally caused by a timer overflow. After the interrupt is enabled, and the stack is not full, and the T0F bit is set, a subroutine call to location 0CH occurs. The related interrupt request flag (T0F) is reset, and the EMI bit is cleared to disable further interrupts. The timer/event counter 1 is operated in the same manner but its related interrupt request flag is T1F (bit 4 of INTC1) and its subroutine call location is 10H.

The time base interrupt is initialized by setting the time base interrupt request flag (TBF; bit 5 of INTC1), that is caused by a regular time base signal. After the interrupt is enabled, and the stack is not full, and the TBF bit is set, a subroutine call to location 14H occurs. The related interrupt request flag (TBF) is reset and the EMI bit is cleared to disable further interrupts.

The real time clock interrupt is initialized by setting the real time clock interrupt request flag (RTF; bit 6 of INTC1), that is caused by a regular real time clock signal. After the interrupt is enabled, and the stack is not full, and the RTF bit is set, a subroutine call to location 18H occurs. The related interrupt request flag (RTF) is reset and the EMI bit is cleared to disable further interrupts.

During the execution of an interrupt subroutine, other interrupt acknowledgments are all held

until the "RETI" instruction is executed or the EMI bit and the related interrupt control bit are set both to 1 (if the stack is not full). To return from the interrupt subroutine, "RET" or "RETI" may be invoked. RETI sets the EMI bit and enables an interrupt service, but RET does not.

Interrupts occurring in the interval between the rising edges of two consecutive T2 pulses are serviced on the latter of the two T2 pulses if the corresponding interrupts are enabled. In the case of simultaneous requests, the priorities in the following table apply. These can be masked by resetting the EMI bit.

| No. | Interrupt Source               | Priority | Vector |
|-----|--------------------------------|----------|--------|
| a   | External interrupt 0           | 1        | 04H    |
| b   | External interrupt 1           | 2        | 08H    |
| c   | Timer/event counter 0 overflow | 3        | 0CH    |
| d   | Timer/event counter 1 overflow | 4        | 10H    |
| e   | Time base interrupt            | 5        | 14H    |
| f   | Real time clock interrupt      | 6        | 18H    |

The timer/event counter 0 interrupt request flag (T0F), external interrupt 1 request flag (EIF1), external interrupt 0 request flag (EIF0), enable timer/event counter 0 interrupt bit (ET0I), enable external interrupt 1 bit (EEI1), enable external interrupt 0 bit (EEI0), and enable master interrupt bit (EMI) make up of the Interrupt Control register 0 (INTC0) which is located at 0BH in the RAM. The real time clock interrupt request flag (RTF), time base interrupt request flag (TBF), timer/event counter 1 interrupt request flag (T1F), enable real time clock interrupt bit (ERTI), and enable time base interrupt bit (ETBI), enable timer/event counter 1 interrupt bit (ET1I) on the other hand, constitute the Interrupt Control register 1 (INTC1) which is located at 1EH in the RAM. EMI, EEI0, EEI1, ET0I, ET1I, ETBI, and ERTI are all used to control the enable/disable status of interrupts. These bits prevent the requested interrupt from being ser-



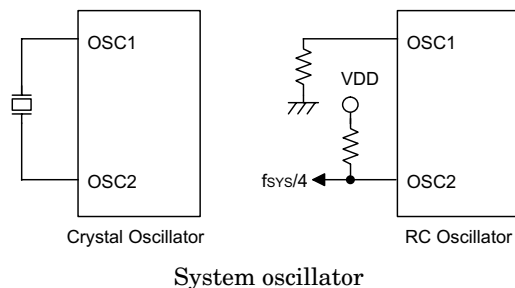
vised. Once the interrupt request flags (RTF, TBF, T0F, T1F, EIF1, EIF0) are all set, they remain in the INTC1 or INTC0 respectively until the interrupts are serviced or cleared by a software instruction.

It is recommended that a program not use the "CALL subroutine" within the interrupt subroutine. It's because interrupts often occur in an unpredictable manner or require to be serviced immediately in some applications. At this time, if only one stack is left, and enabling the interrupt is not well controlled, operation of the "call" in the interrupt subroutine may damage the original control sequence.

**Oscillator configuration**

The HT49C50 provides two oscillator circuits for system clocks, i.e., RC oscillator and crystal oscillator, determined by mask option. No matter what type of oscillator is selected, the signal is used for the system clock. The HALT mode stops the system oscillator and ignores external signal to conserve power.

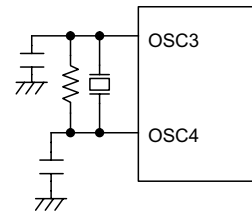
Of the two oscillators, if the RC oscillator is used, an external resistor between OSC1 and VSS is required, and the range of the resistance should be from 51kΩ to 1MΩ. The system clock, divided by 4, is available on OSC2 with pull-high resistor, which can be used to synchronize external logic. The RC oscillator provides the most cost effective solution. However, the frequency of the oscillation may vary with VDD, temperature, and the chip itself due to process variations. It is therefore, not suitable for timing sensitive operations where accurate oscillator frequency is desired.



On the other hand, if the crystal oscillator is selected, a crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for the oscillator, and no other external components are required. A resonator may be connected between OSC1 and OSC2 to replace the crystal and to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required.

There is another oscillator circuit designed for the real time clock. In this case, only the 32.768kHz crystal oscillator can be applied. The crystal should be connected between OSC3 and OSC4, and two external capacitors along with one external resistor are required for the oscillator circuit in order to get a stable frequency.

The RTC oscillator circuit can be controlled to oscillate quickly by setting the "QOSC" bit (bit 4 of RTCC). It is recommended to turn on the quick oscillating function upon power on, and turn it off after 2 seconds.



RTC oscillator

The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Although the system enters the power down mode, the system clock stops, and the WDT oscillator still works with a period of approximately 78μs. The WDT oscillator can be disabled by mask option to conserve power.



**Watchdog timer – WDT**

The WDT clock source is implemented by a dedicated RC oscillator (WDT oscillator) or an instruction clock (system clock/4) or a real time clock oscillator (RTC oscillator). The timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The WDT can be disabled by mask option. But if the WDT is disabled, all executions related to the WDT lead to no operation.

After the WDT clock source is selected, the time-out period is  $f_s/2^{15} \sim f_s/2^{16}$ .

If the WDT clock source chooses the internal WDT oscillator, the time-out period may vary with temperature, VDD, and process variations. On the other hand, if the clock source selects the instruction clock and the "halt" instruction is executed, WDT may stop counting and lose its protecting purpose, and the logic can only be restarted by an external logic.

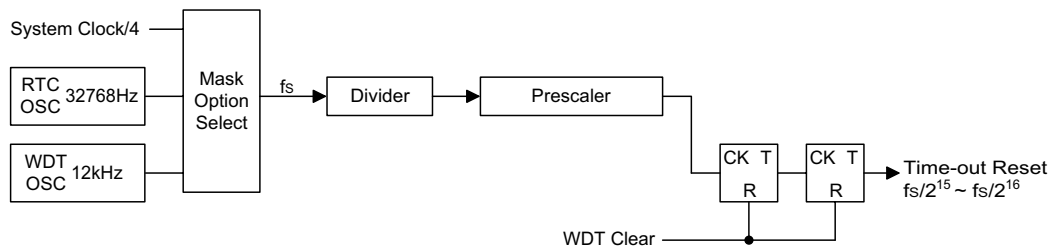
When the device operates in a noisy environment, using the on-chip RC oscillator (WDT OSC) is strongly recommended, since the HALT can stop the system clock.

The WDT overflow under normal operation initializes a "chip reset" and sets the status bit "TO". In the HALT mode, the overflow initializes a "warm reset", and only the PC and SP are reset to zero. To clear the contents of the WDT, there are three methods to be adopted,

i.e., external reset (a low level to  $\overline{RES}$ ), software instruction, and a "HALT" instruction. There are two types of software instructions; "CLR WDT" and the other set – "CLR WDT1" and "CLR WDT2". Of these two types of instruction, only one type of instruction can be active at a time depending on the mask option – "CLR WDT times selection option". If the "CLR WDT" is selected (i.e., CLR WDT times equal one), any execution of the "CLR WDT" instruction clears the WDT. In the case that "CLR WDT1" and "CLR WDT2" are chosen (i.e., CLR WDT times equal two), these two instructions have to be executed to clear the WDT; otherwise, the WDT may reset the chip due to time-out.

