2K 8-Bit OTP MCU with OPA & Comparator

Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0075E MCU Reset and Oscillator Circuits Application Note

Features

- Operating voltage:
  f_{SYS}= 4MHz: 2.2V~5.5V
  f_{SYS}= 8MHz: 3.3V~5.5V
  f_{SYS}= 16MHz: 4.5V~5.5V
- OTP Program Memory: 2K×15 ROM
- RAM Data Memory: 128×8 RAM
- 16 bidirectional I/O lines
- External interrupt input shared with an I/O line
- Single 8-bit programmable Timer/Event Counters with overflow interrupt and 7-stage prescaler
- Single 16-bit programmable Timer/Event Counter with overflow interrupt
- External crystal and RC oscillator
- Watchdog Timer function
- PFD for audio frequency generation
- Power down and wake-up functions to reduce power consumption
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}= 5V
- 4-level subroutine nesting
- Bit manipulation instruction
- Table read instructions
- 63 powerful instructions
- All instructions executed in one or two machine cycles
- Low voltage reset function
- Three integrated operational amplifiers
  - one with programmable gain control
- Single comparator with interrupt function
- Embedded 5V LDO in HT46RS03P
- 128×8 EEPROM in HT46RS03E or HT46RS03PE
- Range of package types

General Description

These devices are 8-bit, high performance, RISC architecture microcontroller device specifically designed for operational amplifier applications that can be used to interface directly to ultrasonic transducers.

The advantages of low power consumption, I/O flexibility, programmable frequency divider, timer functions, oscillator options, internal comparator, internal operational amplifiers, power-down and wake-up functions, enhance the versatility of these devices to suit a wide range of application possibilities such as remote control appliances, car reversing systems, level meters, etc.

EEPROM memory is incorporated into HT46RS03E or HT46RS03PE device, which is useful for applications that require an area of non-volatile memory, perhaps to store information such as calibration parameters, part numbers etc. Most features are common to all devices, however, they differ in the provision of an LDO and EEPROM as well as package types.

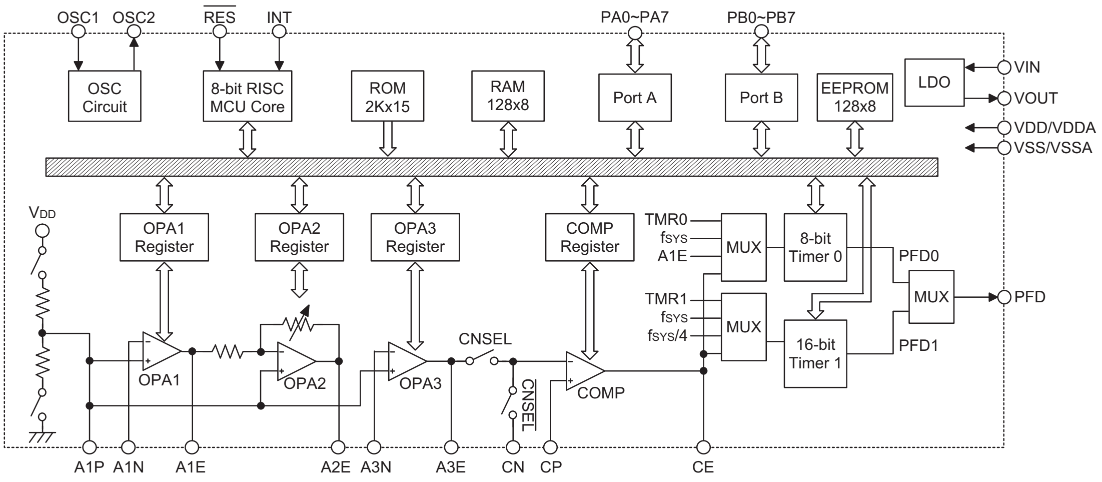
Selection Table

Most features are common to all devices, the main feature distinguishing them are the inclusion of LDO and EEPROM functions, as well as available package types. The following table summarises the main features of each device.

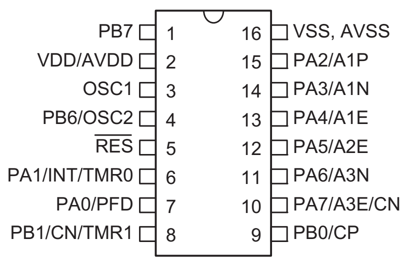
Part No.	VDD	VIN	Program Memory	Data Memory		I/O	Timer		Interrupt		LDO	OPA	CMP	PFD	Stack	Package Types
				SRAM	EEPROM		8-bit	16-bit	Ext.	Int.						
HT46RS03	2.2V~ 5.5V	—	2K×15	128×8	—	16	1	1	1	3	—	3	1	√	4	16NSOP, 16/20DIP, 16/20SSOP
HT46RS03E					128×8											20SOP
HT46RS03P	5.0V	5.5V~ 24V	2K×15	128×8	—	15	1	1	1	3	√	3	1	√	4	16NSOP, 16/20DIP, 16/20SSOP
HT46RS03PE					128×8	16										20/24SOP

Block Diagram

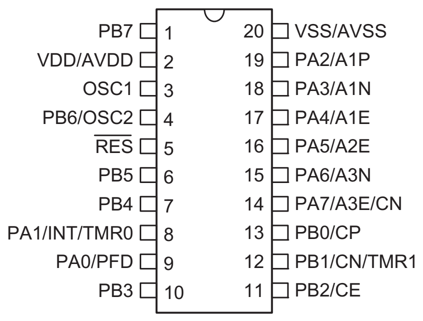
The following block diagram illustrates the main functional blocks.



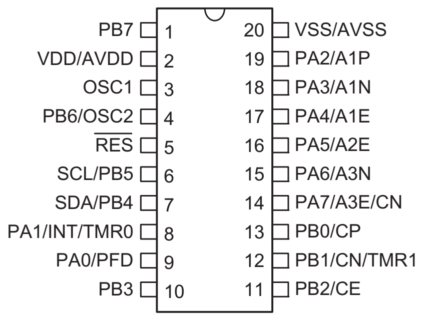
Pin Assignment



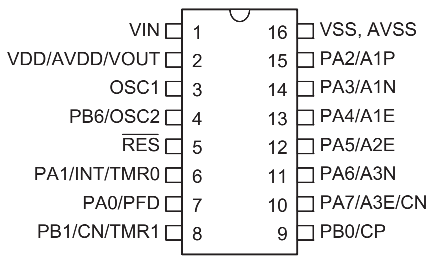
HT46RS03
16 DIP-A/NSOP-A/SSOP-A



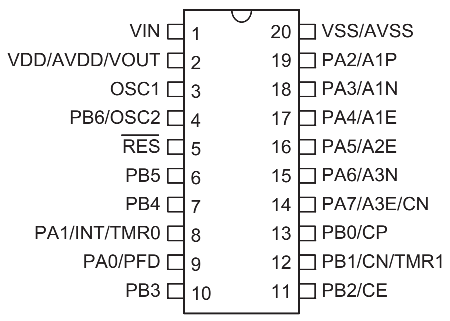
HT46RS03
20 DIP-A/SSOP-A



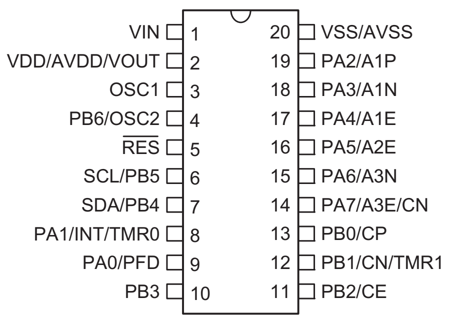
HT46RS03E
20 SOP-A



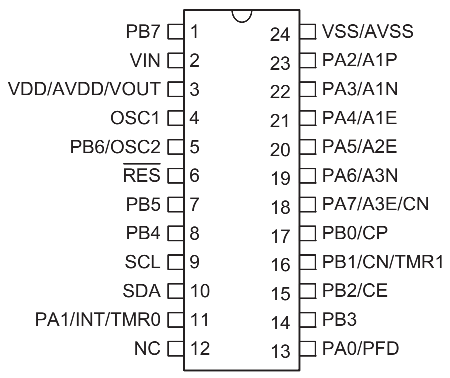
HT46RS03P
16 DIP-B/NSOP-A/SSOP-B



HT46RS03P
20 DIP-B/SSOP-B



HT46RS03PE
20 SOP-A



HT46RS03PE
24 SOP-A

Pin Description

Pin Name	I/O	Configuration Options	Description
PA0/PFD PA1/INT/TMR0 PA2/A1P PA3/A1N PA4/A1E PA5/A2E PA6/A3N PA7/A3E/CN	I/O	Pull-high Wake-up PA0 or PFD	Bidirectional 8-bit input/output port. Each individual pin on this port can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine which pins on the port have pull-high resistors. Pins PA0 and PA1 are pin-shared with PFD and INT/TMR0, respectively. Pins PA2, PA3 and PA4 are pin-shared with the non-inverting input pin A1P, the inverting input pin A1N and the output pin A1E of the 1st OPA, respectively. Pins PA5, PA6 and PA7 are pin-shared with the output pin A2E of the 2nd OPA, the inverting input pin A3N and the output pin of the 3rd OPA, respectively. Pin PA7 is also pin-shared with the inverting input pin CN of Comparator. Software options determine these pins have I/O or analog OPA/Comparator functions. Once selected as analog functions, the I/O functions and pull-high resistors are disabled automatically.
PB0/CP PB1/CN/TMR1 PB2/CE PB3~PB5 PB7	I/O	Pull-high	Bidirectional 7-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 PB0, PB1 and PB2 are pin-shared with the non-inverting input pin CP, the inverting input pin CN and the output pin CE of the Comparator, respectively. Pin PB1 is also pin-shared with TMR1. Software options determine these pins have I/O or analog Comparator functions. Once selected as analog functions, the I/O functions and pull-high resistors are disabled automatically.
OSC1 PB6/OSC2	I I/O	RC or Crystal	OSC1 and 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/PB6 can be used to output the 1/4 system clock or to function as a bidirectional 1-bit input/output line with/without a pull-high resistor by configuration options. At external crystal mode, the pull-high resistor function is disabled automatically.
$\overline{\text{RES}}$	I	—	Schmitt Trigger reset input. Active low.
VIN	—	—	LDO input voltage - Up to 24V
VOUT	—	—	3.3V or 5.0V output which bonds to same pin with VDD, AVDD
VSS	—	—	Negative power supply, ground
AVSS	—	—	Analog negative power supply, for OPAs & Comparator. AVSS and VSS are bonded to the same pin.
VDD	—	—	Positive power supply
AVDD	—	—	Analog positive power supply for OPAs & Comparator. AVDD and VDD are bonded to the same pin internally.

Note: Pin SDA and SCL of the EEPROM are internally connected to pins PB4 and PB5 on the 20-pin package of the HT46RS03PE, respectively.

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

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
		—	f_{SYS}=16MHz	4.5	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=4MHz	—	1	2	mA
		5V		—	2.5	5	mA
I_{DD2}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=8MHz	—	2	4	mA
		5V		—	4	8	mA
I_{DD3}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=16MHz	—	6	12	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 PA, TMR0, TMR1 and INT	—	—	0	—	0.3V_{DD}	V
V_{IH1}	Input High Voltage for PA, TMR0, TMR1 and 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_{LVR1}	Low Voltage Reset 1	—	Configuration option: 2.1V	1.98	2.10	2.22	V
V_{LVR2}	Low Voltage Reset 2	—	Configuration option: 3.15V	2.98	3.15	3.32	V
V_{LVR3}	Low Voltage Reset 3	—	Configuration option: 4.2V	3.98	4.2	4.42	V
I_{OL}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V		10	20	—	mA
I_{OH}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	–2	–4	—	mA
		5V		–5	–10	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V		10	30	50	kΩ
V_{OPA+}	0.5V_{DD} OPAs Bias Voltage	3V~5.5V	—	0.475V_{DD}	0.5V_{DD}	0.525V_{DD}	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
R_{OPA}	Resistance between A1P and VDD. Resistance between A1P and GND.	5V	—	40	100	160	kΩ

LDO – D.C. Characteristics Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{IN}	Conditions				
V_{OUT}	Output Voltage	7V	I_{OUT}=10mA	4.85	5	5.15	V
I_{OUT}	Output Current	7V	—	20	30	—	mA
ΔV_{OUT}	Load Regulation	7V	1mA≤I_{OUT}≤30mA	—	60	100	mV
V_{DIF}	Voltage Drop	—	I_{OUT}=1mA	—	100	—	mV
I_{SS}	Current Consumption	7V	No load	—	2.5	5	μA
$\frac{\Delta V_{OUT}}{\Delta V_{IN} \times V_{OUT}}$	Line Regulation	—	6V≤V_{IN}≤24V I_{OUT}=1mA	—	0.2	—	%/V
V_{IN}	Input Voltage	—	—	—	—	24	V

EEPROM – D.C. Characteristics Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
I_{CC1}	Operating Current	5V	Read at 100kHz	—	—	2	mA
I_{CC2}	Operating Current	5V	Write at 100kHz	—	—	5	mA
V_{IL}	Input Low Voltage	—	—	–1	—	0.3V_{DD}	V
V_{IH}	Input High Voltage	—	—	0.7V_{DD}	—	V_{DD}+0.5	V
V_{OL}	Output Low Voltage	2.4V	I_{OL}=2.1mA	—	—	0.4	V
ILI	Input Leakage Current	5V	V_{IN}=0 or V_{DD}	—	—	1	μA
I_{LO}	Output Leakage Current	5V	V_{OUT}=0 or V_{DD}	—	—	1	μA
I_{STB1}	Standby Current	5V	V_{IN}=0 or V_{DD}	—	—	4	μA
I_{STB2}	Standby Current	2.4V	V_{IN}=0 or V_{DD}	—	—	3	μA
C_{IN}	Input Capacitance (Note)	—	f=1MHz 25°C	—	—	6	pF
C_{OUT}	Output Capacitance (Note)	—	f=1MHz 25°C	—	—	8	pF

Note: These parameters are periodically sampled but not 100% tested

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS1}	System Clock – Crystal OSC	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
		—	4.5V~5.5V	400	—	16000	kHz
f_{SYS2}	System Clock – RC OSC	—	4.5V~5.5V	—	8000	—	kHz
		—	4.5V~5.5V	—	16000	—	kHz
f_{16MRC}	16MHz External RC OSC	5V	R=130kΩ, 25°C	15.2	16.0	16.8	MHz
		4.5V~ 5.5V	R=130kΩ, –40°C~85°C (Note)	14.4	16	17.2	MHz
f_{TIMER}	Timer I/P Frequency – TMR	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
		—	4.5V~5.5V	0	—	24000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{WDT1}	Watchdog Time-out Period WDT Internal Clock Source	—	With prescaler WS2, WS1, WS0 = 111	—	2^{15}	—	t_{WDTOSC}
t_{WDT2}	Watchdog Time-out Period System Clock Source	—	With prescaler WS2, WS1, WS0 = 111	—	2^{17}	—	$*t_{SYS}$
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from Power-down	—	1024	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1	2	ms

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

16MHz RC oscillator, 130kΩ resistor to VDD. factory calibration.

