A/D Type 8-Bit OTP MCU with OPA and 8×8 LED Driver

Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0003E Communicating between the HT48 & HT46 Series MCUs and the HT93LC46 EEPROM
  - HA0075E MCU Reset and Oscillator Circuits Application Note
  - HA0114E Calibrating the OPA Input Voltage Offset on the HT46R32/322/34/342, HT45R32/34 and HT45RM03

Features

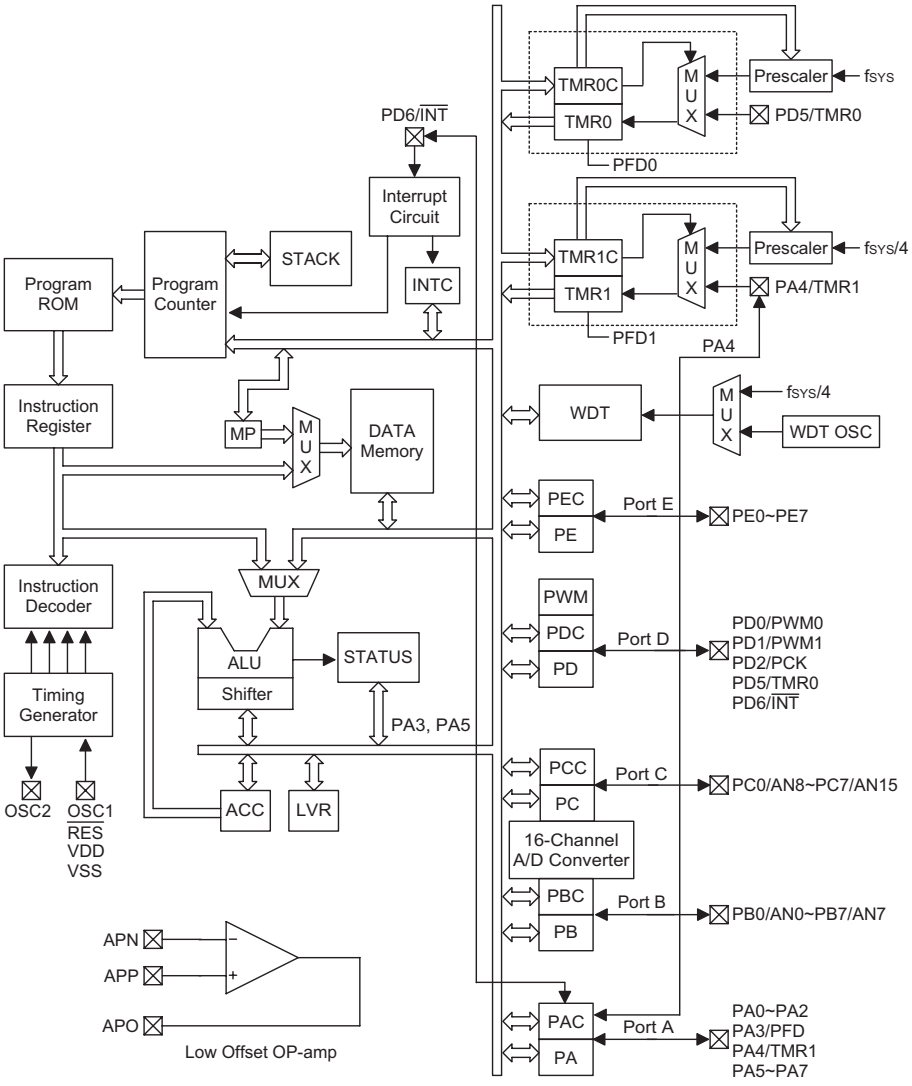
- Operating voltage:
  f_{SYS}=4MHz: 2.2V~5.5V
  f_{SYS}=8MHz: 3.3V~5.5V
- 37 bidirectional I/O lines (max.)
- support 8×8 LED driver
- Single interrupt input shared with an I/O line
- Two 8-bit programmable timer/event counter with overflow interrupt
- Integrated crystal and RC oscillator
- Watchdog Timer
- 4096×15 Program Memory capacity
- 192×8 Data Memory capacity
- Integrated PFD function for sound generation
- Power-down and wake-up functions reduce power consumption
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- 6-level subroutine nesting
- 16 channel 12-bit resolution A/D converter
- Integrated single operational amplifier or comparator selectable via configuration option
- Peripheral clock output – PCK
- Dual 8-bit PWM outputs shared with I/O lines
- Bit manipulation instruction
- Full table read instruction
- 63 powerful instructions
- All instructions executed in one or two machine cycles
- Low voltage reset function
- 44-pin QFP package

General Description

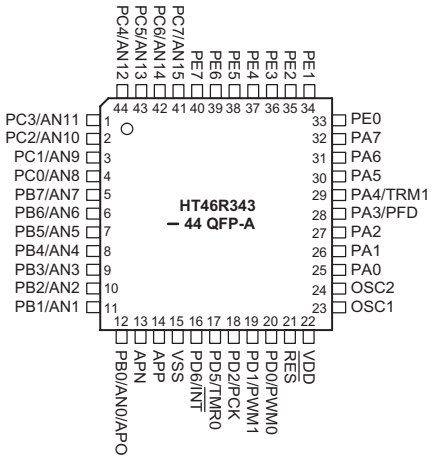
The HT46R343 is 8-bit, high performance, RISC architecture microcontroller devices. With their fully integrated A/D converter they are especially suitable for applications which interface to analog signals, such as those from sensors. The addition of an internal operational amplifier/comparator and PWM modulation functions further adds to the analog capability of these devices.

With the comprehensive features of low power consumption, I/O flexibility, programmable frequency divider, timer functions, oscillator options, multi-channel A/D Converter, OP/Comparator, Pulse Width Modulation function, LED driver, Power-down and wake-up functions etc, the application scope of these devices is broad and encompasses areas such as sensor signal processing, motor driving, industrial control, consumer products, subsystem controllers, etc.

Block Diagram



Pin Assignment



Pin Description

Pin Name	I/O	Options	Description
PA0~PA2 PA3/PFD PA4/TMR1 PA5~PA7	I/O	Pull-high Wake-up PA3 or PFD Timer 0 or Timer 1 (PFD Clock Source)	Bidirectional 8-bit input/output port. Each pin can be configured as wake-up input by configuration options. 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. The PFD and TMR1 pins are pin-shared with PA3 and PA4. PA0~PA7 can be used as LED driver (source end).
PB0/AN0/APO, PB1/AN1~ PB7/AN7, PC0/AN8~ PC7/AN15	I/O	Pull-high	16 lines AD inut pun-shared with PB and PC. 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. Pins PB, PC are pin-shared with the A/D input pins. The A/D inputs are selected via software instructions. Once selected as an A/D input, the I/O function and pull-high resistor are disabled automatically. About PB0/AN0/APO: If the pin is PB0 (setting by ADCS.PCR3~0=00), ADC and OP amp should be power off automatically. If the pin is AN0 (setting by ADCS.PCR3~0), APO is connected with AN0 pin together and OP amp on/off is controlled by OPAC.OPAEN.
PD0/PWM0 PD1/PWM1 PD2/PCK PD5/TMR0 PD6/$\overline{INT}$	I/O	Pull-high PD0 or PWM0 PD1 or PWM1 PD2 or PCK	Bi-directional 4-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pins on this port have pull-high resistors. PD0/PD1 are pin-shared with the PWM0/PWM1 outputs selected via configuration option. PD0, PD1, PD2, PD5, PD6 are pin-shared with the PWM0, PWM1, PCK, TMR0, $\overline{INT}$ output selected via configuration option.
PE0~PE7	I/O	Pull-high	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. A configuration option determines if all pins on the port have pull-high resistors. PE0~PE7 can be used as LED driver (sink end).
APN APP	I I	—	APN and APP are the internal operational amplifier, negative input pin and positive input pin respectively .
$\overline{RES}$	I	—	Schmitt trigger reset input. Active low.
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground.

