

**Technical Document**

- [Tools Information](#)
- [FAQs](#)
- [Application Note](#)
  - [HA0003E Communicating between the HT48 & HT46 Series MCUs and the HT93LC46 EEPROM](#)
  - [HA0004E HT48 & HT46 MCU UART Software Implementation Method](#)
  - [HA0005E Controlling the I2C bus with the HT48 & HT46 MCU Series](#)
  - [HA0007E Using the MCU Look Up Table Instructions](#)
  - [HA0049E Read and Write Control of the HT1380](#)
  - [HA0075E MCU Reset and Oscillator Circuits Application Note](#)

**Features**

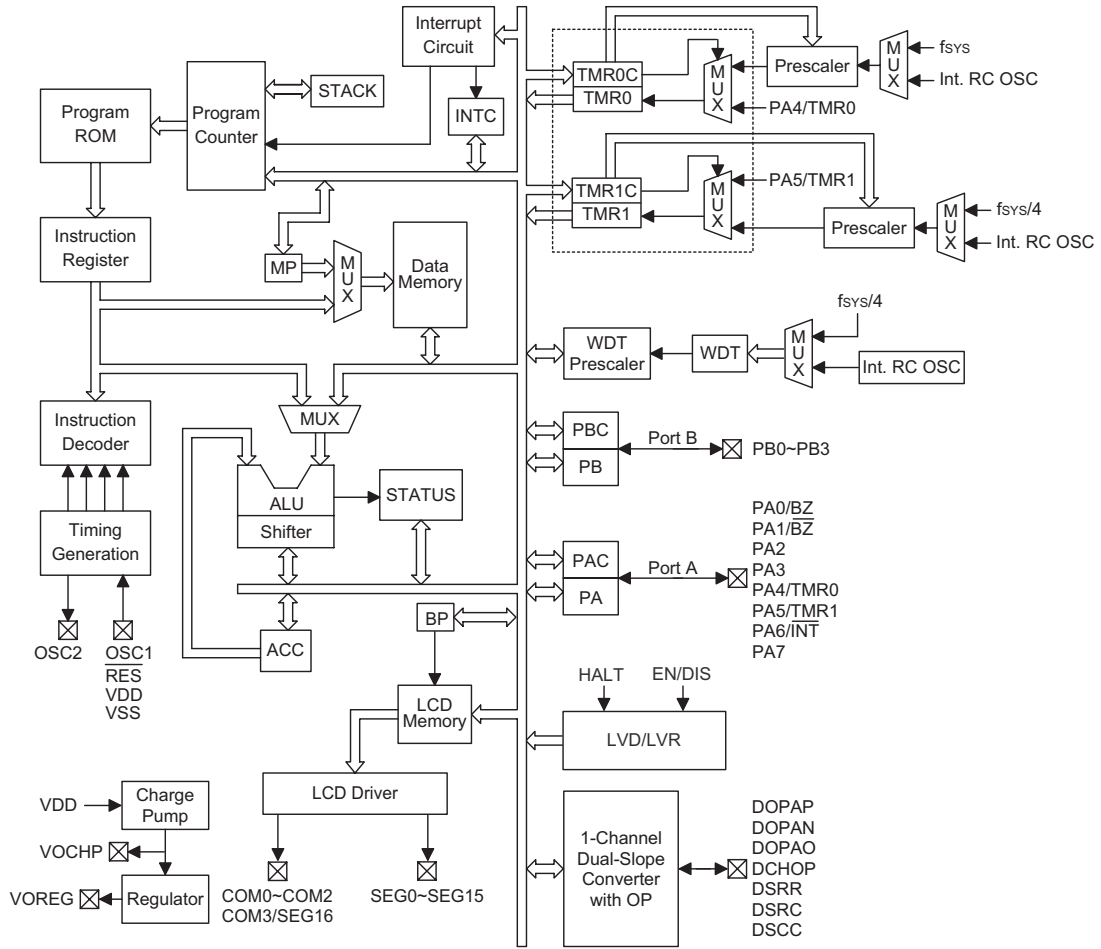
- Operating voltage:  
f<sub>SYS</sub>=4MHz: 2.2V~5.5V  
f<sub>SYS</sub>=8MHz: 3.3V~5.5V
- 12 bidirectional I/O lines and two ADC input
- One external interrupt input shard with an I/O lines
- One 8-bit and one 16-bit programmable timer/event counter with overflow interrupt a 8-stage pre-scaler
- LCD driver with 16×4, 17×3 or 17×2 segments
- 2K×15 program memory with partial lock function
- 96×8 data memory RAM
- Single differential input channel dual slope Analog to Digital Converter with Operational Amplifier.
- Watchdog Timer with regulator power
- Buzzer output
- Internal 12kHz RC oscillator
- On-chip RC or crystal oscillator
- HALT function and wake-up feature reduce power consumption
- Voltage regulator (3.3V) and charge pump
- Embedded voltage reference generator (1.5V)
- 4-level subroutine nesting
- Bit manipulation instruction
- 14-bit table read instruction
- Up to 0.5μs instruction cycle with 8MHz system clock at V<sub>DD</sub>=5V
- 63 powerful instructions
- All instructions in 1 or 2 machine cycles
- Low voltage reset/detector function
- 52-pin QFP package

**General Description**

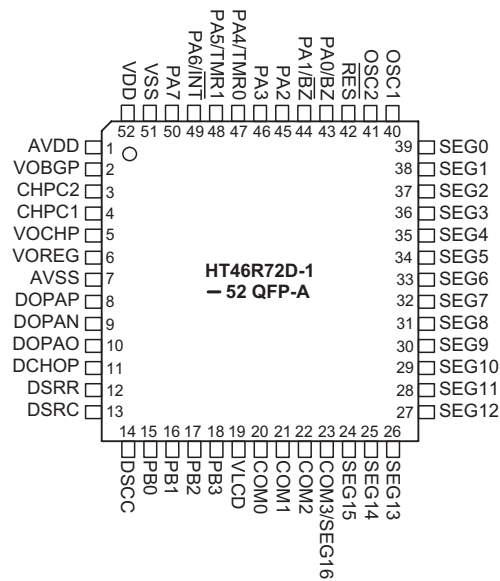
The HT46R72D-1 is an 8-bit high performance, RISC architecture microcontroller device specifically designed for A/D with LCD applications that interface directly to analog signals, such as those from sensors. The advantages of low power consumption, I/O flexibility, timer functions, oscillator options, Dual slope A/D

converter, LCD display, HALT and wake-up functions, watchdog timer, as well as low cost, enhance the versatility of these devices to suit for a wide range of AD with LCD application possibilities such as sensor signal processing, scales, consumer products, subsystem controllers, etc.

**Block Diagram**



**Pin Assignment**



**Pin Description**

Pin Name	I/O	Options	Description
PA0/BZ PA1/BZ PA2 PA3 PA4/TMR0 PA5/TMR1 PA6/INT PA7	I/O	Wake-up Pull-high Buzzer	Bidirectional 8-bit input/output port. Each individual bit on this port can be configured to have a wake-up function using a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pins on this port have pull-high resistors. The BZ, $\overline{\text{BZ}}$ , TMR0, TMR1 and INT are pin-shared with PA0, PA1, PA4, PA5 and PA6 respectively.
PB0~PB3	I/O	Pull-high	Bidirectional 4-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pins on this port have pull-high resistors.
VLCD	I	—	LCD power supply
COM0~COM2 COM3/SEG16	O	1/2, 1/3 or 1/4 Duty	COM0~COM3 are the LCD common outputs. An LCD duty-cycle configuration option determines. When 1/3, 1/2 duty is selected, the COM3/SEG16 is configured as SEG16.
SEG0~SEG15	O	Segment Output	LCD driver outputs for the LCD panel segments.
VOBGP	AO	—	Band gap voltage output pin. (for internal use)
VOREG	O	—	Regulator output 3.3V
VOCHP	O	—	Charge pump output (a capacitor is required to be connected)
CHPC1	—	—	Charge pump capacitor, positive
CHPC2	—	—	Charge pump capacitor, negative
DOPAN, DOPAP, DOPAO, DCHOP	AI/AO	—	Dual Slope converter pre-stage OPA related pins. DOPAN is the OPA Negative input pin, DOPAP is the OPA Positive input pin, DOPAO is the OPA output pin and DCHOP is the OPA Chopper pins.
DSRR, DSRC, DSCC	AI/AO	—	Dual slope AD converter main function RC circuit. DSRR is the input or reference signal, DSRC is the Integrator negative input, and DSCC is the comparator negative input.
OSC1 OSC2	I O	Crystal or RC	OSC1, OSC2 are connected to an external RC network or crystal for the internal system clock. The OSC2 pin can be used to monitor the system clock at 1/4 frequency.
$\overline{\text{RES}}$	I	—	Schmitt trigger reset input, active low
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground
AVSS	—	—	Analog negative power supply, ground
AVDD	—	—	Analog positive power supply

**Absolute Maximum Ratings**

Supply Voltage .....	$V_{SS}-0.3V$ to $V_{SS}+6.0V$	Storage Temperature .....	$-50^{\circ}\text{C}$ to $125^{\circ}\text{C}$
Input Voltage .....	$V_{SS}-0.3V$ to $V_{DD}+0.3V$	Operating Temperature .....	$-40^{\circ}\text{C}$ to $85^{\circ}\text{C}$
$I_{OL}$ Total .....	150mA	$I_{OH}$ Total .....	$-100\text{mA}$
Total Power Dissipation .....	500mW		

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**

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	f <sub>SYS</sub> =4MHz	2.2	—	5.5	V
		—	f <sub>SYS</sub> =8MHz	3.3	—	5.5	V
I <sub>DD1</sub>	Operating Current (Crystal)	5V	No load, f <sub>SYS</sub> =8MHz, analog block off	—	4	8	mA
I <sub>DD2</sub>	Operating Current (Crystal OSC or RC OSC)	3V	No load, f <sub>SYS</sub> =4MHz, ADC block off	—	0.8	1.5	mA
		5V	ADC block off	—	2.5	4	mA
I <sub>DD3</sub>	Operating Current (Crystal OSC or RC OSC)	3V	No load, f <sub>SYS</sub> =2MHz, ADC block off	—	0.5	1	mA
		5V	ADC block off	—	1.5	3	mA
I <sub>DD4</sub>	Operating Current (Crystal OSC or RC OSC)	5V	V <sub>REGO</sub> =3.3V, f <sub>SYS</sub> =4MHz, ADC on, ADCCCLK=125kHz (all other analog devices off)	—	3	5	mA
I <sub>STB1</sub>	Standby Current (WDT Disable)	3V	No load, system HALT, LCD off at HALT	—	—	1	μA
		5V	LCD off at HALT	—	—	2	μA
I <sub>STB2</sub>	Standby Current (WDT Enable)	3V	No load, system HALT, LCD off at HALT, ADC off	—	2.5	5	μA
		5V	LCD off at HALT, ADC off	—	8	15	μA
I <sub>STB3</sub>	Standby Current (WDT Disable Internal RC 12kHz OSC ON)	3V	No load, system HALT, LCD off at HALT, ADC off	—	2	5	μA
		5V	LCD off at HALT, ADC off	—	6	10	μA
I <sub>STB4</sub>	Standby Current (WDT Disable Internal RC 12kHz OSC ON)	3V	No load, system HALT, LCD on at HALT, 1/2 bias, V <sub>LCD</sub> =V <sub>DD</sub>	—	17	30	μA
		5V	V <sub>LCD</sub> =V <sub>DD</sub>	—	34	60	μA
I <sub>STB5</sub>	Standby Current (WDT Disable Internal RC 12kHz OSC ON)	3V	No load, system HALT LCD on at HALT, 1/3 bias, V <sub>LCD</sub> =V <sub>DD</sub>	—	13	25	μA
		5V	V <sub>LCD</sub> =V <sub>DD</sub>	—	28	50	μA
V <sub>IL1</sub>	Input Low Voltage for I/O Ports, TMR0, TMR1 and INT	—	—	0	—	0.3V <sub>DD</sub>	V
V <sub>IH1</sub>	Input High Voltage for I/O Ports, TMR0, TMR1 and INT	—	—	0.7V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>IL2</sub>	Input Low Voltage (RES)	—	—	0	—	0.4V <sub>DD</sub>	V
V <sub>IH2</sub>	Input High Voltage (RES)	—	—	0.9V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>LCD</sub>	LCD Highest Voltage	—	—	0	—	V <sub>DD</sub>	V
V <sub>LVR</sub>	Low Voltage Reset	—	—	2	2.1	2.2	V
V <sub>LVD</sub>	Low Voltage Detector	—	—	2.2	2.3	2.4	V
I <sub>OL1</sub>	I/O Port Segment Logic Output Sink Current	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	4	8	—	mA
		5V		10	20	—	mA
I <sub>OH1</sub>	I/O Port Segment Logic Output Source Current	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-2	-4	—	mA
		5V		-5	-10	—	mA
I <sub>OL2</sub>	LCD Common and Segment Current	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	210	420	—	μA
		5V		350	700	—	μA
I <sub>OH2</sub>	LCD Common and Segment Current	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-80	-160	—	μA
		5V		-180	-360	—	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
R <sub>PH</sub>	Pull-high Resistance of I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
<b>Charge Pump and Regulator</b>							
V <sub>CHPI</sub>	Input Voltage	—	Charge pump on	2.2	—	3.6	V
			Charge pump off	3.7	—	5.5	V
V <sub>REGO</sub>	Output Voltage	—	No load	3	3.3	3.6	V
V <sub>REGDP1</sub>	Regulator Output Voltage Drop (Compare with No Load)	—	V <sub>DD</sub> =3.7V~5.5V Charge pump off Current≤10mA	—	100	—	mV
V <sub>REGDP2</sub>			V <sub>DD</sub> =2.4V~3.6V Charge pump on Current≤6mA	—	100	—	mV
<b>Dual Slope AD, Amplifier and Band Gap</b>							
V <sub>RFGO</sub>	Reference Generator Output	—	@3.3V	1.45	1.5	1.55	V
V <sub>RFGTC</sub>	Reference Generator Temperature Coefficient	—	@3.3V	—	50	—	Ppm/C
V <sub>ICMR</sub>	Common Mode Input Range	—	Amplifier, no load	0.2	—	V <sub>REGO</sub> -1	V
			Integrator, no load	1	—	V <sub>REGO</sub> -0.2	V
V <sub>ADOFF</sub>	Input Offset Range	—	—	—	500	800	μV