EEPROM – A.C. Characteristics Ta=25°C

Symbol	Parameter	Remark	Standard Mode*		V_{CC}=5V±10%		Unit
			Min.	Max.	Min.	Max.	
f_{SK}	Clock Frequency	—	—	100	—	400	kHz
t_{HIGH}	Clock High Time	—	4000	—	600	—	ns
t_{LOW}	Clock Low Time	—	4700	—	1200	—	ns
t_r	SDA and SCL Rise Time	Note	—	1000	—	300	ns
t_f	SDA and SCL Fall Time	Note	—	300	—	300	ns
$t_{HD:STA}$	START Condition Hold Time	After this period the first clock pulse is generated	4000	—	600	—	ns
$t_{SU:STA}$	START Condition Setup Time	Only relevant for repeated START condition	4000	—	600	—	ns
$t_{HD:DAT}$	Data Input Hold Time	—	0	—	0	—	ns
$t_{SU:DAT}$	Data Input Setup Time	—	200	—	100	—	ns
$t_{SU:STO}$	STOP Condition Setup Time	—	4000	—	600	—	ns
t_{AA}	Output Valid from Clock	—	—	3500	—	900	ns
t_{BUF}	Bus Free Time	Time in which the bus must be free before a new trans-mission can start	4700	—	1200	—	ns
t_{SP}	Input Filter Time Constant (SDA and SCL Pins)	Noise suppression time	—	100	—	50	ns
t_{WR}	Write Cycle Time	—	—	5	—	5	ms

Note: These parameters are periodically sampled but not 100% tested

* The standard mode means V_{CC}=2.2V to 5.5V

Operational Amplifier Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
D.C. Characteristics							
	Operating Voltage	—	—	3.0	—	5.5	V
	Power Down Current	5V	—	—	—	0.1	μA
V_{OPOS1}	Input Offset Voltage	5V	AxOF3~0=1000	−15	—	15	mV
V_{OPOS2}	Input Offset Voltage	5V	After calibration	−4	—	4	mV
V_{CM}	Common Mode Voltage Range	—	—	VSS	—	V_{DD}− 1.4V	V
PSRR	Power-supply Rejection Ratio	—	—	59	80	—	dB
CMRR	Common Mode Rejection Ratio	5V	V_{CM}=0~V_{DD}−1.4V	59	80	—	dB
A.C. Characteristics							
A_{OL}	Open Loop Gain	—	—	80	100	—	dB
SR	Slew rate +, Slew rate -	—	No load	1.8	2.5	—	V/μs
GBW	Gain Bandwidth	—	R_L=1MΩ, C_L=100pF	—	500	—	kHz

Comparator Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
	Comparator Operating Voltage	—	—	3.0	—	5.5	V
	Comparator Operating Current	5V	—	—	—	200	μA
	Comparator Power Down Current	5V	Comparator disabled	—	—	0.1	μA
V_{CMPOS1}	Comparator Input Offset Voltage	5V	COF3~0=1000	−15	—	15	mV
V_{CMPOS2}	Comparator Input Offset Voltage	5V	After calibration	−4	—	4	mV
V_{CM}	Comparator Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}− 1.4V	V
t_{PD}	Comparator Response Time	—	With 2mV overdrive	—	—	2	μs

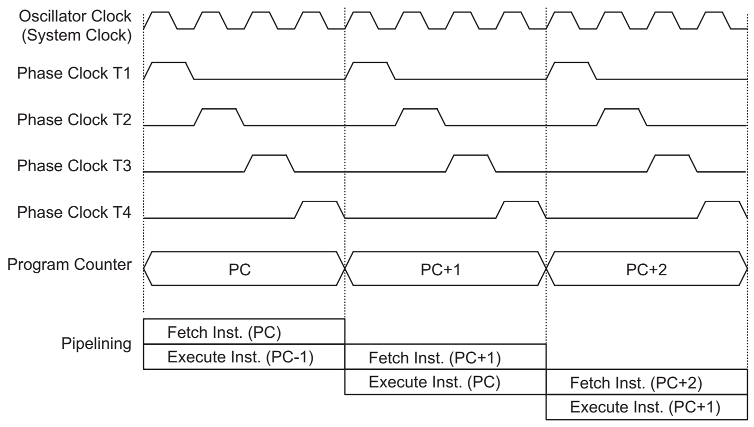
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to the internal system architecture. The range of devices take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions. An 8-bit wide ALU is used in practically all operations of the instruction set. It carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O control system with maximum reliability and flexibility.

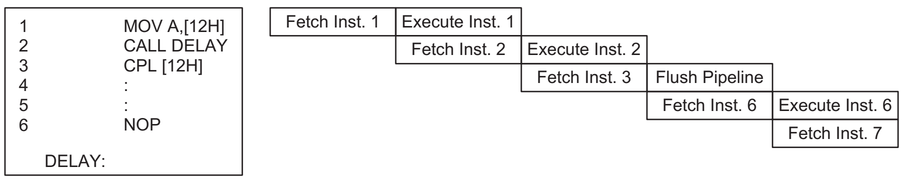
Clocking and Pipelining

The main system clock, derived from either a Crystal/Resonator or RC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the decoding and execution functions. In this way, one T1~T4 clock cycle forms one instruction cycle. Although the fetching and execution of instructions takes place in consecutive instruction cycles, the pipelining structure of the microcontroller ensures that instructions are effectively executed in one instruction cycle. The exception to this are instructions where the contents of the Program Counter are changed, such as subroutine calls or jumps, in which case the instruction will take one more instruction cycle to execute.

For instructions involving branches, such as jump or call instructions, two machine cycles are required to complete instruction execution. An extra cycle is required as the program takes one cycle to first obtain the actual jump or call address and then another cycle to actually execute the branch. The requirement for this extra cycle should be taken into account by programmers in timing sensitive applications



System Clocking and Pipelining



```
1        MOV A,[12H]
2        CALL DELAY
3        CPL [12H]
4        :
5        :
6        NOP

  DELAY:
```

Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demand a jump to a non-consecutive Program Memory address. Note that the Program Counter width varies with the Program Memory capacity depending upon which device is selected. However, it must be noted that only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the user.

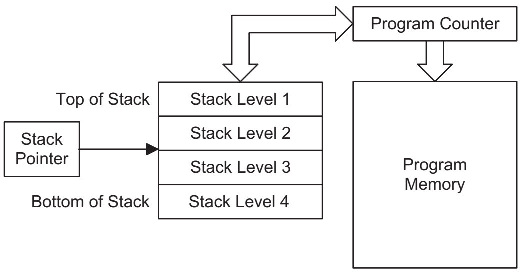
When executing instructions requiring jumps to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc., the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition has been met, the next instruction, which has already been fetched during the present instruction execution, is discarded and a dummy cycle takes its place while the correct instruction is obtained.

The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available for manipulation, the jumps are limited to the present page of memory, that is 256 locations. When such program jumps are executed it should also be noted that a dummy cycle will be inserted.

The lower byte of the Program Counter is fully accessible under program control. Manipulating the PCL might cause program branching, so an extra cycle is needed to pre-fetch. Further information on the PCL register can be found in the Special Function Register section.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack has 4 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 SP, which is also neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, 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, 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 an enabled interrupt takes place, the interrupt request flag will be recorded but the acknowledge signal will be inhibited. When the Stack Pointer is decremented, by RET or RETI, the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. However, when the stack is full, a CALL subroutine in-

Mode	Program Counter Bits										
	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0
External Interrupt	0	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 1 Overflow	0	0	0	0	0	0	0	1	1	0	0
Comparator Interrupt	0	0	0	0	0	0	1	0	0	0	0
Skip	Program Counter + 2										
Loading PCL	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Note: PC10~PC8: Current Program Counter bits
@7~@0: PCL bits
#10~#0: Instruction code address bits
S10~S0: Stack register bits

struction can still be executed which will result in a stack overflow. Precautions should be taken to avoid such cases which might cause unpredictable program branching.

Arithmetic and Logic Unit – ALU

The arithmetic-logic unit or ALU is a critical area of the microcontroller that carries out arithmetic and logic operations of the instruction set. Connected to the main microcontroller data bus, the ALU receives related instruction codes and performs the required arithmetic or logical operations after which the result will be placed in the specified register. As these ALU calculations or operations may result in carry, borrow or other status changes, the status register will be correspondingly updated to reflect these changes. The ALU supports the following functions:

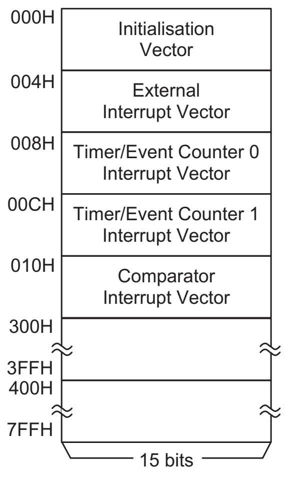
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement INCA, INC, DECA, DEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

Program Memory

The Program Memory is the location where the user code or program is stored. These devices are supplied with One-Time Programmable, OTP, memory where users can program their application code into the device. By using the appropriate programming tools, OTP devices offer users the flexibility to freely develop their applications which may be useful during debug or for products requiring frequent upgrades or program changes. OTP devices are also applicable for use in applications that require low or medium volume production runs.

Structure

The Program Memory has a capacity of 2K by 15. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by separate table pointer registers.


000H
Initialisation Vector
004H
External Interrupt Vector
008H
Timer/Event Counter 0 Interrupt Vector
00CH
Timer/Event Counter 1 Interrupt Vector
010H
Comparator Interrupt Vector
300H
3FFH
400H
7FFH
15 bits

Program Memory Structure

Special Vectors

Within the Program Memory, certain locations are reserved for special usage such as reset and interrupts.

- Location 000H
  This vector is reserved for use by the device reset 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. If the external interrupt pin on the device receives an edge transition, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full. The external interrupt active edge transition type, whether high to low, low to high or both is specified in the MISC register.
- Location 008H
  This internal vector is used by the Timer/Event Counter 0. If a counter overflow occurs, the program will jump to this location and begin execution if the timer/event counter interrupt is enabled and the stack is not full.
- Location 00CH
  This internal vector is used by the Timer/Event Counter 1. If a counter overflow occurs, the program will jump to this location and begin execution if the timer/event counter interrupt is enabled and the stack is not full.
- Location 010H
  This internal vector is used by the comparator interrupt. If the comparator output experiences an edge transition, the program will jump to this location and begin execution if the comparator interrupt is enabled and the stack is not full. The active edge transition type, whether high to low or both is specified by the MISC register CMPES bit.

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the lower order address of the look up data to be retrieved in the table pointer register, TBLP. This register defines the lower 8-bit address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the current Program Memory page or last Program Memory page using the ″TABRDC[m]″ or ″TABRDL [m]″ instructions, respectively. When these instructions are executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as ″0″.

The following diagram illustrates the addressing/data flow of the look-up table:

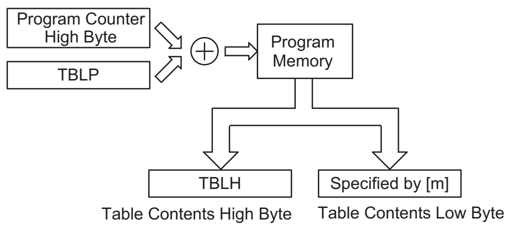


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is ″700H″ which refers to the start address of the last page within the 2K Program Memory. The table pointer is setup here to have an initial value of ″06H″. This will ensure that the first data read from the data table will be at the Program Memory address ″706H″ or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the ″TABRDC [m]″ instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the ″TABRDL [m]″ instruction is executed.

```
tempreg1    db     ?       ; temporary register #1
tempreg2    db     ?       ; temporary register #2
            :
            :
mov         a,06h          ; initialise table pointer - note that this address
                           ; is referenced
mov         tblp,a         ; to the last page or present page
            :
            :
tabrdl      tempreg1       ; transfers value in table referenced by table pointer
                           ; to tempregl
                           ; data at prog. memory address ″706H″ transferred to
                           ; tempreg1 and TBLH
dec         tblp           ; reduce value of table pointer by one
tabrdl      tempreg2       ; transfers value in table referenced by table pointer
                           ; to tempreg2
                           ; data at prog.memory address ″705H″ transferred to
                           ; tempreg2 and TBLH
                           ; in this example the data ″1AH″ is transferred to
                           ; tempreg1 and data ″0FH″ to register tempreg2
                           ; the value ″00H″ will be transferred to the high byte
                           ; register TBLH
            :
            :
org         700h           ; sets initial address of last page
dc          00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
            :
            :
```

Instruction	Table Location Bits										
	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC[m]	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: PC10~PC8: Current Program Counter bits
@7~@0: Table Pointer TBLP bits

Because the TBLH register is a read-only register and cannot be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use the table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of TBLH and subsequently cause errors if used again by the main routine. As a rule it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

Data Memory

The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored. Divided into two sections, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.

Structure

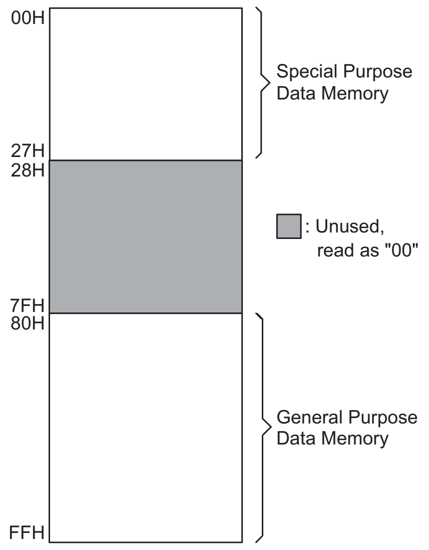
The two sections of Data Memory, the Special Purpose and General Purpose Data Memory are located at consecutive locations. All are implemented in RAM and are 8 bits wide. The start address of the Data Memory is the address ″00H″. Registers which are common to all microcontrollers, such as ACC, PCL, etc., have the same Data Memory address.

General Purpose Data Memory

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user program for both read and write operations. By using the ″SET [m].i″ and ″CLR [m].i″ instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

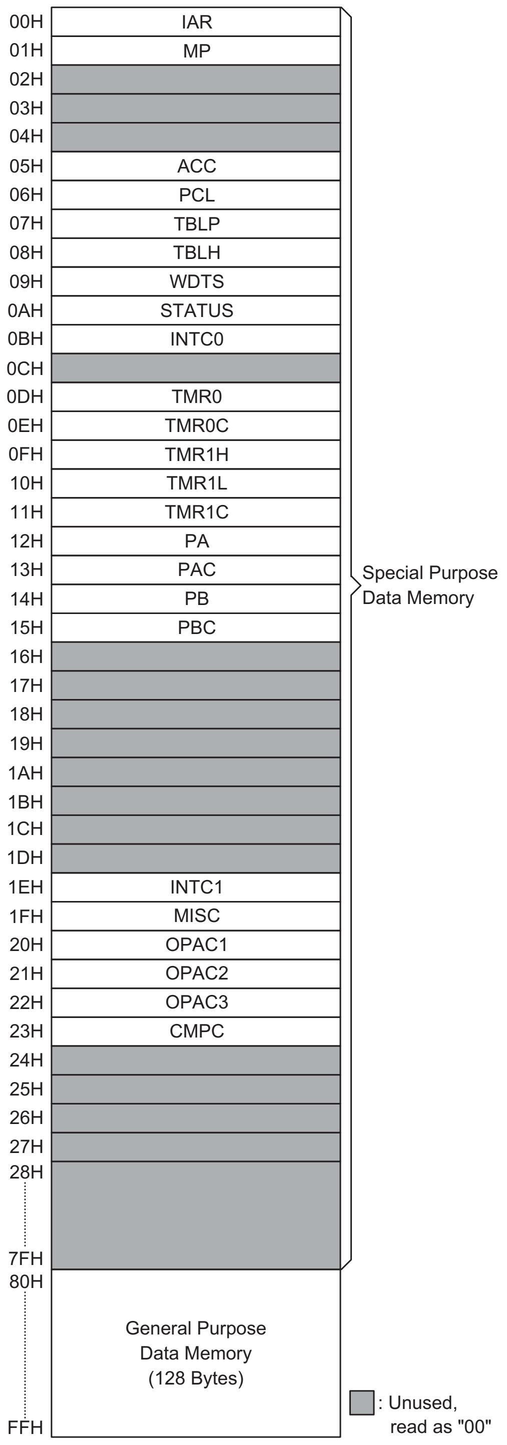
Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and writable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section. Note that for locations that are unused, any read instruction to these addresses will return the value ″00H″.



Data Memory Structure

Note: Most of the Data Memory bits can be directly manipulated using the ″SET [m].i″ and ″CLR [m].i″ with the exception of a few dedicated bits. The Data Memory can also be accessed through the memory pointer register MP.



Special Purpose Data Memory

Special Function Registers

To ensure successful operation of the microcontroller, certain internal registers are implemented in the Data Memory area. These registers ensure correct operation of internal functions such as timers, interrupts, etc., as well as external functions such as I/O data control and comparator operation. The location of these registers within the Data Memory begins at the address 00H. Any unused Data Memory locations between these special function registers and the point where the General Purpose Memory begins is reserved and attempting to read data from these locations will return a value of 00H.

Indirect Addressing Register – IAR

The Indirect Addressing Register, IAR, although having its location in normal RAM register space, does not actually physically exist as a normal register. The method of indirect addressing for RAM data manipulation uses the Indirect Addressing Register and Memory Pointer, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR register will result in no actual read or write operation to this register but rather to the memory location specified by the Memory Pointer, MP. As the Indirect Addressing Register is not physically implemented, reading the Indirect Addressing Register indirectly will return a result of ″00H″ and writing to the registers indirectly will result in no operation.

Memory Pointers – MP

One Memory Pointer, known as MP, is provided. This Memory Pointer is physically implemented in the Data Memory and can be manipulated in the same way as a normal register providing a convenient way with which to address and track data. When any operation to the Indirect Addressing Register is carried out, the actual address that the microcontroller is directed to, is the address specified by the Memory Pointer.

```
data .section  'data'
adres1       db ?
adres2       db ?
adres3       db ?
adres4       db ?
block        db ?
code .section at 0 'code'
org   00h

start:
            mov a,04h           ; setup size of block
            mov block,a
            mov a,offset adres1; Accumulator loaded with first RAM address
            mov mp,a            ; setup memory pointer with first RAM address

loop:
            clr IAR             ; clear the data at address defined by MP
            inc mp              ; increment memory pointer
            sdz block           ; check if last memory location has been cleared
            jmp loop

continue:
```

The important point to note here is that in the example shown above, no reference is made to specific Data Memory addresses.