Pin Name	I/O	Options	Description
OSC1 OSC2	I O	Crystal or RC	OSC1, OSC2 are connected to an external RC network or external crystal, determined by configuration option, for the internal system clock. If the RC system clock option is selected, pin OSC2 can be used to measure the system clock at 1/4 frequency.

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 150mA
Total Power Dissipation 500mW

Storage Temperature ..... –50°C to 125°C
Operating Temperature ..... –40°C to 85°C
I_{OH} Total ..... –100mA

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

Operating Temperature: 40°C~+85°C, 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}=8MHz	3.3	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC)	3V	No load, f_{SYS}=4MHz ADC disable	—	0.6	1.5	mA
		5V		—	2	4	mA
I_{DD2}	Operating Current (RC OSC)	3V	No load, f_{SYS}=4MHz ADC disable	—	0.8	1.5	mA
		5V		—	2.5	4	mA
I_{DD3}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=8MHz ADC disable	—	3	5	mA
I_{STB1}	Standby Current (WDT Enabled)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	Standby Current (WDT Disabled)	3V	No load, system HALT	—	—	1	μA
		5V		—	—	2	μA
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	—	—	2.7	3.0	3.3	V
I_{OL1}	I/O Port Sink Current (PA, PB, PC, PD)	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V	V_{OL}=0.1V_{DD}	10	20	—	mA
I_{OH1}	I/O Port Source Current (PB, PC, PD, PE)	3V	V_{OH}=0.9V_{DD}	–2	–4	—	mA
		5V	V_{OH}=0.9V_{DD}	–5	–10	—	mA
I_{OL2}	PE Ports Sink Current for LED Driver	3V	V_{OL}=0.1V_{DD}	8	16	—	mA
		5V	V_{OL}=0.1V_{DD}	20	40	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH2}	PA Ports Source Current for LED Driver	3V	V_{OH}=0.9V_{DD}	–4	–8	—	mA
		5V	V_{OH}=0.9V_{DD}	–10	–20	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V_{AD}	A/D Input Voltage	—	—	0	—	V_{DD}	V
I_{ADC}	Additional Power Consumption if A/D Converter is Used	3V	—	—	0.5	1	mA
		5V		—	1.5	3	mA
DNL	ADC Differential Non-Linearity	5V	t_{AD}=1μs	—	—	±2	mA
INL	ADC Integral Non-Linearity	5V	t_{AD}=1μs	—	±2.5	4	mA
RESOLU	Resolution	—	—	—	—	12	Bits

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (Crystal OSC, RC OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f_{TIMER}	Timer I/P Frequency (TMR)	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{WDT1}	Watchdog Time-out Period (WDT OSC)	—	—	2^{15}	—	2^{16}	t_{WDTOSC}
t_{WDT2}	Watchdog Time-out Period (System Clock)	—	—	2^{17}	—	2^{18}	t_{SYS}
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	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_{SYS}

OP Amplifier Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
D.C. Electrical Characteristic							
V_{DD}	Operating Voltage	—	—	3	—	5.5	V
V_{OPOS1}	Input Offset Voltage	5V	—	−10	—	10	mV
V_{OPOS2}	Input Offset Voltage	5V	By Calibration	−2	—	2	mV
V_{CM}	Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}-1.4V	V
PSRR	Power Supply Rejection Ratio	—	—	60	80	—	dB
CMRR	Common Mode Rejection Ratio	5V	V_{CM}=0~V_{DD}−1.4V	60	80	—	dB
t_{RES}	Response Time (Comparator)	—	Input overdrive=±10mV	—	—	2	μs
A.C. Electrical Characteristic							
V_{OPOS1}	Open Loop Gain	—	—	60	80	—	dB
SR	Slew Rate +, Slew Rate -	—	No load	—	0.1	—	V/μs
GBW	Gain Band Width	—	R_L=1M, C_L=100p	—	—	100	kHz

Functional Description

Execution Flow

The system clock for the microcontroller is derived from either a crystal or an RC oscillator. The system clock 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 an instruction cycle while decoding and execution takes the next instruction cycle. However, the pipelining scheme causes each instruction to effectively execute in a cycle. If an instruction changes the program counter, two cycles are required to complete the instruction.

Program Counter – PC

The program counter controls the sequence in which the instructions stored in program memory are executed and whose contents specify full range of program memory.

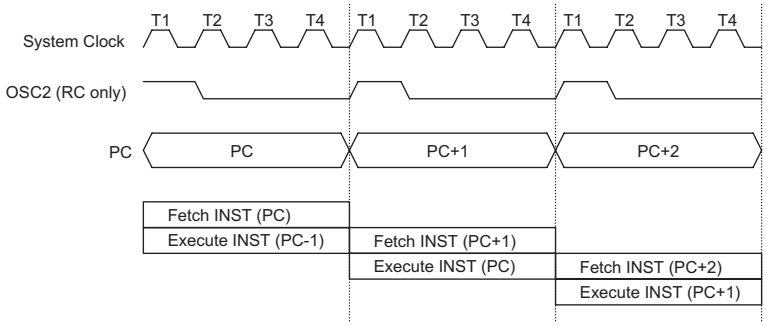
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

When executing a jump instruction, conditional skip execution, loading PCL register, subroutine call, initial reset, internal interrupt, external interrupt or return from 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 the proper instruction. Otherwise proceed with the next instruction.

The lower byte of the program counter, PCL, is a readable and writeable register. Moving data into the PCL performs a short jump. The destination will be within 256 locations.

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



Execution Flow

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

Program Counter

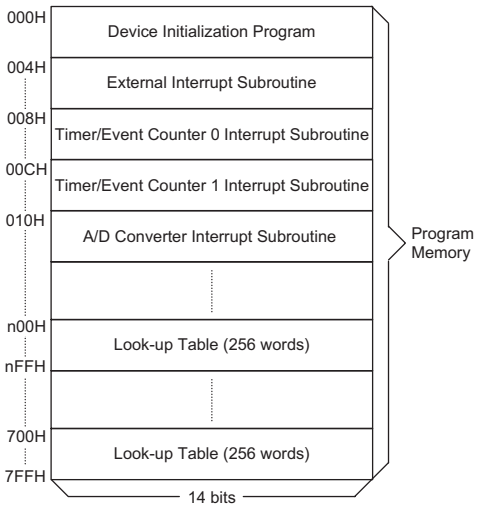
Note: PC11~PC8: Current Program Counter bits
#11~#0: Instruction Code bits
S11~S0: Stack register bits
@7~@0: PCL bits

Program Memory – ROM

The program memory is used to store the program instructions which are to be executed as well as table data and interrupt entries. It is structured into 4K×15 bits device, which can be addressed by both the program counter and table pointer.

Certain locations in the program memory are reserved for use by the reset and by the interrupt vectors.

