HT46R652

A/D with LCD Type 8-Bit OTP MCU

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
  - HA0047E An PWM application example using the HT46 series of MCUs

Features

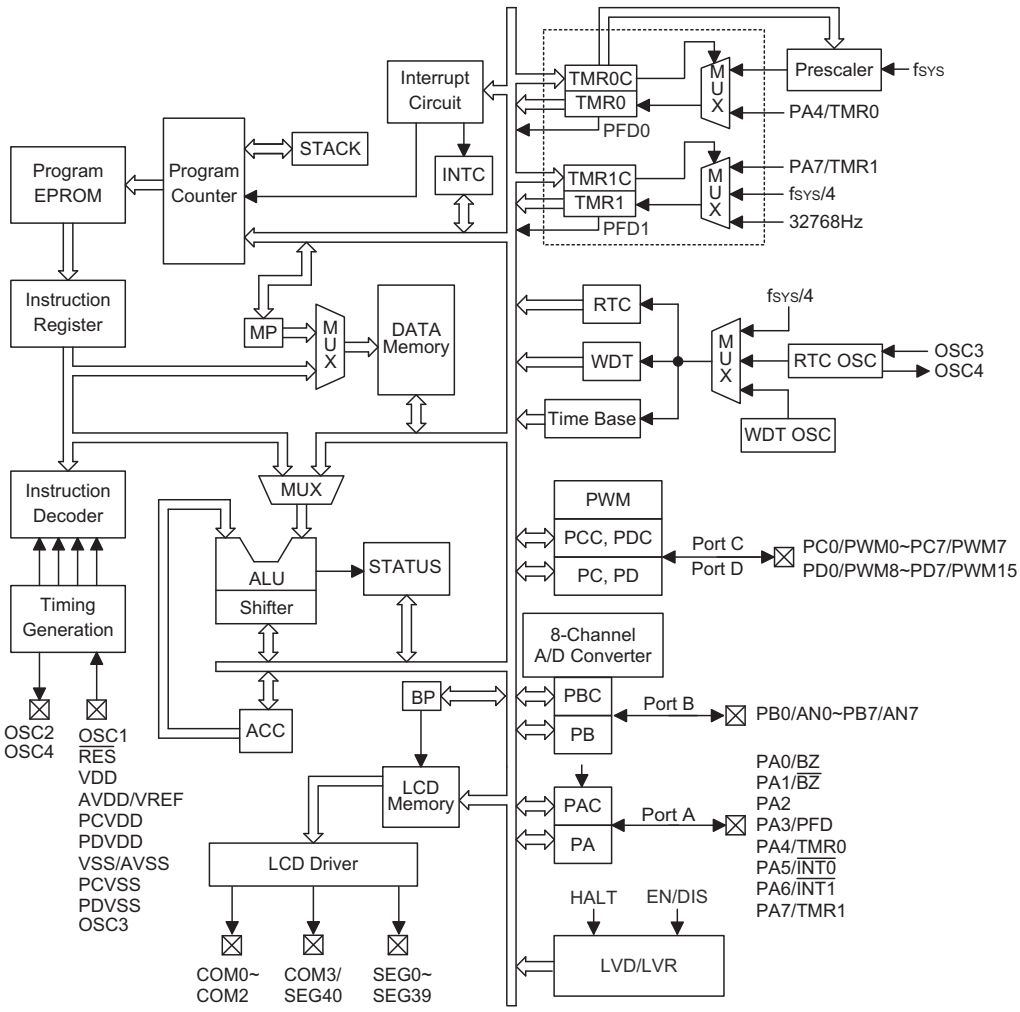
- Operating voltage:
  f_{SYS}=4MHz: 2.2V~5.5V
  f_{SYS}=12MHz: 3.3V~5.5V
- 32 bidirectional I/O lines
- Two external interrupt inputs
- Two 16-bit programmable timer/event counters with PFD (programmable frequency divider) function
- LCD driver with 41×3 or 40×4 segments (logical output option for SEG0~SEG23)
- 8K×16 program memory
- 384×8 data memory RAM
- PFD for sound generation
- Real Time Clock (RTC)
- 8-bit RTC prescaler
- Watchdog Timer
- Buzzer output
- On-chip crystal, RC and 32768Hz crystal oscillator
- Power-down function and wake-up features reduce power consumption
- 16-level subroutine nesting
- 8-channel 12-bit resolution A/D converter
- 16-channel 8-bit PWM output shared with 16 I/O lines
- Bit manipulation instruction
- 16-bit table read instruction
- Up to 0.33μs instruction cycle with 12MHz system clock
- 63 powerful instructions
- All instructions in 1 or 2 machine cycles
- Low voltage reset/detector function
- 100-pin LQFP package

General Description

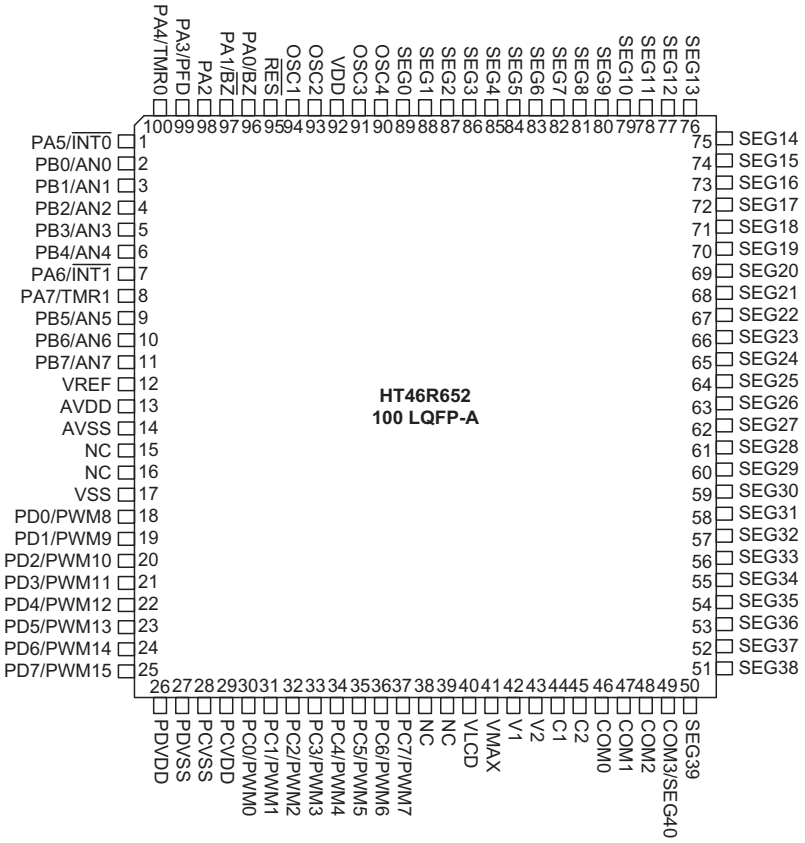
The HT46R652 is an 8-bit, high performance, RISC architecture microcontroller devices specifically designed for A/D product applications that interface directly to analog signals and which require an LCD Interface.

The advantages of low power consumption, I/O flexibility, timer functions, oscillator options, multi-channel A/D Converter, Pulse Width Modulation function, Power-down and wake-up functions, in addition to a flexible and configurable LCD interface enhance the versatility of these devices to control a wide range of applications requiring analog signal processing and LCD interfacing, such as electronic metering, environmental monitoring, handheld measurement tools, motor driving, etc. for both the industrial and home appliance application areas.

Block Diagram



Pin Assignment



Pin Description

Pin Name	I/O	Options	Description
PA0/BZ PA1/BZ PA2 PA3/PFD PA4/TMR0 PA5/INT0 PA6/INT1 PA7/TMR1	I/O	Wake-up Pull-high Buzzer PFD	Bidirectional 8-bit input/output port. Each individual pin on this port can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pins on the port have pull-high resistors. Pins PA0, PA1 and PA3 are pin-shared with BZ, BZ and PFD respectively. Pins PA5, PA6, PA4 and PA7 are pin-shared with INT0, INT1, TMR0 and TMR1 respectively.
PB0/AN0~ PB7/AN7	I/O	Pull-high	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pins on the port have pull-high resistors. PB is pin-shared with the A/D input pins. The A/D inputs are selected via software instructions. Once a PB line is selected as an A/D input, the I/O function and pull-high resistor functions are disabled automatically.
PC0/PWM0~ PC7/PWM7	I/O	Pull-high PWM	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine if all pins on the port have pull-high resistors. A configuration option determines if all of the pins on this port are to be used as PWM outputs. Individual pins cannot be selected to have a PWM function.
PD0/PWM8~ PD7/PWM15	I/O	Pull-high PWM	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pins on the port have pull-high resistors. A configuration option for each pin on this port determines if each pin is to be used as a PWM output.
VLCD	I	—	LCD power supply
VMAX	I	—	IC maximum voltage connect to VDD, VLCD or V1
V1, V2, C1, C2	I	—	Voltage pump
COM0~COM2 COM3/SEG40	O	1/3 or 1/4 Duty	SEG40 can be set as a segment or as a common output driver for LCD panel by options. COM0~COM2 are outputs for the LCD panel.
SEG0~SEG39	O	Logical Output	LCD driver outputs for the the LCD panel segments. SEG0~SEG23 can be configured as logical outputs via a configuration option.
OSC1 OSC2	I O	Crystal or RC	OSC1 and OSC2 are connected to an RC network or external crystal (determined by a configuration option) for the internal system clock. If the RC system clock is selected, OSC2 can be used to measure the system clock at 1/4 frequency. The system clock may also be sourced from the RTC oscillator, in which case these two pins can be left floating.
OSC3 OSC4	I O	RTC or System Clock	Real time clock oscillator. OSC3 and OSC4 are connected to a 32768Hz crystal oscillator for timing purposes or to form a system clock source, depending on configuration options.
RES	I	—	Schmitt trigger reset input, active low
VDD	—	—	Positive power supply
AVDD/VREF			Analog positive power supply and A/D converter reference input voltage.
PCVDD	—	—	Port C positive power supply
PDVDD	—	—	Port D positive power supply
VSS/AVSS	—	—	Negative power supply and analog negative power supply, ground
PCVSS	—	—	Port C negative power supply, ground
PDVSS	—	—	Port D negative power supply, ground

Note: Individual pins on PC cannot be selected as a PWM output, if the PWM configuration option is selected for this port then all pins on PC will be setup as PWM outputs.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}–0.3V to V_{SS}+6.0V
Input Voltage ..... V_{SS}–0.3V to V_{DD}+0.3V
I_{OL} Total 300mA
Total Power Dissipation 500mW

Storage Temperature ..... –50°C to 125°C
Operating Temperature ..... –40°C to 85°C
I_{OH} Total ..... –200mA