Accumulator – ACC

The Accumulator is central to the operation of any microcontroller and is closely related with operations carried out by the ALU. The Accumulator is the place where all intermediate results from the ALU are stored. Without the Accumulator it would be necessary to write the result of each calculation or logical operation such as addition, subtraction, shift, etc., to the Data Memory resulting in higher programming and timing overheads. Data transfer operations usually involve the temporary storage function of the Accumulator; for example, when transferring data between one user defined register and another, it is necessary to do this by passing the data through the Accumulator as no direct transfer between two registers is permitted.

Program Counter Low Register – PCL

To provide additional program control functions, the low byte of the Program Counter is made accessible to programmers by locating it within the Special Purpose area of the Data Memory. By manipulating this register, direct jumps to other program locations are easily implemented. Loading a value directly into this PCL register will cause a jump to the specified Program Memory location, however, as the register is only 8-bit wide, only jumps within the current Program Memory page are permitted. When such operations are used, note that a dummy cycle will be inserted.

Look-up Table Registers – TBLP, TBLH

These two special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP is the table pointer and indicates the location where the table data is located. Its value must be setup before any table read commands are executed. Its value can be changed, for example using the "INC" or "DEC" instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

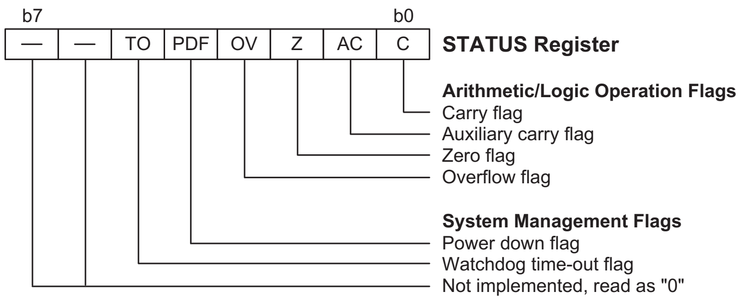
Status Register – STATUS

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). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

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 due to the different instruction operations. The TO flag can be affected only by a system power-up, a WDT time-out or by executing the "CLR WDT" or "HALT" instruction. The PDF flag is affected only by executing the "HALT" or "CLR WDT" instruction or during a system power-up.

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

- **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.
- **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.



Status Register

- **Z** is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- **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.
- **PDF** is cleared by a system power-up or executing the ″CLR WDT″ instruction. PDF is set by executing the ″HALT″ instruction.
- **TO** is cleared by a system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.

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

Interrupt Control Registers – INTC0, INTC1

These 8-bit registers, known as INTC0 and INTC1, control the operation of external, internal timer and comparator interrupts. By setting various bits within this register using standard bit manipulation instructions, the enable/disable function of each interrupt can be independently controlled. A master interrupt bit within this register, the EMI bit, acts like a global enable/disable and is used to set all of the interrupt enable bits on or off. This bit is cleared when an interrupt routine is entered to disable further interrupt and is set by executing the ″RETI″ instruction.

Timer/Event Counter Registers

The devices contains two Timer/Event Counters. One is 8-bits wide and known as Timer/Event Counter 0, while the other is 16-bits wide and known as Timer/Event Counter 1. Timer/Event Counter 0, has an associated timer register known as TMR0, and Timer/Event Counter 1 has two associated timer registers known as TMR1L and TMR1H. These are the register locations where the timer values are located. Two associated control registers, known as TMR0C and TMR1C, contain the setup information for the two individual Timer/Event Counters

Input/Output Ports and Control Registers

Within the area of Special Function Registers, the port PA data I/O register and its associated control register PAC play a prominent role. These registers are mapped to specific addresses within the Data Memory as shown in the Data Memory table. The PA data I/O register, is used to transfer the appropriate output or input data on the PA port. The PAC control register specifies which pins of PA are set as inputs and which are set as outputs. To setup a pin as an input, the corresponding bit of the control register must be set high, for an output it must be set low. During program initialisation, it is important to first setup the control registers to specify which pins are outputs and which are inputs before reading data from or writing data to the I/O ports. One flexible feature of these registers is the ability to directly program single bits using the ″SET [m].i″ and ″CLR [m].i″ instructions. The ability to change I/O pins from output to input and vice versa by manipulating specific bits of the I/O control registers during normal program operation is a useful feature of these devices.

Comparator Control Register – CMPC

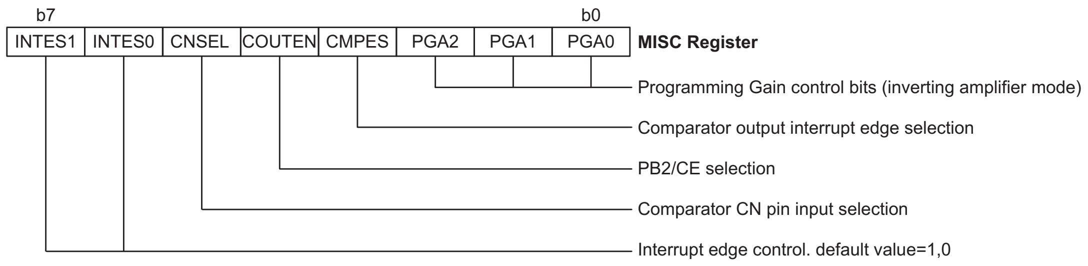
This register is used to control the internal comparator in the device. The internal bits within this register are used to enable and disable the comparator, monitor the output and control the interrupt edge of the comparator.

Miscellaneous Register – MISC

This register is used to provide control over certain internal functions including operational amplifier gain ratio, comparator interrupt transition type, comparator output flag, comparator output or I/O selection and external interrupt transition type control.

Operational Amplifier Control Registers – OPAC1, OPAC2, OPAC3

These three registers are used to control the three internal operational amplifiers in the device. The internal bits within these three registers are used to enable and disable the operational amplifier, monitor the output and control the offset cancellation function.


b7
b0
INTES1
INTES0
CNSEL
COUTEN
CMPES
PGA2
PGA1
PGA0
MISC Register
Programming Gain control bits (inverting amplifier mode)
Comparator output interrupt edge selection
PB2/CE selection
Comparator CN pin input selection
Interrupt edge control. default value=1,0

INTES1	INTES0	
0	0	disable
0	1	rising edge trigger
1	0	falling edge trigger
1	1	dual edge trigger

Miscellaneous Control Register − MISC

EEPROM Data Memory

The HT46RS03PE device contains an internal 1K capacity EEPROM memory with a 128×8 bit structure. An EEPROM, which stands for Electrically Erasable Programmable Read Only Memory, is by its nature a non-volatile form of memory, with data retention even when its power supply is removed. By incorporating this kind of data memory, a whole new host of application possibilities are made available to the designer. The availability of EEPROM storage allows information such as product identification numbers, calibration values, specific user data, system setup data or other product information to be stored directly within the product microcontroller.

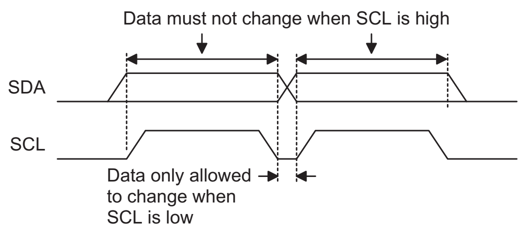
Accessing the EEPROM Data Memory

The internal EEPROM Data Memory has an I^2C structure and as such is accessed via a 2-line serial interface for data transfer. These two lines are the Serial Data line on pin SDA, and the Serial Clock line on pin SCL. The SDA line is bi-directional and is the line where the data is written to and read from the EEPROM. The SCL line is an input line and is the clock signal for both the reading and writing of data. These two EEPROM pins are shared with I/O pins as shown in the table. Any pull-high resistors configuration options for these pin shared pins also remain valid for the EEPROM. Care must be taken if these pins are used as normal I/O pins, as any signals on the pins may be seen by the EEPROM as a valid read or write operation command. If this happens the EEPROM may inadvertently generate signals on its SDA line which could create unexpected programming errors. The Internal EEPROM can be directly controlled using the pin-shared I/O pins or it can be directly connected to an external I^2C bus and controlled by some other external master device. In this latter case care should be taken to ensure that the pin-shared I/Os for the SDA and SCL lines are both setup as inputs.

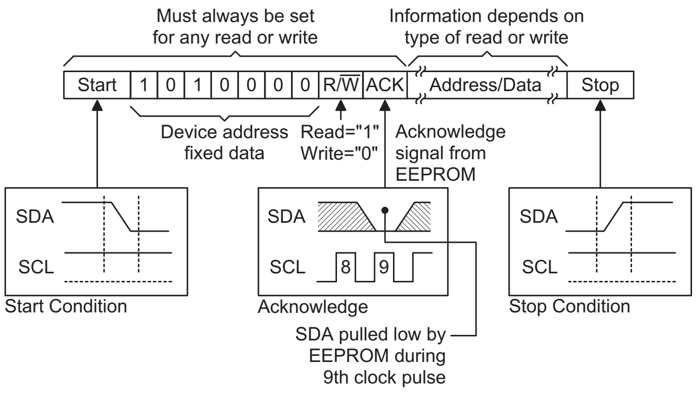
Function	HT46RS03PE			
	20-Pin		24-Pin	
EEPROM Pin	SDA	SCL	SDA	SCL
I/O Pin	PB4	PB5	—	—
Capacity	128×8 bits			

EEPROM I/O Shared Pins

- SCL Pin
  This is the clock input pin to the EEPROM. Writing data into the EEPROM is implemented on the low to high edge of this pin. Reading data out of the EEPROM is implemented on the high to low edge. Any data on the SDA line that is to be sent to the EEPROM on the next rising clock edge is only allowed to change state when the SCL line is low. If the SCL line is high and the data on the SDA line changes, this will be interpreted as a START or STOP condition.
- SDA Pin
  This is the EEPROM data pin. As the pin is used for both reading and writing of data it is bi-directional. As the pin has an open-drain output, it can be wire-OR connected to other external open-drain or open-collector outputs. Otherwise it must be connected to a pull-high resistor for correct operation, which can be implemented using the pull-high configuration option for the corresponding shared I/O pin.


Data must not change when SCL is high
SDA
SCL
Data only allowed to change when SCL is low

Clock/Data Relationship



Must always be set for any read or write
Information depends on type of read or write
Start
1 0 1 0 0 0 0
R/W̄
ACK
Address/Data
Stop
Device address fixed data
Read="1" Write="0"
Acknowledge signal from EEPROM
SDA
SCL
Start Condition
8
9
Acknowledge
Stop Condition
SDA pulled low by EEPROM during 9th clock pulse

Data Transfer Protocol

The sequence of events to read or write data to the EEPROM follows the same pattern. All operations must begin with a START condition and end with a STOP condition. Inserted between the START and STOP conditions are the device address, a read/write bit and address and data information. All control information, addresses and data is sent in 8-bit format to the EEPROM, if successfully received the EEPROM will respond with an acknowledge signal allowing the next 8-bits to be received or transmitted.

- START Condition
  A start condition must be transmitted to the EEPROM prior to transmitting the device address and before any other address or data information is transmitted. A start condition is implemented by a high to low transition on the SDA line with the SCL line high.

- Device Address
  This must always immediately follow the transmitted START condition and is implemented by clocking into the EEPROM a ″1010000″ 7-bit sequence. Clocking the device address into the EEPROM is implemented on the low to high edge of the SCL line. The data on the SDA line must therefore be stable before the SCL line changes from low to high. Any changes on the SDA line when the SCL line is high could be interpreted as a START or STOP condition.

- R/$\overline{W}$ Bit
  This follows the device address sequence and informs the EEPROM if a read or write operation is to be implemented. For a read operation, this bit should be high, for a write operation the bit should be low.

- Acknowledge
  After the EEPROM has successfully received any 8-bits of information, it will transmit an acknowledge signal by pulling the SDA line low. A clock pulse for this EEPROM generated acknowledge signal, which will be the ninth clock pulse, must be supplied on the SCL pin. Therefore after the 7-bit device address and the R/$\overline{W}$ bit, which constitutes a total of 8-bits, has been transmitted to the EEPROM, on the next clock cycle the EEPROM will respond with an acknowledge signal. After this, the 8-bit data address information can then be sent to the EEPROM, after which again the EEPROM will respond with an acknowledge, on the ninth clock pulse. Data information can then be transmitted or received in a similar way.

- Data Address
  Although the EEPROM internal data structure is 128×8 bits and as such requires a 7-bit address to access the data, however an 8-bit address must be transmitted to the EEPROM. The address is transmitted in an MSB bit first format. As the 8th bit, which will be the MSB and the first bit to be transmitted is redundant, its value can be either zero or one. Note that the address is clocked into the EEPROM on the low to high edge of the SCL clock line.

- STOP Condition
  A stop condition must be transmitted to the EEPROM at the end of any read or write operation to terminate the operation. The successful reception of a stop condition by the EEPROM will cause it to enter its Power Down Mode and await the next start bit. A stop condition is implemented by a low to high transition on the SDA line with the SCL line high.

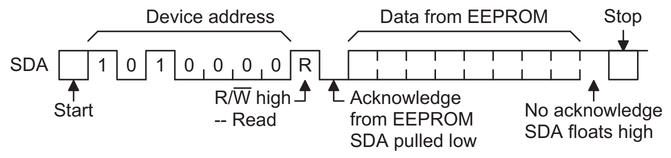
Read Operations

There are three kinds of read operation. These are current address read, random read and sequential read.

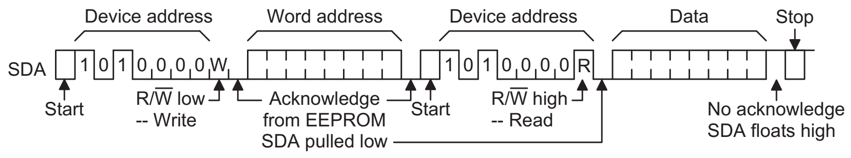
- Current Address Read
  Inside the EEPROM is an internal counter which will point to the present EEPROM address. This internal counter will increment by one each time a read or write operation is executed. The value of the counter will be stored as long as the EEPROM is powered up, only when power is removed will the counter lose its value. If it is required to read the data at the address that this internal counter is pointing to, then it is not necessary to send any address information to the EEPROM. Therefore for a current address read operation to be executed, after the device address and read/write bit information has been sent, if the read/write bit is set high, then after the EEPROM sends its acknowledge signal by pulling SDA low on the ninth clock pulse, it will transmit its internal 8-bits of data at this address on the following eight clock pulses. After the 8-bits of data have been received, no acknowledge is sent but a STOP condition should be transmitted by the receiving device to end the read execution.

- Random Read
  A random read operation allows data at any address within the EEPROM to be read out. For this to happen, a start condition, device address and read/write bit must first be transmitted. Then after the usual acknowledge signal is received from the EEPROM, the required EEPROM internal address information must be transmitted. In this case, as an address is being written to the EEPROM, the read/write bit should be low. After the 8-bit address has been transmitted, in an MSB first format, the EEPROM will respond with another acknowledge signal. At this point, the internal counter is now pointing to the requested address. Now the data can be read out by sending another start, device address and read/write bit information, but this time with the read/write bit set high to indicate a read operation. After the EEPROM acknowledges this in the usual way, the 8-bits of data at the requested address can be read out. As this read operation is the same as the current address read operation, no acknowledge signal will be generated but a stop condition must be sent to the EEPROM to end the read execution.

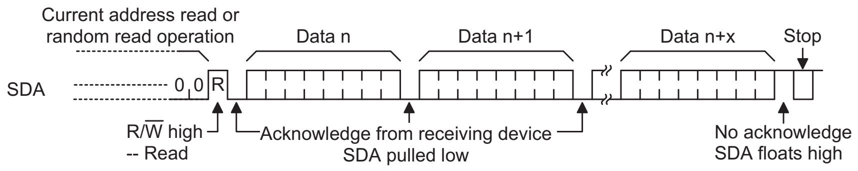
- Sequential Read
  A sequential read allows more than one byte of data to be read out of the EEPROM sequentially. It utilises the EEPROM internal counter, which points to the EEPROM internal address, and which will increment by one automatically after each byte of data has been read out. A sequential read operation is started by first executing a random read or current read to setup the start address of the data to be read out. After the first byte of data has been read out of the EEPROM using the current read or random read, instead of sending a STOP condition, an acknowledge is sent instead. This is achieved by the receiving device pulling the SDA line low on the clock pulse after the eighth data bit. When the EEPROM receives this acknowledge it will automatically increment its internal counter allowing the next byte of data to be read out. Note that when the address counter reaches its maximum value its next value will automatically roll over to zero.