- Location 000H
  This vector is reserved for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.
- Location 004H
  This vector is used by the external interrupt $\overline{INT}$. If the external interrupt pin on the device receives a low going edge, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.
- Location 008H
  This vector is used by the Timer/Event Counter 0. If a timer overflow occurs, the program will jump to this location and begin execution if the timer interrupt is enabled and the stack is not full.
- Location 00CH
  This vector is used by the Timer/Event Counter 1. If a timer overflow occurs, the program will jump to this location and begin execution if the timer interrupt is enabled and the stack is not full.
- Location 010H
  This vector is used by the A/D converter. When an A/D cycle conversion is complete, the program will jump to this location and begin execution if the A/D interrupt is enabled and the stack is not full.
- Table location
  Any location in the Program Memory space 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 higher-order byte to TBLH. Only the destination of the lower-order byte in the table is well-defined, the other bits of the table word are transferred to the lower portion of TBLH, and the remaining bits are read as "0". The Table Higher-order byte register (TBLH) is read only. The table pointer (TBLP) is a read/write register, which indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH register is read only and cannot be restored. If the main routine and the ISR, Interrupt Service Routine, both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR. In such a case errors can occur. Therefore, using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be used in both the main routine and the ISR, the interrupt is should be disabled prior to the table read instruction. It should not be re-enabled until the TBLH has been backed up. All table related instructions require two cycles to complete their operation. These areas may function as normal program memory depending upon requirements.



Program Memory

Stack Register – STACK

This is a special part of the memory which is used to save the contents of the program counter only. The stack is organized into 6 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer, known as stack pointer, and is also neither readable nor writeable. At a subroutine call or interrupt acknowledgment, the contents of the program counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction,

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

Table Location

Note: *11~*0: Table location bits
P11~P8: Current program counter bits
@7~@0: Table pointer bits

RET or RETI, the program counter is restored to its previous value from the stack. After a device reset, the stack pointer will point to the top of the stack.

If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but the acknowledgment will be inhibited. When the stack pointer is decremented, using a RET or RETI instruction, the interrupt will be serviced. This feature prevents a stack overflow allowing the programmer to use the structure more easily. In a similar case, if the stack is full and a "CALL" is subsequently executed, stack overflow occurs and the first entry will be lost. Only the most recent 6 return addresses are stored.

Data Memory – RAM

The data memory is divided into two functional groups: special function registers and general purpose data memory. The general purpose memory has a structure of 192×8 bits. Most locations are read/write, but some are read only.

The remaining space between the end of the Special Purpose Data Memory and the beginning of the General Purpose Data Memory is reserved for future expanded usage, reading these locations will obtain a result of "00H". The general purpose data memory, addressed from 40H to FFH, is used for user data and control information under instruction commands. All of the data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by the "SET [m].i" and "CLR [m].i" instructions. They are also indirectly accessible through memory pointer register, MP.

Indirect Addressing Register

Location 00H is an indirect addressing register that is not physically implemented. Any read/write operation on [00H] accesses data memory pointed to by the MP register. Reading location 00H itself indirectly will return the result 00H. Writing indirectly results in no operation.

The memory pointer register, MP, is a 8-bit register.

Accumulator

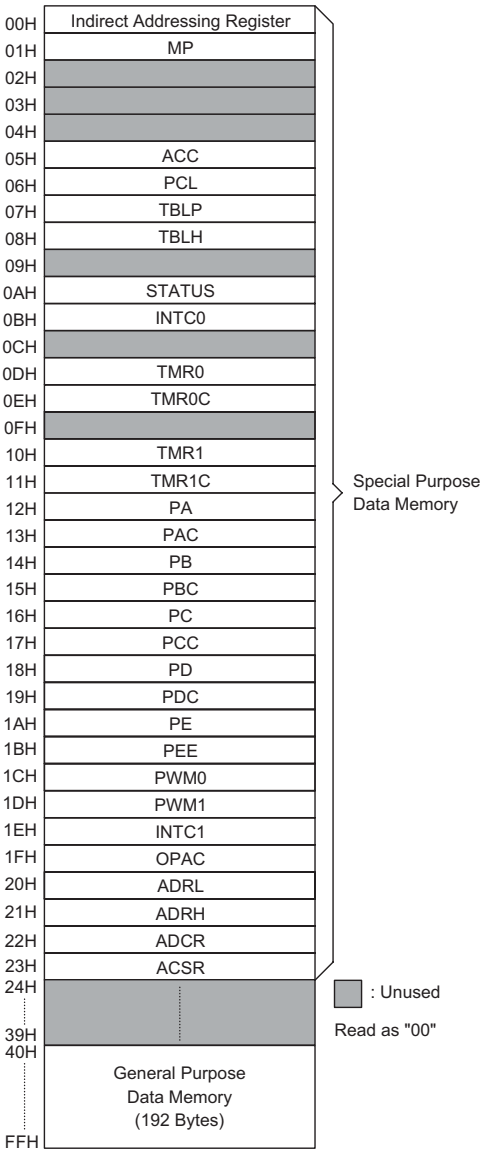
The accumulator is closely related to ALU operations and can carry out immediate data operations. Any data movement between two data memory locations must pass through the accumulator.

Arithmetic and Logic Unit – ALU

This circuit performs 8-bit arithmetic and logic operations. The ALU 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 ....

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



RAM Mapping

Status Register – STATUS

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). It also records the status information and controls the operation sequence.

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition operations related to the status register may give different results from those intended. The TO flag can be affected only by system power-up, a WDT time-out or executing the ″CLR WDT″ or ″HALT″ instruction. The PDF flag can be affected only by executing the ″HALT″ or ″CLR WDT″ instruction or a system power-up.

The Z, OV, AC and C flags generally reflect the status of the latest operations.

In addition, on entering the interrupt sequence or executing the subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status are important and if the subroutine can corrupt the status register, precautions must be taken to save it properly.

Interrupt

The devices provide an external interrupt, an internal timer/event counter interrupt and an A/D converter interrupt. The Interrupt Control Register, INTC, contains the interrupt control bits to set the enable or disable and the interrupt request flags.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked by clearing the EMI bit. This scheme may prevent any further interrupt nesting. Other interrupt requests may happen during this interval but only the interrupt request flag is recorded. If a certain interrupt requires servicing within the service routine, the EMI bit and the corresponding bit in INTC0/INTC1 may be set 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 must be prevented from becoming full.

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 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

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1=enabled; 0=disabled)
1	EEI	Controls the external interrupt (1=enabled; 0=disabled)
2	ET0I	Controls the Timer/Event Counter 0 interrupt (1=enabled; 0=disabled)
3	ET1I	Controls the Timer/Event Counter 1 interrupt (1=enabled; 0=disabled)
4	EIF	External interrupt request flag (1=active; 0=inactive)
5	T0F	Internal Timer/Event Counter 0 request flag (1=active; 0=inactive)
6	T1F	Internal Timer/Event Counter 1 request flag (1=active; 0=inactive)
7	—	Unused bit, read as ″0″

INTC (0BH) Register

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

INTC1 (1EH) Register

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

External interrupts are triggered by a high to low transition on the $\overline{INT}$ pin, which will set the related interrupt request flag, EIF, which is bit 4 of INTC0. When the interrupt is enabled, the stack is not full and the external interrupt is active, a subroutine call to location 04H will occur. The interrupt request flag, EIF, and EMI bits will be cleared to disable other interrupts.

The internal timer/event counter 0 interrupt is initialised by setting the timer/event counter interrupt request flag, T0F, which is bit 5 of INTC0, caused by a timer overflow. When the interrupt is enabled, the stack is not full and the T0F bit is set, a subroutine call to location 008H will occur. The related interrupt request flag, T0F, will be reset and the EMI bit cleared to disable further interrupts.

The internal timer/event counter 1 interrupt is initialised by setting the timer/event counter interrupt request flag, T1F, which is bit 6 of INTC1, caused by a timer overflow. When the interrupt is enabled, the stack is not full and the T1F bit is set, a subroutine call to location 00CH will occur. The related interrupt request flag, T1F, will be reset and the EMI bit cleared to disable further interrupts.

The A/D converter interrupt is initialised by setting the A/D converter request flag, ADF, which is bit 4 of INTC1, caused by an end of A/D conversion. When the interrupt is enabled, the stack is not full and the ADF bit is set, a subroutine call to location 010H will occur. The related interrupt request flag, ADF, will be reset and the EMI bit cleared to disable further interrupts.