**Multi-function timer**

The HT49C50 provides a multi-function timer for the WDT, time base and RTC but with different time-out periods. The multi-function timer consists of a 7-stage divider and an 8-bit prescaler, with the clock source coming from the WDT OSC or RTC OSC or the instruction clock (i.e., system clock divided by 4). The multi-function timer also provides a selectable frequency signal (ranges from  $f_s/2^2$  to  $f_s/2^8$ ) for LCD driver circuits, and a selectable frequency signal (ranges from  $f_s/2^2$  to  $f_s/2^9$ ) for the buzzer output by mask option. It is recommended to select a near 4kHz signal to LCD driver circuits for proper display.



**Time base**

The time base offers a periodic time-out period to generate a regular internal interrupt. Its time-out period ranges from  $f_s/2^{12}$  to  $f_s/2^{15}$  selected by mask option. If time base time-out occurs, the related interrupt request flag (TBF; bit 5 of INTC1) is set. But if the interrupt is enabled, and the stack is not full, a subroutine call to location 14H occurs. The time base time-out signal also can be applied to be a clock source of timer/event counter 1 for getting a longer timer-out period.

| RT2 | RT1 | RT0 | RTC Clock Divided Factor |
|-----|-----|-----|--------------------------|
| 0   | 0   | 0   | $2^8$                    |
| 0   | 0   | 1   | $2^9$                    |
| 0   | 1   | 0   | $2^{10}$                 |
| 0   | 1   | 1   | $2^{11}$                 |
| 1   | 0   | 0   | $2^{12}$                 |
| 1   | 0   | 1   | $2^{13}$                 |
| 1   | 1   | 0   | $2^{14}$                 |
| 1   | 1   | 1   | $2^{15}$                 |

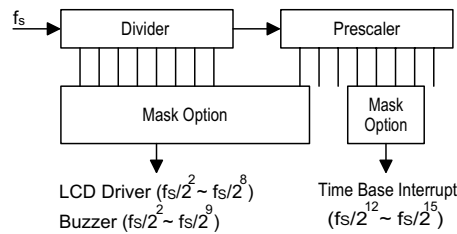
**Real time clock – RTC**

The real time clock (RTC) is operated in the same manner as the time base that is used to supply a regular internal interrupt. Its time-out period ranges from  $f_s/2^8$  to  $f_s/2^{15}$  by software programming. Writing data to RT2, RT1 and RT0 (bit2, 1, 0 of RTCC;09H) yields various time-out periods. If the RTC time-out occurs, the related interrupt request flag (RTF; bit 6 of INTC1) is set. But if the interrupt is enabled, and the stack is not full, a subroutine call to location 18H occurs. The real time clock time-out signal also can be applied to be a clock source of timer/event counter 0 for getting a longer time-out period.

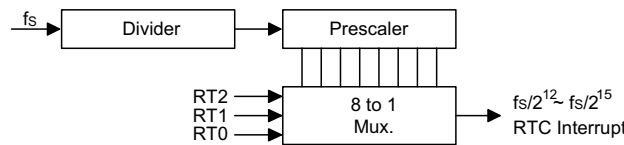
**Power down operation – HALT**

The HALT mode is initialized by the "HALT" instruction and results in the following.

- The system oscillator turns off but the WDT oscillator keeps running (if the WDT oscillator or the real time clock is selected).
- The contents of the on-chip RAM and of the registers remain unchanged.
- The WDT is cleared and start recounting (if the WDT clock source is from the WDT oscillator or the real time clock oscillator).



Time base



Real time clock

- All I/O ports maintain their original status.
- The PD flag is set but the TO flag is cleared.
- LCD driver is still running (if the WDT OSC or RTC OSC is selected).

The system quits the HALT mode by an external reset, an interrupt, an external falling edge signal on port A, or a WDT overflow. An external reset causes device initialization, and the WDT overflow performs a "warm reset". After examining the TO and PD flags, the reason for chip reset can be determined. The PD flag is cleared by system power-up or by executing the "CLR WDT" instruction, and is set by executing the "HALT" instruction. On the other hand, the TO flag is set if WDT time-out occurs, and causes a wake-up that only resets the PC (Program Counter) and SP, and leaves the others at their original state.

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake up the device by mask option. Awakening from an I/O port stimulus, the program resumes execution of the next instruction. On the other hand, awakening from an interrupt, two sequences may occur. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program resumes execution at the next instruction. But if the interrupt is enabled, and the stack is not full, the regular interrupt response takes place.

When an interrupt request flag is set before entering the "halt" status, the system cannot be awoken using that interrupt.

If wake-up events occur, it takes  $1024 t_{SYS}$  (system clock period) to resume normal operation. In other words, a dummy period is inserted after the wake-up. If the wake-up results from an interrupt acknowledgment, the actual interrupt subroutine execution is delayed by more than one cycle. However, if the Wake-up results in the next instruction execution, the execution will be performed immediately after the dummy period is finished.

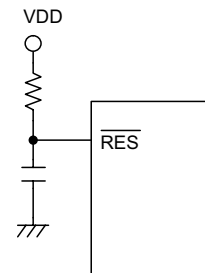
To minimize power consumption, all the I/O pins should be carefully managed before entering the HALT status.

**Reset**

There are three ways in which reset may occur.

- $\overline{RES}$  is reset during normal operation
- $\overline{RES}$  is reset during HALT
- WDT time-out is reset during normal operation

The WDT time-out during HALT differs from other chip reset conditions, for it can perform a "warm reset" that resets only the PC and SP and leaves the other circuits at their original state. Some registers remain unaffected during any other reset conditions. Most registers are reset to the "initial condition" once the reset conditions are met. Examining the PD and TO flags, the program can distinguish between different "chip resets".



Reset circuit

| TO | PD | RESET Conditions                               |
|----|----|--|
| 0  | 0  | $\overline{RES}$ reset during power-up         |
| u  | u  | $\overline{RES}$ reset during normal operation |
| 0  | 1  | $\overline{RES}$ Wake-up HALT                  |
| 1  | u  | WDT time-out during normal operation           |
| 1  | 1  | WDT Wake-up HALT                               |

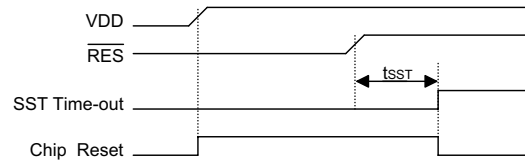
Note: "u" means "unchanged"

To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system awakes from the HALT state or during power up. Awaking from the HALT state or system power-up, the SST delay is added.

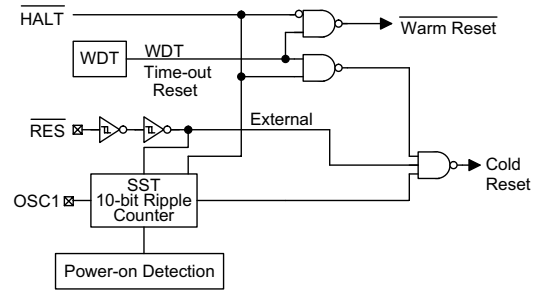
An extra SST delay is added during the power-up period, and any wake-up from the HALT may enable only the SST delay.

The functional unit chip reset status is shown below.

|                     |  |
|---------------------|--|
| PC                  | 000H   |
| Interrupt           | Disabled   |
| Prescaler, Divider  | Cleared  |
| WDT, RTC, Time base | Cleared. After master reset, WDT starts counting |
| Timer/event counter | Off  |
| Input/output ports  | Input mode                                       |
| SP                  | Points to the top of the stack                   |



Reset timing chart



Reset configuration

The states of the registers are summarized below:

| Register        | Reset<br>(Power On) | WDT<br>Time-out<br>(Normal<br>Operation) | $\overline{\text{RES}}$ Reset<br>(Normal<br>Operation) | $\overline{\text{RES}}$ Reset<br>(HALT) | WDT<br>Time-out<br>(HALT)* |
|-----------------|---------------------|--|--|---|----------------------------|
| TMR0            | xxxx xxxx           | uuuu uuuu                                | uuuu uuuu  | uuuu uuuu                               | uuuu uuuu                  |
| TMR0C           | 0000 1---           | 0000 1---                                | 0000 1---  | 0000 1---                               | uuuu u---                  |
| TMR1            | xxxx xxxx           | uuuu uuuu                                | uuuu uuuu  | uuuu uuuu                               | uuuu uuuu                  |
| TMR1C           | 0000 1---           | 0000 1---                                | 0000 1---  | 0000 1---                               | uuuu u---                  |
| Program Counter | 000H                | 000H                                     | 000H   | 000H                                    | 000H                       |
| MP0             | xxxx xxxx           | uuuu uuuu                                | uuuu uuuu  | uuuu uuuu                               | uuuu uuuu                  |
| MP1             | xxxx xxxx           | uuuu uuuu                                | uuuu uuuu  | uuuu uuuu                               | uuuu uuuu                  |
| ACC             | xxxx xxxx           | uuuu uuuu                                | uuuu uuuu  | uuuu uuuu                               | uuuu uuuu                  |
| TBLP            | xxxx xxxx           | uuuu uuuu                                | uuuu uuuu  | uuuu uuuu                               | uuuu uuuu                  |
| TBLH            | -xxx xxxx           | -uuu uuuu                                | -uuu uuuu  | -uuu uuuu                               | -uuu uuuu                  |
| STATUS          | --00 xxxx           | --1u uuuu                                | --uu uuuu  | --01 uuuu                               | --11 uuuu                  |
| INTC0           | -000 0000           | -000 0000                                | -000 0000  | -000 0000                               | -uuu uuuu                  |
| INTC1           | -000 -000           | -000 -000                                | -000 -000  | -000 -000                               | -uuu -uuu                  |
| RTCC            | --00 0111           | --00 0111                                | --00 0111  | --00 0111                               | --uu uuuu                  |
| PA              | 1111 1111           | 1111 1111                                | 1111 1111  | 1111 1111                               | uuuu uuuu                  |
| PB              | 1111 1111           | 1111 1111                                | 1111 1111  | 1111 1111                               | uuuu uuuu                  |
| PC              | ---- 1111           | ---- 1111                                | ---- 1111  | ---- 1111                               | ---- uuuu                  |

Notes: "\*" refers to warm reset  
 "u" means unchanged  
 "x" means unknown

**Timer/event counter**

Two timer/event counters are implemented in the HT49C50. Both of them contain an 8-bit programmable count-up counter.

The timer/event count 0 clock source may come from the system clock or system clock/4 or RTC time-out signal or external source. System clock source or system clock/4 is selected by mask option.

The timer/event count 1 clock source may come from TMR0 overflow or system clock or time base time-out signal or system clock/4 or external source, and the three former clock source is selected by mask option.

The external clock input allows the user to count external events, measure time intervals or pulse widths, or to generate an accurate time base.

The two timer/event counters are operated almost in the same manner, except the clock source and related registers.

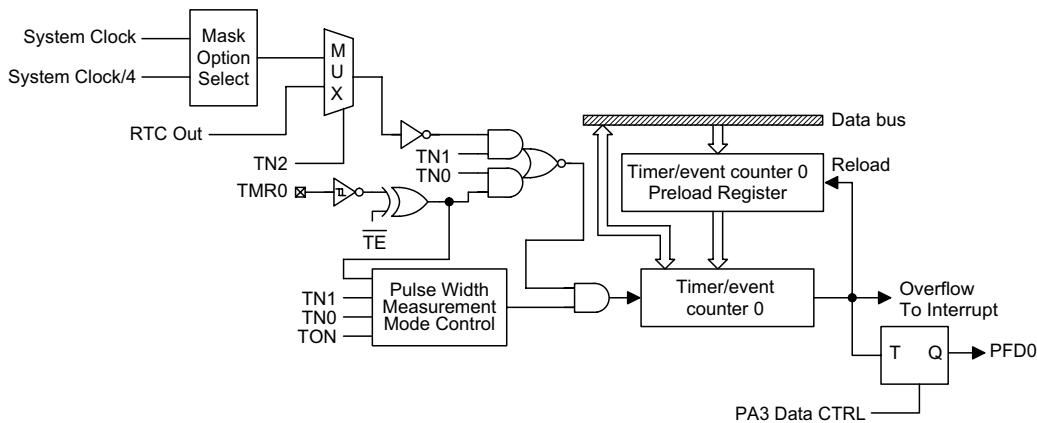
There are two registers related to the timer/event counter 0, i.e., TMR0 ([0DH]) and TMR0C ([0EH]), and two registers related to the timer/event counter 1, i.e., TMR1 ([10H], and TMR1C ([11H]). There are also two physical registers are mapped to TMR0 (TMR1) location; writing TMR0 (TMR1) places the starting value in the timer/event counter preload register, while reading it yields the contents of the timer/event counter. TMR0C and TMR1C are

timer/event counter control registers used to define some options.

The TN0 and TN1 bits define the operation mode. The event count mode is used to count external events, which means that the clock source is from an external (TMR0, TMR1) pin. The timer mode functions as a normal timer with the clock source coming from the internal selected clock source. Finally, the pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR0, TMR1), and the counting is based on the internal selected clock source.

In the event count or timer mode, the timer/event counter starts counting at the current contents in the timer/event counter and ends at FFH. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag (TOF; bit 6 of INTC0, T1F; bit 4 of INTC1).

In the pulse width measurement mode with the values of the TON and TE bits equal to one, after the TMR0 (TMR1) has received a transient from low to high (or high to low if the TE bit is "0"), it will start counting until the TMR0 (TMR1) returns to the original level and resets the TON. The measured result remains in the timer/event counter even if the activated transient occurs again. In other words, only one cycle measurement can be made until the TON is set. The cycle measurement will re-function as long as it receives



Timer/event counter 0

| Label (TMR0C) | Bits   | Function   |
|---------------|--------|--|
| —             | 0~2    | Unused bits, read as "0"   |
| TE            | 3      | To define the TMR0 active edge of timer/event counter (0=active on low to high; 1=active on high to low)   |
| TON           | 4      | To enable/disable timer counting (0=disabled; 1=enabled)   |
| TN2           | 5      | 2 to 1 multiplexer control inputs to select the timer/event counter clock source (0=RTC outputs; 1= system clock or system clock/4)  |
| TN0<br>TN1    | 6<br>7 | To define the operating mode (TN1, TN0)<br>01= Event count mode (External clock)<br>10= Timer mode (Internal clock)<br>11= Pulse Width measurement mode (External clock)<br>00= Unused |

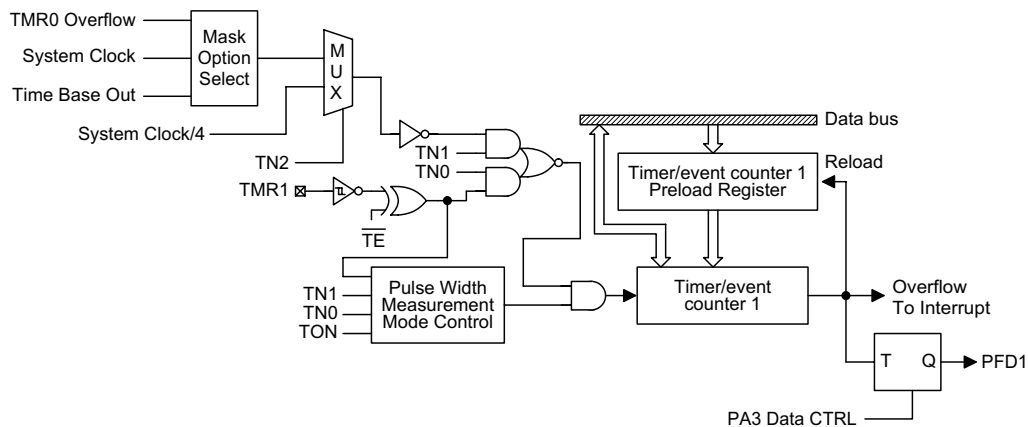
TMR0C register

further transient pulse. In this operation mode, the timer/event counter begins counting according not to the logic level but to the transient edges. In the case of counter overflows, the counter is reloaded from the timer/event counter preload register and issues an interrupt request, as in the other two modes, i.e., event and timer modes.

To enable the counting operation, the Timer ON bit (TON; bit 4 of TMR0C or TMR1C) should be set to 1. In the pulse width measurement mode, the TON is automatically cleared after the measurement cycle is completed. But in the other two modes, the TON can only be reset by

instructions. The overflow of the timer/event counter 0/1 is one of the wake-up sources and can also be applied to a PFD (Programmable Frequency Divider) output at PA3 by mask option. Only one PFD (PFD0 or PFD1) can be applied to PA3 by mask option. No matter what the operation mode is, writing a 0 to ET0I or ET1I disables the related interrupt service. When the PFD function is selected, executing "CLR [PA].3" instruction to enable PFD output and executing "SET [PA].3" instruction to disable PFD output.