**A.C. Characteristics**

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
f <sub>SYS</sub>	System Clock (RC OSC)	—	2.2V~5.5V	400	—	4000	kHz
	System Clock (Crystal OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f <sub>INRC</sub>	Internal RC OSC	3V	—	—	12	—	kHz
		5V		—	15	—	kHz
f <sub>TIMER</sub>	Timer I/P Frequency (TMR0/TMR1)	—	2.2V~5.5V	0	—	4000	kHz
t <sub>WDTOSC</sub>	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t <sub>RES</sub>	External Reset Low Pulse Width	—	—	1	—	—	μs
t <sub>SST</sub>	System Start-up Timer Period	—	Power-up or wake-up from HALT	—	1024	—	t <sub>sys</sub>
t <sub>LVR</sub>	Low Voltage Width to Reset	—	—	0.25	1	2	ms
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 external 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 makes it possible for each instruction to be effectively executed 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 11 bits wide and it controls the sequence in which the instructions stored in the program ROM are executed. The contents of the PC can specify a maximum of 2048 addresses.

After accessing a program memory word to fetch an instruction code, the value of the PC is incremented by 1.

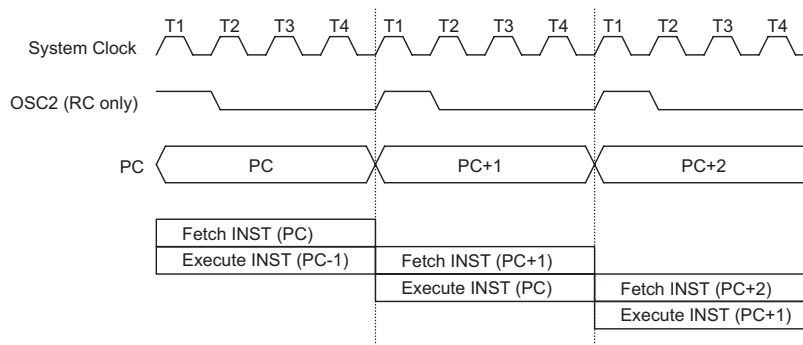
The PC then points to the memory word containing the next instruction code.

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 to 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.



**Execution Flow**

Mode	Program Counter										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0
External Interrupt	0	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 1 Overflow	0	0	0	0	0	0	0	1	1	0	0
ADC Interrupt	0	0	0	0	0	0	1	0	0	0	0
Skip	Program Counter+2										
Loading PCL	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return From Subroutine	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

**Program Counter**

Note: \*10~\*0: Program counter bits  
#10~#0: Instruction code bits

S10~S0: Stack register bits  
@7~@0: PCL bits

**Program Memory – EPROM**

The program memory (EPROM) is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 2048×15 bits which are addressed by the program counter and table pointer.

Certain locations in the ROM are reserved for special usage:

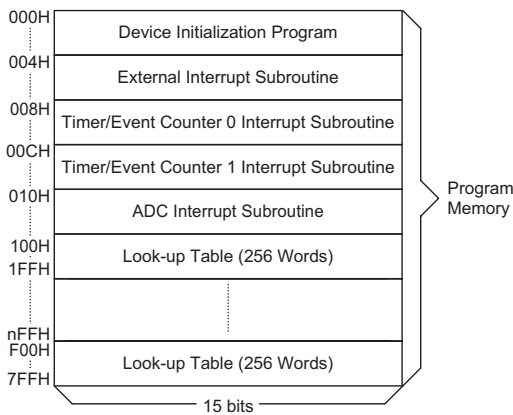
- Location 000H  
Location 000H is reserved for program initialization. After chip reset, the program always begins execution at this location.
- Location 004H  
Location 004H is reserved for the external interrupt service program. If the INT input pin is activated, and the interrupt is enabled, and the stack is not full, the program begins execution at location 004H.
- Location 008H  
Location 008H is reserved for the 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 008H.
- Location 00CH  
Location 00CH is reserved for the Timer/Event Counter 1 interrupt service program. If a timer interrupt results from a Timer/Event Counter 1 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 ADC interrupt service program. If an ADC interrupt occurs, and if the interrupt is enabled and the stack is not full, the program begins execution at this location.
- 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. 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 program counter. The stack is organized into 4 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 the start of a subroutine call or an interrupt acknowledgment, the contents of the program counter 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 program counter 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 4 return addresses are stored).



Note: n ranges from 1 to 6

**Program Memory**

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

**Table Location**

Note: \*10~\*0: Table location bits  
@7~@0: Table pointer bits

P10~P8: Current program counter bits

### Data Memory – RAM

Bank 0 of the data memory has a capacity of 123×8 bits, and is divided into two functional groups, namely the special function registers of 27×8 bit capacity and the general purpose data memory of 96×8 bit capacity. Most locations are readable/writable, although some are read only. The special function register are overlapped in all banks.

Any unused space before 20H is reserved for future expanded usage, reading these locations will get "00H". The general purpose data memory, addressed from 20H to 7FH, is used for data and control information under instruction commands. All of the data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by the "SET [m].i" and "CLR [m].i" instructions. They are also indirectly accessible through the memory pointer registers, MP0 and MP1.

00H	Indirect Addressing Register 0
01H	MP0
02H	Indirect Addressing Register 1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	MODE
0AH	STATUS
0BH	INTC0
0CH	
0DH	TMR0
0EH	TMR0C
0FH	TMR1H
10H	TMR1L
11H	TMR1C
12H	PA
13H	PAC
14H	PB
15H	PBC
16H	
17H	
18H	ADCR
19H	Reserved
1AH	ADCD
1BH	
1CH	WDTC
1DH	WTD
1EH	INTC1
1FH	CHPRC
20H	General Purpose Data Memory (96 Bytes)
21H	
22H	
23H	
24H	
25H	
26H	
27H	
28H	
29H	
2AH	
2BH	
2CH	
2DH	
2EH	
2FH	
30H	
31H	
32H	
33H	
34H	
35H	
36H	
37H	
38H	
39H	
3AH	
3BH	
3CH	
3DH	
3EH	
3FH	
40H	
41H	
42H	
43H	
44H	
45H	
46H	
47H	
48H	
49H	
4AH	
4BH	
4CH	
4DH	
4EH	
4FH	
50H	
51H	
52H	
53H	
54H	
55H	
56H	
57H	
58H	
59H	
5AH	
5BH	
5CH	
5DH	
5EH	
5FH	
60H	
61H	
62H	
63H	
64H	
65H	
66H	
67H	
68H	
69H	
6AH	
6BH	
6CH	
6DH	
6EH	
6FH	
70H	
71H	
72H	
73H	
74H	
75H	
76H	
77H	
78H	
79H	
7AH	
7BH	
7CH	
7DH	
7EH	
7FH	

**RAM Mapping**

Bank 1 contains the LCD Data Memory locations. After first setting up BP to the value of "01H" to access Bank 1 this bank must then be accessed indirectly using the Memory Pointer MP1. With BP set to a value of "01H", using MP1 to indirectly read or write to the data memory areas with addresses from 40H~50H will result in operations to Bank 1. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of BP.

### 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 memory pointer register (MP0, MP1) are 7-bit registers.

The function of data movement between two indirect addressing registers is not supported. The memory pointer registers, MP0 and MP1, are both 7-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 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 (PDF), and a watchdog time-out flag (TO). It also records the status information and controls the operation sequence.

Except for the TO and PDF 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 PDF flags. Operations related to the status

Bit No.	Label	Function
0	C	C is set if an 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.
1	AC	AC is set if an 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.
2	Z	Z is set if the result of an arithmetic or logic operation is zero; otherwise Z is cleared.
3	OV	OV is set if an 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.
4	PDF	PDF is cleared by either a system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction.
5	TO	TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.
6~7	—	Unused bit, read as "0"

#### Status (0AH) Register

register, however, may yield different results from those intended. The TO and PDF 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 device provides one external interrupts, two internal timer/event counter interrupts and the ADC 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.

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

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1=enabled; 0=disabled)
1	E EI	Controls the external interrupt (1=enabled; 0=disabled)
2	ET0I	Controls the Timer/Event Counter 0 interrupt (1=enabled; 0=disabled)
3	ET1I	Controls the Timer/Event Counter 1 interrupt (1=enabled; 0=disabled)
4	EIF	External interrupt request flag (1=active; 0=inactive)
5	T0F	Internal Timer/Event Counter 0 request flag (1=active; 0=inactive)
6	T1F	Internal Timer/Event Counter 1 request flag (1=active; 0=inactive)
7	—	For test mode used only. Must be written as "0"; otherwise may result in unpredictable operation.

#### INTC 0 (0BH) Register

Bit No.	Label	Function
0	EADI	Controls the ADC interrupt (1=enabled; 0:disabled)
1~3, 5~7	—	Unused bit, read as "0"
4	ADF	ADC request flag (1=active; 0=inactive)

#### INTC 1 (1EH) Register

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 interrupts will provide a wake-up function. As an interrupt is serviced, a control transfer occurs by pushing the contents of the program counter onto the stack followed by a branch to a subroutine at the specified location in the Program Memory. Only the contents of the program counter is pushed onto the stack. If the contents of the register or of the status register is altered by the interrupt service program which corrupts the desired control sequence, the contents should be saved in advance.

An external interrupt is triggered by an edge transition on INT (A configuration option selects: high to low, low to high, both low to high and high to low), and the related interrupt request flag (EIF; bit 4 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 occurs. The interrupt request flag (EIF) and EMI bits are all cleared to disable other maskable interrupts.

The internal Timer/Event Counter 0 interrupt is initialized by setting the Timer/Event Counter 0 interrupt request flag (T0F; bit 5 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 08H occurs. The related interrupt request flag (T0F) is reset, and the EMI bit is cleared to disable other maskable interrupts. Timer/Event Counter 1 is operated in the same manner but its related interrupt request flag is T1F (bit 6 of INTC0) and its subroutine call location is 0CH.

The A/D Converter interrupt is initialized by setting the A/D Converter clock interrupt request flag (ADF; bit 4 of INTC1), that is caused by an A/D conversion done signal. After the interrupt is enabled, and the stack is not full, and the ADF bit is set, a subroutine call to location 10H occurs. The related interrupt request flag (ADF) is reset and the EMI bit is cleared to disable further maskable interrupts.

During the execution of an interrupt subroutine, other maskable 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.