During the execution of an interrupt subroutine, other interrupt acknowledgments are held until the RETI instruction is executed or the EMI bit and the related interrupt control bit are set to 1. Of course, the stack must not be full. To return from the interrupt subroutine, a RET or RETI instruction may be executed. A RETI instruction will set the EMI bit to enable an interrupt service, but a RET instruction will not.

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

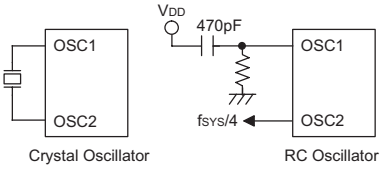
Interrupt Source	Priority	Vector
External Interrupt	1	004H
Timer/Event Counter 0 Overflow	2	008H
Timer/Event Counter 1 Overflow	3	00CH
A/D Converter Interrupt	4	010H

Once the interrupt request flags, T0F/T1F, EIF, ADF, are set, they will remain in the INTC0/INTC1 register until the interrupts are serviced or cleared by a software instruction. It is recommended that a program does not use the CALL subroutine within the interrupt subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications. If only one stack is left and enabling the interrupt is not well controlled, the original control sequence will be damaged once the "CALL" operates in the interrupt subroutine.

Oscillator Configuration

There are two oscillator circuits in the microcontroller, namely an RC oscillator and a crystal oscillator, the choice of which is determined by a configuration option. When the system enters the Power-down mode the system oscillator stops and ignores external signals to conserve power.

If an RC oscillator is used, an external resistor between OSC1 and VSS is required whose resistance value must 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 signal can be used to synchronise external logic. The RC oscillator provides the most cost effective solution, however the frequency of



System Oscillator

oscillation may vary with VDD, temperature and the process variations. It is, therefore, not suitable for timing sensitive operations where an accurate oscillator frequency is desired.

If the Crystal oscillator is used, a crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for the oscillator; no other external components are required. Instead of a crystal, a resonator can also be connected between OSC1 and OSC2 to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required, If the oscillating frequency is less than 1MHz.

The WDT oscillator is a free running on-chip RC oscillator, and requires no external components. Even if the system enters the power down mode, the system clock is stopped, but the WDT oscillator keeps running with a period of approximately 65μs at 5V. The WDT oscillator can be disabled by a configuration option to conserve power.

Watchdog Timer – WDT

The WDT clock source comes from either its own integrated RC oscillator, known as the WDT oscillator, or the instruction clock, which is the system clock divided by 4. The choice of which one is used is decided by a configuration option. This timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The Watchdog Timer can be disabled by a configuration option. If the Watchdog Timer is disabled, all the executions related to the WDT result in no operation.

Once the internal WDT oscillator (RC oscillator with a period of 65μs at 5V nominal) is selected, it is divided by 32768~65536 to get a time-out period of approximately 2.1s~4.3s. This time-out period may vary with temperatures, VDD and process variations. 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 state the WDT may stop counting and lose its protecting purpose. In this situation the logic can only be restarted by external logic.

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

The WDT overflow under normal operation will initialise a "chip reset" and set the status bit "TO". But in the Power-down mode, the overflow will initialisze a "warm reset", and only the program counter and SP are reset to zero. To clear the contents of the WDT, three methods are adopted; external reset (a low level on the $\overline{RES}$ pin), a software instruction and a HALT instruction. The software instruction include "CLR WDT" and the other set – "CLR WDT1" and "CLR WDT2". Of these two types of instruction, only one can be active depending on the configuration option – "CLR WDT times selection option". If the "CLR WDT" is selected (i.e. CLR WDT times equal one), any execution of the "CLR WDT" instruction will clear the WDT. In the case that "CLR WDT1" and "CLR WDT2" are chosen (i.e. CLR WDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip as a result of time-out.

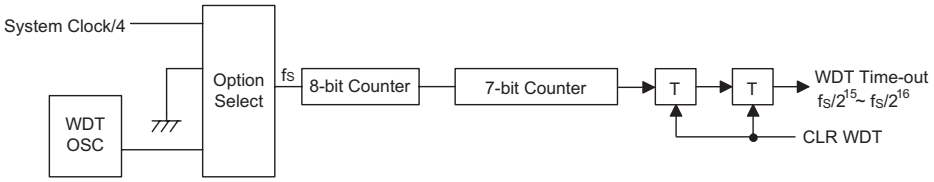
Power Down Operation – HALT

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

- The system oscillator will be turned off but the WDT oscillator keeps running (if the WDT oscillator is selected).
- The contents of the on chip Data Memory and registers remain unchanged.
- WDT will be cleared and start counting again (if the WDT clock is from the WDT oscillator).
- All of the I/O ports maintain their original status.
- The PDF flag is set and the TO flag is cleared.

The system can leave the HALT mode by means of an external reset, an interrupt, an external falling edge signal on port A or a WDT overflow. An external reset causes a device initialisation and the WDT overflow performs a "warm reset". After the TO and PDF flags are examined, the reason for the chip reset can be determined. The PDF flag is cleared by a system power-up or executing the "CLR WDT" instruction and is set when executing the "HALT" instruction. The TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the program counter and Stack Pointer; the others keep their original status.

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 the options. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it is awakening from an interrupt, two



Watchdog Timer

sequences may happen. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program will resume execution at the next instruction. If the interrupt is enabled and the stack is not full, the regular interrupt response takes place. If an interrupt request flag is set to "1" before entering the HALT mode, the wake-up function of the related interrupt will be disabled. Once a wake-up event occurs, it takes 1024 t_{SYS} (system clock period) to resume normal operation. In other words, a dummy period will be inserted after wake-up. If the wake-up results from an interrupt acknowledgment, the actual interrupt subroutine execution will be delayed by one or more cycles. If the wake-up results in the next instruction execution, this will be executed immediately after the dummy period is finished.

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

Reset

There are three ways in which a reset can occur:

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

The WDT time-out during HALT is different from other chip reset conditions, since it can perform a "warm reset" that resets only the program counter and stack pointer, leaving the other circuits in their original state. Some registers remain unchanged during other reset conditions. Most registers are reset to the "initial condition" when the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between different "chip resets".

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

Note: "u" means "unchanged"

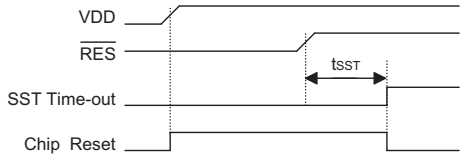
To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system reset (power-up, WDT time-out or $\overline{RES}$ reset) or the system awakes from the HALT state.

When a system reset occurs, the SST delay is added during the reset period. Any wake-up from HALT will enable the SST delay.

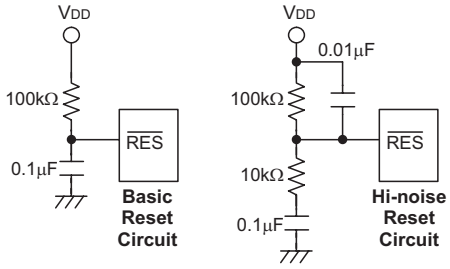
An extra option load time delay is added during system reset (power-up, WDT time-out at normal mode or $\overline{RES}$ reset).

The functional unit chip reset status are shown below.