In the case of timer/event counter OFF condition, writing data to the timer/event counter preload register also reloads that data to the



Timer/event counter 1

| Label (TMR1C) | Bits   | Function  |
|---------------|--------|---|
| —             | 0~2    | Unused bits, read as "0"  |
| TE            | 3      | To define the TMR1 active edge of timer/event counter (0= active on low to high; 1= active on high to low)  |
| TON           | 4      | To enable/disable timer counting (0= disabled; 1= enabled)  |
| TN2           | 5      | 2 to 1 multiplexer control inputs to select the timer/event counter clock source (0= mask option clock source; 1= system clock/4)   |
| TN1<br>TN0    | 7<br>6 | To define the operating mode<br>01= Event count mode (External clock)<br>10= Timer mode (Internal clock)<br>11= Pulse Width measurement mode (External clock)<br>00= Unused |

TMR1C register

timer/ event counter. But if the timer/event counter is turn on, data written to the timer/event counter is kept only in the timer/event counter preload register. The timer/event counter still continues its operation until an overflow occurs.

When the timer/event counter (reading TMR0/TMR1) is read, the clock is blocked to avoid errors. As this may results in a counting error, blocking of the clock should be taken into account by the programmer.

It is strongly recommended to load a desired value into the TMR0/TMR1 register first, then turn on the related timer/event counter for proper operation. Because the initial value of TMR0/TMR1 is unknown.

Due to the timer/event scheme, the programmer should pay special attention on the instruction to enable then disable the timer for the first time, whenever there is a need to use the timer/event function, to avoid unpredictable result. After this procedure, the timer/event function can be operated normally. The example given below, using two 8-bit width Timer's (timer 0 ;timer 1) cascade into 16-bit width.

START:

```

mov  a, 09h  ; Set ET0I&EMI bits to
mov  intc0, a ; enable timer 0 and
                ; global interrupt

mov  a, 01h  ; Set ET1I bit to enable
mov  intc1, a ; timer 1 interrupt

mov  a, 80h  ; Set operating mode as
mov  tmr1c, a ; timer mode and select mask
                ; option clock source

mov  a, 0a0h ; Set operating mode as timer
mov  tmr0c, a ; mode and select system
                ; clock/4

set  tmr1c.4 ; Enable then disable timer 1
clr  tmr1c.4 ; for the first time

mov  a, 00h  ; Load a desired value into
mov  tmr0, a ; the TMR0/TMR1 register
mov  a, 00h  ;
mov  tmr1, a ;

set  tmr0c.4 ; Normal operating
set  tmr1c.4 ;

```

END



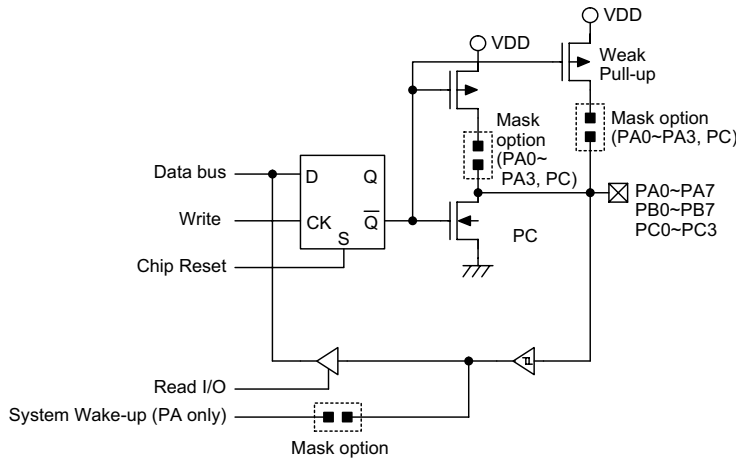
**Input/output ports**

There are a 12-bit bidirectional input/output port, an 8-bit input port in the HT49C50, labeled PA, PB and PC which are mapped to [12H], [14H] and [16H] of the RAM, respectively. PA0~PA3 can be configured as CMOS (output) or NMOS (input/output) with or without pull-high resistor by mask option. PA4~PA7 are always pull-high and NMOS (input/output). If you choose NMOS (input), each bit on the port (PA0~PA7) can be configured as a wake-up input. PB can only be used for input operation, and each bit on the port can be configured with pull-high resistor. PC can be configured as CMOS output or NMOS input/output with or without pull-high resistor by mask option. All the port for the input operation (PA, PB and PC), these ports are non-latched, that is, the inputs should be ready at the T2 rising edge of the instruction "MOV A, [m]" (m=12H or 14H). For PA, PC output operation, all data are latched and remain unchanged until the output latch is rewritten.

When the PA and PC structures are open drain NMOS type, it should be noted that, before reading data from the pads, a "1" should be written to the related bits to disable the NMOS device. That is executing first the instruction "SET [m].i" (i=0~7 for PA) to disable related NMOS device, and then "MOV A, [m]" to get stable data.

After chip reset, these input lines remain at the high level or are left floating (by mask option). Each bit of these output latches can be set or cleared by the "SET [m].i" and "CLR [m].i" (m=12H or 16H) instructions.

Some instructions first input data and then follow the output operations. For example, "SET [m].i", "CLR [m].i", "CPL [m]", "CPLA [m]" read the entire port states into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or to the accumulator.

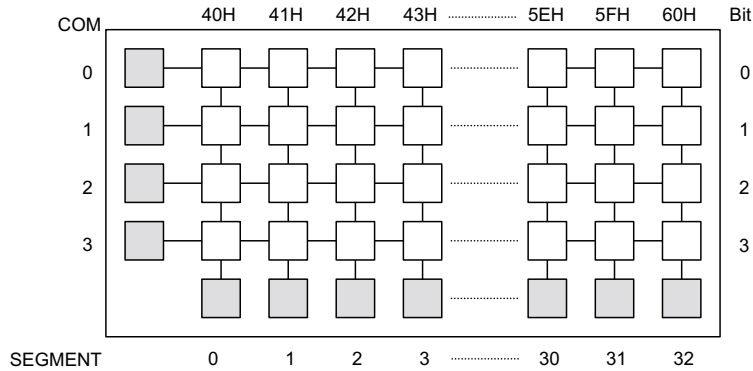


Input/output ports

**LCD display memory**

The HT49C50 provides an area of embedded data memory for LCD display. This area is located from 40H to 60H of the RAM at Bank 1. Bank pointer (BP; located at 04H of the RAM) is the switch between the RAM and the LCD display memory. When the BP is set as "1", any data written into 40H~60H will effect the LCD display. When the BP is cleared to "0", any data written into 40H~60H means to access the general purpose data memory. The LCD display

memory can be read and written to only by indirect addressing mode using MP1. When data is written into the display data area, it is automatically read by the LCD driver which then generates the corresponding LCD driving signals. To turn the display on or off, a "1" or a "0" is written to the corresponding bit of the display memory, respectively. The figure illustrates the mapping between the display memory and LCD pattern for the HT49C50.



Display memory

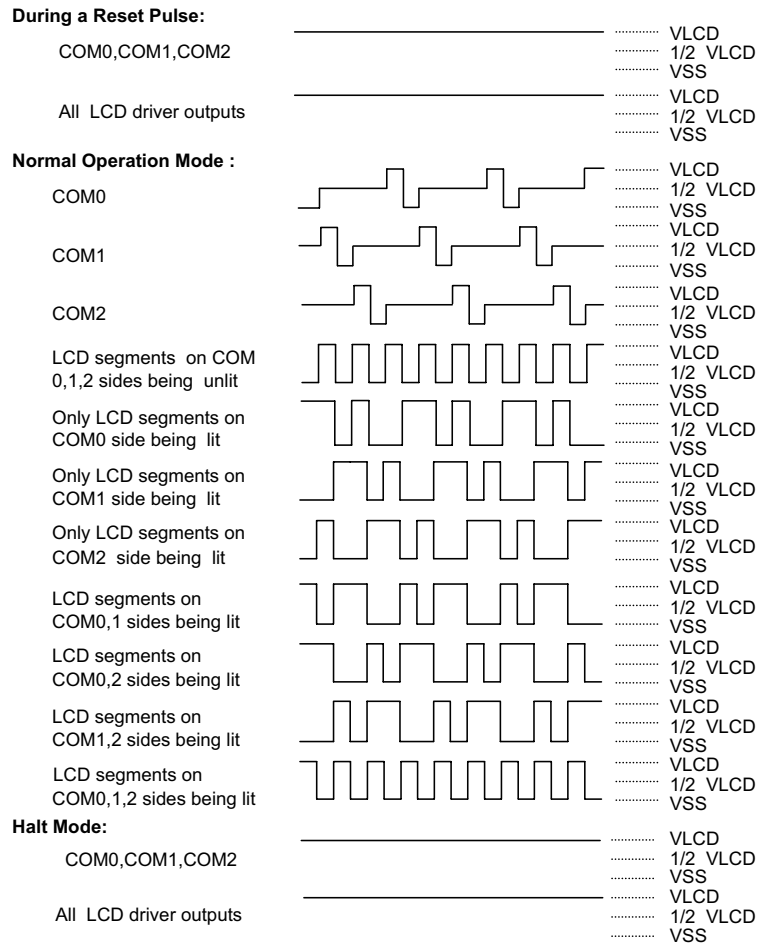
**LCD driver output**

The output number of the HT49C50 LCD driver can be 33×2 or 33×3 or 32×4 by mask option (i.e., 1/2 duty or 1/3 duty or 1/4 duty). The bias type of LCD driver can be "R" type or "C" type. If the "R" bias type is selected, no external capacitor is required. If the "C" bias type is selected, a capacitor mounted between C1 and C2 pins is needed. The bias voltage of LCD driver can be 1/2 bias or 1/3 bias by mask option. If 1/2 bias is selected, a capacitor mounted between V2 pin and ground is required. If 1/3 bias is selected, two capacitors are needed for V1 and V2 pins. Refer to application diagram.