Current Address Read



Random Read


Current address read or random read operation
Data n
Data n+1
Data n+x
Stop
SDA
0 0 R
R/W̄ high -- Read
Acknowledge from receiving device SDA pulled low
No acknowledge SDA floats high

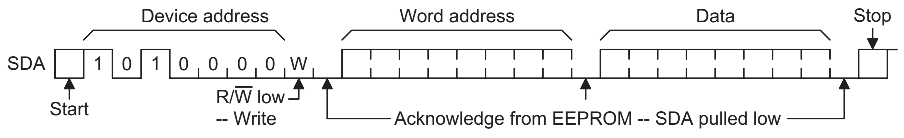
Sequential Read

Write Operations

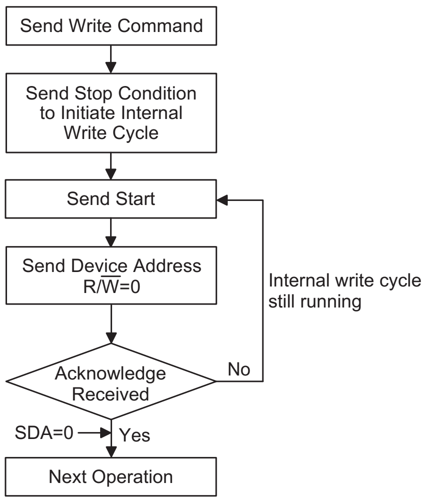
There is only one write operation which is a byte write.

- Byte Write
  A byte write operation allows a single byte of data to be written into the EEPROM. For this to happen, a start condition, device address and read/write bit must first be transmitted. Then after the usual acknowledge signal is received from the EEPROM, the required EEPROM internal address information where the data is to be written must be transmitted. In this case, as an address is being written to the EEPROM, the read/write bit should be low. After the 8-bit address has been transmitted in an MSB format, the EEPROM will respond with another acknowledge signal. At this point the internal counter is now pointing to the requested address. Now the 8-bits of data to be written, in an MSB format, can be transmitted to the EEPROM. When the last bit has been received the EEPROM will respond with the usual acknowledge signal, after which a STOP condition should be transmitted to the EEPROM. After the EEPROM receives the STOP condition it will enter its internal write cycle. The internal write cycle is fully controlled and timed by the EEPROM and when running, no other EEPROM operations can be executed.

- Internal Write Cycle
  After a write operation is executed and when the EEPROM receives the final STOP condition at the end of the write operation, it will enter its internally controlled and timed internal write cycle. When the internal write cycle is executing, no other operations can be carried out on the EEPROM. Before executing further operations on the EEPROM, therefore, a time delay must be provided, whose value should be equal to the maximum write cycle time, t_{WR}, as specified in the EEPROM A.C. Characteristics table. However as only a maximum time is provided, a better method, using polling to determine when the write cycle has finished, can be used. To do this a START condition, followed by a device address and read/write bit low, is transmitted to the EEPROM. If the write cycle has completed the EEPROM will respond with an acknowledge signal, which means the SDA line will be pulled low. If the write cycle has not completed then no acknowledge signal will be generated and the SDA line will remain high.



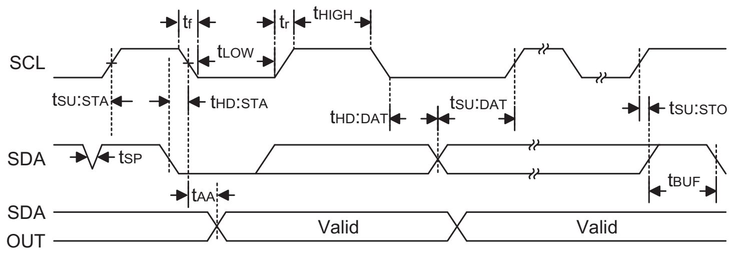
Byte Write


Send Write Command
Send Stop Condition to Initiate Internal Write Cycle
Send Start
Send Device Address R/W=0
Internal write cycle still running
Acknowledge Received
No
SDA=0
Yes
Next Operation

Internal Write Cycle Busy Polling

EEPROM Timing Diagram

The Timing Diagram shows in more detail the timing relationship between SCL and SDA. The specific timing values are provided in the EEPROM A.C. Characteristics table. These timings must be carefully managed by the programmer during application program development, especially if the device is used at higher clock speeds.



Timing Diagram

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. There are sixteen I/O pins whose input or output designation is under user program control and an additional input pin. Additionally, as there are pull-high resistor and wake-up software configurations for each pin, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

Each device has a single I/O port known as Port A, which has a corresponding data register known as PA. This register is mapped to the Data Memory with an addresses as shown in the Special Purpose Data Memory table. Seven of these I/O lines can be used for input and output operations and one line as an input only. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction ″MOV A,[m]″, where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, I/O pins PA0~PA7, PB0~PB7 when configured as an input have the capability of being connected to an internal pull-high resistor. Exception happens when crystal oscillator is selected, PB6 pull-high is always disabled. These pull-high resistors are selectable via configuration option. The pull-high resistors are implemented using weak PMOS transistors.

Port A Wake-up

If the HALT instruction is executed, the device will enter the Power Down Mode, where the system clock will stop resulting in power being conserved, a feature that is important for battery and other low-power applications. Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the PA0~PA7 pins from high to low. After a HALT instruction forces the microcontroller into entering the Power Down Mode, the processor will remain idle or in a low-power state until the logic condition of the selected wake-up pin on Port A changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Note that pins PA0 to PA7 can be selected individually to have this wake-up feature using a configuration option.

I/O Port Control Register

The ports have their own control register, known as PAC and PBC, which controls the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. Pins PA0 to PA7 and PB0 to PB7 are directly mapped to a bit in the port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a ″1″. This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a ″0″, the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

Pin-shared Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For some pins, the chosen function of the multi-function I/O pins is set by configuration options while for others the function is set by application program control.

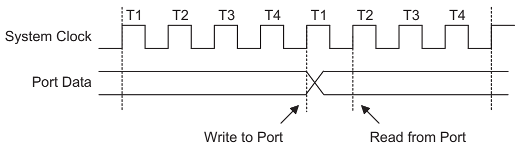
- External interrupt input
  The external interrupt pin INT is pin-shared with the I/O pin PA1. For applications not requiring an external interrupt input, the pin-shared external interrupt pin can be used as a normal I/O pin.
- External Timer/Event Counter input
  Each Timer/Event Counter has an external input pin, known as TMR0 or TMR1 which are pin-shared with I/O pins PA1 and PB1 respectively. For these shared pins to be used as a Timer/Event Counter input, the corresponding Timer/Event Counter must be configured to be in the Event Counter or Pulse Width Measurement Mode. This is achieved by setting the appropriate bits in the relevant Timer/Event Counter Control Register. The pin must also be setup as an input by setting the appropriate bit in the Port Control Register. If the shared pin is to be used as a normal I/O pin, then the external timer input function must be disabled, by ensuring that the corresponding Timer/Event Counter is configured to be in the Off Mode or Timer Mode.
- PFD output
  Each device contains a PFD function whose single output is pin-shared with PA0. The PFD output function of this pin along with the timer source is chosen via configuration option. Note that the corresponding bit of the port control register, PAC.0, must setup the pin as an output to enable the PFD output. If the PAC port control register has setup the pin as an input, then the pin will function as a normal logic input with the usual pull-high option, even if the PFD has been selected.
- Comparator input/outputs
  The device has two comparator inputs and a single comparator output, pin-shared with PB0, PB1 and PB2. The comparator function together with the comparator interrupt transition type is selected via bits in the MISC register. If used as I/O pins then full pull-high resistor selections remain, however if used as comparator inputs then any pull-high resistor selections will be automatically disconnected.

I/O Pin Structures

The diagrams illustrate the I/O pin internal structures. As the exact internal logical construction of the I/O pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins.

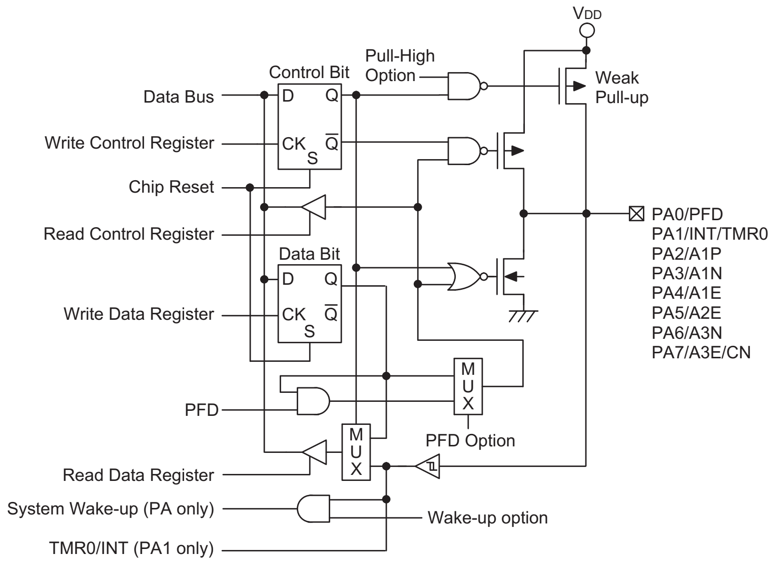
Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, the data registers and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high options have been selected. If the port control registers, are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data register is first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct value into the port control register or by programming individual bits in the port control register using the ″SET [m].i″ and ″CLR [m].i″ instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output port.

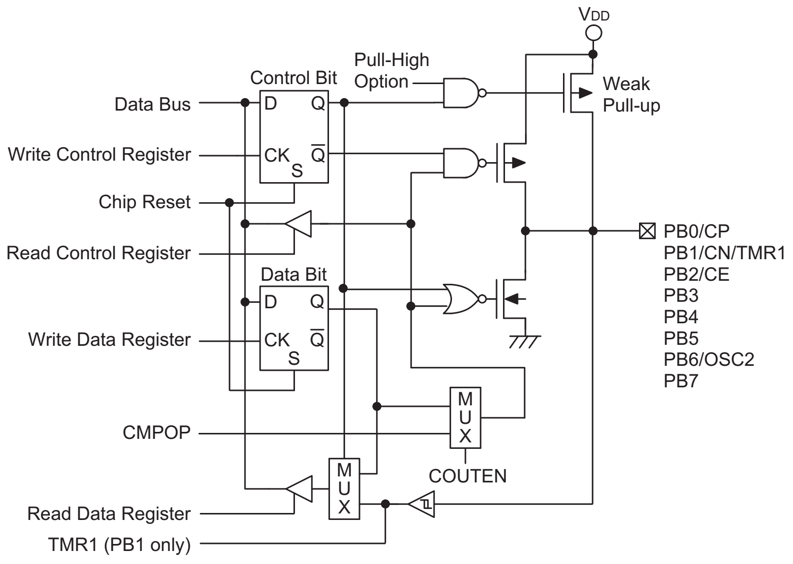


Read/Write Timing

Pins PA0 to PA7 each have a wake-up functions, selected via configuration option. When the device is in the Power Down Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the these pins. Single or multiple pins on Port A can be setup to have this function.



PA0~PA7 Input/Output Ports



PB0~PB7 Input/Output Port

Timer/Event Counters

The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. These devices contain two count-up timers. As each timer has three or four different operating modes, they can be configured to operate as a general timer, an external event counter, a comparator event counter or a pulse width measurement device. The provision of an internal prescaler to the clock circuitry of one of the timer/event counters gives added range to the timer.

There are two types of registers related to the Timer/Event Counters. The first is the register that contains the actual value of the timer and into which an initial value can be preloaded. Reading from this register retrieves the contents of the Timer/Event Counter. The second type of associated register is the Timer Control Register which defines the timer options and determines how the timer is to be used. All timers can have the timer clock configured to come from the internal clock source. In addition, the timer clock source can also be configured to come from an external timer pin.

An external clock source is used when the timer is in the event counting mode, the clock source being provided on the external timer pin, known as TMR0 or TMR1 depending on which timer is selected. These external timer pins are pin-shared with other I/O pins. Depending upon the condition of the T0E or T1E bit in the corresponding Timer Control Register, each high to low, or low to high transition on the external timer input pin will increment the counter by one.

Configuring the Timer/Event Counter Input Clock Source

The internal timer′s clock can originate from various sources, depending upon which timer is chosen. The system clock input timer source is used when the timer is in the timer mode or in the pulse width measurement mode. For Timer/Event Counter 0 this system clock timer source is first divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register bits T0PSC0~T0PSC2.

An external clock source is used when the timer is in the event counter mode, the clock source being provided on an external timer pin TMR0, TMR1, or from the comparator output. Depending upon the condition of the T0E or T1E bit, each high to low, or low to high transition on the external timer pin, comparator or operational amplifier 1 output transition will increment the counter by one. The comparator output clock source is selected using the T0M0/T0M1, COUTSEL, T1M0/T1M1 bit.

Timer Registers – TMR0, TMR1L, TMR1H

The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. These registers are known as TMR0, TMR1L and TMR1H, depending upon which device is used. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin, comparator or operational amplifier 1 output transition. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit counter, or FFFFH for the 16-bit timer, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting.

Note that to achieve a maximum full range count of FFH for the 8-bit counter or FFFFH for the 16-bit counter, the preload registers must first be cleared to all zeros. It should be noted that after power-on, the preload registers will be in an unknown condition. Note that if the Timer/Event Counters are in an OFF condition and data is written to their preload registers, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data registers during this period will remain in the preload registers and will only be written into the actual counter the next time an overflow occurs.

For the 16-bit Timer/Event Counter, which has both low and high byte timer registers, accessing these registers is carried out in a specific way. It must be noted that when using instructions to preload data into the TMR1L low byte register, the data will only be placed in a low byte buffer and not directly into the low byte register. The actual transfer of the data into the low byte register is only carried out when a write to its associated TMR1H high byte register is executed. Using instructions to preload data into the high byte timer register will result in the data being directly written to the high byte register. At the same time the data in the low byte buffer will be transferred into its associated low byte register. For this reason, when preloading data into the 16-bit timer registers, the low byte should be written first. It must also be noted that to read the contents of the low byte register, a read to the high byte register must first be executed to latch the contents of the low byte buffer from its associated low byte register. After this has been done, the low byte register can be read in the normal way. Note that reading the low byte timer register directly will only result in reading the previously latched contents of the low byte buffer and not the actual contents of the low byte timer register.

Timer Control Registers – TMR0C, TMR1C

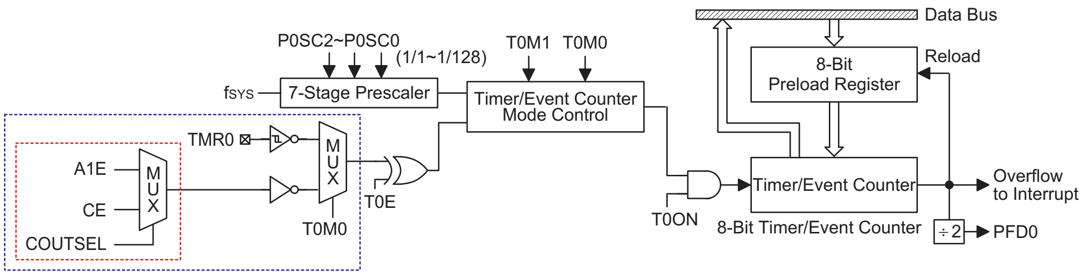
The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in four different modes, the options of which are determined by the contents of their respective control register.

There are two Timer Control Registers, known as TMR0C and TMR1C. It is the Timer Control Register together with their corresponding timer registers that control the full operation of the Timer/Event Counters. Before the timers can be used, it is essential that the appropriate Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

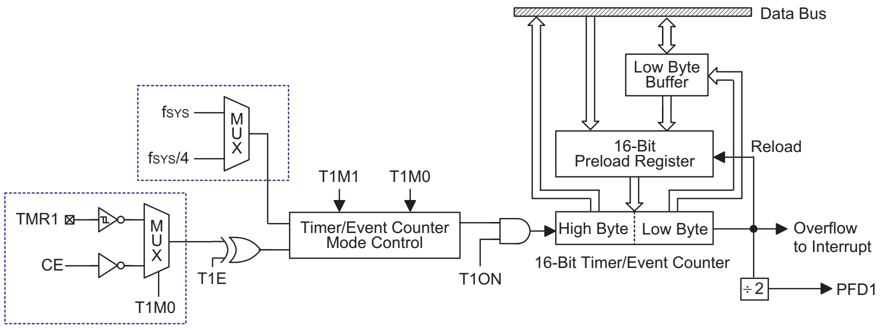
To choose which of the modes the timer is to operate in, either in the timer mode, the event counting mode, the comparator/OPA counting mode or the pulse width measurement mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair T0M1/T0M0 or T1M1/T1M0 respectively, depending upon which timer is used, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as T0ON or T1ON, depending upon which timer is used, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. For Timer/Event Counter 0, which has a prescaler, bits 0~2 of the Timer Control Register determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external or comparator/OPA output transition clock source is used. If the timer is in the event count or pulse width measurement mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as T0E or T1E, depending upon which timer is used.