Program Counter	000H
Interrupt	Disable
WDT	Clear. After master reset, WDT begins counting
Timer/Event Counter	Off
Input/Output Ports	Input mode
Stack Pointer	Points to the top of the stack

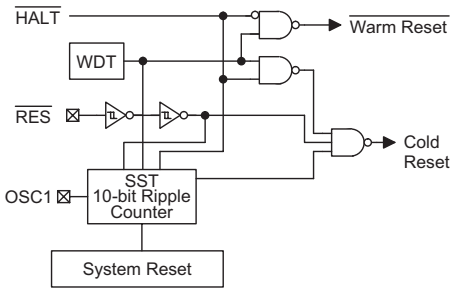


Reset Timing Chart



Reset Circuit

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



Reset Configuration

The registers′ 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 Times-out (HALT)*
MP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
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	-11- -111	-11- -111	-11- -111	-11- -111	-uu- -uuu
PDC	-11- -111	-11- -111	-11- -111	-11- -111	-uu- -uuu
PE	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PWM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
INTC1	---0 ---0	---0 ---0	---0 ---0	---0 ---0	---u ---u
OPAC	0000 1000	0000 1000	0000 1000	0000 1000	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	1-00 --00	1-00 --00	1-00 --00	1-00 --00	u-uu --uu

Note: ″*″ stands for warm reset
″u″ stands for unchanged
″x″ stands for unknown

Timer/Event Counter

Two timer/event counter (TMR) are implemented in the microcontroller. The timer/event counter contains an 8-bit programmable count-up counter and the clock may come from external source or an internal clock source. An internal clock source comes from fsys (f_{SYS}/4). Using an external clock input allows the user to count external events, measure time internals or pulse widths, or generate an accurate time base. While using the internal clock allows the user to generate an accurate time base.

There are two registers related to the Timer/Event Counter; TMR0 (TMR1), TMR0C (TMR1C). Writing TMR0 (TMR1) will transfer the specified data to Timer/Event Counter registers. Reading the TMR0 (TMR1) will read the contents of the Timer/Event Counter. The TMRC0 (TMR1C) is a control register, which defines the operating mode, counting enable or disable and an active edge.

The T0TM0 & T0TM1 (T1TM0 & T1TM1) bits define the operation mode. The event count mode is used to count external events, which means that the clock source is from an external TMR0 (TMR1) pin. The timer mode functions as a normal timer with the clock source coming from the internal selected clock source. Finally, the pulse width measurement mode can be used to count the high or low-level duration of the external signal TMR0 (TMR1), and the counting is based on the internal selected clock source.

In the event count or timer mode, the timer/event counter starts counting at the current contents in the timer/event counter and ends at FFH. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag T0F (T1F). In the pulse width measurement mode with the values of the T0ON & T0E (T1ON &T1E) equal to 1, after the TMR0(TMR1) 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(TMR1) returns to the original level and resets the T0ON(T1ON). The measured result remains in the timer/event counter even if the activated transient occurs again. In other words, only 1-cycle measurement can be made until the T0ON (T1ON) is set. The cycle measurement will re-function as long as it receives further transient pulse. In this operation mode, the timer/event counter begins counting not according to the logic level but to the transient edges. In the case of counter overflow, the counter is reloaded from the timer/event counter register and issues an interrupt request, as in the other two modes, i.e., event and timer modes.

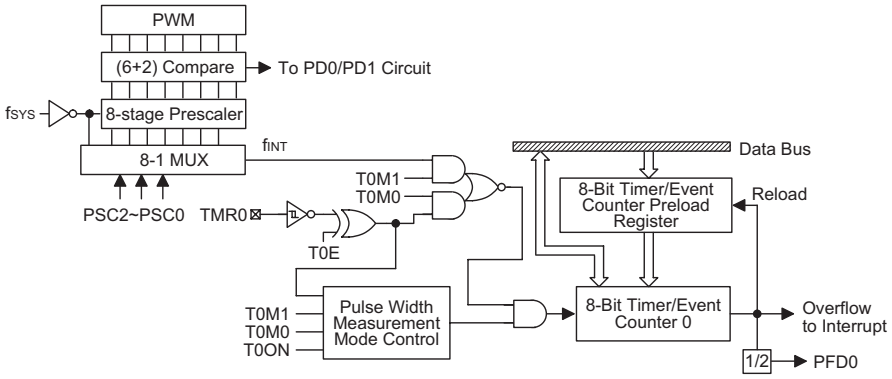
To enable the counting operation, the Timer ON bit T0ON (T1ON) should be set to 1. In the pulse width measurement mode, the T0ON (T1ON) is automatically cleared after the measurement cycle is completed. But in the other two modes, the T0ON (T1ON) can only be reset by instructions. The overflow of the Timer/Event Counter is one of the wake-up sources and can also be applied to a PFD (Programmable Frequency Divider)

Bit No.	Label	Function
0 1 2	T0PSC0 T0PSC1 T0PSC2	Defines the prescaler stages, PSC2, PSC1, PSC0= 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 TMR 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 or disable the timer counting (0=disable; 1=enable)
5	—	Unused bits, 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 00=Unused

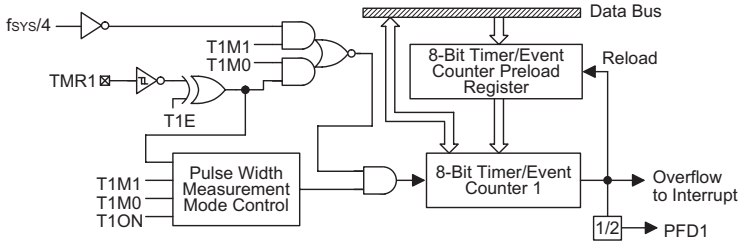
TMR0C (0EH) Register

Bit No.	Label	Function
0~2	—	Unused bits, read as "0"
3	T1E	Defines the TMR 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 or disable the timer counting (0=disable; 1=enable)
5	—	Unused bits, read as "0"
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 00=Unused

TMR1C (11H) Register



Timer/Event Counter 0



Timer/Event Counter 1

output by options. When the PFD function is selected, executing "SET [PA].3" instruction to enable PFD output and executing "CLR [PA].3" instruction to disable PFD output.

In the case of a timer/event counter OFF condition, writing data to the timer/event counter preload register will also reload that data to the timer/event counter. But if the timer/event counter is turned on, data written to it will only be kept in the timer/event counter preload register. The timer/event counter will still operate until an overflow occurs.

When the timer/event counter TMR0 (TMR1) is read, the clock is blocked to avoid errors, as this may results in a counting error. Blocking of the clock issue should be taken into account by the programmer. It is strongly recommended to load a desired value into the TMR0 (TMR1) register first, before turning on the related timer/event counter, for proper operation since the initial value of TMR0 (TMR1) is unknown. Due to the timer/event scheme, the programmer should pay special attention on the instruction to enable then disable the timer for the first time, whenever there is a need to use the timer/event function, to avoid unpredictable result. After this procedure, the timer/event function can be operated normally.

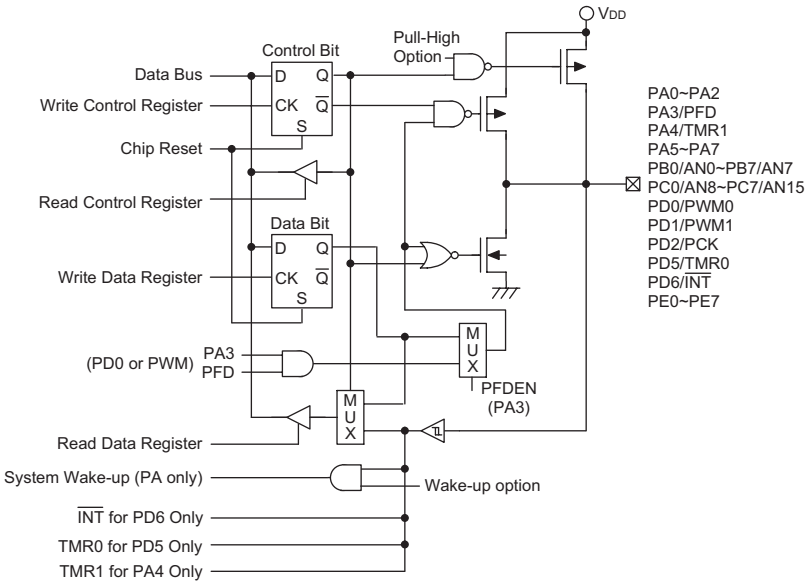
The definitions are as shown. The overflow signal of timer/event 0/1 counter can be used to generate the PFD signal. The timer prescaler is also used as the PWM counter.