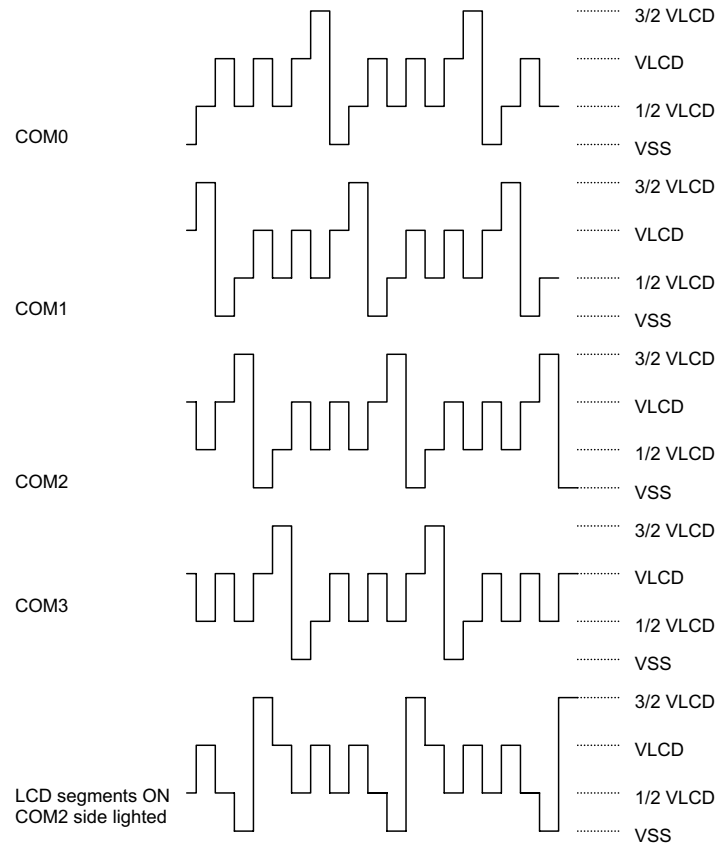
**Buzzer**

HT49C50 provides a pair of buzzer output BZ and BZ̄, which share pins with PA0 and PA1 respectively, and determined by mask option. Its output frequency can be selected by mask option.

When the buzzer function is selected, setting the PA.0 and PA.1 "0" simultaneously, will enable the buzzer output and sets the PA.0 "1" to disable the buzzer output.



LCD driver output (1/3 duty, 1/2 bias, R/C type)



LCD driver output (1/4 duty, 1/3 bias, C type)

| Register      | Bit No. | Label             | Read/Write | Reset | Function   |
|---------------|---------|-------------------|------------|-------|--|
| RTCC<br>(09H) | 0~2     | RT0<br>RT1<br>RT2 | R/W        | 0     | 8 to 1 multiplexer control inputs to select the real time clock prescaler output |
|               | 3       | —                 | —          | —     | Unused bits, this bit must be set to "0"   |
|               | 4       | QOSC              | R/W        | 0     | Control the RTC OSC to oscillate quickly<br>"0" enable<br>"1" disable            |
|               | 5~7     | —                 | —          | —     | Unused bits, read as "0"   |

RTCC register

**Mask option**

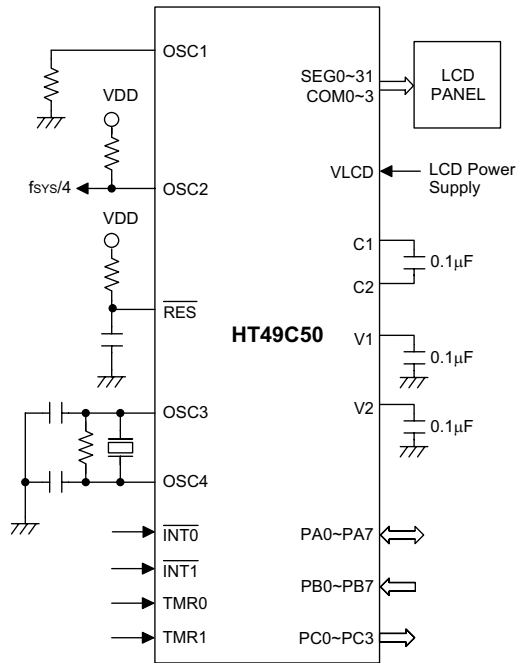
The following shows 18 kinds of mask options defined in order to ensure proper system functioning in the HT49C50. All these options should be defined.

| No. | Mask Option   |
|-----|---|
| 1   | OSC type selection. This option is to decide if an RC or Crystal oscillator is chosen as system clock.  |
| 2   | WDT Clock source selection. RTC and Time Base. There are three types of selection: system clock/4 or RTC OSC or WDT OSC.  |
| 3   | WDT enable/disable selection. WDT can be enabled or disabled by mask option.  |
| 4   | CLR WDT times selection. This option defines how to clear the WDT by instruction. "One time" means that the "CLR WDT" can clear the WDT. "Two times" means only if both of the "CLR WDT1" and "CLR WDT2" have been executed, the WDT can be cleared.  |
| 5   | Time Base time-out period selection. The Time Base time-out period ranges from $\text{clock}/2^{12}$ to $\text{clock}/2^{15}$ . "Clock" means the clock source selected by mask option.   |
| 6   | Buzzer output frequency selection. There are eight types of frequency signals for buzzer output: $\text{Clock}/2^2 \sim \text{Clock}/2^9$ . "Clock" means the clock source selected by mask option.   |
| 7   | Wake-up selection. This option defines the wake-up capability. External I/O pins (PA only) all have the capability to wake-up the chip from a HALT by a falling edge.   |
| 8   | Pull-high selection. This option is to decide whether the pull-high resistance is visible or not on the PA0~PA3 and PC. (PB and PA4~PA7 are always pull-high)   |
| 9   | PA0~PA3 and PC CMOS or NMOS selection.<br>The structure of PA0~PA3 and PC each 4 bits can be selected as CMOS or NMOS individually. When the CMOS is selected, the related pins only can be used for output operations. When the NMOS is selected, the related pins can be used for input or output operations. (PA4~PA7 are always NMOS) |

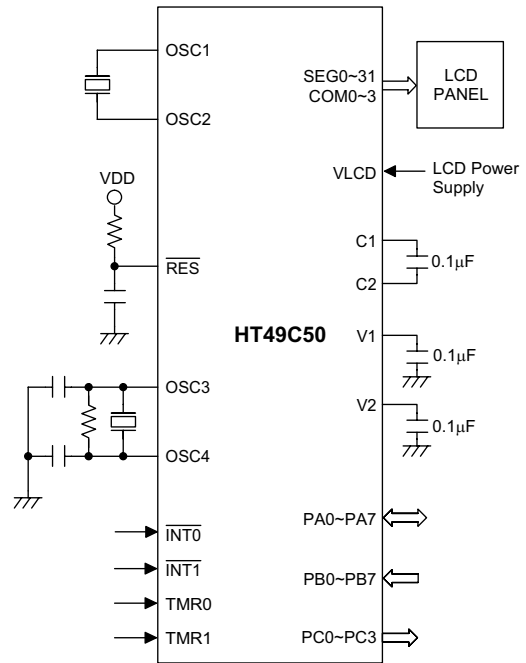
| No. | Mask Option  |
|-----|--|
| 10  | Clock source selection of timer/event counter 0. There are two types of selection: system clock or system clock/4.   |
| 11  | Clock source selection of timer/event counter 1. There are three types of selection: TMR0 overflow, system clock or Time Base overflow.  |
| 12  | I/O pins share with other functions selection.<br>PA0/BZ, PA1/BZ: PA0 and PA1 can be set as I/O pins or buzzer outputs.<br>PA3/PFD: PA3 can be set as I/O pins or PFD output.  |
| 13  | LCD common selection. There are three types of selection: 2 common (1/2 duty) or 3 common (1/3 duty) or 4 common (1/4 duty). If the 4 common is selected, the segment output pin "SEG32" will be set as a common output.   |
| 14  | LCD bias power supply selection.<br>There are two types of selection: 1/2 bias or 1/3 bias.  |
| 15  | LCD bias type selection.<br>This option is to decide what kind of bias is selected, R type or C type.  |
| 16  | LCD driver clock selection. There are seven types of frequency signals for the LCD driver circuits: $f_S/2^2 \sim f_S/2^6$ . "f <sub>S</sub> " means the clock source selection by mask option.  |
| 17  | PFD selection.<br>If PA3 is set as PFD output, there are two types of selection; One is PFD0 as the PFD output, the other is PFD1 as the PFD output. PFD0, PFD1 are the timer overflow signals of the timer/event counter 0, timer/event counter 1 respectively. |