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_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
		—	f_{SYS}=12MHz	3.3	—	5.5	V
AV_{DD}	Analog Operating Voltage*	—	V_{REF}=AV_{DD}	3.0	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC)	3V	No load, ADC off f_{SYS}=4MHz	—	1	2	mA
		5V		—	3	5	mA
I_{DD2}	Operating Current (RC OSC)	3V	No load, ADC off f_{SYS}=4MHz	—	1	2	mA
		5V		—	3	5	mA
I_{DD3}	Operating Current (Crystal OSC, RC OSC)	5V	No load, ADC off f_{SYS}=12MHz	—	4	8	mA
I_{DD4}	Operating Current (f_{SYS}=32768Hz)	3V	No load, ADC off	—	0.3	0.6	mA
		5V		—	0.6	1	mA
I_{STB1}	Standby Current (*f_S=T1)	3V	No load, system HALT LCD off at HALT	—	—	1	μA
		5V		—	—	2	μA
I_{STB2}	Standby Current (*f_S=32.768kHz OSC)	3V	No load, system HALT LCD on at HALT, C type	—	2.5	5	μA
		5V		—	10	20	μA
I_{STB3}	Standby Current (*f_S=WDT RC OSC)	3V	No load, system HALT LCD on at HALT, C type	—	2	5	μA
		5V		—	6	10	μA
I_{STB4}	Standby Current (*f_S=32.768kHz OSC)	3V	No load, system HALT LCD on at HALT, R type, 1/2 bias, VLCD=VDD (Low bias current option)	—	17	30	μA
		5V		—	34	60	μA
I_{STB5}	Standby Current (*f_S=32.768kHz OSC)	3V	No load, system HALT LCD on at HALT, R type, 1/3 bias, VLCD=VDD (Low bias current option)	—	13	25	μA
		5V		—	28	50	μA
I_{STB6}	Standby Current (*f_S=WDT RC OSC)	3V	No load, system HALT LCD on at HALT, R type, 1/2 bias, VLCD=VDD (Low bias current option)	—	14	25	μA
		5V		—	26	50	μA
I_{STB7}	Standby Current (*f_S=WDT RC OSC)	3V	No load, system HALT LCD on at HALT, R type, 1/3 bias, VLCD=VDD (Low bias current option)	—	10	20	μA
		5V		—	19	40	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL1}	Input Low Voltage for I/O Ports, TMR and $\overline{INT}$	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR and $\overline{INT}$	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
V_{LVR}	Low Voltage Reset Voltage	—	—	2.7	3.0	3.3	V
V_{LVD}	Low Voltage Detector Voltage	—	—	3.0	3.3	3.6	V
I_{OL1}	I/O Port (PA, PB) and Segment Logic Output Sink Current	3V	$V_{OL}=0.1V_{DD}$	6	12	—	mA
		5V		10	25	—	mA
I_{OH1}	I/O Port (PA, PB) and Segment Logic Output Source Current	3V	$V_{OH}=0.9V_{DD}$	–2	–4	—	mA
		5V		–5	–8	—	mA
I_{OL2}	I/O Port (PC, PD) Sink Current	3V	$V_{OL}=0.1V_{DD}$	10	20	—	mA
		5V		25	40	—	mA
I_{OH2}	I/O Port (PC, PD) Source Current	3V	$V_{OH}=0.9V_{DD}$	–10	–20	—	mA
		5V		–25	–40	—	mA
I_{OL3}	LCD Common and Segment Current	3V	$V_{OL}=0.1V_{DD}$	210	420	—	μA
		5V		350	700	—	μA
I_{OH3}	LCD Common and Segment Current	3V	$V_{OH}=0.9V_{DD}$	–80	–160	—	μA
		5V		–180	–360	—	μA
R_{PH}	Pull-high Resistance of I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V_{AD}	A/D Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	ADC Input Reference Voltage Range	—	AV_{DD}=3V	1.3	—	A_{VDD}	V
			AV_{DD}=5V	1.5	—	A_{VDD}	V
DNL	ADC Differential Non-Linear	—	—	—	—	±2	LSB
INL	ADC Integral Non-Linear	—	—	—	±2.5	±4	LSB
RESOLU	Resolution	—	—	—	—	12	Bits
I_{ADC}	Additional Power Consumption if A/D Converter is Used	3V	—	—	0.5	1	mA
		5V		—	1.5	3	mA

Note: "*f_S" please refer to clock option of WDT

"*" Voltage level of AV_{DD} and V_{DD} must be the same.

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS1}	System Clock	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	12000	kHz
f_{SYS2}	System Clock (32768Hz Crystal OSC)	—	2.2V~5.5V	—	32768	—	Hz
f_{RTCOSC}	RTC Frequency	—	—	—	32768	—	Hz
f_{TIMER}	Timer I/P Frequency (TMR0/TMR1)	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	12000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Power-up or wake-up from HALT	—	1024	—	*t_{SYS}
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1	2	ms
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{AD}	A/D Clock Period	—	—	1	—	—	μs
t_{ADC}	A/D Conversion Time	—	—	—	80	—	t_{AD}
t_{ADCS}	A/D Sampling Time	—	—	—	32	—	t_{AD}

Note: *t_{SYS}= 1/f_{SYS1} or 1/f_{SYS2}

Functional Description

Execution Flow

The system clock is derived from either a crystal or an RC oscillator or a 32768Hz crystal 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.

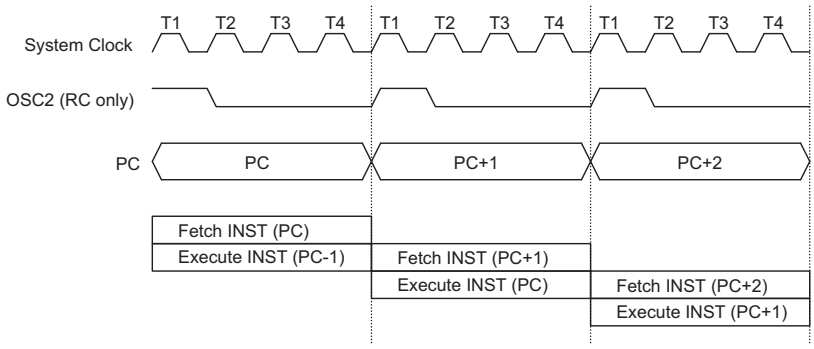
Program Counter – PC

The program counter, PC, is 13 bits wide and it controls the sequence in which the instructions stored in the Program Memory are executed. The contents of the PC can specify a maximum of 8192 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.



Execution Flow

Mode	Program Counter												
	*12	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0	0
External Interrupt 0	0	0	0	0	0	0	0	0	0	0	1	0	0
External Interrupt 1	0	0	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	0	0	1	1	0	0
Timer/Event Counter 1 Overflow	0	0	0	0	0	0	0	0	1	0	0	0	0
Time Base Interrupt	0	0	0	0	0	0	0	0	1	0	1	0	0
RTC Interrupt	0	0	0	0	0	0	0	0	1	1	0	0	0
Skip	Program Counter+2												
Loading PCL	*12	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#12	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Note: *12~*0: Program counter bits
#12~#0: Instruction code bits
S12~S0: Stack register bits
@7~@0: PCL bits

The lower byte of the PC, known as PCL is a readable and writeable register. Moving data into the PCL performs a short jump. The destination is within 256 locations.

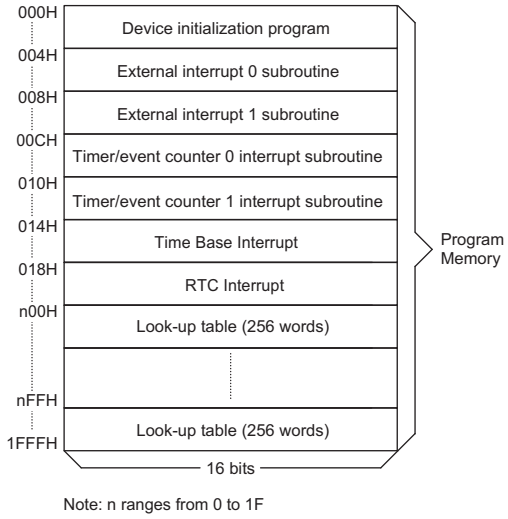
When a control transfer takes place, an additional dummy cycle is required.

Program Memory

The program memory is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 8192×16 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 a device reset, the program always begins execution at this location.
- Location 004H
  Location 004H is reserved for the external interrupt service program. If the $\overline{INT0}$ input pin is activated, and the interrupt is enabled, and the stack is not full, the program begins execution at location 004H.



Note: n ranges from 0 to 1F

Program Memory

- Location 008H
  Location 008H is reserved for the external interrupt service program also. If the $\overline{INT1}$ input pin is activated, and the interrupt is enabled, and the stack is not full, the program begins execution at location 008H.
- Location 00CH
  Location 00CH is reserved for the Timer/Event Counter 0 interrupt service program. If a timer interrupt results from a Timer/Event Counter 0 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.
- Location 010H
  Location 010H is reserved for the Timer/Event Counter 1 interrupt service program. If a timer interrupt results from a Timer/Event Counter 1 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 010H.
- Location 014H
  Location 014H is reserved for the Time Base interrupt service program. If a Time Base interrupt occurs, and the interrupt is enabled, and the stack is not full, the program begins execution at location 014H.
- Location 018H
  Location 018H is reserved for the real time clock interrupt service program. If a real time clock interrupt occurs, and the interrupt is enabled, and the stack is not full, the program begins execution at location 018H.
- Table location
  Any location in the Program Memory 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 which is the Table Higher-order byte register. 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 register is read only, and the table pointer, TBLP, is a read/write register, 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 Program Memory depending upon the user's requirements.

Instruction(s)	Table Location												
	*12	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P12	P11	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: *12~*0: Table location bits
P12~P8: Current program counter bits
@7~@0: Table pointer bits

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 16 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, which 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 a device 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 will be serviced. This feature prevents a 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 as only the most recent sixteen return addresses are stored.

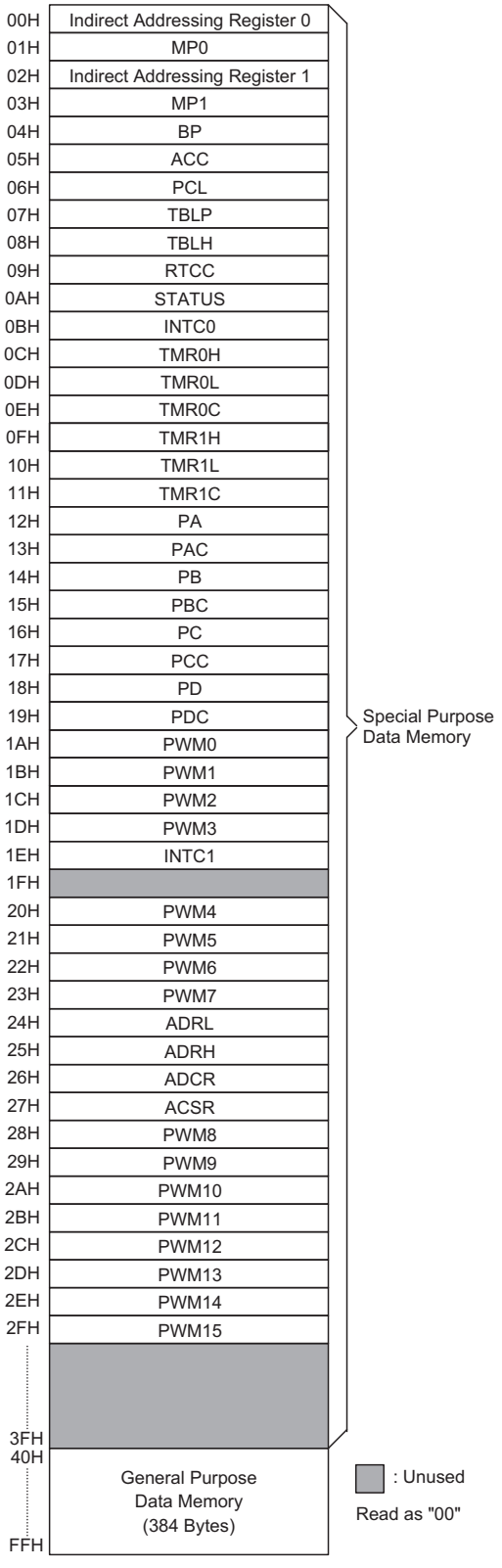
Data Memory – RAM

The data memory has a structure of 431×8 bits, and is divided into two functional groups, namely the special function registers, 47×8 bits, and the general purpose data memory, Bank0: 192×8 bits and Bank2: 192×8 bits most of which are readable/writeable, although some are read only. The special function registers are overlapped in every bank.