Input/Output Ports

There are 37 bidirectional input/output lines in the microcontroller, labeled as PA, PB, PC, PD and PE, which are mapped to the data memory of [12H], [14H], [16H], [18H] and [1AH] respectively. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]" (m=12H, 14H, 16H, 18H or 1AH). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PCC, PDC, PEC) to control the input/output configuration. With this control register, CMOS output or Schmitt trigger input with or without pull-high resistor structures can be reconfigured dynamically (i.e. on-the-fly) under software control. To function as an input, the corresponding latch of the control register must write "1". The input source also depends on the control register. If the control register bit is "1", the input will read the pad state. If the control register bit is "0", the contents of the latches will move to the internal bus. The latter is possible in the "read-modify-write" instruction.

After a device reset, the input/output lines will default to inputs and remain at a high level or floating state, dependent upon 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" (m=12H, 14H, 16H, 18H or 1AH) instructions.



Input/Output Ports

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 line has a pull-high option. Once the pull-high configuration option is selected, the I/O line has a pull-high resistor, otherwise, there's none. Take note that a non-pull-high I/O line operating in input mode will cause a floating state.

Pin PA3 is pin-shared with the PFD signal. If the PFD configuration option is selected, the output signal in the output mode for PA3 will be the PFD signal generated by the timer/event counter overflow signal. The input mode always retains its original functions. Once the PFD option is selected, the PFD output signal is controlled by the PA3 data register only. Writing a "1" to the PA3 data register will enable the PFD output function and writing "0" will force the PA3 to remain at "0". The I/O functions for PA3 are shown below.

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 PD6, PD5 and PA4 are pin-shared with $\overline{INT}$, TMR0 and TMR1 pins respectively.

The PB and PC can also be used as A/D converter inputs. The A/D function will be described later. There are two PWM functions shared with pins PD0 and PD1. If the PWM functions are enabled, the PWM signals will appear on PD0 and PD1, the pins are setup as outputs. Writing a "1" to the PD0 or PD1 data register will enable the PWM outputs to function while writing a "0" will force the PD0 and PD1 outputs to remain at "0". The I/O functions of PD0 and PD1 are as shown.

I/O Mode	I/P (Normal)	O/P (Normal)	I/P (PWM)	O/P (PWM)
PD0 PD1	Logical Input	Logical Output	Logical Input	PWM0 PWM1

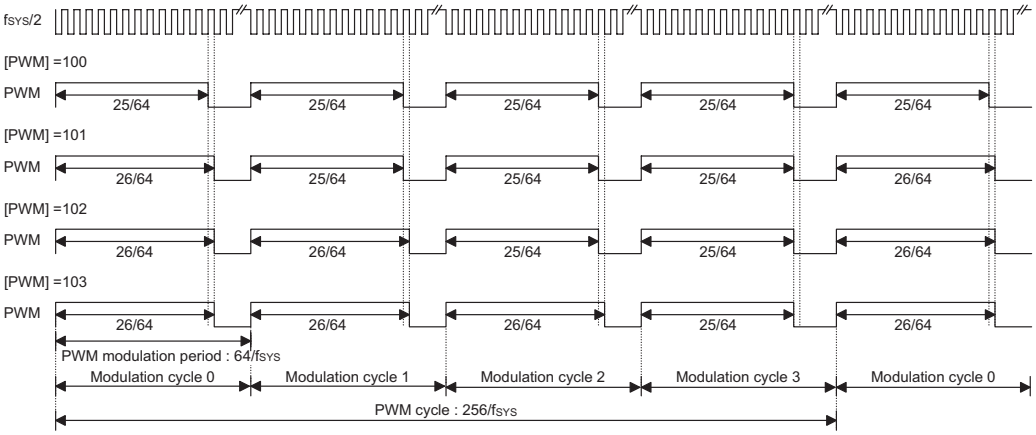
It is recommended that unused I/O lines should be setup as output pins by software instructions to avoid consuming power under input floating states.

PWM

The microcontroller provides a 2 channel (6+2) bits PWM0/PWM1 output shared with PD0/PD1. The PWM channel has its data register denoted as PWM0 and PWM1. The frequency source of the PWM counter comes from f_{SYS}. The PWM register is an eight bit register. Once PD0/PD1 are selected as PWM outputs and the output function of PD0/PD1 is enabled (PDC.0="0" or PDC.1="0"), writing a 1 to the PD0/PD1 data register will enable the PWM output function while writing a "0" will force the PD0/PD1 outputs to stay at "0".

A PWM cycle is divided into four modulation cycles (modulation cycle 0~modulation cycle 3). Each modulation cycle has 64 PWM input clock period. In a (6+2) bit PWM function, the contents of the 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.2.

Group 2 is denoted by AC which is the value of PWM.1~PWM.0.



6+2 PWM Mode

In a PWM cycle, the duty cycle of each modulation cycle is shown in the table.

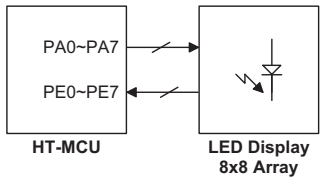
Parameter	AC (0~3)	Duty Cycle
Modulation cycle i (i=0~3)	i<AC	$\frac{DC+1}{64}$
	i≥AC	$\frac{DC}{64}$

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

PWM Modulation Frequency	PWM Cycle Frequency	PWM Cycle Duty
$f_{SYS}/64$	$f_{SYS}/256$	[PWM]/256

LED Driver

The device provides a maximum of 8×8 LED drivers which uses I/O Ports PA and PE with double the usual sink/source current drive capabilities. To use the LED driver function, PA and PE must be setup as outputs.



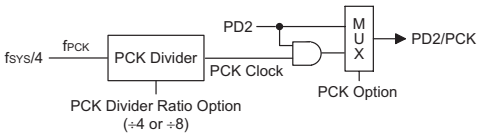
Peripheral Clock Output – PCK

The device also provides a Peripheral Clock Output (PCK) which is pin-shared with PD2. Once the PD2 is selected as the PCK outputs and the output function of PD2 is enabled (PDC.2 =″0″), writing 1 to PA0 data register will enable the PCK output function and writing ″0″ will force the PD2 to stay at ″0″.

The PCK clock frequency can be optional : $f_{SYS}/16$ or $f_{SYS}/32$ (by configuration option).

PCK output = 500kHz or 250kHz if f_{SYS}=8MHz.

Recommend to clear PD2 before entering HALT mode.



A/D Converter

The 16 channel 12-bit resolution A/D converter is implemented in the microcontrollers. The reference voltage for the A/D is VDD. The A/D converter contains 4 special registers, which are; ADRL, ADRH, ADCR and ACSR.