## Application Circuits

### RC oscillator application



### Crystal oscillator application



**Instruction Set Summary**

| <b>Mnemonic</b>                | <b>Description</b>  | <b>Flag Affected</b> |
|--------------------------------|---|----------------------|
| <b>Arithmetic</b>              |   |                      |
| ADD A,[m]                      | Add data memory to ACC  | Z,C,AC,OV            |
| ADDM A,[m]                     | Add ACC to data memory  | Z,C,AC,OV            |
| ADD A,x                        | Add immediate data to ACC   | Z,C,AC,OV            |
| ADC A,[m]                      | Add data memory to ACC with carry                                   | Z,C,AC,OV            |
| ADCM A,[m]                     | Add ACC to register with carry                                      | Z,C,AC,OV            |
| SUB A,x                        | Subtract immediate data from ACC                                    | Z,C,AC,OV            |
| SUB A,[m]                      | Subtract data memory from ACC                                       | Z,C,AC,OV            |
| SUBM A,[m]                     | Subtract data memory from ACC with result in data memory            | Z,C,AC,OV            |
| SBC A,[m]                      | Subtract data memory from ACC with carry                            | Z,C,AC,OV            |
| SBCM A,[m]                     | Subtract data memory from ACC with carry with result in data memory | Z,C,AC,OV            |
| DAA [m]                        | Decimal adjust ACC for addition with result in data memory          | C                    |
| <b>Logic Operation</b>         |   |                      |
| AND A,[m]                      | AND data memory to ACC  | Z                    |
| OR A,[m]                       | OR data memory to ACC   | Z                    |
| XOR A,[m]                      | Exclusive-OR data memory to ACC                                     | Z                    |
| ANDM A,[m]                     | AND ACC to data memory  | Z                    |
| ORM A,[m]                      | OR ACC to data memory   | Z                    |
| XORM A,[m]                     | Exclusive-OR ACC to data memory                                     | Z                    |
| AND A,x                        | AND immediate data to ACC   | Z                    |
| OR A,x                         | OR immediate data to ACC  | Z                    |
| XOR A,x                        | Exclusive-OR immediate data to ACC                                  | Z                    |
| CPL [m]                        | Complement data memory  | Z                    |
| CPLA [m]                       | Complement data memory with result in ACC                           | Z                    |
| <b>Increment and Decrement</b> |   |                      |
| INCA [m]                       | Increment data memory with result in ACC                            | Z                    |
| INC [m]                        | Increment data memory   | Z                    |
| DECA [m]                       | Decrement data memory with result in ACC                            | Z                    |
| DEC [m]                        | Decrement data memory   | Z                    |



| <b>Mnemonic</b>      | <b>Description</b>  | <b>Flag Affected</b> |
|----------------------|---|----------------------|
| <b>Rotate</b>        |   |                      |
| RRA [m]              | Rotate data memory right with result in ACC               | None                 |
| RR [m]               | Rotate data memory right                                  | None                 |
| RRCA [m]             | Rotate data memory right through carry with result in ACC | C                    |
| RRC [m]              | Rotate data memory right through carry                    | C                    |
| RLA [m]              | Rotate data memory left with result in ACC                | None                 |
| RL [m]               | Rotate data memory left                                   | None                 |
| RLCA [m]             | Rotate data memory left through carry with result in ACC  | C                    |
| RLC [m]              | Rotate data memory left through carry                     | C                    |
| <b>Data Move</b>     |   |                      |
| MOV A,[m]            | Move data memory to ACC                                   | None**               |
| MOV [m],A            | Move ACC to data memory                                   | None                 |
| MOV A,x              | Move immediate data to ACC                                | None                 |
| <b>Bit Operation</b> |   |                      |
| CLR [m].i            | Clear bit of data memory                                  | None                 |
| SET [m].i            | Set bit of data memory                                    | None                 |
| <b>Branch</b>        |   |                      |
| JMP addr             | Jump unconditionally                                      | None                 |
| SZ [m]               | Skip if data memory is zero                               | None                 |
| SZA [m]              | Skip if data memory is zero with data movement to ACC     | None                 |
| SZ [m].i             | Skip if bit i of data memory is zero                      | None                 |
| SNZ [m].i            | Skip if bit i of data memory is not zero                  | None                 |
| SIZ [m]              | Skip if increment data memory is zero                     | None                 |
| SDZ [m]              | Skip if decrement data memory is zero                     | None                 |
| SIZA [m]             | Skip if increment data memory is zero with result in ACC  | None                 |
| SDZA [m]             | Skip if decrement data memory is zero with result in ACC  | None                 |
| CALL addr            | Subroutine call   | None                 |
| RET                  | Return from subroutine                                    | None                 |
| RET A,x              | Return from subroutine and load immediate data to ACC     | None                 |
| RETI                 | Return from interrupt                                     | None                 |
| <b>Table Read</b>    |   |                      |
| TABRDC [m]           | Read ROM code (current page) to data memory and TBLH      | None                 |
| TABRDL [m]           | Read ROM code (last page) to data memory and TBLH         | None                 |

| <b>Mnemonic</b> | <b>Description</b>                             | <b>Flag Affected</b> |
|-----------------|--|----------------------|
| Miscellaneous   |  |                      |
| NOP             | No operation                                   | None                 |
| CLR [m]         | Clear data memory                              | None                 |
| SET [m]         | Set data memory                                | None                 |
| CLR WDT         | Clear Watchdog timer                           | TO,PD                |
| CLR WDT1        | Pre-clear Watchdog timer                       | TO*,PD*              |
| CLR WDT2        | Pre-clear Watchdog timer                       | TO*,PD*              |
| SWAP [m]        | Swap nibbles of data memory                    | None                 |
| SWAPA [m]       | Swap nibbles of data memory with result in ACC | None                 |
| HALT            | Enter power down mode                          | TO,PD                |

Notes: x: 8-bit immediate data

m: 7-bit data memory address

A: accumulator

i: 0~7 number of bits

addr: 10-bit program memory address

√: Flag(s) is affected

–: Flag(s) is not affected

\*: Flag(s) may be affected by the execution status

\*\* : For the old version of the E.V. chip, the zero flag (Z) can be affected by executing the MOV A,[M] instruction.

For the new version of the E.V. chip, the zero flag cannot be changed by executing the MOV A,[M] instruction.

**Instruction Definition**

**ADC A,[m]** Add data memory and carry to the accumulator  
 Description The contents of the specified data memory, accumulator and the carry flag are added simultaneously, leaving the result in the accumulator.

Operation  $ACC \leftarrow ACC+[m]+C$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | √  | √ | √  | √ |

**ADCM A,[m]** Add the accumulator and carry to data memory  
 Description The contents of the specified data memory, accumulator and the carry flag are added simultaneously, leaving the result in the specified data memory.

Operation  $[m] \leftarrow ACC+[m]+C$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | √  | √ | √  | √ |

**ADD A,[m]** Add data memory to the accumulator  
 Description The contents of the specified data memory and the accumulator are added. The result is stored in the accumulator.

Operation  $ACC \leftarrow ACC+[m]$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | √  | √ | √  | √ |

**ADD A,x** Add immediate data to the accumulator  
 Description The contents of the accumulator and the specified data are added, leaving the result in the accumulator.