Any unused remaining space before 40H is reserved for future expanded usage and if read will return a ″00H″ value. The Data Memory space before 40H will overlap in each bank.

The general purpose data memory, addressed from 40H to FFH (Bank0; BP=0 or Bank2; BP=2), 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 ″SET [m].i″ and ″CLR [m].i″. They are also indirectly accessible through memory pointer registers, MP0 and MP1.

After first setting up BP to the value of ″01H″ or ″02H″ to access either bank 1 or bank 2 respectively, these banks must then be accessed indirectly using the Memory Pointer MP1. With BP set to a value of either ″01H″ or ″02H″, using MP1 to indirectly read or write to the data memory areas with addresses from 40H~FFH, will result in operations to either bank 1 or bank 2. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of BP.



RAM Mapping

Indirect Addressing Register

Locations 00H and 02H are for indirect addressing registers that are not physically implemented. Any read/write operation to locations [00H] and [02H] accesses the Data Memory locations pointed to by MP0 and MP1 respectively. Reading location 00H or 02H indirectly will return a result of 00H. Writing indirectly will lead to no operation.

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

Accumulator – ACC

The accumulator, ACC, is related to the ALU operations. It is also mapped to location 05H in the Data Memory 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 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 register, however, may yield different results from those intended. The TO and PDF flags can only be changed by a Watchdog Timer overflow, a device 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 a 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 precautions should be taken to save it properly.

Interrupts

The device provides two external interrupts, two internal timer/event counter interrupts, an internal time base interrupt and an internal real time clock interrupt. The interrupt control register 0, INTC0, and the interrupt control register 1, INTC1, 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 as the EMI bit is automatically cleared, which may prevent any further interrupt nesting. Other interrupt requests may take place during this interval, but only the interrupt request flag will be recorded. If a certain interrupt requires servicing within the

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

service routine, the EMI bit and the corresponding bit of INTC0 or of INTC1 may be set in order to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the stack pointer is decremented. If immediate service is desired, the stack should be prevented from becoming full.

All these interrupts will generate 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.

External interrupts are triggered by a an edge transition on $\overline{INT0}$ or $\overline{INT1}$, when the related interrupt request flag, EIF0; bit 4 of INTC0, EIF1; bit 5 of INTC0, is set. The trigger edge type, high to low, low to high, or both low to high and or high to low is determined by configuration option. After the interrupt is enabled, if the stack is not full, and the external interrupt is active, a subroutine call to location 04H or 08H occurs. The interrupt request flag, EIF0 or EIF1, and EMI bits are all cleared to disable other maskable interrupts.

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

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

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

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1=enabled; 0=disabled)
1	EEI0	Controls the external interrupt 0 (1=enabled; 0=disabled)
2	EEI1	Controls the external interrupt 1 (1=enabled; 0=disabled)
3	ET0I	Controls the Timer/Event Counter 0 interrupt (1=enabled; 0=disabled)
4	EIF0	External interrupt 0 request flag (1=active; 0=inactive)
5	EIF1	External interrupt 1 request flag (1=active; 0=inactive)
6	T0F	Internal Timer/Event Counter 0 request flag (1=active; 0=inactive)
7	—	For test mode used only. Must be written as ″0″; otherwise may result in unpredictable operation.

INTC0 (0BH) Register

Bit No.	Label	Function
0	ET1I	Controls the Timer/Event Counter 1 interrupt (1=enabled; 0=disabled)
1	ETBI	Controls the time base interrupt (1=enabled; 0:disabled)
2	ERTI	Controls the real time clock interrupt (1=enabled; 0:disabled)
3, 7	—	Unused bit, read as ″0″
4	T1F	Internal Timer/Event Counter 1 request flag (1=active; 0=inactive)
5	TBF	Time base request flag (1=active; 0=inactive)
6	RTF	Real time clock request flag (1=active; 0=inactive)

INTC1 (1EH) Register

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,a "RET" or "RETI" instruction may be executed. 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 0	1	04H
External interrupt 1	2	08H
Timer/Event Counter 0 overflow	3	0CH
Timer/Event Counter 1 overflow	4	10H
Time base interrupt	5	14H
Real time clock interrupt	6	18H

The EMI, EEI0, EEI1, ET0I, ET1I, ETBI, and ERTI bits are all used to control the enable/disable status of the interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags, RTF, TBF, T0F, T1F, EIF1, EIF0 are set, they remain in the INTC1 or INTC0 register respectively until the interrupts are serviced or cleared by a software instruction.

It is recommended that a program should not use a "CALL" instruction within the interrupt subroutine. This is 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, execution of a "CALL" in the interrupt subroutine may damage the original control sequence.

Oscillator Configuration

The device provides three oscillator circuits for the system clock, namely an RC oscillator, a crystal oscillator and an RTC 32768Hz crystal oscillator, the choice of which is determined by configuration option. When the device enters the Power Down mode, the RC or crystal oscillator will cease running to conserve power. The 32768Hz crystal oscillator, however, will keep running when the device is in the Power Down mode. If the 32768Hz crystal oscillator is selected as the system oscillator, when the device enters the Power Down mode, the system oscillator keeps running, but instruction execution will cease. Since the 32768Hz oscillator is also designed for timing purposes, the internal timing functions, RTC, time base and WDT, continue to operate even when the system enters the Power Down mode.

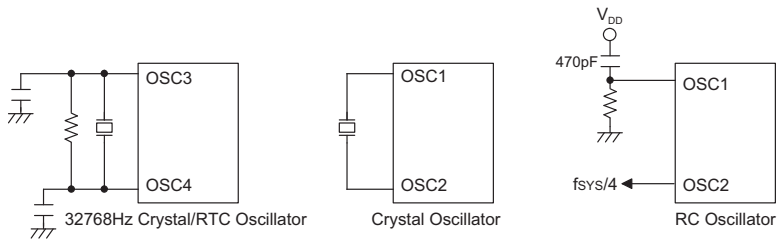
If the RC oscillator is used, an external resistor connected between pins OSC1 and VSS is required, whose value should range from 24kΩ to 1MΩ. The system clock, divided by 4, can be monitored on pin OSC2 if a pull-high resistor is connected. This pin can be used to synchronise external logic. The RC oscillator provides the most cost effective solution. However, as the frequency may vary with VDD, temperature, and process variations, it is therefore not suitable for timing sensitive operations where an accurate oscillator frequency is desired.

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

The other oscillator circuit, which is a real time clock, requires a 32768Hz crystal oscillator to be connected between OSC3 and OSC4.

The RTC oscillator circuit can be controlled to start-up quickly by setting the "QOSC" bit, which is bit 4 of RTCC. It is recommended to turn on the quick start-up function during power on, and then turn it off again after 2 seconds.

The WDT oscillator is a free running on-chip RC oscillator, which does not require external components. Although when the system enters the Power Down mode and the system clock stops, the WDT oscillator still operates with a nominal period of approximately 65μs at 5V. The WDT oscillator can be disabled by a configuration option to conserve power.



System Oscillator

Note: *32768Hz crystal enable condition: for WDT clock source or for system clock source.

Watchdog Timer – WDT

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

If the internal WDT oscillator, which is an RC oscillator with a nominal period of 65μs at 5V, is selected, it is divided by 2^{12}~2^{15} , the actual ratio chosen by configuration option, to get the WDT time-out period. The minimum period for the WDT time-out period is about 300ms~600ms. This time-out period may vary with temperature, VDD and process variations. By selection the WDT configuration option, longer time-out periods can be realised. If the WDT time-out is selected as 2^{15}, the maximum time-out period is divided by 2^{15}~2^{16} about 2.1s~4.3s. If the WDT oscillator is disabled, the WDT clock may still come from the instruction clock and operate in the same manner except that in the Power Down mode the WDT will stop counting and lose its protecting function. If the device operates in a noisy environment, using the WDT internal RC oscillator is strongly recommended, since the HALT instruction will stop the system clock.

The WDT overflow under normal operation initiates a device reset which sets the status bit "TO". In the Power Down mode, the overflow initiates a warm reset, in which only the Program Counter and Stack Pointer are reset to zero. To clear the contents of the WDT, there are three methods that can be adopted. These are, an external reset, which is a low level on the $\overline{RES}$ pin, a software instruction and a "HALT" instruction. There are two types of software instructions; a 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 a 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. In the case where the two "CLR WDT1" and "CLR WDT2" instruction are chosen (i.e., CLR WDT times equal two), these two instructions have to be executed to clear the WDT; otherwise, the WDT may reset the chip due to a time-out.

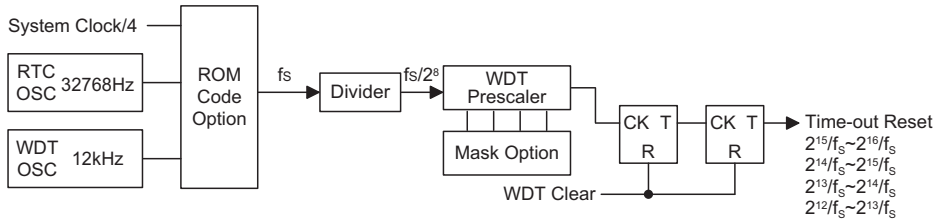
Multi-function Timer

The device provides a multi-function timer for the WDT, time base and RTC but with different time-out periods. The multi-function timer consists of an 8-stage divider and a 7-bit prescaler, with the clock source coming from the WDT OSC, the RTC OSC or the instruction clock, which is the system clock divided by 4. The multi-function timer also provides a selectable frequency signal, whose division ratio ranges from $f_S/2^2$ to $f_S/2^8$, for LCD driver circuits, and a selectable frequency signal, ranging from $f_S/2^2$ to $f_S/2^9$, for the buzzer output selectable via configuration options.