Configuring the Timer Mode

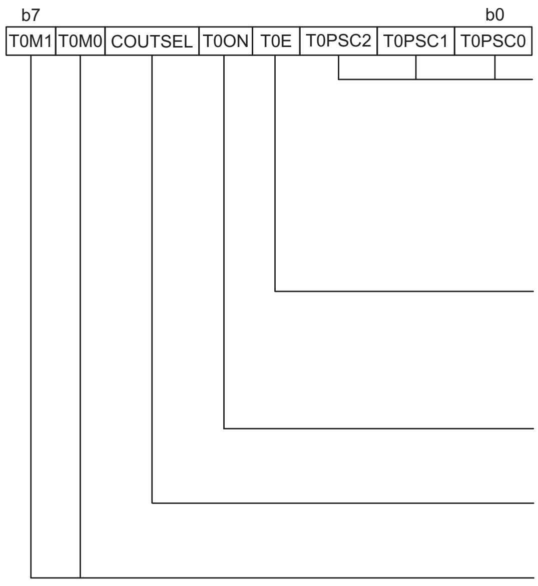
In this mode, the timer can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the counter overflows. To operate in this mode, the bit pair, T0M1/T0M0 or T1M1/T1M0 depending upon which timer is used, must be set to 1 and 0 respectively. In this mode the internal clock is used as the timer clock. Note that for Timer/Event Counter 0, the timer input clock source is f_{SYS}. However, this timer clock source is further divided by a prescaler, the value of which is determined by the bits T0PSC2~T0PSC0 in the Timer Control Register 0. For Timer/Event Counter 1, the timer input clock source $f_{SYS}/4$. There is no prescaler function for Timer/Event Counter 1. The timer-on bit, T0ON or T1ON, depending upon which timer is used, must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one; when the timer is full and overflows, an interrupt signal is generated and the timer will preload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupt is one of the wake-up sources, however, the internal interrupts can be disabled by ensuring that the ET0I or ET1I bits of the INTC0 register are reset to zero.



8-bit Timer/Event Counter 0 Structure



16-bit Timer/Event Counter 1 Structure



TMR0C Register

Timer prescaler rate select

T0PSC2	T0PSC1	T0PSC0	Timer Rate
0	0	0	1:1
0	0	1	1:2
0	1	0	1:4
0	1	1	1:8
1	0	0	1:16
1	0	1	1:32
1	1	0	1:64
1	1	1	1:128

Event Counter active edge select
1: count on falling edge
0: count on rising edge

Pulse Width Measurement active edge select
1: start counting on rising edge, stop on falling edge
0: start counting on falling edge, stop on rising edge

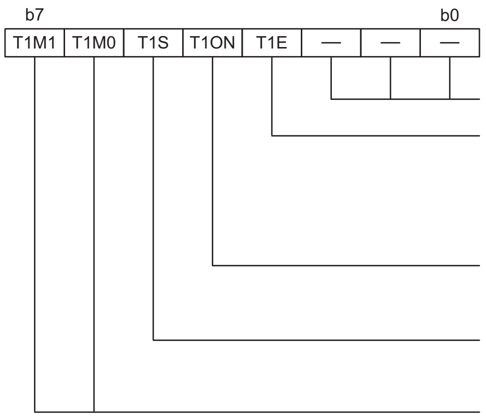
Timer/Event Counter counting enable
1: enable
0: disable

Timer0 input clock sources when (T0M1, T0M0)=(0,0)
1: A1E
0: CE

Operating mode select

T0M1	T0M0	
0	0	comparator/OPA counter mode (from comparator output or operational amplifier output)
0	1	event counter mode (from TMR0)
1	0	timer mode
1	1	pulse width measurement mode

Timer/Event Counter 0 Control Register



TMR1C Register

Not implemented, read as "0"

Event Counter active edge select
1: count on falling edge
0: count on rising edge
Pulse Width Measurement active edge select
1: start counting on rising edge, stop on falling edge
0: start counting on falling edge, stop on rising edge

Timer/Event Counter counting enable
1: enable
0: disable

Clock source select
1: f_{SYS}
0: $f_{SYS}/4$

Operating mode select

T0M1	T0M0	
0	0	Comparator counter mode (Comparator output)
0	1	event counter mode (from TMR1)
1	0	timer mode
1	1	pulse width measurement mode

Timer/Event Counter 1 Control Register

Configuring the Event Counter Mode

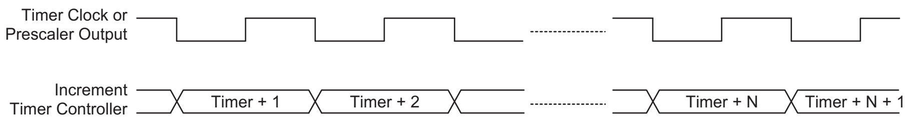
In this mode, a number of externally changing logic events, occurring on the external timer pin, can be recorded by the internal timer. For the timer to operate in the event counter mode, the bit pair, T0M1/T0M0 or T1M1/T1M0, depending upon which timer is used, must be set to 0 and 1 respectively. The timer-on bit T0ON or T1ON, depending upon which timer is used, must be set high to enable the timer to count. Depending upon which counter is used, if T0E or T1E is low, the counter will increment each time the external timer pin receives a low to high transition. If T0E or T1E is high, the counter will increment each time the external timer pin receives a high to low transition. As in the case of the other modes, when the counter is full, the timer will overflow and generate an internal interrupt signal. The counter will then preload the value already loaded into the preload register. As the external timer pins are pin-shared with other I/O pins, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the T0M1/T0M0 or T1M1/T1M0 bits place the Timer/Event Counter in the event counting mode, the second is to ensure that the port control register configures the pin as an input. It should be noted that a timer overflow is one of the interrupt and wake-up sources. Note that the timer interrupts can be disabled by ensuring that the ET0I or ET1I bits in the INTC0 register are reset to zero.

Configuring the Comparator/OPA Counter Mode

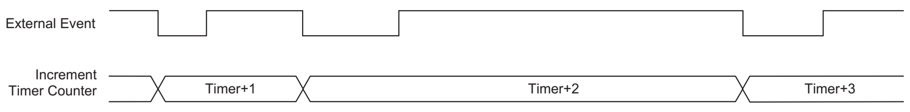
For the timer to operate in the comparator/OPA counter mode, the bit pair, T0M1/T0M0, must both be cleared to zero. The timer-on bit T0ON, must be set high to enable the timer to count. If T0E is low, the counter will increment each time the comparator/OPA output experiences a low to high transition. If T0E is high, the counter will increment each time the comparator/OPA output experiences a high to low transition. As in the case of the other modes, when the counter is full, the timer will overflow and generate an internal interrupt signal. The counter will then preload the value already loaded into the preload register. As the external timer pins are pin-shared with other I/O pins, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the T0M1/T0M0 bits place the Timer/Event Counter 0 in the comparator/OPA counter mode. It should be noted that a timer overflow is one of the interrupt and wake-up sources. Note that the timer interrupts can be disabled by ensuring that the ET0I or ET1I bits in the INTC0 register are reset to zero.

Configuring the Pulse Width Measurement Mode

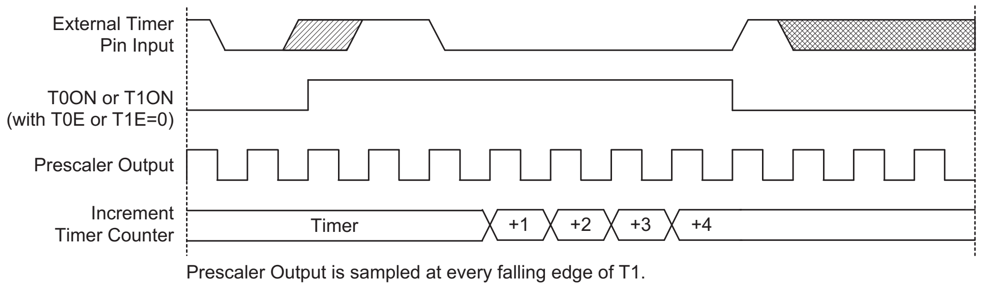
In this mode, the width of external pulses applied to the external timer pin can be measured. In the Pulse Width Measurement Mode the timer clock source is supplied by the internal clock. For the timer to operate in this mode, the bit pair, T0M1/T0M0 or T1M1/T1M0, depending upon which timer is used, must both be set high. Depending upon which counter is used, if the T0E or T1E bit is low, once a high to low transition has been received on the external timer pin, the timer will start counting until the external timer pin returns to its original high level. At this point the T0ON or T1ON bit, depending upon which counter is used, will be automatically reset to zero and the timer will stop counting. If the T0E or T1E bit is high, the timer will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the T0ON or T1ON bit will be automatically reset to zero and the timer will stop counting. It is important to note that in the Pulse Width Measurement Mode, the T0ON or T1ON bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the T0ON or T1ON bit can only be reset to zero under program control. The residual value in the timer, which can now be read by the program, therefore represents the length of the pulse received on the external timer pin. As the T0ON or T1ON bit has now been reset, any further transitions on the external timer pin, will be ignored. Not until the T0ON or T1ON bit is again set high by the program can the timer begin further pulse width measurements. In this way, single shot pulse measurements can be easily made. It should be noted that in this mode the counter is controlled by logical transitions on the external timer pin and not by the logic level. As in the case of the other two



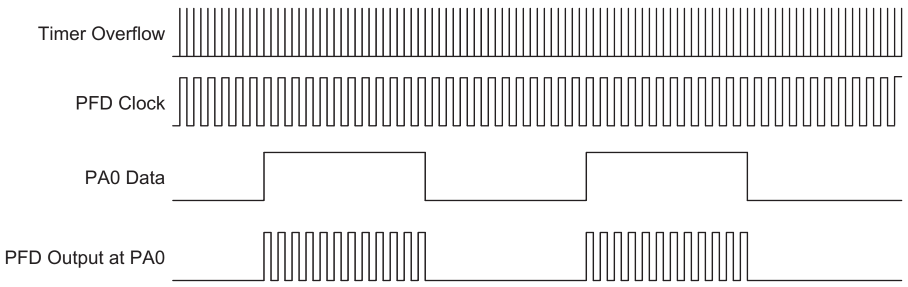
Timer Mode Timing Chart



Event Counter Mode Timing Chart


External Timer Pin Input
T0ON or T1ON (with T0E or T1E=0)
Prescaler Output
Increment Timer Counter
Timer
+1
+2
+3
+4
Prescaler Output is sampled at every falling edge of T1.

Pulse Width Measure Mode Timing Chart



PFD Output Control

modes, when the counter is full, the timer will overflow and generate an internal interrupt signal. The counter will also be reset to the value already loaded into the preload register. If the external timer pin is pin-shared with other I/O pins, to ensure that the pin is configured to operate as a pulse width measuring input pin, two things have to happen. The first is to ensure that the T0M1/T0M0 or T1M1/T1M0 bits place the Timer/Event Counter in the pulse width measuring mode, the second is to ensure that the port control register configures the pin as an input. It should be noted that a timer overflow and corresponding timer interrupt is one of the wake-up sources. Note that the timer interrupts can be disabled by ensuring that the ET0I or ET1I bits in the INTC0 register are reset to zero.

Programmable Frequency Divider – PFD

The PFD output is pin-shared with the I/O pin PA0. The PFD on/off function and its timer source are selected via configuration option, however, if not selected, the pin can operate as a normal I/O pin. The timer overflow signal is the clock source for the PFD circuit. The output frequency is controlled by loading the required values into the timer register and if available the timer prescaler registers to give the required frequency. The timer/event counter, driven by the system clock and if applicable, divided by the prescaler value, will begin to count-up from this preloaded register value until full, at which point an overflow signal will be generated, causing the PFD output to change state. The counter will then be automatically reloaded with the preload register value and once again continue counting-up.

For the PFD output to function, it is essential that the corresponding bit of the Port A control register PAC bit 0 is setup as an output. If setup as an input the PFD output will not function, however, the pin can still be used as a normal input pin. The PFD output will only be activated if bit PA0 is set to ″1″. This output data bit is used as the on/off control bit for the PFD output. Note that the PFD output will be low if the PA0 output data bit is cleared to ″0″.

Using this method of frequency generation, and if a crystal oscillator is used for the system clock, very precise values of frequency can be generated.

Prescaler

Bits T0PSC0~T0PSC2 of the TMR0C register can be used to define a division ratio for the internal clock source of Timer/Event Counter 0 enabling longer time out periods to be setup.

I/O Interfacing

The Timer/Event Counters, when configured to run in the event counter or pulse width measurement mode, require the use of the external timer pins for their operation. As these pins are shared pins they must be configured correctly to ensure that they are setup for use as Timer/Event Counter input pins and not as a normal I/O pins. This is achieved by ensuring that the mode select bits in the Timer/Event Counter control register, select either the event counter or pulse width measurement mode. Additionally the corresponding PAC Port Control Register bits must be set high to ensure that the pins are setup as inputs. Any pull-high resistor register options on these pins will remain valid even if the pin is used as a Timer/Event Counter input.

Programming Considerations

When configured to run in the timer mode, the internal system clock is used as the timer clock source and is therefore synchronised with the overall operation of the microcontroller. In this mode when the appropriate timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector. For the pulse width measurement mode, the internal system clock is also used as the timer clock source but the timer will only run when the correct logic condition appears on the external timer input pin. As this is an external event and not synchronised with the internal timer clock, the microcontroller will only see this external event when the next timer clock pulse arrives. As a result, there may be small differences in measured values requiring programmers to take this into account during programming. The same applies if the timer is configured to be in the event counting mode, which again is an external event and not synchronised with the internal system or timer clock.

When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, this should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialised before using them for the first time. The associated timer enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The edge select, timer mode and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application. It is also important to ensure that an initial value is first loaded into the timer registers before the timer is switched on; this is because after power-on the initial values of the timer registers are unknown. After the timer has been initialised the timer can be turned on and off by controlling the enable bit in the timer control register.

When the Timer/Event counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the timer interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a Timer/Event counter overflow will also generate a wake-up signal if the device is in a Power-down condition. This situation may occur if the Timer/Event Counter is in the Event Counting Mode and if the external signal continues to change state. In such a case, the Timer/Event Counter will continue to count these external events and if an overflow occurs the device will be woken up from its Power-down condition. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the ″HALT″ instruction to enter the Power Down Mode.

Timer Program Example

The following example program section contains one internal 8-bit Timer/Event Counter, one internal 16-bit Timer/Event Counter. The program shows how the Timer/Event Counter registers are setup along with how the interrupts are enabled and managed. Note how the Timer/Event Counter is turned on, by setting bit 4 of the respective Timer Control Register. The Timer/Event Counter can be turned off in a similar way by clearing the same bit. This example program sets the Timer/Event Counters to be in the timer mode, which uses the internal system clock as their clock source.

```
:                       :
org 04h             ; external interrupt vector
reti
org 08h             ; Timer Counter 0 interrupt vector
jmp tmr0int         ; jump here when Timer 0 overflows
org 0ch             ; Timer Counter 1 interrupt vector
jmp tmr1int         ; jump here when Timer 1 overflows
:                       :
org 20h             ; main program
:                       :
;internal Timer 0 interrupt routine
tmr0int:
:
; Timer 0 main program placed here
:
reti
:
;internal Timer 1 interrupt routine
tmr1int:
:
;Timer 1 main program placed here
:
reti
:
begin:
;setup Timer 0 registers
mov a,09bh          ; setup Timer 0 preload value
mov tmr0,a
mov a,081h          ; setup Timer 0 control register
mov tmr0c,a         ; timer mode and prescaler set to /2
;setup Timer 1 registers
clr tmr1l
clr tmr1h           ; clear timer register to give maximum count value
mov a,080h          ; setup Timer 1 control register
mov tmr1c,a         ; timer mode - Timer 1 has no prescaler
;setup interrupt register
mov a,00dh          ; enable master interrupt and both timer
; interrupts
mov intc0,a
:                       :
set tmr0c.4         ; start Timer 0
set tmr1c.4         ; start Timer 1
:                       :
```

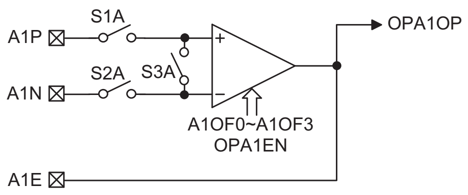
Operational Amplifiers

There are three fully integrated Operational Amplifiers in the device, OPA1, OPA2 and OPA3. These OPAs can be used for signal amplification according to specific user requirements. The OPAs can be disabled or enabled entirely under software control using internal registers. Operation Amplifier OPA2 includes a programmable gain control function which is setup using an internal register.

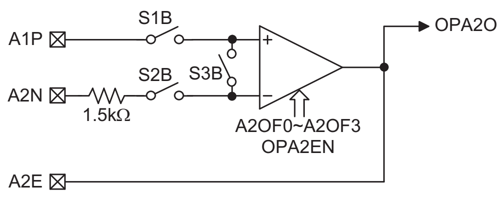
Operational Amplifier Registers

The internal Operational Amplifiers are fully under the control of three internal registers, OPAC1, OPAC2 and OPAC3, one for each amplifier. These control the enable/disable function, polarity and calibration function.