Operation  $ACC \leftarrow ACC+x$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | √  | √ | √  | √ |

**ADDM A,[m]** Add the accumulator to the data memory  
 Description The contents of the specified data memory and the accumulator are added. The result is stored in the data memory.  
 Operation  $[m] \leftarrow ACC + [m]$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | √  | √ | √  | √ |

**AND A,[m]** Logical AND accumulator with data memory  
 Description Data in the accumulator and the specified data memory perform a bitwise logical\_AND operation. The result is stored in the accumulator.  
 Operation  $ACC \leftarrow ACC \text{ "AND" } [m]$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**AND A,x** Logical AND immediate data to the accumulator  
 Description Data in the accumulator and the specified data perform a bitwise logical\_AND operation. The result is stored in the accumulator.  
 Operation  $ACC \leftarrow ACC \text{ "AND" } x$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**ANDM A,[m]** Logical AND data memory with the accumulator  
 Description Data in the specified data memory and the accumulator perform a bitwise logical\_AND operation. The result is stored in the data memory.  
 Operation  $[m] \leftarrow ACC \text{ "AND" } [m]$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**CALL addr** Subroutine call  
**Description** The instruction unconditionally calls a subroutine located at the indicated address. The program counter increments once to obtain the address of the next instruction, and pushes this onto the stack. The indicated address is then loaded. Program execution continues with the instruction at this address.

**Operation** Stack ← PC+1  
 PC ← addr

**Affected flag(s)**

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**CLR [m]** Clear data memory  
**Description** The contents of the specified data memory are cleared to zero.

**Operation** [m] ← 00H

**Affected flag(s)**

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**CLR [m].i** Clear bit of data memory  
**Description** The bit i of the specified data memory is cleared to zero.

**Operation** [m].i ← 0

**Affected flag(s)**

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**CLR WDT** Clear watchdog timer  
**Description** The WDT is cleared (re-counting from zero). The power down bit (PD) and time-out bit (TO) are cleared.

**Operation** WDT ← 00H  
 PD and TO ← 0

**Affected flag(s)**

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | 0  | 0  | —  | — | —  | — |

**CLR WDT1**

Preclear watchdog timer

## Description

The TD, PD flags and WDT are all cleared (re-counting from zero), if the other preclear WDT instruction has been executed. Only execution of this instruction without the other preclear instruction sets the indicated flag which implies that this instruction has been executed and the TO and PD flags remain unchanged.

## Operation

WDT  $\leftarrow$  00H\*  
 PD and TO  $\leftarrow$  0\*

## Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | 0* | 0* | —  | — | —  | — |

**CLR WDT2**

Preclear watchdog timer

## Description

The TO, PD flags and WDT are all cleared (re-counting from zero), if the other preclear WDT instruction has been executed. Only execution of this instruction without the other preclear instruction sets the indicated flag which implies that this instruction has been executed and the TO and PD flags remain unchanged.

## Operation

WDT  $\leftarrow$  00H\*  
 PD and TO  $\leftarrow$  0\*

## Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | 0* | 0* | —  | — | —  | — |

**CPL [m]**

Complement data memory

## Description

Each bit of the specified data memory is logically complemented (1's complement). Bits which previously contained a one are changed to zero and vice-versa.

## Operation

[m]  $\leftarrow$   $\overline{[m]}$

## Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**CPLA [m]** Complement data memory and place result in the accumulator  
 Description Each bit of the specified data memory is logically complemented (1's complement). Bits which previously contained a one are changed to zero and vice-versa. The complemented result is stored in the accumulator and the contents of the data memory remain unchanged.

Operation  $ACC \leftarrow \overline{[m]}$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**DAA [m]** Decimal-Adjust accumulator for addition  
 Description The accumulator value is adjusted to the BCD (Binary Code Decimal) code. The accumulator is divided into two nibbles. Each nibble is adjusted to the BCD code and an internal carry (AC1) will be done if the low nibble of the accumulator is greater than 9. The BCD adjustment is done by adding 6 to the original value if the original value is greater than 9 or a carry (AC or C) is set; otherwise the original value remains unchanged. The result is stored in the data memory and only the carry flag (C) may be affected.

Operation  
 If  $ACC.3 \sim ACC.0 > 9$  or  $AC=1$   
 then  $[m].3 \sim [m].0 \leftarrow (ACC.3 \sim ACC.0) + 6, AC1 = \overline{AC}$   
 else  $[m].3 \sim [m].0 \leftarrow (ACC.3 \sim ACC.0), AC1 = 0$   
 and  
 If  $ACC.7 \sim ACC.4 + AC1 > 9$  or  $C=1$   
 then  $[m].7 \sim [m].4 \leftarrow ACC.7 \sim ACC.4 + 6 + AC1, C=1$   
 else  $[m].7 \sim [m].4 \leftarrow ACC.7 \sim ACC.4 + AC1, C=C$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | √ |

**DEC [m]** Decrement data memory  
 Description Data in the specified data memory is decremented by one.

Operation  $[m] \leftarrow [m] - 1$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**DECA [m]** Decrement data memory and place result in the accumulator  
 Description Data in the specified data memory is decremented by one, leaving the result in the accumulator. The contents of the data memory remain unchanged.  
 Operation  $ACC \leftarrow [m]-1$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**HALT** Enter power down mode  
 Description This instruction stops program execution and turns off the system clock. The contents of the RAM and registers are retained. The WDT and prescaler are cleared. The power down bit (PD) is set and the WDT time-out bit (TO) is cleared.  
 Operation  $PC \leftarrow PC+1$   
 $PD \leftarrow 1$   
 $TO \leftarrow 0$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | 0  | 1  | —  | — | —  | — |

**INC [m]** Increment data memory  
 Description Data in the specified data memory is incremented by one.  
 Operation  $[m] \leftarrow [m]+1$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**INCA [m]** Increment data memory and place result in the accumulator  
 Description Data in the specified data memory is incremented by one, leaving the result in the accumulator. The contents of the data memory remain unchanged.  
 Operation  $ACC \leftarrow [m]+1$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |



**JMP addr** Directly jump  
 Description The contents of the program counter are replaced with the directly-specified address unconditionally, and control is passed to this destination.  
 Operation  $PC \leftarrow \text{addr}$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**MOV A,[m]** Move data memory to the accumulator  
 Description The contents of the specified data memory are copied to the accumulator.  
 Operation  $ACC \leftarrow [m]$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**MOV A,x** Move immediate data to the accumulator  
 Description The 8-bit data specified by the code is loaded into the accumulator.  
 Operation  $ACC \leftarrow x$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**MOV [m],A** Move the accumulator to data memory  
 Description The contents of the accumulator are copied to the specified data memory (one of the data memories).  
 Operation  $[m] \leftarrow ACC$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**NOP** No operation  
 Description No operation is performed. Execution continues with the next instruction.  
 Operation  $PC \leftarrow PC+1$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**OR A,[m]** Logical OR accumulator with data memory  
 Description Data in the accumulator and the specified data memory (one of the data memories) perform a bitwise logical\_OR operation. The result is stored in the accumulator.

Operation  $ACC \leftarrow ACC \text{ "OR" } [m]$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**OR A,x** Logical OR immediate data to the accumulator  
 Description Data in the accumulator and the specified data perform a bitwise logical\_OR operation. The result is stored in the accumulator.

Operation  $ACC \leftarrow ACC \text{ "OR" } x$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**ORM A,[m]** Logical OR data memory with the accumulator  
 Description Data in the data memory (one of the data memories) and the accumulator perform a bitwise logical\_OR operation. The result is stored in the data memory.

Operation  $[m] \leftarrow ACC \text{ "OR" } [m]$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**RET** Return from subroutine  
 Description The program counter is restored from the stack. This is a two-cycle instruction.

Operation  $PC \leftarrow \text{Stack}$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**RET A,x** Return and place immediate data in the accumulator  
 Description The program counter is restored from the stack and the accumulator loaded with the specified 8-bit immediate data.

Operation PC ← Stack  
 ACC ← x

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**RETI** Return from interrupt  
 Description The program counter is restored from the stack, and interrupts are enabled by setting the EMI bit. EMI is the enable master (global) interrupt bit (bit 0; register INTC).

Operation PC ← Stack  
 EMI ← 1

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**RL [m]** Rotate data memory left  
 Description The contents of the specified data memory are rotated one bit left with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; [m].i:bit i of the data memory (i=0~6)  
 [m].0 ← [m].7

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**RLA [m]** Rotate data memory left and place result in the accumulator  
 Description Data in the specified data memory is rotated one bit left with bit 7 rotated into bit 0, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC.(i+1) ← [m].i; [m].i:bit i of the data memory (i=0~6)  
 ACC.0 ← [m].7

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**RLC [m]** Rotate data memory left through carry  
 Description The contents of the specified data memory and the carry flag are rotated one bit left. Bit 7 replaces the carry bit; the original carry flag is rotated into the bit 0 position.

Operation  $[m].(i+1) \leftarrow [m].i$ ;  $[m].i$ :bit  $i$  of the data memory ( $i=0\sim6$ )  
 $[m].0 \leftarrow C$   
 $C \leftarrow [m].7$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | √ |

**RLCA [m]** Rotate left through carry and place result in the accumulator  
 Description Data in the specified data memory and the carry flag are rotated one bit left. Bit 7 replaces the carry bit and the original carry flag is rotated into bit 0 position. The rotated result is stored in the accumulator but the contents of the data memory remain unchanged.