The ADRH and ADRL registers are the A/D conversion result register higher-order byte and lower-order byte and are read-only. After the A/D conversion has completed, the ADRL and ADRH registers should be read to get the conversion result data. The ADCR register 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. It is used to start an A/D conversion, define the PB configuration, select the converted analog channel, and give the START bit a raising edge and a falling edge (0→1→0). At the end of an A/D conversion, the EOCB bit is cleared and an A/D converter interrupt occurs, if the A/D converter interrupt is enabled. The ACSR register is an 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. Bit2~bit0 of the ADCR register and bit 4 of ADCS are used to select an analog input channel. There are a total of 16 channels to select. Bit5~bit3 of the ADCR register and bit 5 of ADCS are used to set the PB, PC configurations. PB, PC can be configured as an analog input or as a digital I/O line decided by these 4 bits. Once a PB, PC line is selected as an analog input, the I/O functions and pull-high resistor of this I/O line are 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 is monitored to check when the A/D conversion has completed. The START bit of the ADCR register is used to initiate the A/D conversion process. When the START bit is provided with a raising edge and then a falling edge, the A/D conversion process will begin. In order to ensure that the A/D conversion is completed, the START should remain at ″0″ until the EOCB flag is cleared to ″0″ which indicates the end of the 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 is 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, Port C 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, Port C channel selection bits being modified. Note that if the Port B, Port C channel selection bits are all cleared to zero then an A/D initialisation is not required.

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 A/D conversion result data bit LSB~MSB.

ADRL (20H), ADRH (21H) Register

Bit No.	Label	Function
0 1 2	ACS0 ACS1 ACS2	ACS3[ACSR], ACS2, ACS1, ACS0: selected A/D channel
3 4 5	PCR0 PCR1 PCR2	Port B, C configuration selection. If PCR0, PCR1, PCR2 and PCR3[ACSR] are all zero, the ADC circuit is power 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= Reset A/D converter and set EOCB to ″1″)

ADCR (22H) Register

ACS3, ACS2, ACS1, ACS0	Channel	ACS3, ACS2, ACS1, ACS0	Channel
0000	AN0	1000	AN8
0001	AN1	1001	AN9
0010	AN2	1010	AN10
0011	AN3	1011	AN11
0100	AN4	1100	AN12
0101	AN5	1101	AN13
0110	AN6	1110	AN14
0111	AN7	1111	AN15

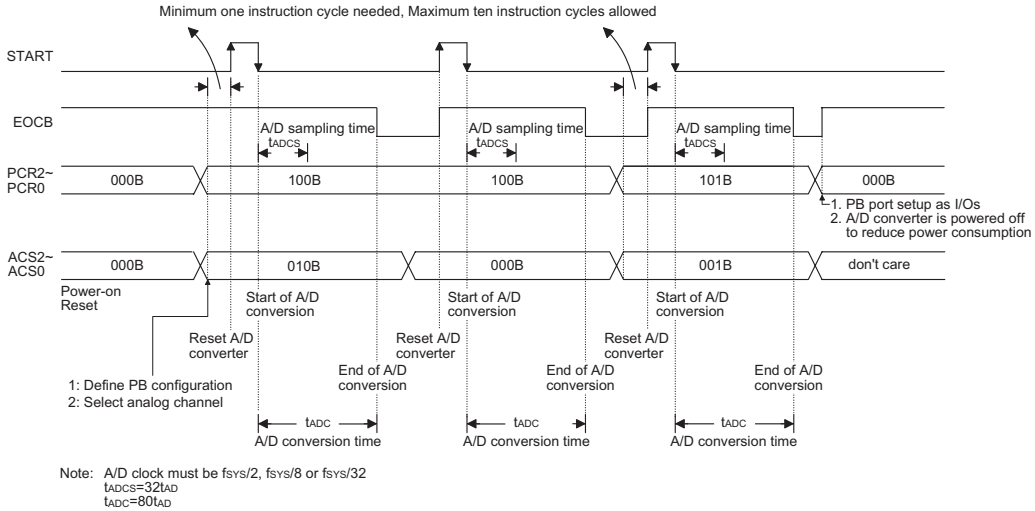
A/D Channel

Bit No.	Label	Function
0 1	ADCS0 ADCS1	Select the A/D converter clock source. 0, 0: $f_{SYS}/2$ 0, 1: $f_{SYS}/8$ 1, 0: $f_{SYS}/32$ 1, 1: Undefined
2~3	—	Unused bit, read as ″0″.
4	ACS3	Bit3 of 4bit A/D channel selection
5	PCR3	Bit3 of 4bit Port B, C configuration selection.
6	—	Unused bit, read as ″0″.
7	TEST	For internal test only.

ACSR (23H) Register

PCR3	PCR2	PCR1	PCR0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0	0	0	1	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	AN0
0	0	1	0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	PB3	PB2	AN1	AN0
0	0	1	1	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	PB3	AN2	AN1	AN0
0	1	0	0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	AN3	AN2	AN1	AN0
0	1	0	1	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	AN4	AN3	AN2	AN1	AN0
0	1	1	0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	AN5	AN4	AN3	AN2	AN1	AN0
0	1	1	1	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	0	0	0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	0	0	1	PC7	PC6	PC5	PC4	PC3	PC2	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	0	1	0	PC7	PC6	PC5	PC4	PC3	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	0	1	1	PC7	PC6	PC5	PC4	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	0	0	PC7	PC6	PC5	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	0	1	PC7	PC6	AN13	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	1	0	PC7	AN14	AN13	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	1	1	AN15	AN14	AN13	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0

Port B, C configuration



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

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the EOCB bit in the ADCR register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion 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 end of A/D conversion
    jmp     polling_EOC         ; continue polling
    mov     a,ADRH              ; read conversion result high byte value from the ADRH register
    mov     adrh_buffer,a       ; save result to user defined memory
    mov     a,ADRL              ; read conversion result low byte value from the ADRL register
    mov     adrl_buffer,a       ; save result to user defined memory
            :
            :
    jmp     start_conversion    ; start next A/D conversion
```

Example: using interrupt 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
    clr     ADF                 ; clear ADC interrupt request flag
    set     EADI                ; enable ADC interrupt
    set     EMI                 ; enable global interrupt
            :
            :
            :
; ADC interrupt service routine
ADC_ISR:
    mov     acc_stack,a         ; save ACC to user defined memory
    mov     a,STATUS
    mov     status_stack,a      ; save STATUS to user defined memory
            :
            :
    mov     a,ADRH              ; read conversion result high byte value from the ADRH register
    mov     adrh_buffer,a       ; save result to user defined register
    mov     a,ADRL              ; read conversion result low byte value from the ADRL register
    mov     adrl_buffer,a       ; save result to user defined register
    clr     START
    set     START               ; reset A/D
    clr     START               ; start A/D
            :
            :
EXIT_INT_ISR:
    mov     a,status_stack
    mov     STATUS,a            ; restore STATUS from user defined memory
    mov     a,acc_stack         ; restore ACC from user defined memory
    reti
```

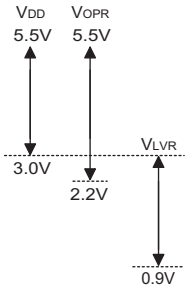
Low Voltage Reset – LVR

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 what happens when changing a battery, the LVR will automatically reset the device internally.