Interrupt Source	Priority	Vector
External interrupt	1	04H
Timer/Event Counter 0 overflow	2	08H
Timer/Event Counter 1 overflow	3	0CH
ADC interrupt	4	10H

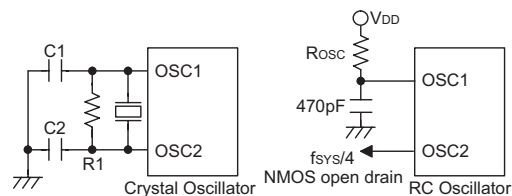
Once the interrupt request flags (ADF, T0F, T1F, EIF) are all set, they remain in the INTC1 or INTC0 respectively until the interrupts are serviced or cleared by a software instruction.

It is recommended that a program should 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. During that period, 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 device provides three oscillator circuits, an external RC oscillator, a crystal oscillator and an internal RC 12kHz oscillator (Int. RCOSC). The external RC oscillator and a crystal oscillator signal are used for the system clock and the internal 12kHz RC oscillator is designed for timing purposes.

In the IDLE mode, the IRC clock source is enabled (IRCC=0), the system oscillator stop running, but the internal RC oscillator (Int.RCOSC) still continuously keep free running. In HALT mode, if the WDT is disabled, the IRC clock source is disabled (IRCC=1), both the system oscillator and internal RC oscillator stop running, but if the WDT is enabled, the internal RC oscillator will always keep free running. The system can be woken-up from either the IDLE or HALT mode by the occurrence of an interrupt, a high to low transition on any of the Port A pins, a WDT overflow or a timer overflow and request flag is set (0→1).



**System Oscillator**

If the crystal oscillator is used, a crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for the oscillator. No other external components are required. Instead of a crystal, a resonator can also be connected between OSC1 and OSC2 to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required (if the oscillator can be disabled by options to conserve power).

If an external RC oscillator is used, an external resistor between OSC1 and VSS is required to achieve oscillation, the value of which must be between 100kΩ to 2.4MΩ. The system clock divided by 4, which can be monitored on pin OSC2, can be used for external logic synchronization purposes.

The Internal RC oscillator (Int.RCOSC) is a free running on-chip RC oscillator, requiring no external components. Even if the system enters the Power Down Mode, and the system clock is stopped, the internal RC oscillator continues to run with a period of approximately 65μs at 5V if either the WDT or IRC clock is enabled. The internal RC oscillator can be disabled by a configuration option and by setting the IRCC bit to "1" to conserve power.

#### Watchdog Timer – WDT

The WDT is implemented using a dedicated internal RC oscillator (Int. RCOSC, note: the WDTOSC described in this document represents the same oscillator as the Int.RCOSC) or the instruction clock which is the system

clock/4. The timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The watchdog timer can be disabled by a configuration option. If the watchdog timer is disabled, the WDT timer will have the same manner as in the enable-mode except that the timeout signal will not generate a chip reset. So in the watchdog timer disable mode, the WDT timer counter can be read out and can be cleared. This function is used for the application program to access the WDT frequency to get the temperature coefficient for analog component adjustment. The WDT oscillator needs to be disabled/enabled by the special function registers(WDTC :WDTOSC), for power saving reasons.

There are 2 registers related to the WDT function, WDTC and WDTD. The WDTC register can control the WDT oscillator enable/disable and the WDT power source. WDTD is the WDT counter readout register.

WDT\_PWR can be used to choose the WDT power source, the default source is VOCHP. The main purpose of the regulator is to be used for the WDT Temperature-coefficient adjustment. In this case, the application program should enable the regulator before switching to the Regulator source. The WDTOSC can be used to enable or disable the WDT OSC (12kHz). If the application does not use the WDT OSC, then it needs to disable it in order to save power. When WDTOSC is disabled, then it is actually turned off, regardless of the IRCC setting. When the WDTOSC is enabled, the Power Down mode situation will be defined by the IRCC registers.

Bit No.	Label	Function
0~1	WDT_PWR0~WDT_PWR1	The WDT Power source selection. (WDT_PWR1:0)= 01: WDT power comes from VOCHP 10: WDT power comes from regulator 00/11: WDT power comes from VOCHP strongly recommend use to use 01 for VOCHP prevent the noise to let the WDT lose the power
2~3	WDTOSC0~WDTOSC1	The WDT oscillator enable/disable (WDTOSC1:0)= 01: WDT OSC disable 10: WDT OSC enable 00/11: WDT OSC enable strongly recommend use to use 10 for WDT OSC enable
4~7	—	Reserved

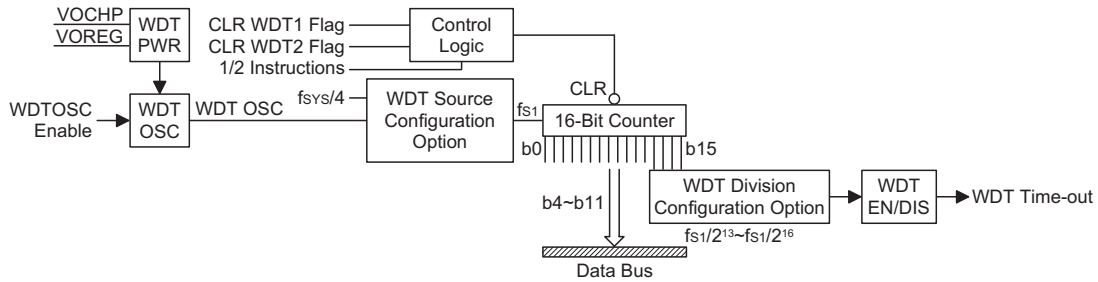
#### WDTC (1CH) Register

Note: WDTOSC registers initial value will be set to enable (1,0), if both "WDT option enable" and "WDT clock option set to WDT", otherwise, it will be set to disable (0,1)

Bit No.	Label	Function
0~7	WDTD0~WDTD7	The WDT counter data value. This register is read only. It's used for temperature adjusting.

#### WDTD (1DH) Register

The WDT clock ( $f_{S1}$ ) is further divided by an internal counter to give longer watchdog time-outs. In this device, the division ratio can be varied by selecting different configuration options to give  $2^{13}$  to  $2^{16}$  division ration range.



Watchdog Timer

Once an internal RC oscillator (Int.RC oscillator with period 65µs normally) is selected, it is divided by  $Max 2^{16}$  to get the time-out period of approximately 4.3s. This time-out period may vary with temperature, VDD and process variations.

The WDT clock source may also come from the instruction clock, in which case the WDT will operate in the same manner except that in the Power Down mode the WDT may stop counting and lose its protecting purpose. In this situation the device can only be restarted by external logic. If the device operates in a noisy environment, using the on-chip RC oscillator (Int.RC OSC) is strongly recommended, since the HALT instruction will stop the system clock.

The WDT overflow under normal operation initializes a "chip reset" and sets the status bit "TO". In the HALT or IDLE mode, the overflow initializes a "warm reset", and only the PC and SP are reset to zero. There are three methods to clear the contents of the WDT, an external reset (a low level on  $\overline{RES}$ ), a software instruction or a "HALT" instruction. There are two types of software instructions; the single "CLR WDT" instruction, or the pair of instructions — "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 configuration 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. If the "CLR WDT1" and "CLR WDT2" option is 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 a time-out.

**Buzzer Output**

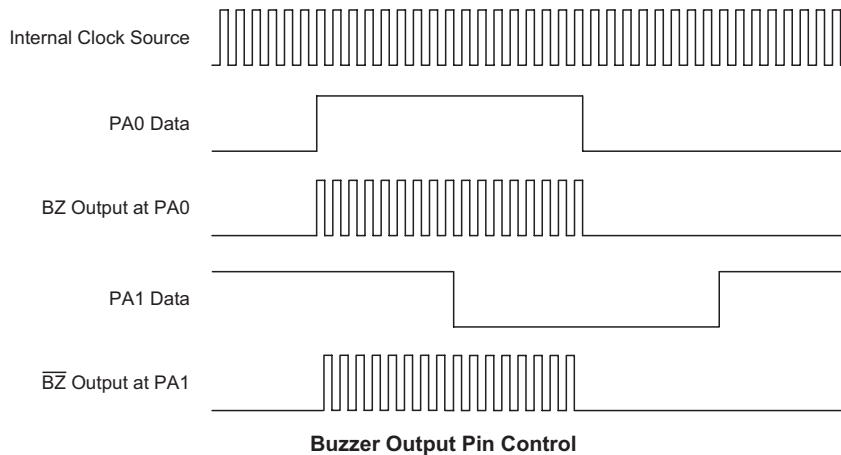
The Buzzer function provides a means of producing a variable frequency output, suitable for applications such as Piezo-buzzer driving or other external circuits that require a precise frequency generator. The BZ and  $\overline{BZ}$  pins form a complimentary pair, and are pin-shared with I/O pins, PA0 and PA1. A configuration option is used to select from one of three buzzer options. The first option is for both pins PA0 and PA1 to be used as normal I/Os, the second option is for both pins to be configured as BZ and  $\overline{BZ}$  buzzer pins, the third option selects only the PA0 pin to be used as a BZ buzzer pin with the PA1 pin retaining its normal I/O pin function. Note that the  $\overline{BZ}$  pin is the inverse of the BZ pin which together generate a differential output which can supply more power to connected interfaces such as buzzers.

The buzzer is driven by the internal clock source,  $f_S$ , which then passes through a divider, the division ratio of which is selected by configuration options to provide a range of buzzer frequencies from  $f_S/2^2$  to  $f_S/2^9$ .

PAC Register PAC.0	PAC Register PAC.1	PA data Register PA.0	PA data Register PA.1	Output Function
0	0	0	X	PA0=0, PA1=0
0	0	1	X	PA0=BZ, PA1= $\overline{BZ}$
0	1	0	X	PA0=0, PA1=Input
0	1	1	X	PA0=BZ, PA1=Input
1	0	0	X	PA0=Input, PA1=0
1	1	X	X	PA0=Input, PA1=Input

PA0/PA1 Pin Function Control

Note: "X" stands for don't care



The clock source that generates  $f_s$ , which in turn controls the buzzer frequency, can originate from two different sources, the Int.RCOSC (Internal RC oscillator) or the System oscillator/4, the choice of which is determined by the  $f_s$  clock source configuration option. Note that the buzzer frequency is controlled by configuration options, which select both the source clock for the internal clock  $f_s$  and the internal division ratio. There are no internal registers associated with the buzzer frequency.

If the configuration options have selected both pins PA0 and PA1 to function as a BZ and  $\overline{BZ}$  complementary pair of buzzer outputs, then for correct buzzer operation it is essential that both pins must be setup as outputs by setting bits PAC0 and PAC1 of the PAC port control register to zero. The PA0 data bit in the PA data register must also be set high to enable the buzzer outputs, if set low, both pins PA0 and PA1 will remain low. In this way the single bit PA0 of the PA register can be used as an on/off control for both the BZ and  $\overline{BZ}$  buzzer pin outputs. Note that the PA1 data bit in the PA register has no control over the  $\overline{BZ}$  buzzer pin PA1.

If configuration options have selected that only the PA0 pin is to function as a BZ buzzer pin, then the PA1 pin can be used as a normal I/O pin. For the PA0 pin to function as a BZ buzzer pin, PA0 must be setup as an output by setting bit PAC0 of the PAC port control register to zero. The PA0 data bit in the PA data register must also be set high to enable the buzzer output, if set low pin PA0 will remain low. In this way the PA0 bit can be used as an on/off control for the BZ buzzer pin PA0. If the PAC0 bit of the PAC port control register is set high, then pin PA0 can still be used as an input even though the configuration option has configured it as a BZ buzzer output.

Note that no matter what configuration option is chosen for the buzzer, if the port control register has setup the pin to function as an input, then this will override the configuration option selection and force the pin to always behave as an input pin. This arrangement enables the

pin to be used as both a buzzer pin and as an input pin, so regardless of the configuration option chosen; the actual function of the pin can be changed dynamically by the application program by programming the appropriate port control register bit.

Note: The above drawing shows the situation where both pins PA0 and PA1 are selected by configuration option to be BZ and  $\overline{BZ}$  buzzer pin outputs. The Port Control Register of both pins must have already been setup as outputs. The data setup on pin PA1 has no effect on the buzzer outputs.

**Power Down Operation – HALT**

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

- The system oscillator turns off but the Internal oscillator (Int.RCOSC) keeps running (if the Internal oscillator is selected).
- The contents of the on-chip RAM and of the registers remain unchanged.
- The WDT is cleared and starts recounting (if the WDT clock source is from the Internal RC oscillator).
- All I/O ports maintain their original status.
- The PDF flag is set but the TO flag is cleared.
- The LCD driver keeps running if the IRC clock is enabled; IRCC=0 and LCD on HALT mode set to ON.