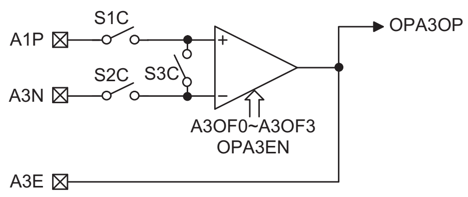
ARS1	A1OFM	S1A	S2A	S3A
0	0	ON	ON	OFF
0	1	OFF	ON	ON
1	0	ON	ON	OFF
1	1	ON	OFF	ON

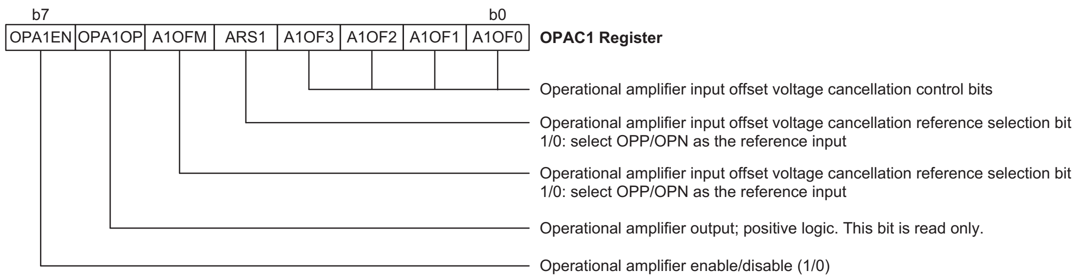


ARS2	A2OFM	S1B	S2B	S3B
0	0	ON	ON	OFF
0	1	OFF	ON	ON
1	0	ON	ON	OFF
1	1	ON	OFF	ON



ARS3	A3OFM	S1C	S2C	S3C
0	0	ON	ON	OFF
0	1	OFF	ON	ON
1	0	ON	ON	OFF
1	1	ON	OFF	ON





OPAC1 Register

b7							b0
OPA1EN	OPA1OP	A1OFM	ARS1	A1OF3	A1OF2	A1OF1	A1OF0

A1OF3~A1OF0: Operational amplifier input offset voltage cancellation control bits

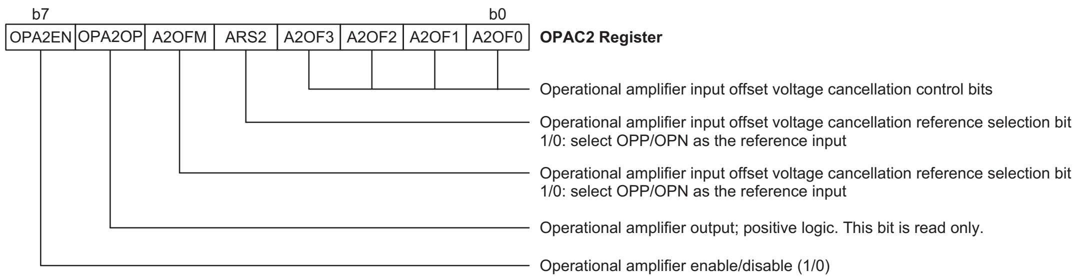
ARS1: Operational amplifier input offset voltage cancellation reference selection bit
1/0: select OPP/OPN as the reference input

A1OFM: Operational amplifier input offset voltage cancellation reference selection bit
1/0: select OPP/OPN as the reference input

OPA1OP: Operational amplifier output; positive logic. This bit is read only.

OPA1EN: Operational amplifier enable/disable (1/0)

Operational Amplifier Control Register – OPAC1



OPAC2 Register

b7							b0
OPA2EN	OPA2OP	A2OFM	ARS2	A2OF3	A2OF2	A2OF1	A2OF0

A2OF3~A2OF0: Operational amplifier input offset voltage cancellation control bits

ARS2: Operational amplifier input offset voltage cancellation reference selection bit
1/0: select OPP/OPN as the reference input

A2OFM: Operational amplifier input offset voltage cancellation reference selection bit
1/0: select OPP/OPN as the reference input

OPA2OP: Operational amplifier output; positive logic. This bit is read only.

OPA2EN: Operational amplifier enable/disable (1/0)

Operational Amplifier Control Register – OPAC2

Operational Amplifier Operation

The OPAs are connected together internally in a specific way as shown in the block diagram. The output of OPA3 can also be connected to the internal comparator as shown in the block diagram. Each of the OPAs has its own control register, with the name OPAC1, OPAC2 and OPAC3 which are used to control the enable/disable function, the output polarity and the calibration procedure. The MISC register is used to control the programmable gain function of OPA2.

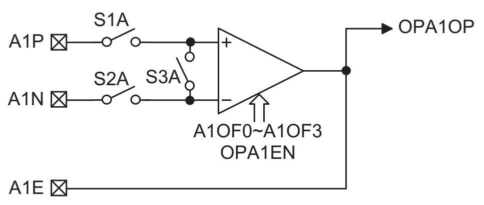
Each of the internal OPAs allows for a commode mode adjustment method of its input offset voltage.

The calibration steps are as following:

1. Set A1OFM=1 to setup the offset cancellation mode, here S3A is closed.
2. Set ARS1 to select which input pin is to be used as the reference voltage - S1 or S2 is closed
3. Adjust A1OF0~A1OF3 until the output status changes
4. Set A1OFM = 0 to restore the normal OPA mode
5. Repeat the same procedure from steps 1 to 4 for OPA2 & OPA3

ARS1	A1OFM	S1A	S2A	S3A
0	0	ON	ON	OFF
0	1	OFF	ON	ON
1	0	ON	ON	OFF
1	1	ON	OFF	ON

There are integrated bias resistors which generate a 1/2VDD voltage reference for the OPAs. A configuration selects if this internal reference is to be used. In addition any of the three OPAs must also be selected to activate this reference. These are enabled by setting OPAxEN =1.



OPAC3 Register

b7							b0
OPA3EN	OPA3OP	A3OFM	ARS3	A3OF3	A3OF2	A3OF1	A3OF0

A3OF3~A3OF0: Operational amplifier input offset voltage cancellation control bits

ARS3: Operational amplifier input offset voltage cancellation reference selection bit
1/0: select OPP/OPN as the reference input

A3OFM: Operational amplifier input offset voltage cancellation reference selection bit
1/0: select OPP/OPN as the reference input

OPA3OP: Operational amplifier output; positive logic. This bit is read only.

OPA3EN: Operational amplifier enable/disable (1/0)

Operational Amplifier Control Register 3

MISC Register

b7							b0
INTES1	INTES0	CNSEL	COUTEN	CMPES	PGA2	PGA1	PGA0

PGA2~PGA0: Programming Gain control bits (inverting amplifier mode)

PGA2	PGA1	PGA0	Gain
0	0	0	1.0
0	0	1	1.0
0	1	0	3.9
0	1	1	6.8
1	0	0	9.7
1	0	1	12.5
1	1	0	15.4
1	1	1	18.3

CMPES: Comparator output interrupt edge selection
0: interrupt on falling edge
1: interrupt on dual edge

COUTEN: PB2/CE selection
0: PB2/CE is as a GPIO pin
1: PB2/CE is comparator output, and the status of CE can be ready by reading the PB2 register

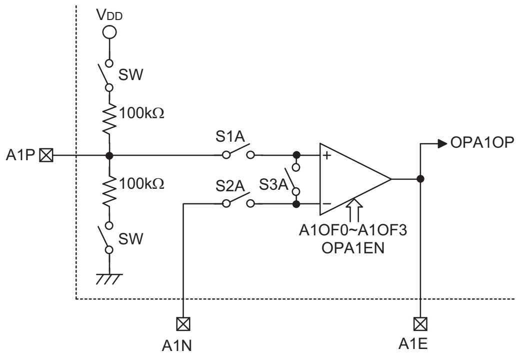
CNSEL: Comparator CN pin input selection
0: from CN pin
1: from A3E

INTES1, INTES0: Interrupt edge control

Miscellaneous Control Register − MISC

When the bias control configuration option is selected for the external bias voltage generator, two internal switches, as shown in the diagram, will be opened allowing the A1P pin to be in a floating state. In this case an external reference voltage reference can be connected according to specific requirements.

R_{OPA} is a 100kΩ resistor. V_{OPA+} is the voltage on the A1P pin.



The switches are closed when (1) & (2) occur.

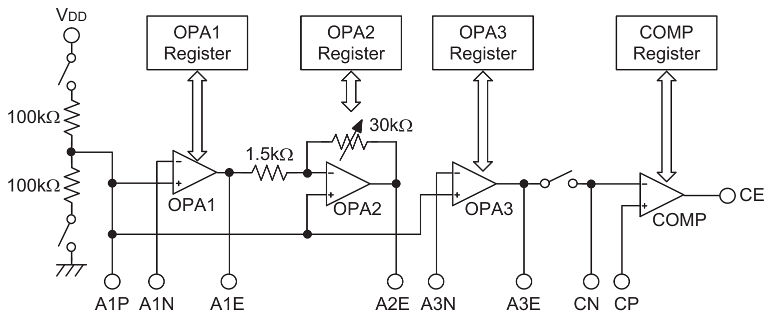
(1) The Configuration option selects the internal bias generator.

(2) The OPA1EN or OPA2EN or OPA3EN bits are set.

SW is opened when (3) or (4) occur.

(3) The Configuration option selects the internal bias generator, but none of the OPA1EN, OPA2EN or OPA3EN bits are set.

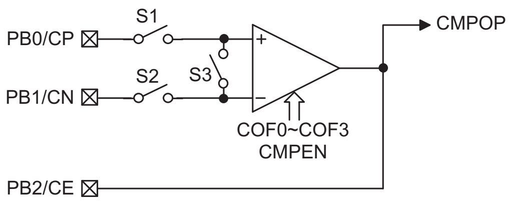
(4) The Configuration option selects the external bias generator.



OPA Block Diagram

Comparator

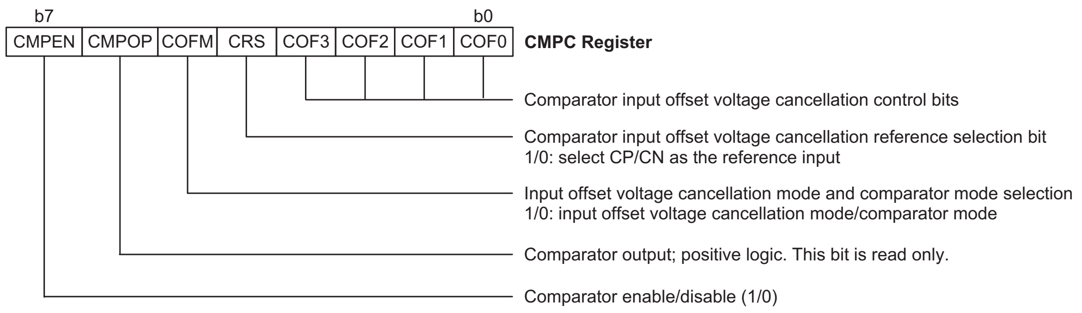
The device contains a fully integrated Comparator whose operation is controlled by the Comparator Control Register, otherwise known as the CMPC register. The CMPEN bit within this register is used as the enable or disable bit for the comparator function. If the CMPEN bit is cleared to ″0″, the Comparator is disabled and the PB0/CP, PB1/CN, PB2/CE will be setup as general purpose I/O pins. If the CMPEN bit is set high, the Comparator function is enabled, resulting in the PB0/CP and PB1/CN pins being setup as Comparator input pins, and the PB2/CE pin being setup as the Comparator output pin. Note that the COUTEN bit in the MISC register must also be set high for the comparator function to be active.



As the CP, CN and CE are pin-shared with PB0, PB1 and PB2, once the Comparator function is enabled, the internal I/O register bits for I/O pins PB0 and PB1 cannot be used, however the I/O register bit for PB2 can be used as an input to read the CE status, if COUTEN is high. Note that the PB0 and PB1 I/O function, the PB2 output function and their pull-high resistors are disabled automatically if COUTEN is high. Once the Comparator function is used, pin PB2 can be used as a General Purpose I/O pin when COUTEN is low. Software instructions determine if the Comparator function is to be used.

CRS	COFM	S1	S2	S3
0	0	ON	ON	OFF
0	1	OFF	ON	ON
1	0	ON	ON	OFF
1	1	ON	OFF	ON

CMPEN	COUTEN	PB0, PB1, PB2
0	x	PB0, PB1, PB2 are setup as normal I/O pins.
1	0	PB0, PB1 is comparator input pins and PB2 is a normal I/O.
1	1	PB0, PB1 are comparator input pins and PB2 is a comparator output pin. PB2 can read the Comparator output status.



Comparator input offset voltage cancellation control bits

Comparator input offset voltage cancellation reference selection bit
1/0: select CP/CN as the reference input

Input offset voltage cancellation mode and comparator mode selection
1/0: input offset voltage cancellation mode/comparator mode

Comparator output; positive logic. This bit is read only.

Comparator enable/disable (1/0)

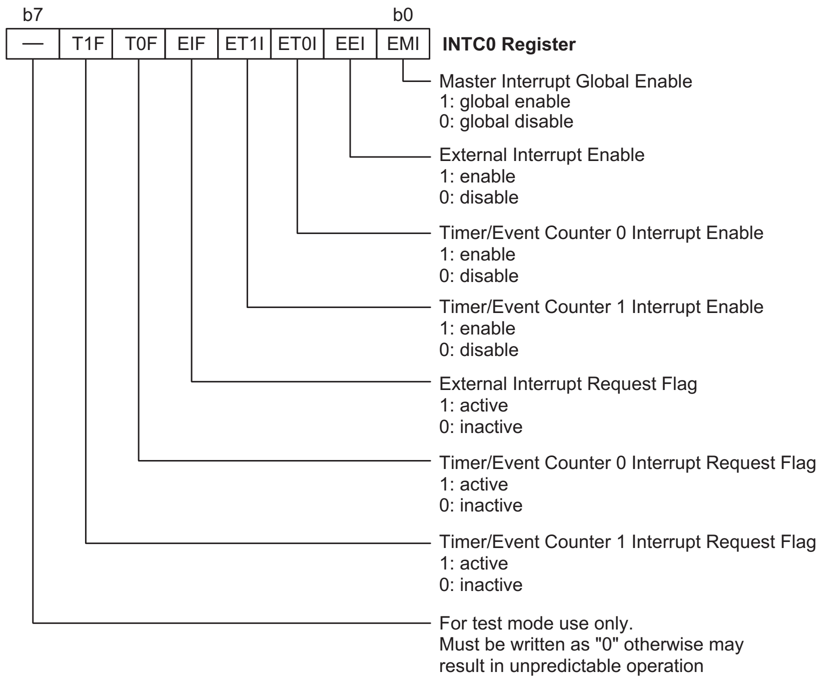
Comparator Control Register

Interrupts

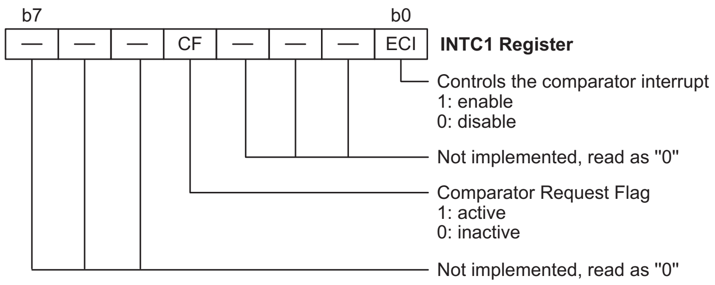
Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter or a comparator requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. Each device contains a single external interrupt and several internal interrupts functions. The external interrupt is controlled by the action of the external INT pin, while the internal interrupts are controlled by the Timer/Event Counter overflows and the comparator interrupt.

Interrupt Registers

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by two INTC0 and INTC1 registers, which are located in the Data Memory. By controlling the appropriate enable bits in these registers each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request flag will be set by the microcontroller. The global enable flag if cleared to zero will disable all interrupts.


b7
b0
—
T1F
T0F
EIF
ET1I
ET0I
EEI
EMI
INTC0 Register
Master Interrupt Global Enable
1: global enable
0: global disable
External Interrupt Enable
1: enable
0: disable
Timer/Event Counter 0 Interrupt Enable
1: enable
0: disable
Timer/Event Counter 1 Interrupt Enable
1: enable
0: disable
External Interrupt Request Flag
1: active
0: inactive
Timer/Event Counter 0 Interrupt Request Flag
1: active
0: inactive
Timer/Event Counter 1 Interrupt Request Flag
1: active
0: inactive
For test mode use only.
Must be written as "0" otherwise may
result in unpredictable operation

Interrupt Control Register 0



Interrupt Control Register 1

Interrupt Operation

A Timer/Event Counter overflow, an active edge on the external interrupt pin or a comparator output transition will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP statement which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI statement, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagram with their order of priority.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded. If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, 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.

Interrupt Priority

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 case of simultaneous requests, the following table shows the priority that is applied. These can be masked by resetting the EMI bit.