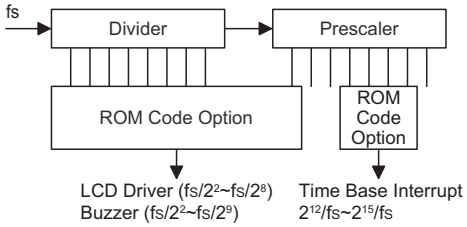
It is recommended to select a frequency as close as possible to 4kHz for the LCD driver circuits to obtain the best display clarity.

Time Base

The time base offers a periodic time-out period to generate a regular internal interrupt. Its time-out period ranges from $2^{12}/f_S$ to $2^{15}/f_S$ selected by a configuration option. If a time base time-out occurs, the related interrupt request flag, TBF; bit 5 of INTC1, will be set. If the interrupt is enabled, and the stack is not full, a subroutine call to location 14H occurs.



Watchdog Timer



Time Base

Real Time Clock – RTC

The real time clock, RTC, is operated in the same manner as the time base in that it is used to supply a regular internal interrupt. Its time-out period ranges from $f_S/2^8$ to $f_S/2^{15}$, the value being setup using software. Writing data to the RT2, RT1 and RT0 bits in the RTCC register, provides various time-out periods. If an RTC time-out occurs, the related interrupt request flag, RTF; bit 6 of INTC1, is set. But if the interrupt is enabled, and the stack is not full, a subroutine call to location 18H occurs.

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

Note: * not recommended to be used

Power Down Operation

The Power Down mode is entered by the execution of a ″HALT″ instruction and results in the following.

- The system will cease to run but the WDT oscillator will keep running if the WDT oscillator or the real time clock is selected.
- The contents of the Memory and registers remain unchanged.
- The WDT will be cleared and starts recounting, if the WDT clock source is sourced from the WDT oscillator or the real time clock oscillator.
- All I/O ports maintain their original status.
- The PDF flag is set but the TO flag is cleared.
- The LCD driver will maintain its function if the WDT OSC or RTC OSC is selected.

The system will wake up from the Power Down mode via an external reset, an interrupt, an external falling edge signal on port A or a WDT overflow. An external reset will generate a device initialisation, while a WDT overflow performs a ″warm reset″. After examining the TO and PDF flags, the reason for the device reset can be determined. The PDF flag is cleared by a system power-up or by executing the ″CLR WDT″ instruction, and is set by executing the ″HALT″ instruction. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and the Stack Pointer, but 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 bit in port A can be independently selected to wake up the device by a configuration option. If awakened by an I/O port stimulus, the program resumes execution at the next instruction following the ″HALT″ instruction. Awakening from an interrupt, two sequence may occur. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program resumes execution at the next instruction. But if the interrupt is enabled, and the stack is not full, a regular interrupt response takes place.

When an interrupt request flag is set before entering the Power Down mode, 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 a wake-up results in the next instruction execution, the execution will be performed immediately after the dummy period is finished.

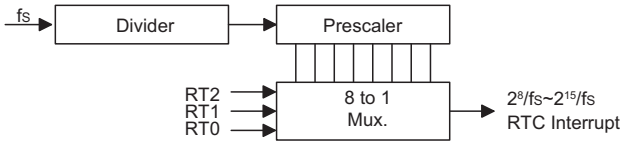
To minimise power consumption, all the I/O pins should be carefully managed before entering the Power Down mode.

Reset

There are three ways in which a reset may occur.

- $\overline{RES}$ pin is pulled low during normal operation
- $\overline{RES}$ pin is pulled low when in the Power Down mode
- A WDT time-out during normal operation

A WDT time-out when the device is in the Power Down mode differs from other device reset conditions, as it will perform a ″warm reset″ that resets only the program counter and the SP but leaves the other circuits in their original state. Some registers remain unaffected during other reset conditions. Most registers are reset to their initial condition once the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between the different types of device resets.



Real Time Clock

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
1	u	WDT time-out during normal operation
1	1	WDT Wake-up

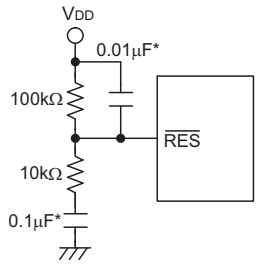
Note: "u" stands for unchanged

To guarantee that the system oscillator is started and stabilised, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system awakes from the Power Down mode or during a power up. When awakened from the Power Down mode or during a system power-up, the SST delay will be added.

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

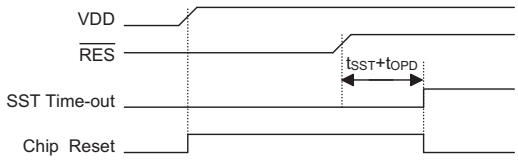
The following table shows how various components of the microcontroller are affected after a power-on reset occurs.

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

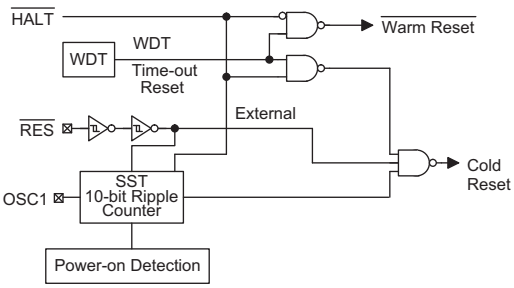


Reset Circuit

Note: "*" Make the length of the wiring, which is connected to the $\overline{RES}$ pin as short as possible, to avoid noise interference.



Reset Timing Chart



Reset Configuration

The register states are summarised in the following table:

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
TMR0H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u 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 1---	0000 1---	0000 1---	0000 1---	uuuu u---
Program Counter	0000H	0000H	0000H	0000H	0000H
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
RTCC	--00 0111	--00 0111	--00 0111	--00 0111	--uu uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PWM0~ PWM15	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	xxxx ----	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	---- --00	---- --00	---- --00	---- --00	---- --uu

Note: "*" stands for warm reset

"u" stands for unchanged

"x" stands for unknown

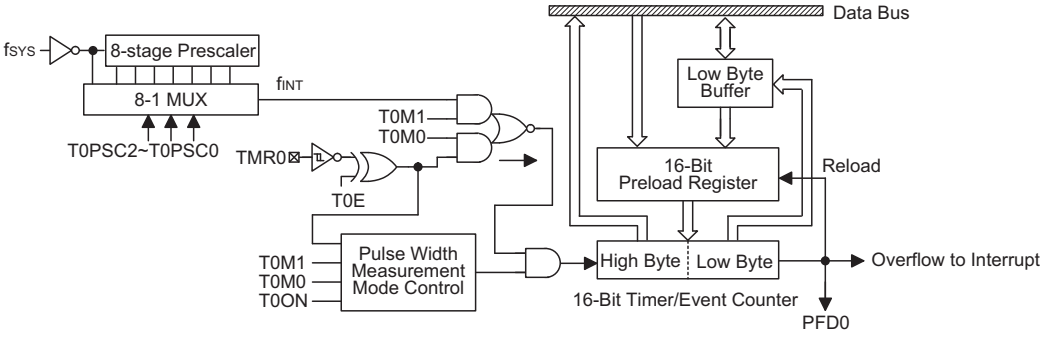
Timer/Event Counter

Two timer/event counters, Timer/Event Counter 0 and Timer/Event Counter,1 are implemented within the microcontroller. Timer/Event Counter 0 is a 16-bit programmable count-up counter whose clock may come from an external or internal source. The internal clock source will come from f_{SYS}. Timer/Event Counter 1 is also a 16-bit programmable count-up counter whose clock may come from an external source or an internal source. The internal clock source comes from $f_{SYS}/4$ or a 32768Hz source, selected by a configuration 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.

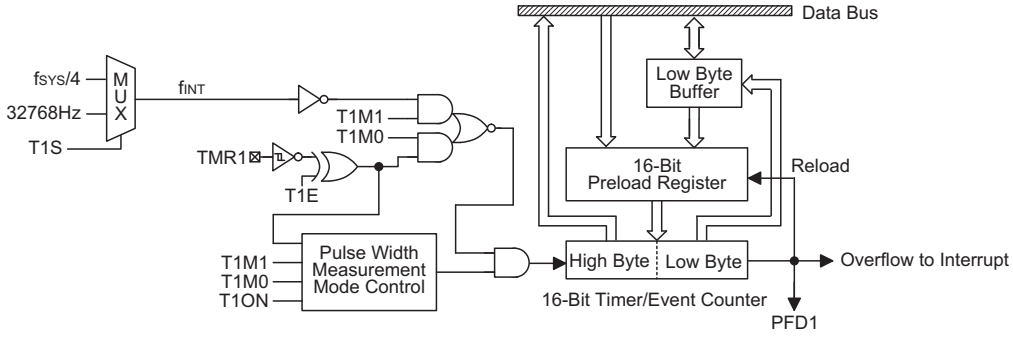
There are three registers associated with Timer/Event Counter 0; TMR0H, TMR0L and TMR0C, and another three for Timer/Event Counter 1; TMR1H, TMR1L and TMR1C. Writing to TMR0L and TMR1L will only put the written data into an internal lower-order byte buffer (8-bit) while writing to TMR0H and TMR1H will transfer the specified data and the contents of the lower-order byte buffer to the TMR0H/TMR1H and TMR0L/TMR1L registers. The Timer/Event Counter 0/1 preload register is changed with each TMR0H/TMR1H write operations. Reading TMR0H/TMR1H will latch the contents of TMR0H/TMR1H and TMR0L/TMR1L counters to the destination and the lower-order byte buffer, respectively. Reading TMR0L/TMR1L will only read the contents of the lower-order byte buffer. The TMR0C and TMR1C registers are the Timer/Event Counter control registers, which control the operating mode, the timer enable or disable and the active edge type.

The T0M0, T0M1 and T1M0, T1M1 bits define the timer operational mode. The event count mode is used to count external events, which requires that the clock source comes from an external TMR0 or TMR1 pin. The timer mode functions as a normal timer with the clock source coming from the internally selected clock source. The pulse width measurement mode can be used to count the high or low level duration of an external signal on pin TMR0 or TMR1, with the count value based on the internally selected clock source.

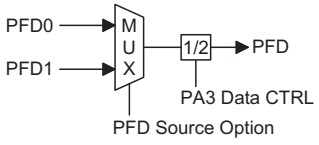
In the event count or timer mode, the timer/event counter starts counting from the current contents in the



Timer/Event Counter 0



Timer/Event Counter 1



PFD Source Option

timer/event counter and ends at FFFFH. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag, T0F; bit 6 of INTC0 and T1F; bit 4 of INTC1. In the pulse width measurement mode with the values of the T0ON/T1ON and T0E/T1E bits equal to 1, after the TMR0 or TMR1 pin has received a transient from low to high, or high to low if the T0E/T1E bit is "0", it will start counting until the TMR0 or TMR1 pin returns to its original level and resets the T0ON/T1ON bit. The measured result remains in the timer/event counter even if the activated transient occurs again. In other words, only a single measurement can be made until the T0ON/T1ON is again set. In this operation mode, the timer/event counter begins counting, not according to the logic level on the pins, but according 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.