The system leaves the HALT or IDLE mode by means of an external reset, an interrupt, an external falling edge signal on port A, or a WDT overflow. An external reset causes device initialisation, and the WDT overflow performs a "warm reset". After examining the TO and PDF flags, the reason for chip reset can be determined. The PDF 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 program counter and SP, and leaves the others in their original state.

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each pin of port A can be independently selected to wake-up the device using configuration options. After awakening from an I/O port stimulus, the program will resume execution at the next instruction. However, if 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 will resume 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 awakened using that interrupt.

If a wake-up events occur, it takes 1024  $t_{SYS}$  (system clock periods) 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 a 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 or IDLE differs from other chip reset conditions, for it can perform a "warm reset" that resets only the program counter 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 their initial conditions once the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between different chip resets.

TO	PDF	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

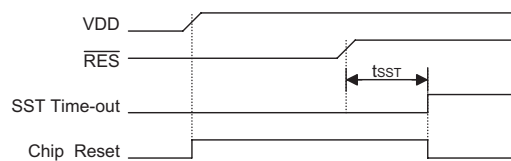
Note: "u" stands for unchanged

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. Awakening from the HALT state or system power-up, the SST delay is added.

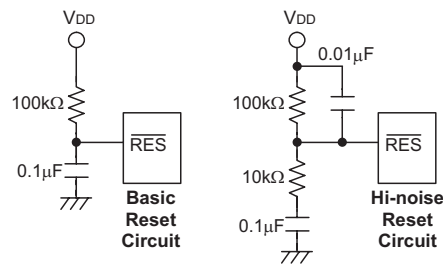
An extra SST delay is added during the power-up period, and any wake-up from HALT may enable only the SST delay.

The functional unit chip reset status is shown below.

Program Counter	000H
Interrupt	Disabled
Prescaler, Divider	Cleared
WDT	Cleared. After master reset, WDT starts counting
Timer/Event Counter	Off
Input/output Ports	Input mode
Stack Pointer	Points to the top of the stack

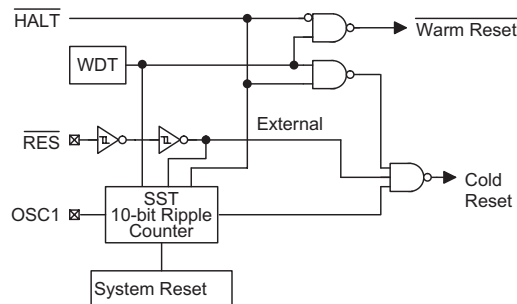


**Reset Timing Chart**



**Reset Circuit**

Note: Most applications can use the Basic Reset Circuit as shown, however for applications with extensive noise, it is recommended to use the Hi-noise Reset Circuit.



**Reset Configuration**

The register states are summarized below:

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (HALT)	WDT Time-out (HALT)*
MP0	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu
MP1	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu
BP	---- --0	---- --0	---- --0	---- --0	---- --u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	0000H	0000H	0000H	0000H	0000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
MODE	--0- 01--	--0- 01--	--0- 01--	--0- 01--	--u- uu--
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PBC	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
ADCR	0000 x000	0000 x000	0000 x000	0000 x000	uuuu xuuu
ADCD	---- -111	---- -111	---- -111	---- -111	---- -uuu
WDTC	---- ss01	---- ss01	---- ss01	---- ss01	---- uuuu
WDTD	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
INTC1	---0 ---0	---0 ---0	---0 ---0	---0 ---0	---u ---u
CHPRC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

Note: "\*" stands for warm reset

"u" stands for unchanged

"x" stands for unknown

"s" for special case, it depends on the option table (please see the WDT chapter for the detail)

**Timer/Event Counter**

Two timer/event counters are implemented in the microcontroller. Timer/Event Counter 0 contains an 8-bit programmable count-up counter whose clock may come from an external source or an internal clock source. An internal clock source comes from  $f_{SYS}$  or the Internal RC. Timer/Event Counter 1 contains a 16-bit programmable count-up counter whose clock may come from an external source or an internal clock source. An internal clock source comes from  $f_{SYS}/4$  or Internal RC selected by a special function register 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.

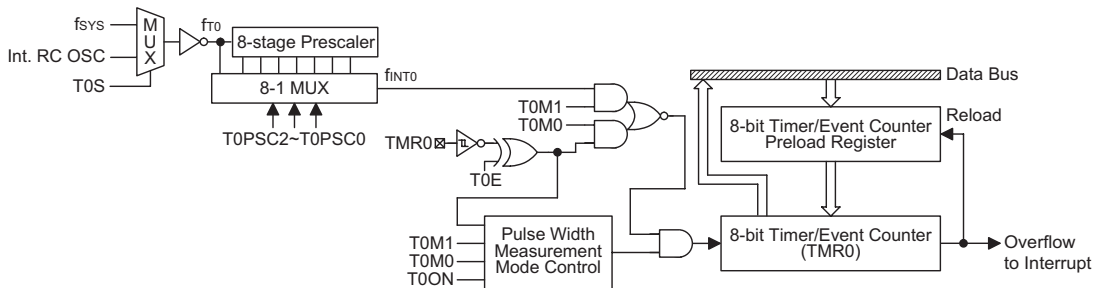
There are two registers related to the Timer/Event Counter 0; TMR0 ([0DH]) and TMR0C ([0EH]). Writing to TMR0 puts the starting value in the Timer/Event Counter 0 register and reading TMR0 reads out the contents of Timer/Event Counter 0. The TMR0C is a timer/event counter control register, which defines some options. There are three registers related to the Timer/Event Counter 1; TMR1H (0FH), TMR1L (10H) and TMR1C (11H). Writing to TMR1L will only put the written data into an internal lower-order byte buffer (8-bit) while writing to TMR1H will transfer the specified data and the contents of the lower-order byte buffer to both the TMR1H and TMR1L registers, respectively.

The Timer/Event Counter 1 preload register is changed every time there is a write operation to TMR1H. Reading TMR1H will latch the contents of TMR1H and TMR1L counters to the destination and the lower-order byte buffer, respectively. Reading TMR1L will read the con-

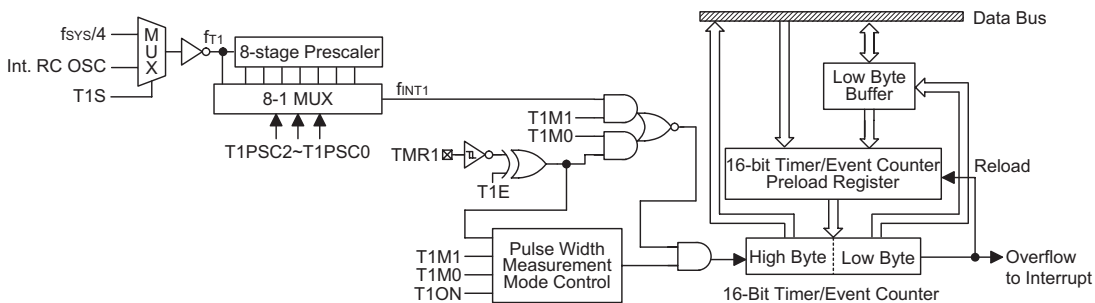
tents of the lower-order byte buffer. TMR1C is the Timer/Event Counter 1 control register, which defines the operating mode, counting enable or disable and an active edge.

The T0M0, T0M1 (TMR0C) and T1M0, T1M1 (TMR1C) bits define the operation mode. The event count mode is used to count external events, which means that the clock source must come 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 a high or low level duration of an external signal on TMR0 or TMR1, with the timing based on the internal selected clock source.

In the event count or timer mode, the Timer/Event Counter 0 (1) starts counting at the current contents in the Timer/Event Counter 0 (1) and ends at FFH (FFFFH). Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag (T0F; bit 5 of INTC0, T1F; bit6 of INTC0). In the pulse width measurement mode with the values of the T0ON/T1ON and T0E/T1E bits equal to 1, 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) pin returns to the original level and resets the T0ON/T1ON bit. The measured result remains in the timer/event counter even if the activated transient occurs again. Therefore, only a 1-cycle measurement can be made until the T0ON/T1ON bit is again set. The cycle measurement will re-function as long as it receives further transient pulses. In this operation mode, the timer/event counter



**Timer/Event Counter 0**



**Timer/Event Counter 1**

Bit No.	Label	Function
0 1 2	T0PSC0 T0PSC1 T0PSC2	To define the prescaler stages, T0PSC2, T0PSC1, T0PSC0= 000: $f_{INT0}=f_{T0}$ 001: $f_{INT0}=f_{T0}/2$ 010: $f_{INT0}=f_{T0}/4$ 011: $f_{INT0}=f_{T0}/8$ 100: $f_{INT0}=f_{T0}/16$ 101: $f_{INT0}=f_{T0}/32$ 110: $f_{INT0}=f_{T0}/64$ 111: $f_{INT0}=f_{T0}/128$
3	T0E	Defines the TMR0 active edge of the timer/event counter: In Event Counter Mode (T0M1,T0M0)=(0,1): 1:count on falling edge; 0:count on rising edge In Pulse Width measurement mode (T0M1,T0M0)=(1,1): 1: start counting on the rising edge, stop on the falling edge; 0: start counting on the falling edge, stop on the rising edge
4	T0ON	Enable/disable timer counting (0=disabled; 1=enabled)
5	T0S	Defines the TMR0 internal clock source (0= $f_{SYS}$ ; 1=Int.RCOSC (Internal RC OSC))
6 7	T0M0 T0M1	Defines the operating mode T0M1, T0M0= 01=Event count mode (External clock) 10=Timer mode (Internal clock) 11=Pulse Width measurement mode (External clock) 00=Unused

**TMR0C (0EH) Register**

Bit No.	Label	Function
0 1 2	T1PSC0 T1PSC1 T1PSC2	To define the prescaler stages, T1PSC2, T1PSC1, T1PSC0= 000: $f_{INT1}=f_{T1}$ 001: $f_{INT1}=f_{T1}/2$ 010: $f_{INT1}=f_{T1}/4$ 011: $f_{INT1}=f_{T1}/8$ 100: $f_{INT1}=f_{T1}/16$ 101: $f_{INT1}=f_{T1}/32$ 110: $f_{INT1}=f_{T1}/64$ 111: $f_{INT1}=f_{T1}/128$
3	T1E	Defines the TMR1 active edge of the timer/event counter: In Event Counter Mode (T1M1,T1M0)=(0,1): 1:count on falling edge; 0:count on rising edge In Pulse Width measurement mode (T1M1,T1M0)=(1,1): 1: start counting on the rising edge, stop on the falling edge; 0: start counting on the falling edge, stop on the rising edge
4	T1ON	Enable/disable timer counting (0=disabled; 1=enabled)
5	T1S	Defines the TMR1 internal clock source (0= $f_{SYS}/4$ ; 1=Int.RCOSC (Internal RC OSC))
6 7	T1M0 T1M1	Defines the operating mode T1M1, T1M0= 01=Event count mode (External clock) 10=Timer mode (Internal clock) 11=Pulse Width measurement mode (External clock) 00=Unused

**TMR1C (11H) Register**

begins counting not according to the logic level but to the transient edges. In the case of counter overflows, the counter is reloaded from the timer/event counter 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 (T0ON; bit 4 of TMR0C or T1ON bit 4 of TMR1C) should be set to 1. In the pulse width measurement mode, the T0ON (T1ON) is automatically cleared after the measurement cycle is completed. But in the other two modes, the T0ON (T1ON) can only be reset by instructions. The overflow of the Timer/Event Counter 0/1 is one of the wake-up sources. No matter what the operation mode is, writing a 0 to ET0I or ET1I disables the related interrupt service.

In the case of a timer/event counter OFF condition, writing data to the timer/event counter preload register also reloads that data to the timer/event counter. But if the timer/event counter is turned 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, however as this may result in a counting error, it should be taken into account by the programmer. It is strongly recommended to load a desired value into the TMR0/TMR1 register first, before turning on the related timer/event counter, for proper operation since the initial value of TMR0/TMR1 is unknown. Due to the timer/event counter scheme, the programmer should pay special attention to the instructions which enables then disables the timer for the first time, whenever there is a need to use the timer/event counter function, to avoid unpredictable results. After this procedure, the timer/event function can be operated normally.

The bit0~bit2 of the TMR0C/TMR1C can be used to define the pre-scaling stages of the internal clock sources of Timer/Event Counter 0/1.