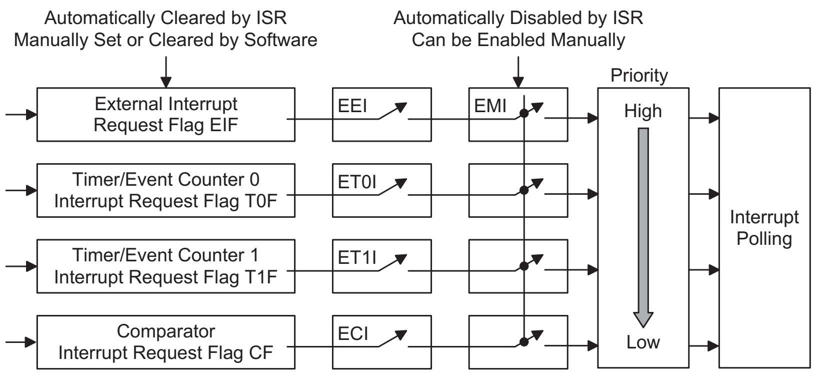
Interrupt Source	Priority
External Interrupt	1
Timer/Event Counter 0 Overflow	2
Timer/Event Counter 1 Overflow	3
Comparator Interrupt	4

In cases where both external and internal interrupts are enabled and where an external and internal interrupt occurs simultaneously, the external interrupt will always have priority and will therefore be serviced first. Suitable masking of the individual interrupts using the interrupt registers can prevent simultaneous occurrences.

External Interrupt

For an external interrupt to occur, the global interrupt enable bit, EMI, and external interrupt enable bit, EEI, must first be set. An actual external interrupt will take place when the external interrupt request flag, EIF, is set, a situation that will occur when an edge transition appears on the external INT line. The type of transition that will trigger an external interrupt, whether high to low, low to high or both is determined by the INTES0 and INTES1 bits, which are bits 6 and 7 respectively, in the MISC control register. These two bits can also disable the external interrupt function.

INTES1	INTES0	Edge Trigger Type
0	0	Disable
0	1	Rising Edge Trigger
1	0	Falling Edge Trigger
1	1	Dual Edge Trigger


Automatically Cleared by ISR
Manually Set or Cleared by Software
Automatically Disabled by ISR
Can be Enabled Manually
Priority
External Interrupt Request Flag EIF
Timer/Event Counter 0 Interrupt Request Flag T0F
Timer/Event Counter 1 Interrupt Request Flag T1F
Comparator Interrupt Request Flag CF
EEI
ET0I
ET1I
ECI
EMI
High
Low
Interrupt Polling

Interrupt Scheme

The external interrupt pin is pin-shared with the I/O pin PA1 and can only be configured as an external interrupt pin if the corresponding external interrupt enable bit in the INTC0 register has been set and the edge trigger type has been selected using the MISC register. The pin must also be setup as an input by setting the corresponding PAC.1 bit in the port control register. When the interrupt is enabled, the stack is not full and a transition appears on the external interrupt pin, a subroutine call to the external interrupt vector at location 04H, will take place. When the interrupt is serviced, the external interrupt request flag, EIF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor connections on this pin will remain valid even if the pin is used as an external interrupt input.

Timer/Event Counter Interrupts

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, ET0I or ET1I, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, T0F or T1F, is set, a situation that will occur when the relevant Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter 0 overflow occurs, a subroutine call to the timer interrupt vector at location 08H, will take place. Similarly a subroutine call to the timer interrupt vector at 0CH will take place when Timer/Event Counter 1 overflows. When the interrupt is serviced, the timer interrupt request flag, T0F or T1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Comparator Interrupt

For an comparator interrupt to occur, the global interrupt enable bit, EMI, and the corresponding interrupt enable bit, ECI, must be first set. An actual comparator interrupt will take place when the comparator request flag, CF, is set, a situation that will occur when the internal comparator output has changed state. When the interrupt is enabled, the stack is not full and the comparator output changes state, a subroutine call to the comparator interrupt vector at location 10H for the device, will take place. When the interrupt is serviced, the comparator interrupt request flag, CF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Programming Considerations

By disabling the interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by a software instruction.

It is recommended that programs do not use the "CALL subroutine" instruction 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 the interrupt is not well controlled, the original control sequence will be damaged once a "CALL subroutine" is executed in the interrupt subroutine.

All of these interrupts have the capability of waking up the processor when in the Power Down Mode.

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 may corrupt the desired control sequence, then the contents should be saved in advance.

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontroller. In this case, internal circuitry will ensure that the microcontroller, after a short delay, will be in a well defined state and ready to execute the first program instruction. After this power-on reset, certain important internal registers will be set to defined states before the program commences. One of these registers is the Program Counter, which will be reset to zero forcing the microcontroller to begin program execution from the lowest Program Memory address.

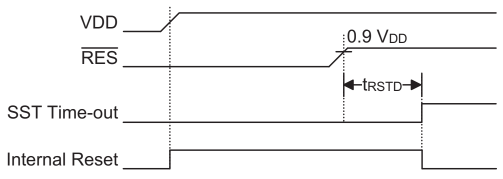
In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup.

Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset is implemented in situations where the power supply voltage falls below a certain threshold.

Reset Functions

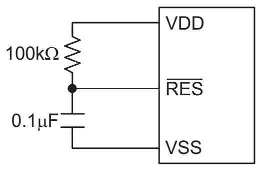
There are five ways in which a microcontroller reset can occur, through events occurring both internally and externally:

- Power-on Reset
  The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.
  Although the microcontroller has an internal RC reset function, if the VDD power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing a proper reset operation. In such cases it is recommended that an external RC network is connected to the $\overline{RES}$ pin, whose additional time delay will ensure that the $\overline{RES}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{RES}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.



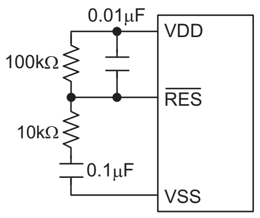
Power-On Reset Timing Chart

For most applications a resistor connected between VDD and the $\overline{RES}$ pin and a capacitor connected between VSS and the $\overline{RES}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{RES}$ pin should be kept as short as possible to minimise any stray noise interference.



Basic Reset Circuit

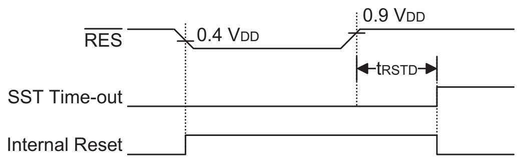
For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



Enhanced Reset Circuit

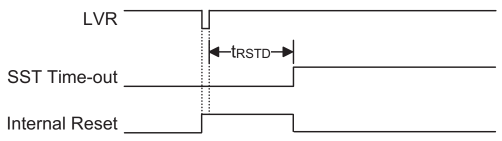
More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

- $\overline{RES}$ Pin Reset
  This type of reset occurs when the microcontroller is already running and the $\overline{RES}$ pin is forcefully pulled low by external hardware such as an external switch. In this case as in the case of other reset, the Program Counter will reset to zero and program execution initiated from this point.



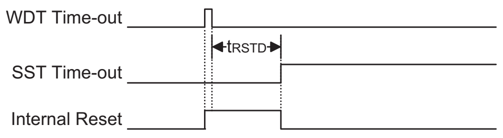
$\overline{RES}$ Reset Timing Chart

- Low Voltage Reset – LVR
  The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function enable/disable, along with a choice of voltages which will activate the LVR function, are selected via configuration options. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the A.C. characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value is selected using a configuration option from a range of supplied fixed voltages.



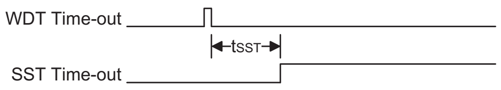
Low Voltage Reset Timing Chart

- Watchdog Time-out Reset during Normal Operation
  The Watchdog time-out Reset during normal operation is the same as a hardware $\overline{\text{RES}}$ pin reset except that the Watchdog time-out flag TO will be set to ″1″.



WDT Time-out Reset during Normal Operation Timing Chart

- Watchdog Time-out Reset during Power Down
  The Watchdog time-out Reset during Power Down is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to ″0″ and the TO flag will be set to ″1″. Refer to the A.C. Characteristics for t_{SST} details.



WDT Time-out Reset during Power Down Timing Chart

Reset Initial Conditions

The different types of reset described affect the reset flags in different ways. These flags, known as PDF and TO are located in the status register and are controlled by various microcontroller operations, such as the Power Down function or Watchdog Timer. The reset flags are shown in the table:

TO	PDF	RESET Conditions
0	0	$\overline{\text{RES}}$ reset during power-on
u	u	$\overline{\text{RES}}$ or LVR reset during normal operation
1	u	WDT time-out reset during normal operation
1	1	WDT time-out reset during Power Down

Note: ″u″ stands for unchanged

The following table indicates the way in which the various components of the microcontroller are affected after a power-on reset occurs.

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT	Clear after reset, WDT begins counting
Timer/Event Counter	Timer Counter will be turned off
Prescaler	The Timer Counter Prescaler will be cleared
Input/Output Ports	I/O ports will be setup as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers.

Register	Reset (Power-on)	WDT time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)*
Program Counter	000H	000H	000H	000H	000H
MP	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	---0 ---0	---0 ---0	---0 ---0	---0 ---0	---u ---u
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00-0 0100	00-0 0100	00-0 0100	00-0 0100	uu-u uuuu
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	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
MISC	1000 0000	1000 0000	1000 0000	1000 0000	uuuu uuuu
OPAC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
OPAC2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
OPAC3	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CMPC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

Note: ″*″ means ″warm reset″
″-″ not implemented
″u″ means ″unchanged″
″x″ means ″unknown″

Oscillator

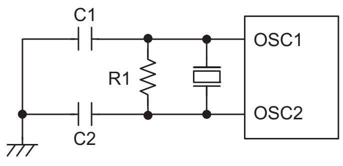
Various oscillator options offer the user a wide range of functions according to their various application requirements. Four types of system clocks can be selected while various clock source options for the Watchdog Timer are provided for maximum flexibility. All oscillator options are selected through the configuration options.

System Clock Configurations

There are two methods of generating the system clock, using an external crystal/ceramic oscillator, an external RC network. One of these two methods must be selected using the configuration options. If the crystal/ceramic oscillator is chosen then another configuration option must be selected if the frequency is below or above 10MHz.

System Crystal/Ceramic Oscillator

After selecting the correct oscillator configuration option, for most crystal oscillator configurations, the simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, without requiring external capacitors. However, for some crystal types and frequencies, to ensure oscillation, it may be necessary to add two small value capacitors, C1 and C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be



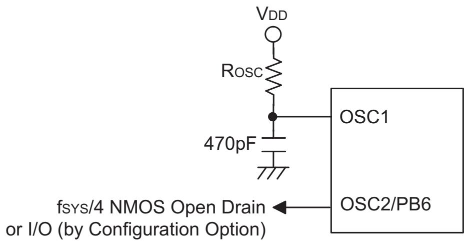
Crystal/Ceramic Oscillator

connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. In most applications, resistor R1 is not required, however for those applications where the LVR function is not used, R1 may be necessary to ensure the oscillator stops running when VDD falls below its operating range.

More information regarding the oscillator is located in Application Note HA0075E on the Holtek website.

External System RC Oscillator

After selecting the correct configuration option, using the external system RC oscillator requires that a resistor, with a value between 24kΩ and 1.5MΩ, is connected between OSC1 and VDD, and a 470pF capacitor is connected to ground. The generated system clock divided by 4 can be provided on pin OSC2/PB6, which can be used for external synchronisation purposes. As the OSC2 pin is also shared with I/O pin PB6, its desired function must first be chosen using a configuration option. Although this is a cost effective oscillator configuration, the oscillation frequency can vary with VDD, temperature and process variations and is therefore not suitable for applications where timing is critical or where accurate oscillator frequencies are required. For the value of the external resistor R_{OSC} refer to the Appendix section for typical RC Oscillator vs. Temperature and VDD characteristics graphics.



RC Oscillator

Note that it is the only microcontroller internal circuitry together with the external resistor, that determine the frequency of the oscillator. The external capacitor shown on the diagram does not influence the frequency of oscillation. The external capacitor is added to improve oscillator stability, especially if the open-drain OSC2 output is utilised in the application circuit.

Watchdog Timer Oscillator

The WDT oscillator is a fully self-contained free running on-chip RC oscillator with a typical period of 65μs at 5V requiring no external components. When the device enters the Power Down Mode, the system clock will stop running but the WDT oscillator continues to free-run and to keep the watchdog active. However, to preserve power in certain applications the WDT oscillator can be disabled via a configuration option.

Power Down Mode and Wake-up

Power Down Mode

All of the Holtek microcontrollers have the ability to enter a Power Down Mode, also sometimes known as the HALT Mode or Sleep Mode. When the device enters this mode, the normal operating current, will be reduced to an extremely low standby current level. This occurs because when the device enters the Power Down Mode, the system oscillator is stopped which reduces the power consumption to extremely low levels, however, as the device maintains its present internal condition, it can be woken up at a later stage and continue running, without requiring a full reset. This feature is extremely important in application areas where the MCU must have its power supply constantly maintained to keep the device in a known condition but where the power supply capacity is limited such as in battery applications.

Entering the Power Down Mode

There is only one way for the device to enter the Power Down Mode and that is to execute the "HALT" instruction in the application program. When this instruction is executed, the following will occur:

- The system oscillator will stop running and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the WDT oscillator. The WDT will stop if its clock source originates from the system clock.
- The I/O ports will maintain their present condition.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Standby Current Considerations

As the main reason for entering the Power Down Mode is to keep the current consumption of the MCU to as low a value as possible, perhaps only in the order of several micro-amps, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised.

Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs.

If the configuration options have enabled the Watchdog Timer internal oscillator then this will continue to run when in the Power Down Mode and will thus consume some power. For power sensitive applications it may be therefore preferable to use the system clock source for the Watchdog Timer.

Wake-up

After the system enters the Power Down Mode, it can be woken up from one of various sources listed as follows:

- An external reset
- An external falling edge on Port A
- A system interrupt
- A WDT overflow

If the system is woken up by an external reset, the device will experience a full system reset, however, if the device is woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flags. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the "HALT" instruction. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

Pins Port A can be setup via an individual configuration option to permit a negative transition on the pin to wake-up the system. When a Port A pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction.

If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set to "1" before entering the Power Down Mode, the wake-up function of the related interrupt will be disabled.

No matter what the source of the wake-up event is, once a wake-up situation occurs, a time period equal to 1024 system clock periods will be required before normal system operation resumes. However, if the wake-up has originated due to an interrupt, the actual interrupt subroutine execution will be delayed by an additional one or more cycles. If the wake-up results in the execution of the next instruction following the "HALT" instruction, this will be executed immediately after the 1024 system clock period delay has ended.

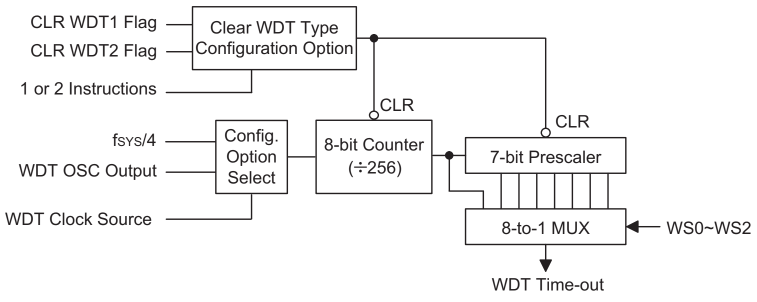
Watchdog Timer

The Watchdog Timer, also known as the WDT, is provided to inhibit program malfunctions caused by the program jumping to unknown locations due to certain uncontrollable external events such as electrical noise. It operates by providing a device reset when the Watchdog Timer counter overflows. Note that if the Watchdog Timer function is not enabled, then any instructions related to the Watchdog Timer will result in no operation.

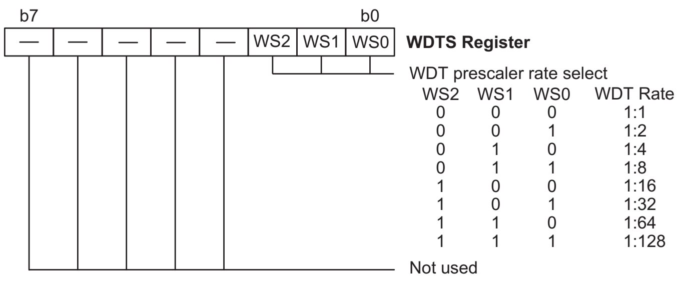
Setting up the various Watchdog Timer options are controlled via the configuration options.

The Watchdog Timer clock can emanate from two different sources, selected by configuration option. These are its own fully integrated dedicated internal oscillator or $f_{SYS}/4$. The Watchdog Timer dedicated internal clock source is an internal oscillator which has an approximate period of 65μs at a supply voltage of 5V. However, it should be noted that this specified internal clock period can vary with VDD, temperature and process variations. The other Watchdog Timer clock source options are the f_{SYS}. It is important to note that when the system enters the Power Down Mode the instruction clock is stopped, therefore if the configuration options have selected $f_{SYS}/4$ as the Watchdog Timer clock source, the Watchdog Timer will cease to function. For systems that operate in noisy environments, using the internal Watchdog Timer internal oscillator as the clock source is therefore the recommended choice. No matter which clock source is selected, it is further divided by 256 via an internal 8-bit counter and then by a 7-bit prescaler to give longer time-out periods. The division ratio of the prescaler is determined by bits 0, 1 and 2 of the WDTS register, known as WS0, WS1 and WS2. If the Watchdog Timer internal clock source is selected and with the WS0, WS1 and WS2 bits of the WDTS register all set high, the prescaler division ratio will be 1:128, which will give a maximum time-out period of about 2.1s.