Bit No.	Label	Function
0 1 2	T0PSC0 T0PSC1 T0PSC2	To define the prescaler stages. T0PSC2, T0PSC1, T0PSC0= 000: $f_{INT}=f_{SYS}$ 001: $f_{INT}=f_{SYS}/2$ 010: $f_{INT}=f_{SYS}/4$ 011: $f_{INT}=f_{SYS}/8$ 100: $f_{INT}=f_{SYS}/16$ 101: $f_{INT}=f_{SYS}/32$ 110: $f_{INT}=f_{SYS}/64$ 111: $f_{INT}=f_{SYS}/128$
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	—	Unused bit, read as "0"
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
0~2	—	Unused bit, read as "0"
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=32768Hz)
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

To enable a counting operation, the Timer ON bit, T0ON or T1ON 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 and can also be used to drive the PFD (Programmable Frequency Divider) output on pin PA3, a function which is selected by a configuration option. If PA3 is setup as a PFD output, there are two types of selections. One is to use PFD0 as the PFD output, the other is to use PFD1 as the PFD output. PFD0 and PFD1 are the timer overflow signals of the Timer/Event Counter 0 and Timer/Event Counter 1 respectively. No matter what the operation mode is, writing a 0 to ET0I or ET1I disables the related interrupt service. When the PFD function is selected, executing a ″SET [PA].3″ instruction will enable the PFD output and executing a ″CLR [PA].3″ instruction will disable the PFD output.

In cases where the timer/event counter is turned off, writing data to the timer/event counter preload register will also reload the new data into the timer/event counter. But if the timer/event counter is turned on, data written to the timer/event counter will only be stored in the timer/event counter preload register. The timer/event counter will continue with its normal operation until an overflow occurs.

When the timer/event counter is read, the clock is blocked to avoid errors, which may result in a counting error and should therefore be taken into account by the programmer. It is strongly recommended to load a desired value into the timer registers first before turning on the related timer/event counter, for proper operation since the initial value of the timer registers is unknown. Due to the timer/event counter scheme, the programmer should pay special attention with instructions to enable then disable 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.

Bits 0~2 of TMR0C can be used to define the pre-scaling stages of the internal clock sources of the timer/event counter. The overflow signal of the timer/event counter can be used to generate the PFD signal. The timer prescaler is also used as the the PWM counter.

Input/Output Ports

There are 32 bidirectional input/output lines in the microcontroller, divided among several ports labeled as PA, PB, PC and PD. 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]″. For output operations, all the data is latched and remains unchanged until the output latch is rewritten.

Each port has has its own Port Control Register, known as PAC, PBC, PCC and PDC to control the input/output configuration. With this control register, a CMOS output or Schmitt Trigger input with or without pull-high resistor structures can be reconfigured dynamically under software control. To function as an input, the corresponding bit of the control register must contain a ″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.

After a device reset, as the Port Control Registers will be set high, the input/output lines will be setup as inputs, and will be at a high level or in a floating state, depending on the pull-high configuration options. Each bit of these input/output latches can be set or cleared by the ″SET [m].i″ and ″CLR [m].i″ 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 pull-high options. Once the pull-high option is selected, the I/O port has a pull-high resistor. It should be noted that a non-pull-high I/O port operating in an input mode will be in a floating condition.

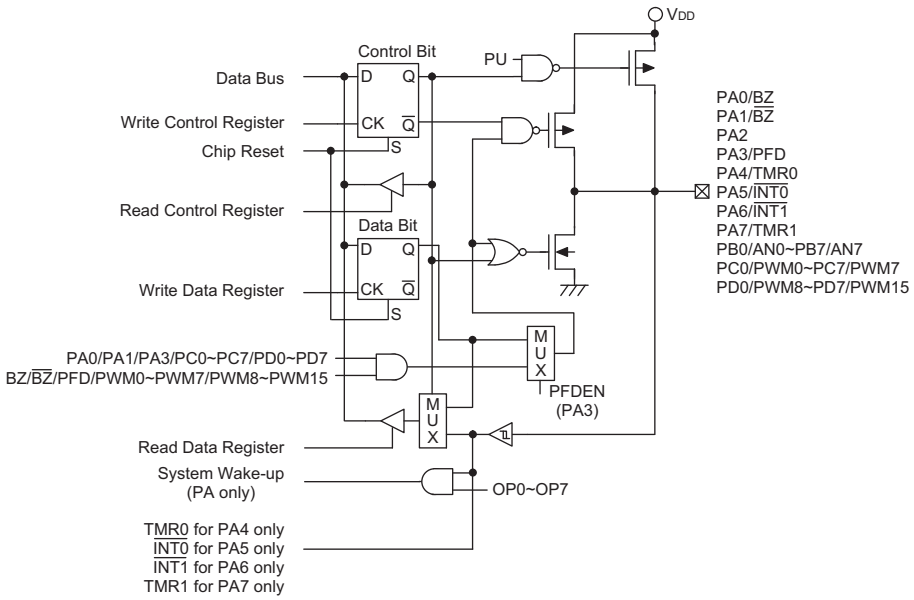
Pin PA3 is pin-shared with the PFD signal. If the PFD configuration option is selected, the output signal for PA3, if it setup as an output, will be the PFD signal generated by the timer/event counter overflow signal. If setup as an input the PA3 will always retain its input function. Once the PFD configuration option is selected, the PFD output signal can be controlled by the PA3 data register. Writing a ″1″ to the PA3 data register will enable the PFD output function while writing a ″0″ will force the PA3 pin to remain in a low condition. The I/O functions of the PA3 pin are shown in the table.

I/O Mode	I/P (Normal)	O/P (Normal)	I/P (PFD)	O/P (PFD)
PA3	Logical Input	Logical Output	Logical Input	PFD (Timer on)

Note: The PFD frequency is the timer/event counter overflow frequency divided by 2.

Pins PA0, PA1, PA3, PA5, PA6, PA4 and PA7 are pin-shared with the BZ, $\overline{BZ}$, PFD, $\overline{INT0}$, $\overline{INT1}$, TMR0 and TMR1 pins respectively.

The PA0 and PA1 pins are pin-shared with the BZ and $\overline{BZ}$ signal, respectively. If the BZ/$\overline{BZ}$ option is selected, the output signal in the output mode of PA0/PA1 will be the buzzer signal, which is generated by the multi-func-



Input/Output Ports

tion timer. If the pins are setup as inputs then they will always retain their input function. Once the BZ/$\overline{BZ}$ configuration option is selected, the buzzer output signal is controlled by the PA0/PA1 data register.

The PA0/PA1 pins I/O functions are shown in the table.

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_0	0	1	0	1
PA1 Data	X	D	X	X	X	D1	D	D	X	X
PA0 Pad Status	I	I	D	0	B	D_0	0	B	0	B
PA1 Pad Status	I	D	I	I	I	D_1	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

The PB port is also used for the A/D converter inputs. The PWM outputs are shared with pins PC0~PC7 and PD0~PD7. If the PWM function is enabled, the PWM0~PWM15 outputs will appear on pins PC0~PC7 and PD0~PD7, if PC0~PC7 and PD0~PD7 are setup as outputs. Writing a ″1″ to the PC0~PC7 and PD0~PD7 data registers will enable the PWM output function while writing a ″0″ will force PC0~PC7 and PD0~PD7 to remain at a ″0″level. The I/O functions of PC0~PC7 and PD0~PD7 are shown in the table.

I/O Mode	I/P (Normal)	O/P (Normal)	I/P (PWM)	O/P (PWM)
PC0~PC7, PD0~PD7	Logical Input	Logical Output	Logical Input	PWM0~ PWM15

Any unused pins must be carefully managed to ensure that there are no floating input lines which will result in increased power consumption. It is therefore recommended that any unused pins are setup as outputs or connected to a pull-high resistor if setup as inputs.

The definitions of the PFD control signals and the PFD output frequencies are listed in the following table.

Timer	Timer Preload Value	PA3 Data Register	PA3 Pad State	PFD Frequency
OFF	X	0	0	X
OFF	X	1	U	X
ON	N	0	0	X
ON	N	1	PFD	$f_{TMR}/[2\times(M-N)]$

Note: ″X″ stands for unused
″U″ stands for unknown
″M″ is ″65536″ for PFD0 or PFD1
″N″ is the timer/event counter preload value
″f_{TMR}″ is the input clock frequency for the timer/event counter

PWM

The microcontroller provides a 16-channel PWM output shared with pins PC0~PC7 and PD0~PD7. Its output signals can be configured as (6+2) or (7+1) type dependent upon configuration options. Each PWM channel has its own 8-bit data register, denoted as PWM0~PWM15. The PWM frequency source comes from f_{SYS}. Once the PC0~PC7 and PD0~PD7 pins are selected as PWM outputs, if the pins are setup as outputs, writing a "1" to the PC0~PC7 and PD0~PD7 data registers will enable the corresponding PWM output function, while writing a "0" will force the PC0~PC7 and PD0~PD7 pins to remain at "0".

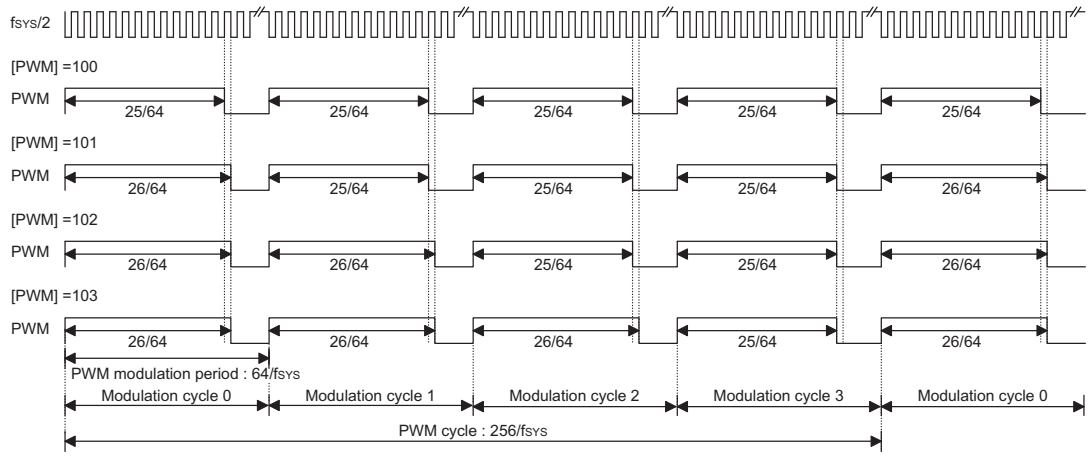
In the (6+2) mode, the PWM cycle is divided into four modulation cycles, modulation cycle 0~modulation cycle 3. Each modulation cycle has 64 PWM input clock periods. In the (6+2) mode, the contents of each PWM register is divided into two groups. Group 1 of the PWM register is denoted by a DC which is the value of PWM.7~PWM.2. Group 2 is denoted by AC which is the value of PWM.1~PWM.0.

In the (6+2) mode, the duty cycle of each modulation cycle is shown in the table.