### Input/Output Ports

There are 12 bidirectional input/output lines in the microcontroller, labeled as PA and PB, which are mapped to the data memory of [12H] and [14H] respectively. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]" (m=12H or 14H). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC) to control the input/output configuration. With this control register, CMOS outputs or Schmitt trigger inputs with or without pull-high resistor structures can be reconfigured dynamically under software control. To function as an in-

put, the corresponding latch of the control register must write "1". The input source also depends on the control register. If the control register bit is "1", the input will read the pad state. If the control register bit is "0", the contents of the latches will move to the internal bus. The latter is possible in the "read-modify-write" instruction.

For output function, CMOS is the only configuration. These control registers are mapped to locations 13H, and 15H.

After a chip reset, these input/output lines remain at high levels or in a floating state, depending upon the pull-high configuration options. Each bit of these input/output latches can be set or cleared by "SET [m].i" and "CLR [m].i" (m=12H or 14H) 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 the accumulator.

Each line of port A has the capability of waking-up the device.

Each I/O port has a pull-high option. Once the pull-high option is selected, the I/O port has a pull-high resistor connected. Take note that a non-pull-high I/O port setup as an input mode will be in a floating condition.

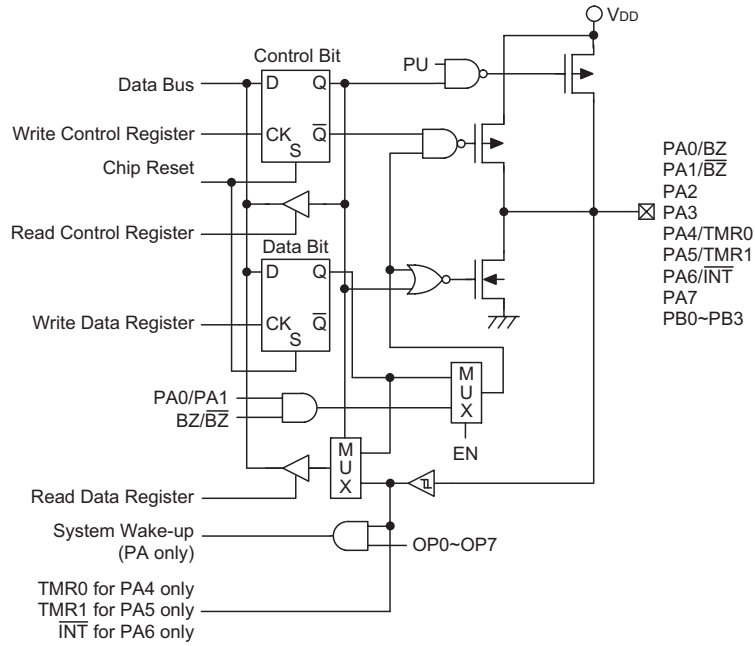
Pins PA0, PA1, PA4, PA5 and PA6 are pin-shared with BZ,  $\overline{BZ}$ , TMR0, TMR1 and  $\overline{INT}$  pins respectively.

PA0 and PA1 are pin-shared with BZ and  $\overline{BZ}$  signal, respectively. If the BZ/ $\overline{BZ}$  configuration option is selected, the output signals in the output mode of PA0/PA1 can be the buzzer signal. The input mode always retains its original function. Once the BZ/ $\overline{BZ}$  configuration option is selected, the buzzer output signals are controlled by the PA0 data register.

The PA0/PA1 I/O function is shown below.

PA0 I/O	I	I	O	O	O	O	O	O	O	O
PA1 I/O	I	O	I	I	I	O	O	O	O	O
PA0 Mode	X	X	C	B	B	C	B	B	B	B
PA1 Mode	X	C	X	X	X	C	C	C	B	B
PA0 Data	X	X	D	0	1	D <sub>0</sub>	0	1	0	1
PA1 Data	X	D	X	X	X	D <sub>1</sub>	D	D	X	X
PA0 Pad Status	I	I	D	0	B	D <sub>0</sub>	0	B	0	B
PA1 Pad Status	I	D	I	I	I	D <sub>1</sub>	D	D	0	B

Note: "I" input; "O" output  
 "D, D0, D1" Data  
 "B" buzzer option, BZ or  $\overline{BZ}$   
 "X" don't care  
 "C" CMOS output



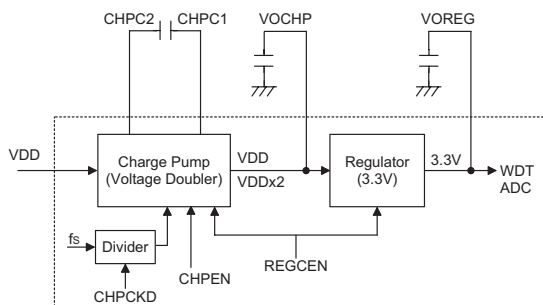
Input/Output Ports

It is recommended that unused or not bonded out I/O lines should be set as output pins using software instructions to avoid consuming power when in an input floating state.

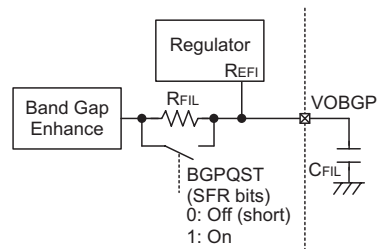
**Charge Pump and Voltage Regulator**

There is one charge pump and one voltage regulator implement in this device.

The charge pump can be enabled/disabled by the application program. The charge pump uses V<sub>DD</sub> as its input, and has the function of doubling the V<sub>DD</sub> voltage. The output voltage of the charge pump will be V<sub>DD</sub>×2. The regulator can generate a stable voltage of 3.3V, for internal WDT, ADC and also can provide an external bridge sensor excitation voltage or supply a reference voltage for other applications. The user needs to guarantee the charge pump output voltage is greater than 3.6V to ensure that the regulator generates the required 3.3V voltage output. The block diagram of this module is shown below.



Additionally, the device also includes a band gap voltage generator for the 1.5V low temperature sensitive reference voltage. This reference voltage is used as the zero adjustment and for a single end type reference voltage.



R<sub>FIL</sub> is about 100kΩ and the recommend C<sub>FIL</sub> is 10μF. Note: VOBGP signal is only for chip internal used. Don't connect to external component except the recommend C<sub>FIL</sub>.

There is a single register associated with this module named CHPRC. The CHPRC is the Charge Pump/Regulator Control register, which controls the charge pump on/off, regulator on/off functions as well as setting the clock divider value to generate the clock for the charge pump.

The CHPCKD4~CHPCKD0 bits are use to set the clock divider to generate the desired clock frequency for proper charge pump operation. The actual frequency is determined by the following formula.

$$\text{Actual Charge Pump Clock} = (f_{\text{SYS}}/16)/(CHPCKD + 1).$$

Bit No.	Label	Function
0	REGCEN	Enable/disable Regulator/Charge-Pump module. (1=enable; 0=disable)
1	CHPEN	Charge Pump Enable/disable setting. (1=enable; 0=disable) Note: this bit will be ignore if the REGCEN is disable
2	BGPQST	Band gap quickly start-up function 0: R short, quickly start 1: R connected, normal RC filter mode Every time when REGCEN change from 0 to 1 (Regulator turn on) This bit should be set to 0 and then set to 1 to make sure the quickly stable. (the minimum 0 keeping time is about 2ms now )
3~7	CHPCKD0~ CHPCKD4	The Charge pump clock divider. This 5 bits can form the clock divide by 1~32. Following the below equation: Charge Pump clock = (f <sub>SY</sub> /16) / (CHPCKD+1)

#### CHPRC (1FH) Register

REGCEN	CHPEN	Charge Pump	VOCHP Pin	Regulator	VOREG Pin	OPA ADC	Description
0	X	OFF	V <sub>DD</sub>	OFF	Hi-Impedance	Disable	The whole module is disable, OPA/ADC will lose the Power
1	0	OFF	V <sub>DD</sub>	ON	3.3V	Active	Use for V <sub>DD</sub> is greater than 3.6V (V <sub>DD</sub> >3.6V)
1	1	ON	2×V <sub>DD</sub>	ON	3.3V	Active	Use for V <sub>DD</sub> is less than 3.6V (V <sub>DD</sub> =2.2V~3.6V)

The suggested charge pump clock frequency is 20kHz. The application needs to set the correct value to get the desired clock frequency. For a 4MHz application, the CHPCKD bits should be set to the value 11, and for a 2MHz application, the bits should be set to 5.

The REGCEN bit in the CHPRC register is the Regulator/ Charge-pump module enable/disable control bit. If this bit is disabled, then the regulator will be disabled and the charge pump will be also be disabled to save power. When REGCEN = 0, the module will enter the Power Down Mode ignoring the CHPEN setting. The ADC and OPA will also be disabled to reduce power.

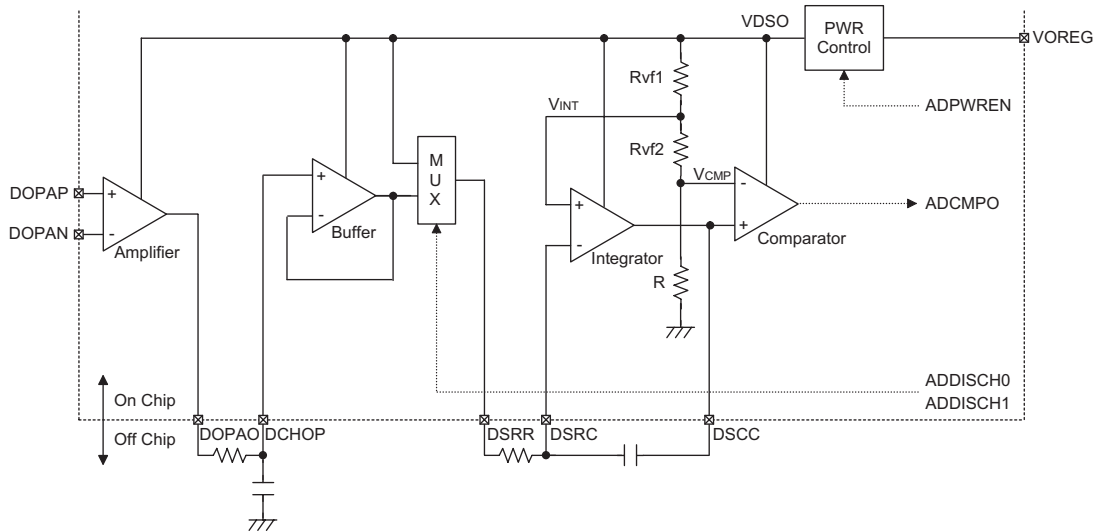
If REGCEN is set to "1", the regulator will be enabled. If CHPEN is enabled, the charge pump will be active and will use V<sub>DD</sub> as its input to generate the double voltage output. This double voltage will be used as the input voltage for the regulator. If CHPEN is set to "0", the charge pump is disabled and the charge pump output will be equal to the charge pump input, V<sub>DD</sub>.

It is necessary to take care of the V<sub>DD</sub> voltage. If the voltage is less than 3.6V, then CHPEN should be set to 1 to enable the charge pump, otherwise CHPEN should be set to zero. If the Charge pump is disabled and V<sub>DD</sub> is less than 3.6V then the output voltage of the regulator will not be guaranteed.

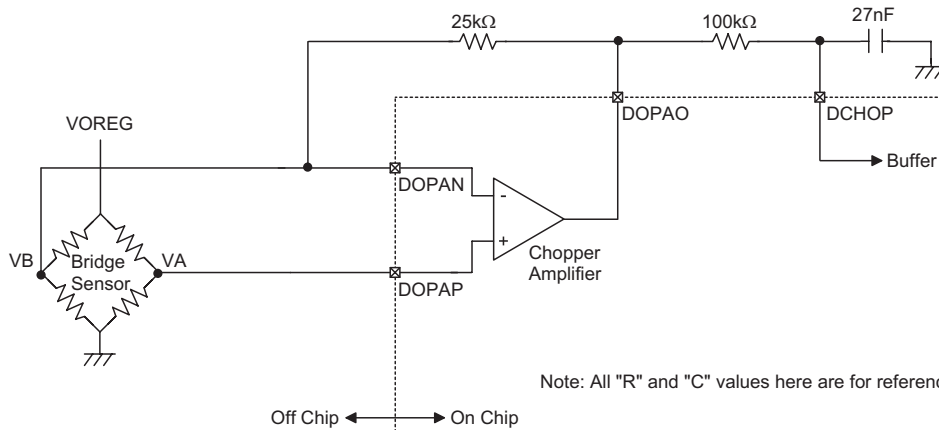
#### ADC – Dual Slope

A Dual Slope A/D converter is implemented in this microcontroller. The dual slope module includes an Operational Amplifier and a buffer for the amplification of differential signals, an Integrator and a comparator for the main dual slope AD converter.