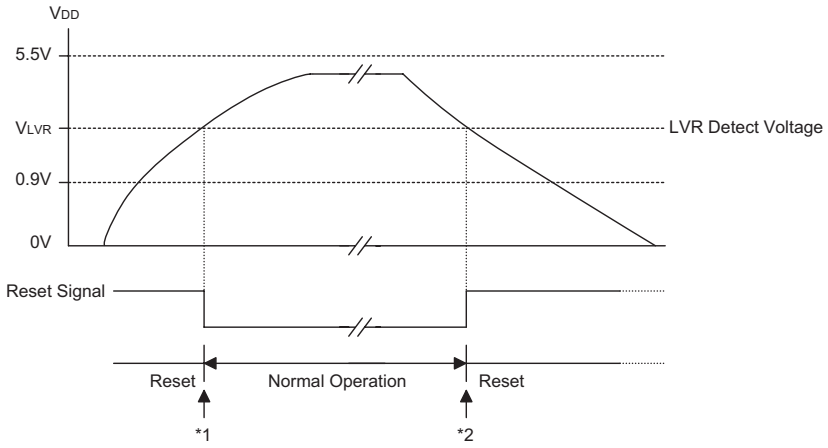
The LVR includes the following specifications:

- The low voltage (0.9V~V_{LVR}) has to remain in its original state to exceed t_{LVR}. If the low voltage state does not exceed t_{LVR}, the LVR will ignore it and will not perform a reset function.
- The LVR uses the "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 at 4MHz system clock.



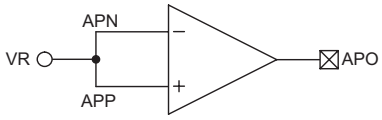
Low Voltage Reset

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

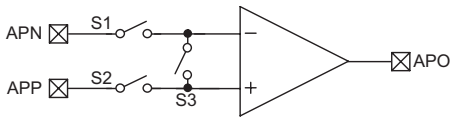
*2: Since the low voltage has to maintain in its original state and exceed t_{LVR}, therefore t_{LVR} delay enter the reset mode.

OP Amplifier/Comparator

The devices include an integrated operational amplifier or comparator, selectable via configuration option. The default is function is comparator. The input voltage offset is adjustable by using a common mode input to calibrate the offset value.



The calibration process is as follows:



- Set bit AOFM=1 to select the offset cancellation mode - this closes switch S3
- Set the ARS bit to select which input pin is the reference voltage - closes either switch S1 or S2
- Adjust bits AOF0~AOF3 until the output status OPAOP has changed.
- Set AOFM=0 to select the normal operating mode

Bit No.	Label	Function
0 1 2 3	AOF0 AOF1 AOF2 AOF3	OP amp/comparator input offset voltage cancellation control bits
4	ARS	OP amp/comparator input offset voltage cancellation reference selection bit 1/0 : select OPP/OPN (CP/CN) as the reference input
5	AOFM	Input offset voltage cancellation mode and OP amp/comparator mode selection 1: input offset voltage cancellation mode 0: OP amp/comparator
6	OPAOP	OP amp/comparator output; positive logic
7	OPAEN	OP amp/comparator enable/disable (1/0) If OP/comparator is disabled, output is floating.

OPAC (1FH) Register

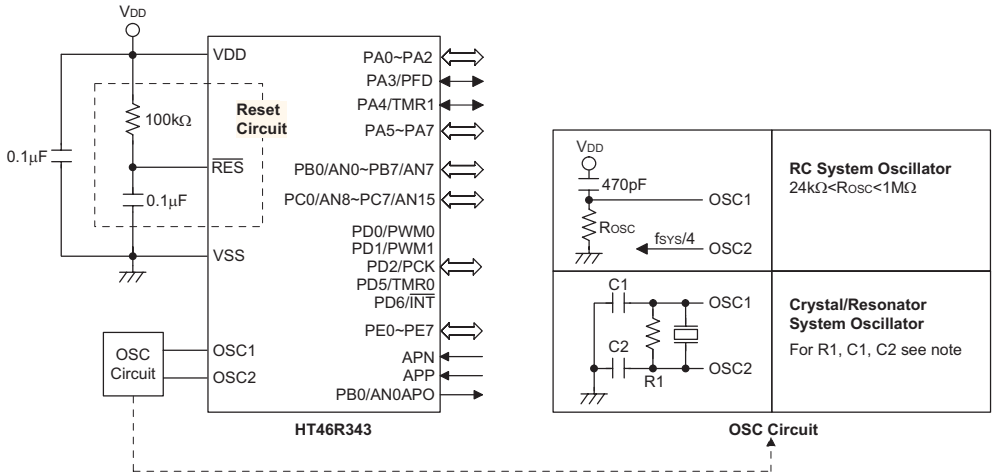
If the OP amp/comparator is disabled, the power consumption will be very small. To ensure that power consumption is minimised when the device is in the Power-down mode, the OP amp/comparator should be switched off by clearing bit OPAEN to 0 before entering the Power-down mode.

Configuration Options

The following table shows the various microcontroller configuration options. All of the configuration options must be properly defined to ensure correct system functioning.

No.	Options
1	WDT clock source: WDTOSC or T1 ($f_{SYS}/4$)
2	WDT function: enable or disable
3	CLRWDT instruction(s): one or two clear WDT instruction(s)
4	System oscillator: RC or crystal
5	Pull-high resistors (PA, PB, PD): none or pull-high
6	PWM enable or disable
7	PA0~PA7 wake-up: enable or disable
8	PFD enable or disable
9	Low voltage reset selection: enable or disable LVR function.
10	Comparator or OP selection

Application Circuits



Note: 1. Crystal/resonator system oscillators
For crystal oscillators, C1 and C2 are only required for some crystal frequencies to ensure oscillation. For resonator applications C1 and C2 are normally required for oscillation to occur. For most applications it is not necessary to add R1. However if the LVR function is disabled, and if it is required to stop the oscillator when VDD falls below its operating range, it is recommended that R1 is added. The values of C1 and C2 should be selected in consultation with the crystal/resonator manufacturer specifications.

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

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

Instruction Set

Introduction

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

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

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs 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] ← $\overline{[m]}$
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 ← $\overline{[m]}$
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] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← ACC + 66H
Affected flag(s)	C
DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [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 ← [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 ← 0 PDF ← 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 ← 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] ← 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] ← [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 ← [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] ← 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 ← 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 ≠ 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

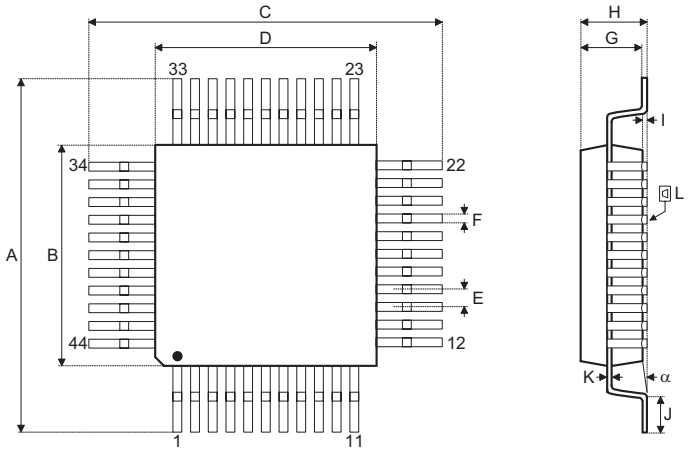
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

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

44-pin QFP (10×10) Outline Dimensions



Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	13	—	13.4
B	9.9	—	10.1
C	13	—	13.4
D	9.9	—	10.1
E	—	0.8	—
F	—	0.3	—
G	1.9	—	2.2
H	—	—	2.7
I	0.25	—	0.5
J	0.73	—	0.93
K	0.1	—	0.2
L	—	0.1	—
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. (Shanghai Sales Office)
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China 200233
Tel: 86-21-6485-5560
Fax: 86-21-6485-0313
http://www.holtek.com.cn

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

Holtek Semiconductor Inc. (Beijing Sales Office)
Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 86-10-6641-0030, 86-10-6641-7751, 86-10-6641-7752
Fax: 86-10-6641-0125

Holtek Semiconductor Inc. (Chengdu Sales Office)
709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016
Tel: 86-28-6653-6590
Fax: 86-28-6653-6591

Holmate Semiconductor, Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holmate.com