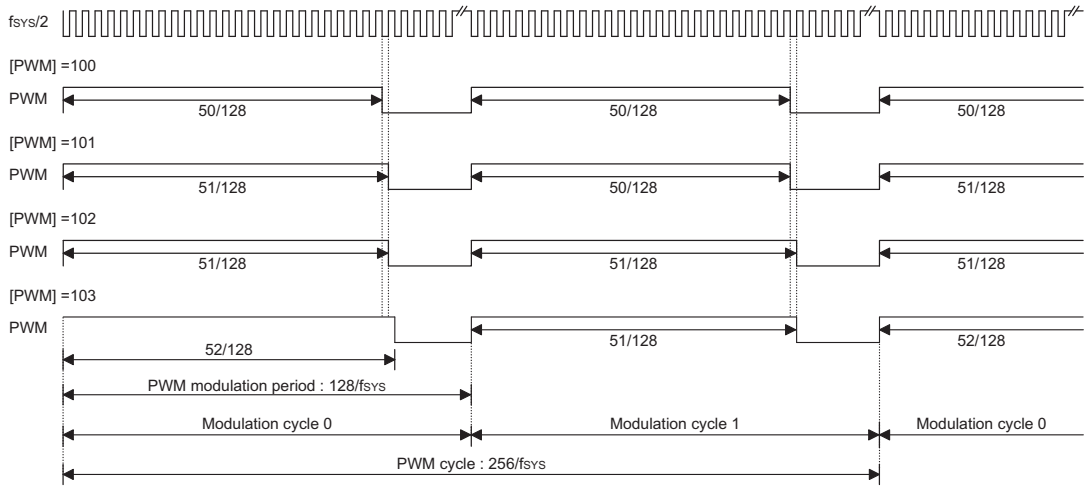
Parameter	AC (0~3)	Duty Cycle
Modulation cycle i (i=0~3)	i<AC	$\frac{DC+1}{64}$
	i≥AC	$\frac{DC}{64}$

The (7+1) mode PWM cycle is divided into two modulation cycles, modulation cycle0~modulation cycle 1. Each modulation cycle has 128 PWM input clock periods. In the (7+1) mode, the contents of each PWM register is divided into two groups. Group 1 of the PWM register is denoted by DC which is the value of PWM.7~PWM.1. Group 2 is denoted by AC which is the value of PWM.0.

In the (7+1) mode, the duty cycle of each modulation cycle is shown in the table.



(6+2) PWM Mode



(7+1) PWM Mode

Parameter	AC (0~1)	Duty Cycle
Modulation cycle i (i=0~1)	i<AC	$\frac{DC+1}{128}$
	i≥AC	$\frac{DC}{128}$

The modulation frequency, cycle frequency and cycle duty of the PWM output signal are summarised in the following table.

PWM Modulation Frequency	PWM Cycle Frequency	PWM Cycle Duty
$f_{SYS}/64$ for (6+2) bits mode $f_{SYS}/128$ for (7+1) bits mode	$f_{SYS}/256$	[PWM]/256

A/D Converter

An 8-channel, 12-bit resolution A/D converter is implemented within the microcontroller. The reference voltage is VDD. The A/D converter contains 4 special registers which are; ADRL, ADRH, ADCR and ACSR. The ADRH and ADRL registers contain the A/D result register higher-order byte and lower-order byte conversion values and are read-only. After the A/D conversion is completed, the ADRH and ADRL should be read to get the conversion result data. The ADCR is an A/D converter control register, which defines the A/D channel number, analog channel select, start A/D conversion control bit and the end of A/D conversion flag. This register is used to start an A/D conversion, define the PB configuration, select the analog channel, and give the START bit a rising and falling edge (0→1→0). At the end of the A/D conversion, the EOCB bit will be automatically cleared to indicate that the conversion process has finished. The ACSR is the A/D clock setting register, which is used to select the A/D clock source.

The A/D converter control register is used to control the A/D converter. Bits 2~0 of ADCR are used to select the analog input channel. There are a total of eight selectable channels. Bits 5~3 of ADCR are used to set the PB configurations. PB can be setup to be either an analog input or digital I/O line decided by these 3 bits. Once a PB line is selected as an analog input, the I/O function and pull-high resistor of the I/O line is disabled and the A/D converter circuit is powered-on. The EOCB bit, bit6 of ADCR, is the end of A/D conversion flag. This bit can be monitored to know when the A/D conversion has finished. The START bit in the ADCR register is used to initiate an A/D conversion process. By providing the START bit with a rising edge and then a falling edge, the A/D conversion process will be initiated. In order to ensure that the A/D conversion has completed, the START bit should remain at ″0″ until the EOCB has cleared to ″0″, which indicates the end of A/D conversion.

Bit 7 of the ACSR register is used for test purposes only and must not be used for other purposes by the application program. Bit1 and bit0 of the ACSR register are used to select the A/D clock source.

When the A/D conversion has completed, the A/D interrupt request flag will be set. The EOCB bit will be set to ″1″ when the START bit is set from ″0″ to ″1″.

Important Note for A/D initialisation:
Special care must be taken to initialise the A/D converter each time the Port B A/D channel selection bits are modified, otherwise the EOCB flag may be in an undefined condition. An A/D initialisation is implemented by setting the START bit high and then clearing it to zero within 10 instruction cycles of the Port B channel selection bits being modified. Note that if the Port B channel selection bits are all cleared to zero then an A/D initialisation is not required.

Bit No.	Label	Function
0 1	ADCS0 ADCS1	Selects the A/D converter clock source 00= system clock/2 01= system clock/8 10= system clock/32 11= undefined
2~7	—	Unused bit, read as ″0″

ACSR (27H) Register

Bit No.	Label	Function
0 1 2	ACS0 ACS1 ACS2	Defines the analog channel select.
3 4 5	PCR0 PCR1 PCR2	Defines the port B configuration select. If PCR0, PCR1 and PCR2 are all zero, the ADC circuit is powered off to reduce power consumption
6	EOCB	Indicates end of A/D conversion. (0 = end of A/D conversion) Each time bits 3~5 change state the A/D should be initialised by issuing a START signal, otherwise the EOCB flag may have an undefined condition. See ″Important note for A/D initialisation″.
7	START	Starts the A/D conversion. (0→1→0= start; 0→1= Resets the A/D converter and sets EOCB to ″1″)

ADCR (26H) Register

PCR2	PCR1	PCR0	7	6	5	4	3	2	1	0
0	0	0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0	0	1	PB7	PB6	PB5	PB4	PB3	PB2	PB1	AN0
0	1	0	PB7	PB6	PB5	PB4	PB3	PB2	AN1	AN0
0	1	1	PB7	PB6	PB5	PB4	PB3	AN2	AN1	AN0
1	0	0	PB7	PB6	PB5	PB4	AN3	AN2	AN1	AN0
1	0	1	PB7	PB6	PB5	AN4	AN3	AN2	AN1	AN0
1	1	0	PB7	PB6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	1	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0

Port B Configuration

ACS2	ACS1	ACS0	Analog Channel
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

Analog Input Channel Selection

Register	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRL	D3	D2	D1	D0	—	—	—	—
ADRH	D11	D10	D9	D8	D7	D6	D5	D4

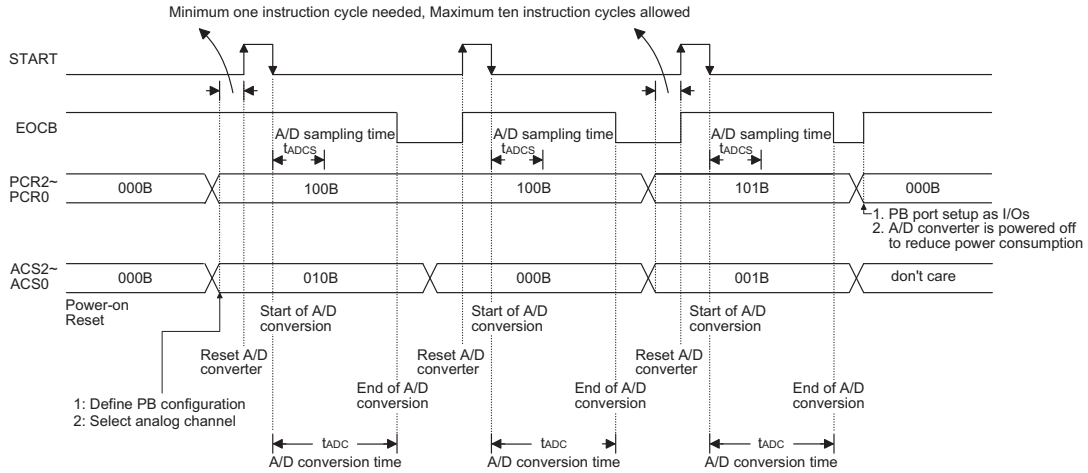
Note: D0~D11 is the A/D conversion result data bit LSB~MSB.

ADRL (24H), ADRH (25H) Register

The following programming example illustrates how to setup and implement an A/D conversion. The method of polling the EOCB bit in the ADCR register is used to detect when the conversion cycle is complete.

Example: using EOCB Polling Method to detect end of conversion

```
    clr     EADI                ; disable ADC interrupt
    mov     a,00000001B
    mov     ACSR,a              ; setup the ACSR register to select fSYS/8 as the A/D clock
    mov     a,00100000B         ; setup ADCR register to configure Port PB0~PB3 as A/D inputs
    mov     ADCR,a              ; and select AN0 to be connected to the A/D converter
            :
            :                   ; As the Port B channel bits have changed the following START
                                ; signal (0-1-0) must be issued within 10 instruction cycles
            :
Start_conversion:
    clr     START
    set     START               ; reset A/D
    clr     START               ; start A/D
Polling_EOC:
    sz      EOCB                ; poll the ADCR register EOCB bit to detect an end of A/D conversion
    jmp     polling_EOC         ; continue polling
    mov     a,ADRH              ; read the conversion result high byte value from the ADRH register
    mov     adrh_buffer,a       ; save result to the user defined memory
    mov     a,ADRL              ; read the conversion result low byte value from the ADRL register
    mov     adrl_buffer,a       ; save the result to the user defined memory
            :
            :
    jmp     start_conversion    ; start the next A/D conversion
```

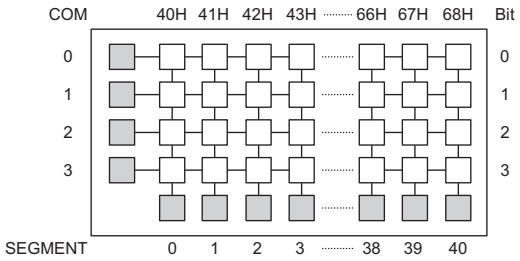


Note: A/D clock must be $f_{SYS}/2$, $f_{SYS}/8$ or $f_{SYS}/32$
$t_{ADCS}=32t_{AD}$
$t_{ADC}=80t_{AD}$