There are 2 special function registers related to this function known as ADCR and ADCD. The ADCR register is the A/D control register, which controls the ADC block power on/off, the chopper clock on/off, the charge/discharge control and is also used to read out the comparator output status. The ADCD register is the A/D Chopper clock divider register, which defines the chopper clock to the ADC module.



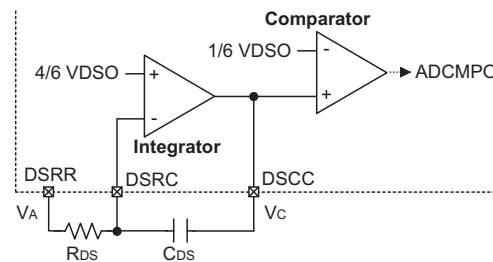
Note:  $V_{INT}$ ,  $V_{CMP}$  signal can come from different R groups which are selected by software registers.



Note: All "R" and "C" values here are for reference only

The ADPWREN bit, defined in ADCR register, is used to control the ADC module on/off function. The ADCKEN bit defined in the ADCR register is used to control the chopper clock on/off function. When ADCKEN is set to "1" it will enable the Chopper clock, with the clock frequency defined by the ADCD registers. The ADC module includes the OPA, buffer, integrator and comparator, however the Bandgap voltage generator is independent of this module. It will be automatically enabled when the regulator is enabled, and also be disabled when the regulator is disabled. The application program should enable the related power to permit them to function and disable them when idle to conserve power. The charge/discharge control bits, ADDISCH1~ ADDISCH0, are used to control the Dual slope circuit charging and discharging behavior. The ADCMPO bit is read only for the comparator output, while the ADINTM bits can set the ADCMPO trigger mode for interrupt generation.

The following descriptions are based the fact that ADRR0=0



The amplifier and buffer combination, form a differential input pre-amplifier which amplifies the sensor input signal.

The combination of the Integrator, the comparator, the resistor Rds, between DSRR and DSRC and the capacitor Cds, between DSRC and DSCC form the main body of the Dual slope ADC.

The Integrator integrates the output voltage increase or decrease and is controlled by the "Switch Circuit" - refer to the block diagram. The integration and de-integration curves are illustrated by the following.

The "comparator" will switch the state from high to low when  $V_C$ , which is the DSCC pin voltage, drops to less than  $1/6 V_{DSO}$ .

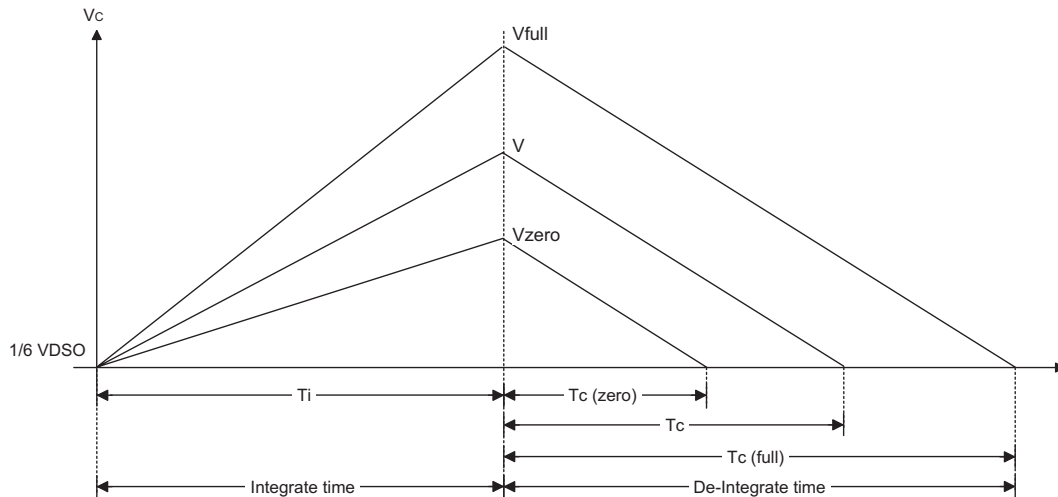
In general applications, the application program will switch the ADC to the charging mode for a fixed time called  $T_i$ , which is the integrating time. It will then switch to the dis-charging mode and wait for  $V_C$  to drop to less

than  $1/6 V_{DSO}$ . At this point the comparator will change state and store the time taken,  $T_c$ , which is the de-integrating time. The following formula 1 can then be used to calculate the input voltage  $V_A$ .

$$\text{formula 1: } V_A = (1/3) \times V_{DSO} \times (2 - T_c / T_i)$$

(Based on  $ADRR0=0$ )

In user applications, it is required to choose the correct value of  $R_{DS}$  and  $C_{DS}$  to determine the  $T_i$  value, to allow the  $V_C$  value to operate between  $5/6 V_{DSO}$  and  $1/6 V_{DSO}$ .  $V_{full}$  cannot be greater than  $5/6 V_{DSO}$  and  $V_{zero}$  cannot be less than  $1/6 V_{DSO}$



Bit No.	Label	Function
0	ADPWREN	Dual slope block (including input OP) power on/off switching. 0: disable Power 1: Power source comes from the regulator.
1~2	ADDISCH0~ ADDISCH1	Defines the ADC discharge/charge. (ADDISCH1:0) 00: reserved 01: charging. (Integrator input connect to buffer output) 10: discharging. (Integrator input connect to VDSO) 11: reserved
3	ADCMPO	Dual Slope ADC - last stage comparator output. Read only bit, write data instructions will be ignored. During the discharging state, when the integrator output is less than the reference voltage, the ADCMPO will change from high to low.
4~5	ADINTM0~ ADINTM1	ADC integrator interrupt mode definition. These two bit define the ADCMPO data interrupt trigger mode: (ADINTM1:0)= 00: no interrupt 01: rising edge 10: falling edge 11: both edge
6	ADCKEN	ADC OP chopper clock source on/off switching. 0: disable 1: enable (clock value is defined by ADCD register)
7	ADRR0	ADC resistors selection 0: ( $V_{INT}, V_{CMP}$ )= (4/6 VOREG, 1/6 VOREG) 1: ( $V_{INT}, V_{CMP}$ )= (4.4/6 VOREG, 1/6 VOREG)

**ADCR (18H) Register**

Bit No.	Label	Function
0	ADCD0	Define the chopper clock (ADCCKEN should be enable), the suggestion clock is around 10kHz. The chopper clock define : 0: clock= (f <sub>SYS</sub> /32)/1 1: clock= (f <sub>SYS</sub> /32)/2 2: clock= (f <sub>SYS</sub> /32)/4 3: clock= (f <sub>SYS</sub> /32)/8 4: clock= (f <sub>SYS</sub> /32)/16 5: clock= (f <sub>SYS</sub> /32)/32 6: clock= (f <sub>SYS</sub> /32)/64 7: clock= (f <sub>SYS</sub> /32)/128
1	ADCD1	
2	ADCD2	
3~7	—	Reserved

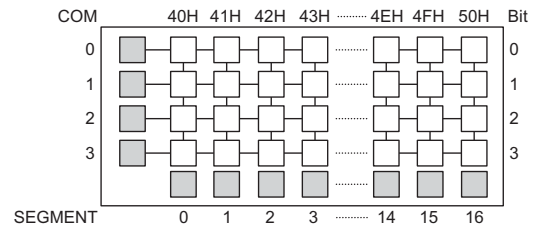
**ADCD (1AH) Register**

**LCD Display Memory**

The device provides an area of embedded data memory for the LCD display. This area is located at 40H to 50H in Bank 1 of the Data Memory. The bank pointer BP, enables either the General Purpose Data Memory or LCD Memory to be chosen. When BP is set to "1", any data written into location range 40H~50H will affect the LCD display. When the BP is cleared to "0", any data written into 40H~50H will access the general purpose data memory. The LCD display memory can be read and written to only indirectly 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 device.

The LCD clock is driven by the IRC clock, which then passes through a divider, the division ratio of which is selected by configuration options to provide a range of

LCD frequencies from Int.RCOSC/3 to Int.RCOSC/4. Note that the LCD frequency is controlled by configuration options, which select the internal division ratio.

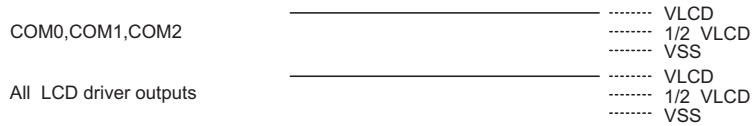


**Display Memory**

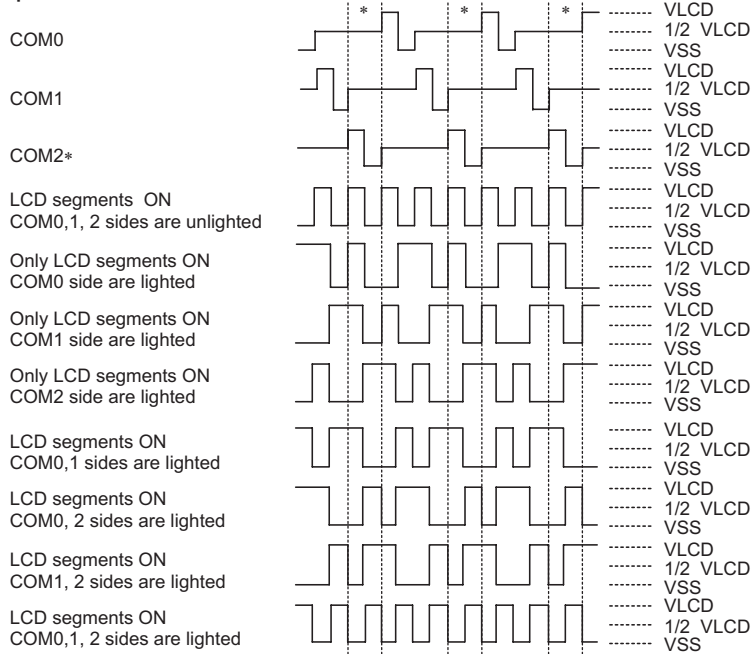
**LCD Driver Output**

The output structure of the device LCD driver can be either 16×4, 17×3 or 17×2 selectable via configuration option (i.e., 1/2 duty, 1/3 duty or 1/4 duty). The bias type LCD driver is R type only. The LCD driver bias voltage can be 1/2 bias or 1/3 bias by option.

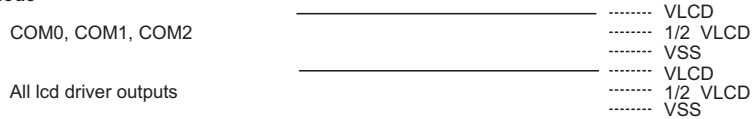
**During a Reset Pulse**



**Normal Operation Mode**

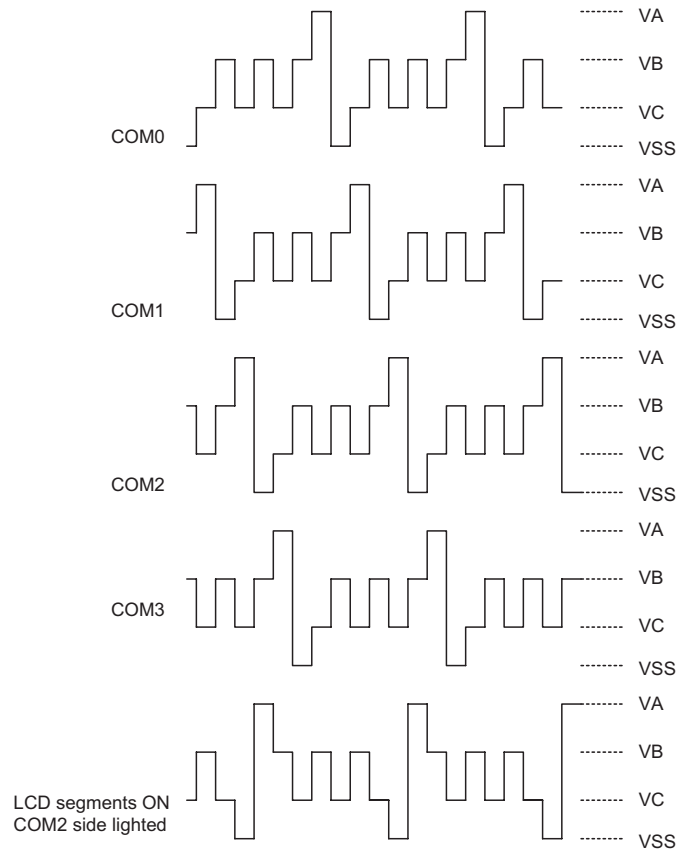


**HALT Mode**



Note: "\*" Omit the COM2 signal, if the 1/2 duty LCD is used.