Operation  $ACC.(i+1) \leftarrow [m].i$ ;  $[m].i$ :bit  $i$  of the data memory ( $i=0\sim6$ )  
 $ACC.0 \leftarrow C$   
 $C \leftarrow [m].7$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | √ |

**RR [m]** Rotate data memory right  
 Description The contents of the specified data memory are rotated one bit right with bit 0 rotated to bit 7.

Operation  $[m].i \leftarrow [m].(i+1)$ ;  $[m].i$ :bit  $i$  of the data memory ( $i=0\sim6$ )  
 $[m].7 \leftarrow [m].0$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**RRA [m]** Rotate right-place result in the accumulator

Description Data in the specified data memory is rotated one bit right with bit 0 rotated into bit 7, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation  $ACC.i \leftarrow [m].(i+1)$ ;  $[m].i$ :bit i of the data memory (i=0~6)  
 $ACC.7 \leftarrow [m].0$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**RRC [m]** Rotate data memory right through carry

Description The contents of the specified data memory and the carry flag are together rotated one bit right. Bit 0 replaces the carry bit; the original carry flag is rotated into the bit 7 position.

Operation  $[m].i \leftarrow [m].(i+1)$ ;  $[m].i$ :bit i of the data memory (i=0~6)  
 $[m].7 \leftarrow C$   
 $C \leftarrow [m].0$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | √ |

**RRCA [m]** Rotate right through carry-place result in the accumulator

Description Data of the specified data memory and the carry flag are rotated one bit right. Bit 0 replaces the carry bit and the original carry flag is rotated into the bit 7 position. The rotated result is stored in the accumulator. The contents of the data memory remain unchanged.

Operation  $ACC.i \leftarrow [m].(i+1)$ ;  $[m].i$ :bit i of the data memory (i=0~6)  
 $ACC.7 \leftarrow C$   
 $C \leftarrow [m].0$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | √ |

**SBC A,[m]** Subtract data memory and carry from the accumulator  
 Description The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the accumulator.  
 Operation  $ACC \leftarrow ACC + [\overline{m}] + C$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | √  | √ | √  | √ |

**SBCM A,[m]** Subtract data memory and carry from the accumulator  
 Description The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the data memory.  
 Operation  $[m] \leftarrow ACC + [\overline{m}] + C$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | √  | √ | √  | √ |

**SDZ [m]** Skip if decrement data memory is zero  
 Description The contents of the specified data memory are decremented by one. If the result is zero, the next instruction is skipped. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).  
 Operation Skip if  $([m]-1)=0$ ,  $[m] \leftarrow ([m]-1)$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**SDZA [m]** Decrement data memory and place result in ACC, skip if zero  
 Description The contents of the specified data memory are decremented by one. If the result is zero, the next instruction is skipped. The result is stored in the accumulator but the data memory remains unchanged. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).  
 Operation Skip if  $([m]-1)=0$ ,  $ACC \leftarrow ([m]-1)$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**SET [m]** Set data memory  
 Description Each bit of the specified data memory is set to one.  
 Operation  $[m] \leftarrow FFH$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**SET [m].i** Set bit of data memory  
 Description Bit "i" of the specified data memory is set to one.  
 Operation  $[m].i \leftarrow 1$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**SIZ [m]** Skip if increment data memory is zero  
 Description The contents of the specified data memory are incremented by one. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).  
 Operation Skip if  $([m]+1)=0$ ,  $[m] \leftarrow ([m]+1)$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**SIZA [m]** Increment data memory and place result in ACC, skip if zero  
 Description The contents of the specified data memory are incremented by one. If the result is zero, the next instruction is skipped and the result is stored in the accumulator. The data memory remains unchanged. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).  
 Operation Skip if  $([m]+1)=0$ ,  $ACC \leftarrow ([m]+1)$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**SNZ [m].i** Skip if bit "i" of the data memory is not zero

Description If bit "i" of the specified data memory is not zero, the next instruction is skipped. If bit "i" of the data memory is not zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).

Operation Skip if [m].i≠0

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**SUB A,[m]** Subtract data memory from the accumulator

Description The specified data memory is subtracted from the contents of the accumulator, leaving the result in the accumulator.

Operation  $ACC \leftarrow ACC + [\bar{m}] + 1$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | √  | √ | √  | √ |

**SUBM A,[m]** Subtract data memory from the accumulator

Description The specified data memory is subtracted from the contents of the accumulator, leaving the result in the data memory.

Operation  $[m] \leftarrow ACC + [\bar{m}] + 1$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | √  | √ | √  | √ |

**SUB A,x** Subtract immediate data from the accumulator

Description The immediate data specified by the code is subtracted from the contents of the accumulator, leaving the result in the accumulator.

Operation  $ACC \leftarrow ACC + \bar{x} + 1$

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | √  | √ | √  | √ |



**SWAP [m]** Swap nibbles within the data memory  
 Description The low-order and high-order nibbles of the specified data memory (one of the data memories) are interchanged.  
 Operation  $[m].3\sim[m].0 \leftrightarrow [m].7\sim[m].4$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**SWAPA [m]** Swap data memory-place result in the accumulator  
 Description The low-order and high-order nibbles of the specified data memory are interchanged, writing the result to the accumulator. The contents of the data memory remain unchanged.  
 Operation  $ACC.3\sim ACC.0 \leftarrow [m].7\sim[m].4$   
 $ACC.7\sim ACC.4 \leftarrow [m].3\sim[m].0$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**SZ [m]** Skip if data memory is zero  
 Description If the contents of the specified data memory are zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).  
 Operation Skip if  $[m]=0$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**SZA [m]** Move data memory to ACC, skip if zero  
 Description The contents of the specified data memory are copied to the accumulator. If the contents is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).  
 Operation Skip if  $[m]=0$ ,  $ACC \leftarrow [m]$   
 Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**SZ [m].i** Skip if bit "i" of the data memory is zero

Description If bit "i" of the specified data memory is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).

Operation Skip if [m].i=0

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**TABRDC [m]** Move the ROM code (current page) to TBLH and data memory

Description The low byte of ROM code (current page) addressed by the table pointer (TBLP) is moved to the specified data memory and the high byte transferred to TBLH directly.

Operation [m] ← ROM code (low byte)  
TBLH ← ROM code (high byte)

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**TABRDL [m]** Move the ROM code (last page) to TBLH and data memory

Description The low byte of ROM code (last page) addressed by the table pointer (TBLP) is moved to the data memory and the high byte transferred to TBLH directly.

Operation [m] ← ROM code (low byte)  
TBLH ← ROM code (high byte)

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | — | —  | — |

**XOR A,[m]** Logical XOR accumulator with data memory

Description Data in the accumulator and the indicated data memory perform a bitwise logical Exclusive\_OR operation and the result is stored in the accumulator.

Operation ACC ← ACC "XOR" [m]

Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**XORM A,[m]**

Logical XOR data memory with the accumulator

## Description

Data in the indicated data memory and the accumulator perform a bitwise logical Exclusive\_OR operation. The result is stored in the data memory. The zero flag is affected.

## Operation

 $[m] \leftarrow \text{ACC} \text{ "XOR" } [m]$ 

## Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**XOR A,x**

Logical XOR immediate data to the accumulator

## Description

Data in the the accumulator and the specified data perform a bitwise logical Exclusive\_OR operation. The result is stored in the accumulator. The zero flag is affected.

## Operation

 $\text{ACC} \leftarrow \text{ACC} \text{ "XOR" } x$ 

## Affected flag(s)

| TC2 | TC1 | TO | PD | OV | Z | AC | C |
|-----|-----|----|----|----|---|----|---|
| —   | —   | —  | —  | —  | √ | —  | — |

**Holtek Semiconductor Inc. (Headquarters)**

No.3 Creation Rd. II, Science-based Industrial Park, Hsinchu, Taiwan, R.O.C.  
Tel: 886-3-563-1999  
Fax: 886-3-563-1189

**Holtek Semiconductor Inc. (Taipei Office)**

5F, No.576, Sec.7 Chung Hsiao E. Rd., Taipei, Taiwan, R.O.C.  
Tel: 886-2-2782-9635  
Fax: 886-2-2782-9636  
Fax: 886-2-2782-7128 (International sales hotline)

**Holtek Microelectronics Enterprises Ltd.**

RM.711, Tower 2, Cheung Sha Wan Plaza, 833 Cheung Sha Wan Rd., Kowloon, Hong Kong  
Tel: 852-2-745-8288  
Fax: 852-2-742-8657

Copyright © 1999 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at <http://www.holtek.com.tw>.