A/D Conversion Timing

LCD Display Memory

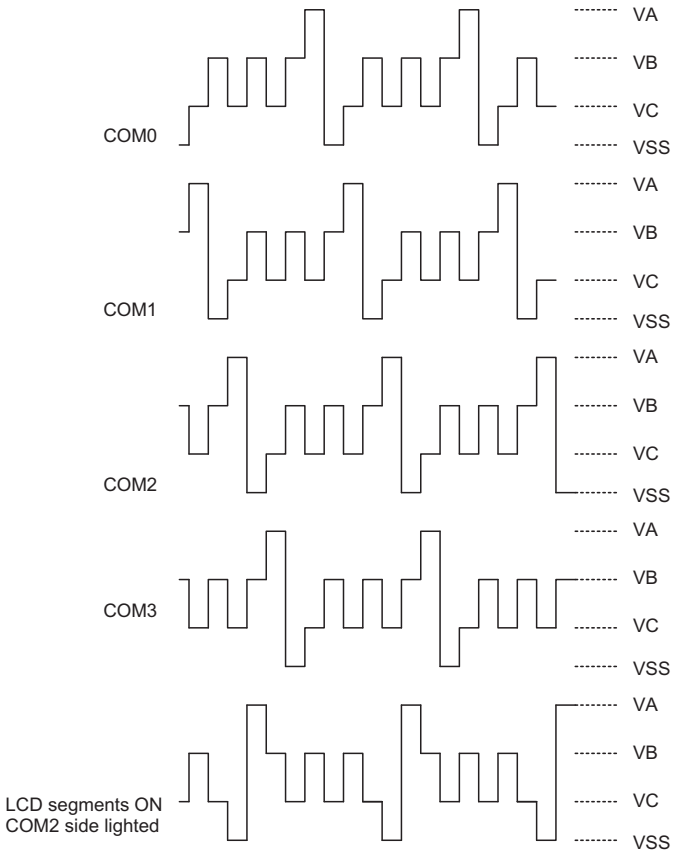
The device provides an area of embedded data memory for the LCD display. This area is located from 40H to 68H of the Data Memory inside Bank 1. The Bank Pointer, known as BP, is used to select the LCD display memory. When the BP is set to ″1″, any data written into locations 40H~68H will affect only the LCD display. When the BP is cleared to ″0″ or set to ″2″, any data written into locations 40H~68H will access the general purpose data memory. The LCD display memory can be read and written to only by an indirect addressing mode using MP1. When data is written into the display data area, it is automatically read by the LCD driver which then generates the corresponding LCD driving signals. To turn the display on or off, a ″1″ or a ″0″ is written to the corresponding bit of the display memory, respectively. The figure illustrates the mapping between the display memory and LCD pattern for the device.



Display Memory

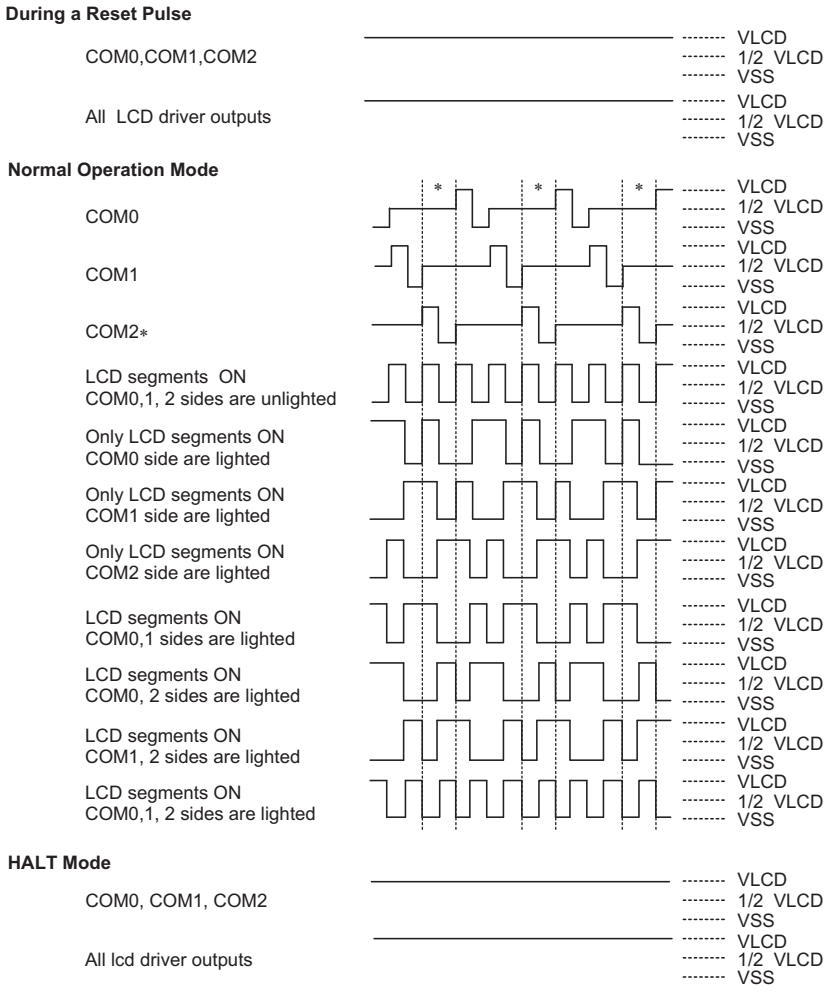
LCD Driver Output

The output number of the device LCD driver can be 41×2 or 41×3 or 40×4 by option, i.e., 1/2 duty, 1/3 duty or 1/4 duty. The LCD driver can either have an ″R″ type or ″C″ bias type. If the ″R″ bias type is selected, no external capacitors are required. If the ″C″ bias type is selected, a capacitor is required to be connected between the C1 and C2 pins. The LCD driver bias voltage can be 1/2 bias or 1/3 bias, selected via a configuration option. If the 1/2 bias is selected, a capacitor must be connected between the V2 pin and ground. If the 1/3 bias is selected, two capacitors are needed for the V1 and V2 pins.



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

LCD Driver Output



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

LCD Driver Output - 1/3 Duty, 1/2 Bias, R/C Type

LCD Segments as Logic Outputs

The SEG0~SEG23 pins can also can be setup as logic outputs via a configuration options. Once an LCD segment is configured as a logic output, the content of bit0 of the related segment address in the LCD RAM will appear on the segment.

Pins SEG0~SEG7 and SEG8~SEG15 are together byte optioned as logic outputs, SEG16~SEG23 are individually bit optioned as logic outputs.

LCD Type	R Type		C Type	
LCD Bias Type	1/2 bias	1/3 bias	1/2 bias	1/3 bias
V_{MAX}	If $V_{DD}>V_{LCD}$, then V_{MAX} connect to V_{DD}, else V_{MAX} connect to V_{LCD}		If $V_{DD}>\frac{3}{2}V_{LCD}$, then V_{MAX} connect to V_{DD}, else V_{MAX} connect to V1	

Low Voltage Reset/Detector Functions

A low voltage detector, LVD, and low voltage reset, LVR, functions are implemented within the microcontroller. These two functions can be enabled/disabled by options. Once the LVD option is enabled, the user can use the RTCC.3 to enable/disable (1/0) the LVD circuit and read the LVD detector status (0/1) from RTCC.5; otherwise, the LVD function is disabled.

The RTCC register definitions are listed below.

Bit No.	Label	Function
0~2	RT0~RT2	8 to 1 multiplexer control inputs to select the real clock prescaler output
3	LVDC	LVD enable/disable (1/0)
4	QOSC	32768Hz OSC quick start-up oscillator 0/1: quickly/slowly start
5	LVDO	LVD detection output (1/0) 1: low voltage detected, read only
6, 7	—	Unused bit, read as ″0″

RTCC (09H) Register

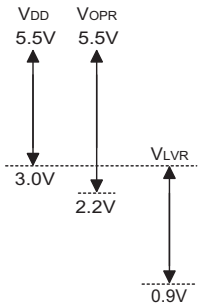
The LVR has the same effect or function as the external RES signal which performs a chip reset. During the HALT state, both LVR and LVD are disabled.

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~V_{LVR}, such as when changing a battery, the LVR will automatically reset the device internally.

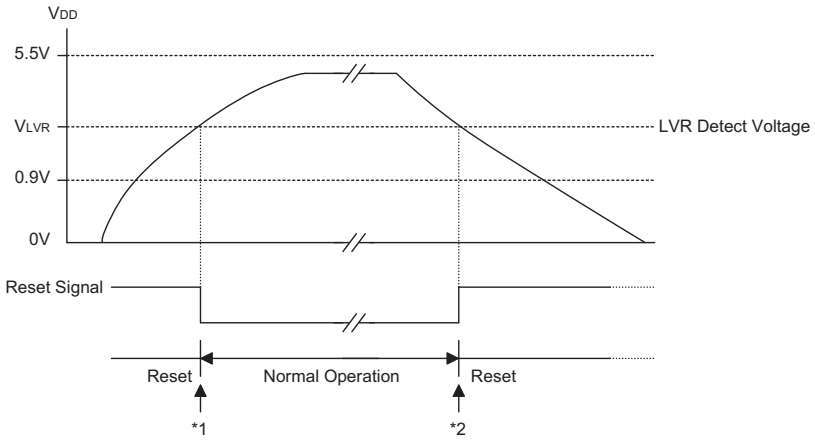
The LVR includes the following specifications:

- The low voltage (0.9V~V_{LVR}) has to remain in its original state for longer than 1ms. If the low voltage state does not exceed 1ms, the LVR will ignore it and will not perform a reset function.
- The LVR uses an ″OR″ function with the external $\overline{RES}$ signal to perform a chip reset.

The relationship between V_{DD} and V_{LVR} is shown below.



Note: V_{OPR} is the voltage range for proper chip operation with a 4MHz system clock.



Low Voltage Reset

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

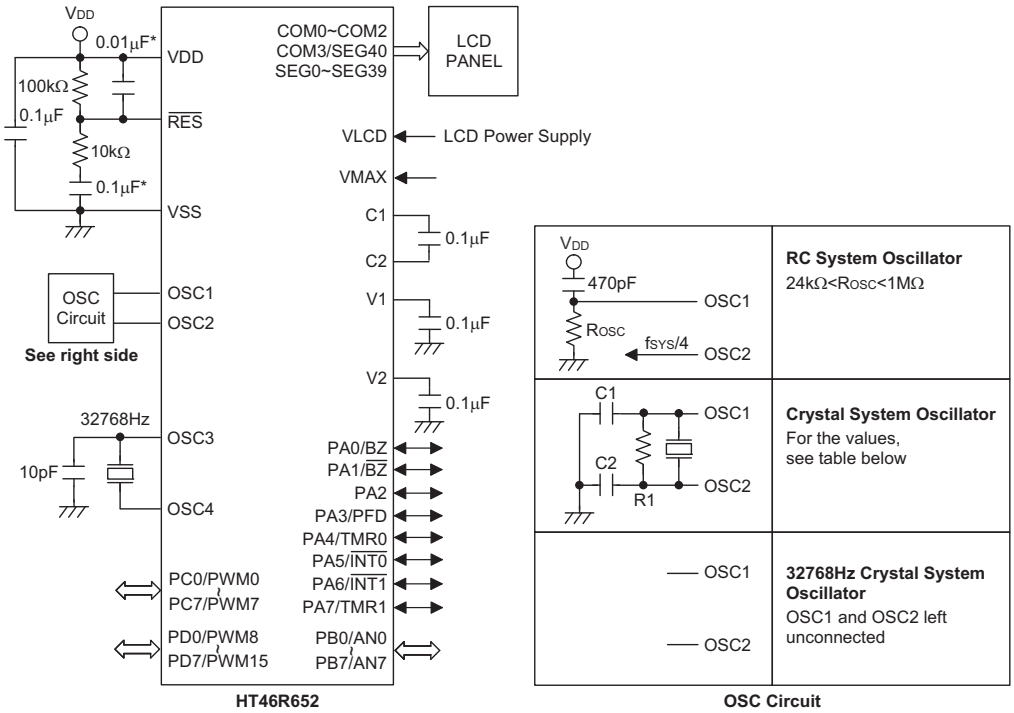
*2: Since a low voltage state has to be maintained its original state for over t_{LVR}, therefore after t_{LVR} delay, the device enters the reset mode.