**LCD Driver Output (1/3 Duty, 1/2 Duty, R Type)**



Note: 1/4 duty, 1/3 bias, R type: "VA" VLCD, "VB" 2/3 VLCD, "VC" 1/3 VLCD

**LCD Driver Output (1/4 Duty)**

**Low Voltage Reset/Detector Functions**

There is a low voltage detector, LVD, and a low voltage reset circuit, LVR, implemented in the microcontroller. These two functions can be enabled/disabled by configuration options. Once the LVD option is enabled, the user can use the MODE.3 bit to enable/disable (1/0) the LVD circuit and read the LVD detector status (0/1) from bit MODE.5.

The MODE register definitions are listed below.

Bit No.	Label	Function
0~1, 4, 6~7	—	Unused bit, read as "unknown"
2	IRCC	In HALT mode, IRC clock enable or disable selection bit. 0: IRC clock enable and Int.RCOSCON. 1: IRC clock disabled.
3	LVDC	LVD enable/disable (1/0)
5	LVDO	LVD detection output (1/0) 1: low voltage detected, read only. 0: low voltage not detected.

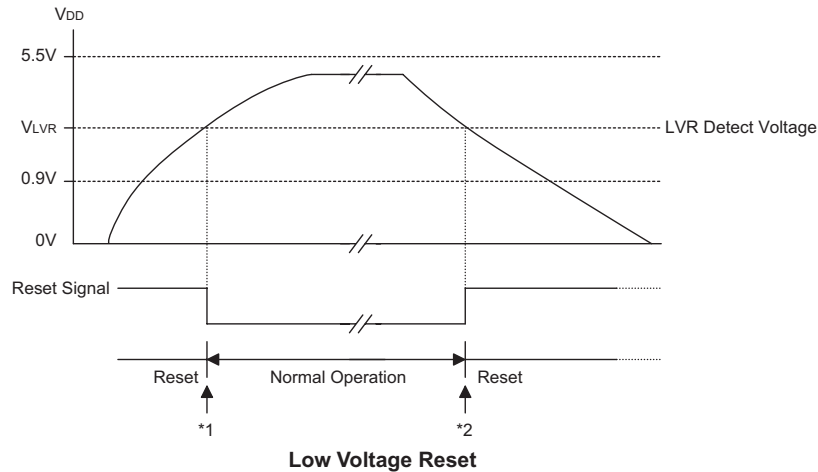
**MODE (09H) Register**

The LVR has the same effect or function as the external RES signal which performs a device reset. When in the Power Down Mode, both LVR and LVD are disabled.

The microcontroller provides a low voltage reset circuit in order to monitor the supply voltage of the device. If the supply voltage of the device is within the range  $0.9V \sim V_{LVR}$ , such as what might happen when changing a battery, the LVR will automatically reset the device internally.

The LVR includes the following specifications:

- The low voltage, which is specified as  $0.9V \sim V_{LVR}$ , has to remain within this range for a period of time greater than 1ms. If the low voltage state does not exceed 1ms, the LVR will ignore it will not perform a reset function.
- The LVR has an "OR" function with the external  $\overline{RES}$  signal to perform a chip reset.



Note: \*1: To make sure that the system oscillator has stabilized, the SST provides an extra delay of 1024 system clock pulses before entering the normal operation.

\*2: Since a low voltage state has to be maintained in its original state for over 1ms, therefore after 1ms delay, the device enters the reset mode.

**Operation Mode**

The device has three operational modes. The system clock comes from external RC or crystal oscillator, and whose operational modes can be either Normal, IDLE or HALT mode. These are all selected by the software.

HALT Instruction	IRCC	WDT	System Oscillator	IRC Clock	System Clock	Mode
Not executed	x	—	On	Enabled	External RC or Crystal Oscillator	Normal
Executed	0	x	Off	Enabled and Int. RCOSC ON	Off	Idle
	1	Enabled	Off	Disabled and Int. RCOSC ON	Off	HALT
		Disabled	Off	Disabled and Int. RCOSC OFF	Off	HALT

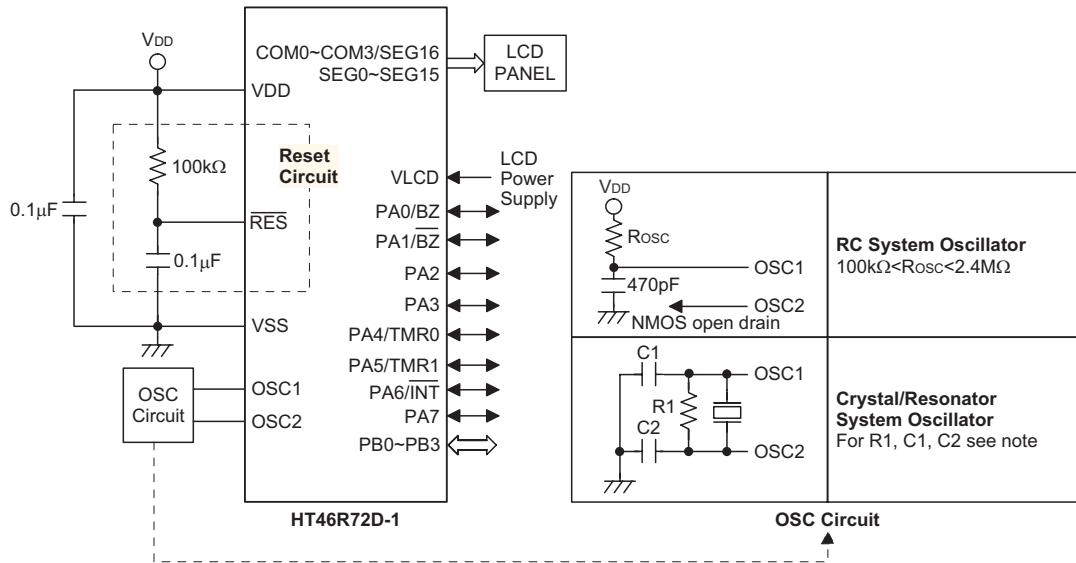
Note: The WDTOSC0:1 register should be set to enable, otherwise, the Int.RCOSC will always be disabled. Refer to the WDT section for the WDTOSC setup details.

**Options**

The following shows the options in the device. All these options should be defined in order to ensure proper functioning system.

<b>Options</b>
<p>OSC type selection. There are two types selection: Crystal OSC or RC OSC</p>
<p><math>f_S</math> clock source. There are two types of selections: Int.RCOSC or <math>f_{SYS}/4</math></p>
<p>WDT clock source selection. There are two types of selections: system clock/4 or Int.RCOSC.</p>
<p>WDT enable/disable selection. WDT can be enabled or disabled by option.</p>
<p>WDT time-out period selection. There are four types of selection: WDT clock source divided by <math>2^{16}/f_S</math>, <math>2^{15}/f_S</math>, <math>2^{14}/f_S</math> or <math>2^{13}/f_S</math></p>
<p>CLR WDT times selection. This option selects the instruction method of clearing the WDT. "One time" means that the "CLR WDT" instruction can clear the WDT. "Two times" means only if both the "CLR WDT1" and "CLR WDT2" instructions have been executed, can the WDT be cleared.</p>
<p>Buzzer output frequency selection. There are eight types of frequency signals for buzzer output: <math>f_S/2^2 \sim f_S/2^9</math>. "<math>f_S</math>" means the configuration option selected clock source.</p>
<p>Wake-up selection. This option defines the wake-up capability. A falling edge on each external pin on PA has the capability to wake-up the device from a Power Down condition. Bit option.</p>
<p>Pull-high selection. Selects pull-high resistors when the I/O in in the input mode. PA and PB ports are bit options.</p>
<p>I/O pins shared with other function selections. PA0/BZ, PA1/<math>\overline{BZ}</math>: PA0 and PA1 can be set as I/O pins or buzzer outputs.</p>
<p>LCD common selection. There are three types of selections: 2 common (1/2 duty), 3 common (1/3 duty) or 4 common (1/4 duty).</p>
<p>LCD bias selection. This option is to determine what kind of bias is selected, 1/2 bias or 1/3 bias.</p>
<p>LCD driver clock frequency selection. There are two types of frequency signals for the LCD driver circuits: Int.RCOSC/3~Int.RCOSC/4.</p>
<p>LCD ON/OFF when in Power Down Mode selection</p>
<p>LVR selection. LVR enable or disable option</p>
<p>LVD selection. LVD enable or disable option</p>
<p><math>\overline{INT}</math> trigger edge selection: disable; high to low; low to high; low to high or high to low</p>

Application Circuits



Note: 1. Crystal/resonator system oscillators

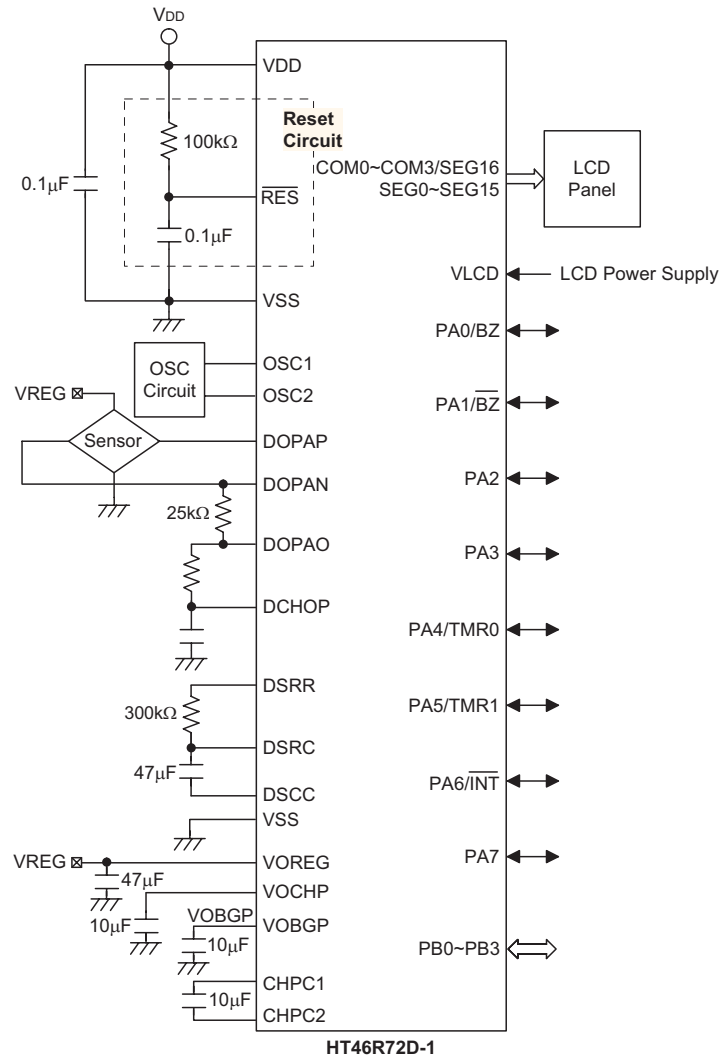
For crystal oscillators, C1 and C2 are only required for some crystal frequencies to ensure oscillation. For resonator applications C1 and C2 are normally required for oscillation to occur. For most applications it is not necessary to add R1. However if the LVR function is disabled, and if it is required to stop the oscillator when VDD falls below its operating range, it is recommended that R1 is added. The values of C1 and C2 should be selected in consultation with the crystal/resonator manufacturer specifications.

2. Reset circuit

The reset circuit resistance and capacitance values should be chosen to ensure that VDD is stable and remains within its operating voltage range before the RES pin reaches a high level. Ensure that the length of the wiring connected to the RES pin is kept as short as possible, to avoid noise interference.

3. For applications where noise may interfere with the reset circuit and for details on the oscillator external components, refer to Application Note HA0075E for more information.

Example



## Instruction Set

### Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontrollers, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

### Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5 $\mu$ s and branch or call instructions would be implemented within 1 $\mu$ s. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

### Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of three kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

### Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and

subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

### Logical and Rotate Operations

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application where rotate data operations are used is to implement multiplication and division calculations.

### Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction RET in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

### Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

### Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

### Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

### Instruction Set Summary

The following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0-7 number of bits

addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
<b>Arithmetic</b>			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1 <sup>Note</sup>	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1 <sup>Note</sup>	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1 <sup>Note</sup>	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1 <sup>Note</sup>	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1 <sup>Note</sup>	C
<b>Logic Operation</b>			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1 <sup>Note</sup>	Z
ORM A,[m]	Logical OR ACC to Data Memory	1 <sup>Note</sup>	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1 <sup>Note</sup>	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1 <sup>Note</sup>	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
<b>Increment &amp; Decrement</b>			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1 <sup>Note</sup>	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1 <sup>Note</sup>	Z

Mnemonic	Description	Cycles	Flag Affected
<b>Rotate</b>			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	↑ <sup>Note</sup>	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	↑ <sup>Note</sup>	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	↑ <sup>Note</sup>	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	↑ <sup>Note</sup>	C
<b>Data Move</b>			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	↑ <sup>Note</sup>	None
MOV A,x	Move immediate data to ACC	1	None
<b>Bit Operation</b>			
CLR [m].i	Clear bit of Data Memory	↑ <sup>Note</sup>	None
SET [m].i	Set bit of Data Memory	↑ <sup>Note</sup>	None
<b>Branch</b>			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	↑ <sup>Note</sup>	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	↑ <sup>note</sup>	None
SZ [m].i	Skip if bit i of Data Memory is zero	↑ <sup>Note</sup>	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	↑ <sup>Note</sup>	None
SIZ [m]	Skip if increment Data Memory is zero	↑ <sup>Note</sup>	None
SDZ [m]	Skip if decrement Data Memory is zero	↑ <sup>Note</sup>	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	↑ <sup>Note</sup>	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	↑ <sup>Note</sup>	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
<b>Table Read</b>			
TABRDC [m]	Read table (current page) to TBLH and Data Memory	2 <sup>Note</sup>	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2 <sup>Note</sup>	None
<b>Miscellaneous</b>			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	↑ <sup>Note</sup>	None
SET [m]	Set Data Memory	↑ <sup>Note</sup>	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	↑ <sup>Note</sup>	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.  
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.  
3. For the "CLR WDT1" and "CLR WDT2" instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both "CLR WDT1" and "CLR WDT2" instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

**Instruction Definition**

<b>ADC A,[m]</b>	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C
<b>ADCM A,[m]</b>	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	$[m] \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C
<b>ADD A,[m]</b>	Add Data Memory to ACC
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)	OV, Z, AC, C
<b>ADD A,x</b>	Add immediate data to ACC
Description	The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC + x$
Affected flag(s)	OV, Z, AC, C
<b>ADDM A,[m]</b>	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	$[m] \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C
<b>AND A,[m]</b>	Logical AND Data Memory to ACC
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)	Z
<b>AND A,x</b>	Logical AND immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC \text{ "AND" } x$
Affected flag(s)	Z
<b>ANDM A,[m]</b>	Logical AND ACC to Data Memory
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)	Z

<b>CALL addr</b>	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack $\leftarrow$ Program Counter + 1 Program Counter $\leftarrow$ addr
Affected flag(s)	None
<b>CLR [m]</b>	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] $\leftarrow$ 00H
Affected flag(s)	None
<b>CLR [m].i</b>	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i $\leftarrow$ 0
Affected flag(s)	None
<b>CLR WDT</b>	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO $\leftarrow$ 0 PDF $\leftarrow$ 0
Affected flag(s)	TO, PDF
<b>CLR WDT1</b>	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repeatedly executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO $\leftarrow$ 0 PDF $\leftarrow$ 0
Affected flag(s)	TO, PDF
<b>CLR WDT2</b>	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repeatedly executing this instruction without alternately executing CLR WDT1 will have no effect.
Operation	WDT cleared TO $\leftarrow$ 0 PDF $\leftarrow$ 0
Affected flag(s)	TO, PDF

<b>CPL [m]</b>	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	$[m] \leftarrow \overline{[m]}$
Affected flag(s)	Z
<b>CPLA [m]</b>	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 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)	Z
<b>DAA [m]</b>	Decimal-Adjust ACC for addition with result in Data Memory
Description	Convert the contents of the Accumulator value to a BCD ( Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.
Operation	$[m] \leftarrow ACC + 00H$ or $[m] \leftarrow ACC + 06H$ or $[m] \leftarrow ACC + 60H$ or $[m] \leftarrow ACC + 66H$
Affected flag(s)	C
<b>DEC [m]</b>	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	$[m] \leftarrow [m] - 1$
Affected flag(s)	Z
<b>DECA [m]</b>	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow [m] - 1$
Affected flag(s)	Z
<b>HALT</b>	Enter power down mode
Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.
Operation	$TO \leftarrow 0$ $PDF \leftarrow 1$
Affected flag(s)	TO, PDF

<b>INC [m]</b>	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	$[m] \leftarrow [m] + 1$
Affected flag(s)	Z
<b>INCA [m]</b>	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow [m] + 1$
Affected flag(s)	Z
<b>JMP addr</b>	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	$Program\ Counter \leftarrow addr$
Affected flag(s)	None
<b>MOV A,[m]</b>	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	$ACC \leftarrow [m]$
Affected flag(s)	None
<b>MOV A,x</b>	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	$ACC \leftarrow x$
Affected flag(s)	None
<b>MOV [m],A</b>	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	$[m] \leftarrow ACC$
Affected flag(s)	None
<b>NOP</b>	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
<b>OR A,[m]</b>	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC \text{ "OR" } [m]$
Affected flag(s)	Z

<b>OR A,x</b>	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "OR" x
Affected flag(s)	Z
<b>ORM A,[m]</b>	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC "OR" [m]
Affected flag(s)	Z
<b>RET</b>	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None
<b>RET A,x</b>	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None
<b>RETI</b>	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the enable master (global) interrupt bit (bit 0; register INTC). If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None
<b>RL [m]</b>	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i = 0~6) [m].0 ← [m].7
Affected flag(s)	None
<b>RLA [m]</b>	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i = 0~6) ACC.0 ← [m].7
Affected flag(s)	None

<b>RLC [m]</b>	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	$[m].(i+1) \leftarrow [m].i; (i = 0\sim 6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C
<b>RLCA [m]</b>	Rotate Data Memory left through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC.(i+1) \leftarrow [m].i; (i = 0\sim 6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C
<b>RR [m]</b>	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	$[m].i \leftarrow [m].(i+1); (i = 0\sim 6)$ $[m].7 \leftarrow [m].0$
Affected flag(s)	None
<b>RRA [m]</b>	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC.i \leftarrow [m].(i+1); (i = 0\sim 6)$ $ACC.7 \leftarrow [m].0$
Affected flag(s)	None
<b>RRC [m]</b>	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	$[m].i \leftarrow [m].(i+1); (i = 0\sim 6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
<b>RRCA [m]</b>	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC.i \leftarrow [m].(i+1); (i = 0\sim 6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C

<b>SBC A,[m]</b>	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C
<b>SBCM A,[m]</b>	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C
<b>SDZ [m]</b>	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] - 1$ Skip if $[m] = 0$
Affected flag(s)	None
<b>SDZA [m]</b>	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m] - 1$ Skip if $ACC = 0$
Affected flag(s)	None
<b>SET [m]</b>	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	$[m] \leftarrow FFH$
Affected flag(s)	None
<b>SET [m].i</b>	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	$[m].i \leftarrow 1$
Affected flag(s)	None

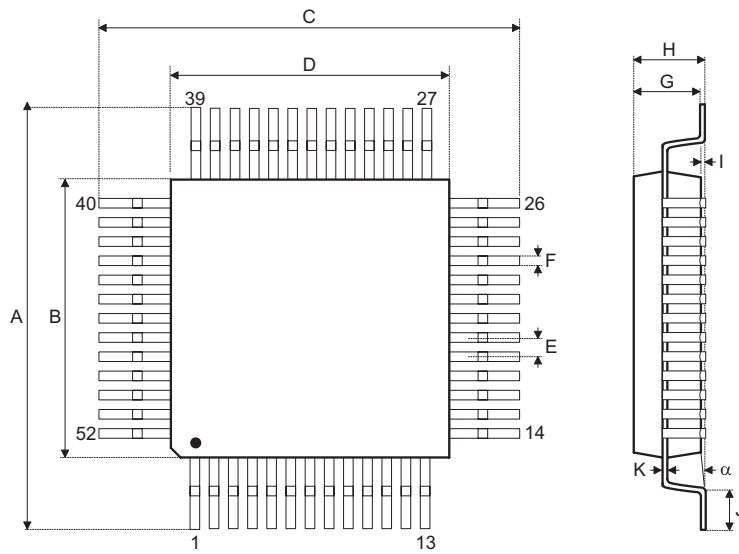
<b>SIZ [m]</b>	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] + 1$ Skip if $[m] = 0$
Affected flag(s)	None
<b>SIZA [m]</b>	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m] + 1$ Skip if $ACC = 0$
Affected flag(s)	None
<b>SNZ [m].i</b>	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if $[m].i \neq 0$
Affected flag(s)	None
<b>SUB A,[m]</b>	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C
<b>SUBM A,[m]</b>	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C
<b>SUB A,x</b>	Subtract immediate data from ACC
Description	The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - x$
Affected flag(s)	OV, Z, AC, C

<b>SWAP [m]</b>	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
Affected flag(s)	None
<b>SWAPA [m]</b>	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in 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)	None
<b>SZ [m]</b>	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if $[m] = 0$
Affected flag(s)	None
<b>SZA [m]</b>	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m]$ Skip if $[m] = 0$
Affected flag(s)	None
<b>SZ [m].i</b>	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if $[m].i = 0$
Affected flag(s)	None
<b>TABRDC [m]</b>	Read table (current page) to TBLH and Data Memory
Description	The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	$[m] \leftarrow$ program code (low byte) $TBLH \leftarrow$ program code (high byte)
Affected flag(s)	None
<b>TABRDL [m]</b>	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	$[m] \leftarrow$ program code (low byte) $TBLH \leftarrow$ program code (high byte)
Affected flag(s)	None

<b>XOR A,[m]</b>	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "XOR" [m]
Affected flag(s)	Z
<b>XORM A,[m]</b>	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC "XOR" [m]
Affected flag(s)	Z
<b>XOR A,x</b>	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "XOR" x
Affected flag(s)	Z

**Package Information**

**52-pin QFP (14×14) Outline Dimensions**



Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	17.3	—	17.5
B	13.9	—	14.1
C	17.3	—	17.5
D	13.9	—	14.1
E	—	1	—
F	—	0.4	—
G	2.5	—	3.1
H	—	—	3.4
I	—	0.1	—
J	0.73	—	1.03
K	0.1	—	0.2
$\alpha$	0°	—	7°

**Holtek Semiconductor Inc. (Headquarters)**

No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan  
Tel: 886-3-563-1999  
Fax: 886-3-563-1189  
<http://www.holtek.com.tw>

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

4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan  
Tel: 886-2-2655-7070  
Fax: 886-2-2655-7373  
Fax: 886-2-2655-7383 (International sales hotline)

**Holtek Semiconductor Inc. (Shanghai Sales Office)**

7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China 200233  
Tel: 86-21-6485-5560  
Fax: 86-21-6485-0313  
<http://www.holtek.com.cn>

**Holtek Semiconductor Inc. (Shenzhen Sales Office)**

5/F, Unit A, Productivity Building, Cross of Science M 3rd Road and Gaoxin M 2nd Road, Science Park, Nanshan District, Shenzhen, China 518057  
Tel: 86-755-8616-9908, 86-755-8616-9308  
Fax: 86-755-8616-9722

**Holtek Semiconductor Inc. (Beijing Sales Office)**

Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031  
Tel: 86-10-6641-0030, 86-10-6641-7751, 86-10-6641-7752  
Fax: 86-10-6641-0125

**Holtek Semiconductor Inc. (Chengdu Sales Office)**

709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016  
Tel: 86-28-6653-6590  
Fax: 86-28-6653-6591

**Holtek Semiconductor (USA), Inc. (North America Sales Office)**

46729 Fremont Blvd., Fremont, CA 94538  
Tel: 1-510-252-9880  
Fax: 1-510-252-9885  
<http://www.holtek.com>

Copyright © 2008 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's products are not authorized for use as critical components in life support devices or systems. 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>.