Configuration Options

The following shows the configuration options in the device. All these options must be defined in order to ensure proper functioning of the microcontroller.

Options
OSC type selection. This option is to decide if an RC or crystal or 32768Hz crystal oscillator is chosen as the system clock.
WDT, RTC and time base clock source selection. There are three types of selections: system clock/4 or RTC OSC or WDT OSC.
WDT enable/disable selection. WDT can be enabled or disabled by option.
WDT time-out period selection. There are four types of selection: WDT clock source divided by $2^{12}/f_S$~$2^{13}/f_S$, $2^{13}/f_S$~$2^{14}/f_S$, $2^{14}/f_S$~$2^{15}/f_S$ or $2^{15}/f_S$~$2^{16}/f_S$.
CLR WDT times selection. This option defines the method to clear the WDT by instruction. ″One time″ means that the ″CLR WDT″ instruction can clear the WDT. ″Two times″ means only if both of the ″CLR WDT1″ and ″CLR WDT2″ instructions have been executed, the WDT can be cleared.
Time Base time-out period selection. The Time Base time-out period ranges from $2^{12}/f_S$ to $2^{15}/f_S$. ″f_S″ means the clock source selected by options.
Buzzer output frequency selection. There are eight types of frequency signals for buzzer output: $f_S/2^2$~$f_S/2^9$. ″f_S″ means the clock source selected by options.
Wake-up selection. This option defines the wake-up capability. External I/O pins (PA only) all have the capability to wake-up the chip from a HALT by a falling edge (bit option).
Pull-high selection. This option is to decide whether the pull-high resistance is visible or not in the input mode of the I/O ports. PA, PB, PC and PD can be independently selected (bit option).
I/O pins shared with other function selections. PA0/$\overline{BZ}$, PA1/BZ: PA0 and PA1 can be set as I/O pins or as buzzer outputs.
LCD common selection. There are three types of selections: 2 common (1/2 duty) or 3 common (1/3 duty) or 4 common (1/4 duty). If the 4 common is selected, the segment output pin ″SEG40″ will be setup as a common output.
LCD bias power supply selection. There are two types of selections: 1/2 bias or 1/3 bias
LCD bias type selection. This option is to determine what kind of bias is selected, R type or C type.
LCD driver clock frequency selection. There are seven types of frequency signals for the LCD driver circuits: $f_S/2^2$~$f_S/2^8$. ″f_S″ stands for the clock source selection by options.
LCD ON/OFF at HALT selection.
LCD Segments as logical output selection, (byte, byte, bit, bit, bit, bit, bit, bit, bit, bit option) [SEG0~SEG7], [SEG8~SEG15], SEG16, SEG17, SEG18, SEG19, SEG20, SEG21, SEG22, or SEG23
LVR selection. LVR enable\disable option
LVD selection. LVD enable\disable option
PFD selection. If PA3 is setup as a PFD output, there are two types of selections; One is PFD0 as the PFD output, the other is PFD1 as the PFD output. PFD0, PFD1 are the timer overflow signals of the Timer/Event Counter 0 and Timer/Event Counter 1, respectively.
PWM selection: (7+1) or (6+2) mode PC0~PC7: General Purpose I/Os or PWM outputs (port option – no individual pin selection) PD0~PD7: General Purpose I/O or PWM output (bit option – individual pin selection)
$\overline{INT0}$ or $\overline{INT1}$ triggering edge selection: disable; high to low; low to high; low to high or high to low.
LCD bias current selection: low/high driving current (for R type only).

Application Circuits



The following table shows the C1, C2 and R1 values corresponding to the different crystal values. (For reference only)

Crystal or Resonator	C1, C2	R1
4MHz Crystal	25pF	12kΩ
4MHz Resonator	10pF	18kΩ
3.58MHz Crystal	25pF	12kΩ
3.58MHz Resonator	10pF	15kΩ
2MHz Crystal & Resonator	25pF	12kΩ
1MHz Crystal	68pF	24kΩ
480kHz Resonator	100pF	12kΩ
455kHz Resonator	200pF	12kΩ
429kHz Resonator	200pF	12kΩ
400kHz Resonator	300pF	10kΩ
The function of the resistor R1 is to ensure that the oscillator will switch off should low voltage conditions occur. Such a low voltage, as mentioned here, is one which is less than the lowest value of the MCU operating voltage. Note however that if the LVR is enabled then R1 can be removed.		

Note: The resistance and capacitance for reset circuit should be designed in such a way as to ensure that the VDD is stable and remains within a valid operating voltage range before bringing $\overline{RES}$ to high.

"*" Make the length of the wiring, which is connected to the $\overline{RES}$ pin as short as possible, to avoid noise interference.

"VMAX" connect to VDD or VLCD or V1 refer to the table.

LCD Type	R Type	C Type	
LCD bias type		1/2 bias	1/3 bias
VMAX	If $V_{DD}>V_{LCD}$, then VMAX connect to V_{DD}, else VMAX connect to V_{LCD}	If $V_{DD} > 3/2V_{LCD}$, then VMAX connect to V_{DD}, else VMAX connect to V1	

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 microcontroller, 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 0.5μs and branch or call instructions would be implemented within 1μs. 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
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	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	1Note	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	1Note	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	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
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	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	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	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z

Mnemonic	Description	Cycles	Flag Affected
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	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
Table Read			
TABRDC [m]	Read table (current page) to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	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	1Note	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

ADC A,[m] 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 ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C

ADCM A,[m] 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] ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C

ADD A,[m] 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 ← ACC + [m]

Affected flag(s) OV, Z, AC, C

ADD A,x 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 ← ACC + x

Affected flag(s) OV, Z, AC, C

ADDM A,[m] 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] ← ACC + [m]

Affected flag(s) OV, Z, AC, C

AND A,[m] 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 ← ACC "AND" [m]

Affected flag(s) Z

AND A,x 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 ← ACC "AND" x

Affected flag(s) Z

ANDM A,[m] 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] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr	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 ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None
CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT1	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. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT2	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. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CPL [m] 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

CPLA [m] 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

DAA [m] 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

DEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

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

Affected flag(s) Z

DECA [m] 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

HALT 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

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z
INCA [m]	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 ← [m] + 1
Affected flag(s)	Z
JMP addr	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 ← addr
Affected flag(s)	None
MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None
MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None
NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
OR A,[m]	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 ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x 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

ORM A,[m] 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

RET 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

RET A,x 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

RETI 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 master interrupt global enable bit. 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

RL [m] 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

RLA [m] 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

RLC [m] 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) ← [m].i; (i = 0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

RLCA [m] 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) ← [m].i; (i = 0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

RR [m] 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 ← [m].(i+1); (i = 0~6)
[m].7 ← [m].0

Affected flag(s) None

RRA [m] 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 ← [m].(i+1); (i = 0~6)
ACC.7 ← [m].0

Affected flag(s) None

RRC [m] 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 ← [m].(i+1); (i = 0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

RRCA [m] 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 ← [m].(i+1); (i = 0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s) C

SBC A,[m] 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 $ACC \leftarrow ACC - [m] - \overline{C}$

Affected flag(s) OV, Z, AC, C

SBCM A,[m] 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 $[m] \leftarrow ACC - [m] - \overline{C}$

Affected flag(s) OV, Z, AC, C

SDZ [m] 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

SDZA [m] 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

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i 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

SIZ [m] 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] ← [m] + 1
Skip if [m] = 0

Affected flag(s) None

SIZA [m] 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 ← [m] + 1
Skip if ACC = 0

Affected flag(s) None

SNZ [m].i 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

SUB A,[m] 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 ← ACC – [m]

Affected flag(s) OV, Z, AC, C

SUBM A,[m] 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] ← ACC – [m]

Affected flag(s) OV, Z, AC, C

SUB A,x 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 ← ACC – x

Affected flag(s) OV, Z, AC, C

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

SWAPA [m] 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 ~ ACC.0 ← [m].7 ~ [m].4
ACC.7 ~ ACC.4 ← [m].3 ~ [m].0

Affected flag(s) None

SZ [m] 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

SZA [m] 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 ← [m]
Skip if [m] = 0

Affected flag(s) None

SZ [m].i 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

TABRDC [m] 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] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

TABRDL [m] 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] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

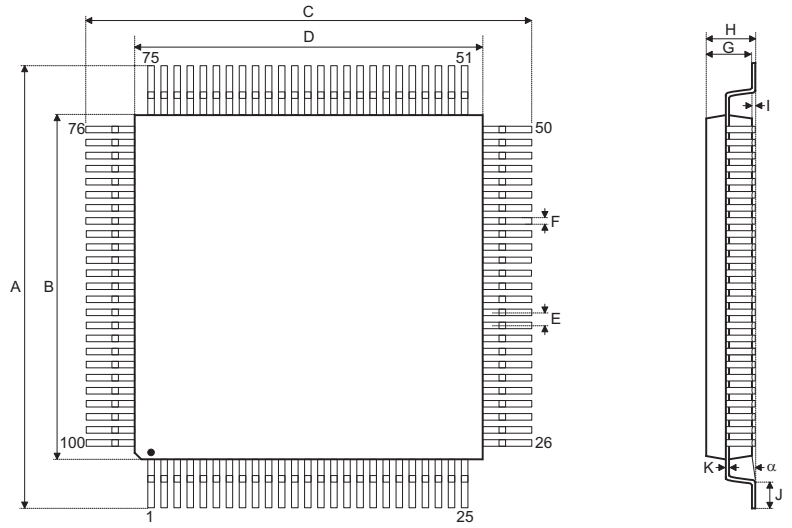
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

XORM A,[m]	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

XOR A,x	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

100-pin LQFP (14mm×14mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.626	—	0.634
B	0.547	—	0.555
C	0.626	—	0.634
D	0.547	—	0.555
E	—	0.020	—
F	—	0.008	—
G	0.053	—	0.057
H	—	—	0.063
I	—	0.004	—
J	0.018	—	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	15.90	—	16.10
B	13.90	—	14.10
C	15.90	—	16.10
D	13.90	—	14.10
E	—	0.50	—
F	—	0.20	—
G	1.35	—	1.45
H	—	—	1.60
I	—	0.10	—
J	0.45	—	0.75
K	0.10	—	0.20
α	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. (Shenzhen Sales Office)
5F, Unit A, Productivity Building, No.5 Gaoxin M 2nd Road, Nanshan District, Shenzhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

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