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the Power Down Mode, when a Watchdog Timer time-out occurs, the device will be woken up, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is an external hardware reset, which means a low level on the external reset pin, the second is using the Clear Watchdog Timer software instructions and the third is when a HALT instruction is executed. There are two methods of using software instructions to clear the Watchdog Timer, one of which must be chosen by configuration option. The first option is to use the single ″CLR WDT″ instruction while the second is to use the two commands ″CLR WDT1″ and ″CLR WDT2″. For the first option, a simple execution of ″CLR WDT″ will clear the Watchdog Timer while for the second option, both ″CLR WDT1″ and ″CLR WDT2″ must both be executed to successfully clear the Watchdog Timer. Note that for this second option, if ″CLR WDT1″ is used to clear the Watchdog Timer, successive executions of this instruction will have no effect, only the execution of a ″CLR WDT2″ instruction will clear the Watchdog Timer. Similarly after the ″CLR WDT2″ instruction has been executed, only a successive ″CLR WDT1″ instruction can clear the Watchdog Timer.



Watchdog Timer



WS2	WS1	WS0	WDT Rate
0	0	0	1:1
0	0	1	1:2
0	1	0	1:4
0	1	1	1:8
1	0	0	1:16
1	0	1	1:32
1	1	0	1:64
1	1	1	1:128

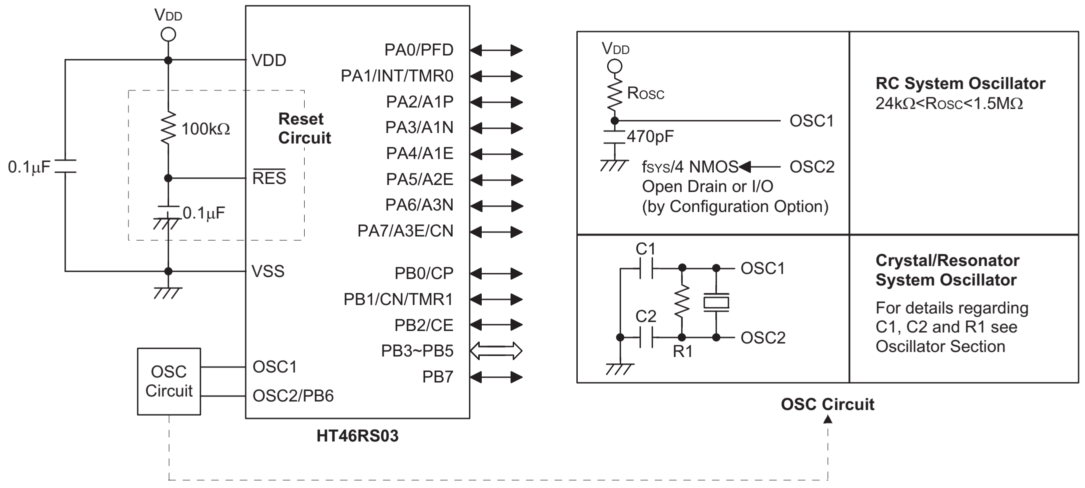
Watchdog Timer Register

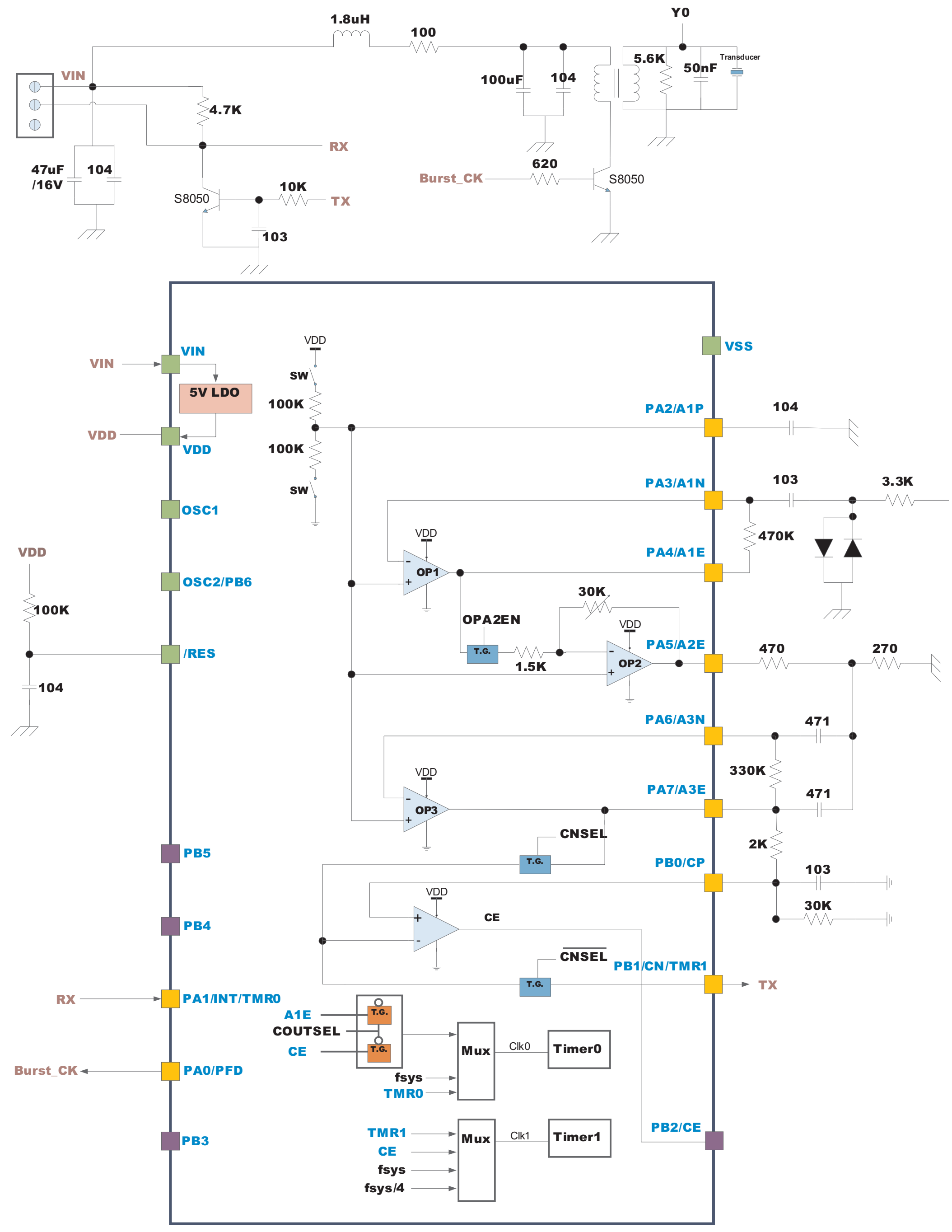
Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the OTP Program Memory device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later by the application software. All options must be defined for proper system function, the details of which are shown in the table.

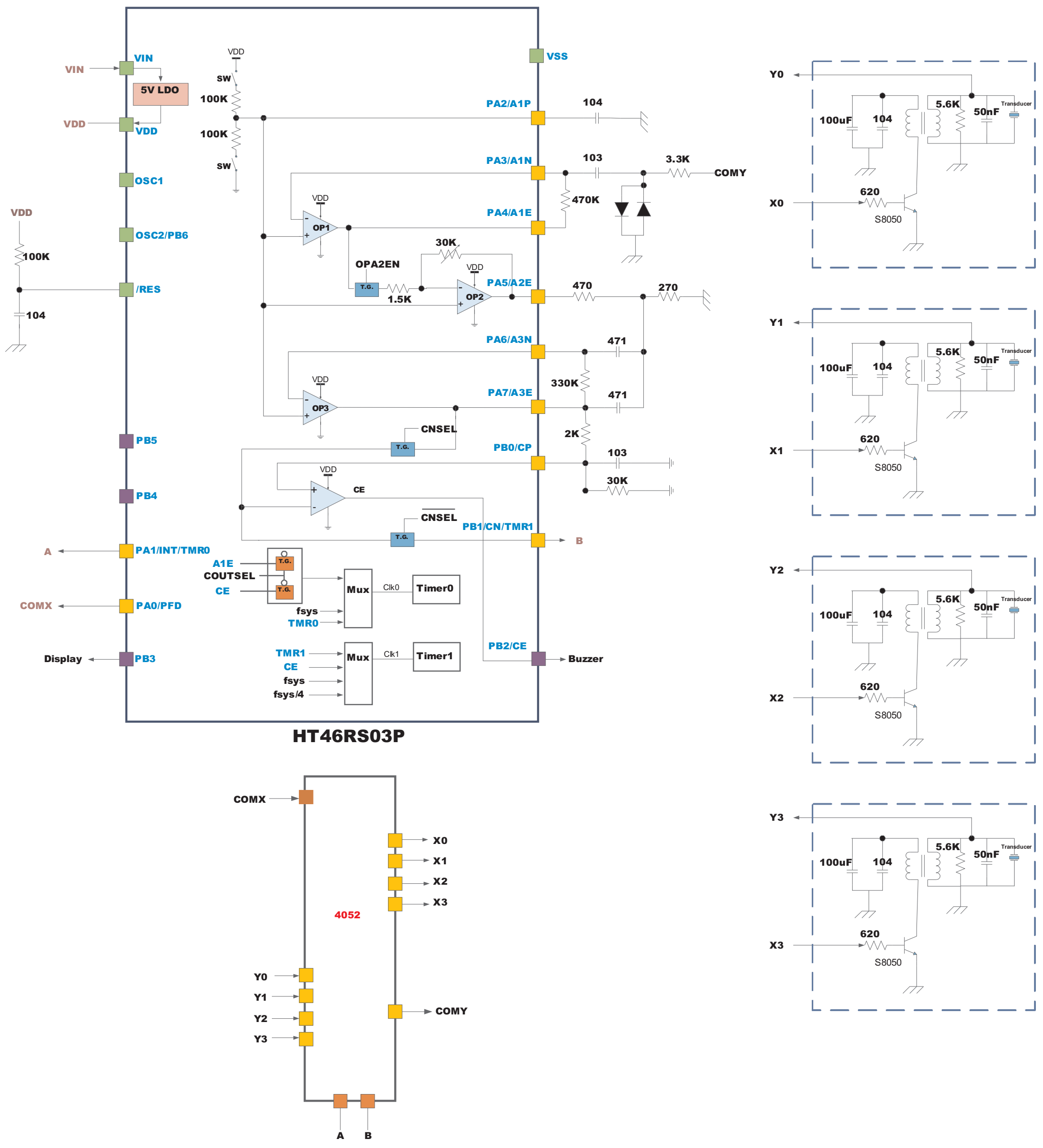
No.	Options
I/O Options	
1	PA7~PA0 wake-up: enable or disable (by bit)
2	PA pull-high enable or disable (by bit)
3	PB pull-high enable or disable (by bit)
4	PA0: normal I/O or PFD output
Oscillator Options	
5	OSC type selection: RC + OSC2 (f_{SYS}/4), RC+I/O (PB6), or XTAL
6	XTAL high or low: (> 10MHz) or (< 10MHz)
PFD Options	
7	PFD: from Timer0 or Timer1
Watchdog Options	
8	Watchdog Timer: enable or disable
9	Watchdog Timer clock source: WDT internal oscillator or f_{SYS}/4
10	CLRWDT instructions: 1 or 2 instructions
OPAs Bias Voltage Options	
11	Bias voltage control: External or internal
LVR Options	
12	LVR function: enable or disable
13	LVR voltage: 2.1V, 3.15V or 4.2V
Lock Options	
14	Lock All
15	Partial Lock

Application Circuits




1.8uH
100
Y0
VIN
4.7K
100uF
104
5.6K
50nF
Transducer
RX
47uF /16V
104
620
Burst_CK
S8050
S8050
10K
TX
103
VIN
VDD
5V LDO
SW
100K
100K
SW
VSS
PA2/A1P
104
VDD
VDD
OSC1
OSC2/PB6
PA3/A1N
103
3.3K
Y0
470K
OP1
PA4/A1E
100K
/RES
104
OPA2EN
30K
T.G.
1.5K
OP2
PA5/A2E
470
270
PA6/A3N
471
330K
OP3
PA7/A3E
471
CNSEL
2K
PB5
PB0/CP
103
30K
PB4
CE
CNSEL
PB1/CN/TMR1
TX
RX
PA1/INT/TMR0
A1E
COUTSEL
CE
Mux
Clk0
Timer0
Burst_CK
PA0/PFD
fsys
TMR0
PB3
TMR1
CE
fsys
fsys/4
Mux
Clk1
Timer1
PB2/CE

HT46RS03P


VIN
VDD
VDD
100K
104
A
COMX
Display
VIN
5V LDO
VDD
OSC1
OSC2/PB6
/RES
PB5
PB4
PA1/INT/TMR0
PA0/PFD
PB3
VDD
SW
100K
100K
SW
OP1
OPA2EN
30K
1.5K
OP2
OP3
CNSEL
CE
CNSEL
A1E
COUTSEL
CE
fsys
TMR0
Mux
Clk0
Timer0
TMR1
CE
fsys
fsys/4
Mux
Clk1
Timer1
VSS
PA2/A1P
PA3/A1N
PA4/A1E
PA5/A2E
PA6/A3N
PA7/A3E
PB0/CP
PB1/CN/TMR1
PB2/CE
104
103
3.3K
COMY
470K
470
270
471
330K
471
2K
103
30K
B
Buzzer
HT46RS03P
COMX
X0
X1
X2
X3
4052
Y0
Y1
Y2
Y3
COMY
A
B
Y0
X0
Y1
X1
Y2
X2
Y3
X3
100uF
104
5.6K
50nF
Transducer
620
S8050

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	1[Note]	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1[Note]	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1[Note]	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1[Note]	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1[Note]	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	1[Note]	Z
ORM A,[m]	Logical OR ACC to Data Memory	1[Note]	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1[Note]	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1[Note]	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	1[Note]	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1[Note]	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	1$^{\text{Note}}$	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1$^{\text{Note}}$	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1$^{\text{Note}}$	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1$^{\text{Note}}$	C
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1$^{\text{Note}}$	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1$^{\text{Note}}$	None
SET [m].i	Set bit of Data Memory	1$^{\text{Note}}$	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1$^{\text{Note}}$	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1$^{\text{note}}$	None
SZ [m].i	Skip if bit i of Data Memory is zero	1$^{\text{Note}}$	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1$^{\text{Note}}$	None
SIZ [m]	Skip if increment Data Memory is zero	1$^{\text{Note}}$	None
SDZ [m]	Skip if decrement Data Memory is zero	1$^{\text{Note}}$	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1$^{\text{Note}}$	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1$^{\text{Note}}$	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	2$^{\text{Note}}$	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2$^{\text{Note}}$	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1$^{\text{Note}}$	None
SET [m]	Set Data Memory	1$^{\text{Note}}$	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	1$^{\text{Note}}$	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

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

Instruction Definition

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

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None
CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF

CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	$[m] \leftarrow \overline{[m]}$
Affected flag(s)	Z
CPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow \overline{[m]}$
Affected flag(s)	Z
DAA [m]	Decimal-Adjust ACC for addition with result in Data Memory
Description	Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.
Operation	[m] ← 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 $\leftarrow$ ACC $-$ [m] $- \overline{C}$
Affected flag(s)	OV, Z, AC, C

SBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] $\leftarrow$ ACC $-$ [m] $- \overline{C}$
Affected flag(s)	OV, Z, AC, C

SDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] $\leftarrow$ [m] $-$ 1 Skip if [m] = 0
Affected flag(s)	None

SDZA [m]	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	ACC $\leftarrow$ [m] $-$ 1 Skip if ACC = 0
Affected flag(s)	None

SET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	[m] $\leftarrow$ FFH
Affected flag(s)	None

SET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	[m].i $\leftarrow$ 1
Affected flag(s)	None

SIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] + 1 Skip if [m] = 0
Affected flag(s)	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] + 1 Skip if ACC = 0
Affected flag(s)	None
SNZ [m].i	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 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

16-pin DIP (300mil) Outline Dimensions

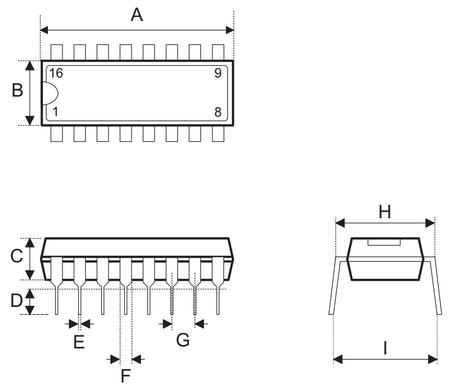


Fig1. Full Lead Packages

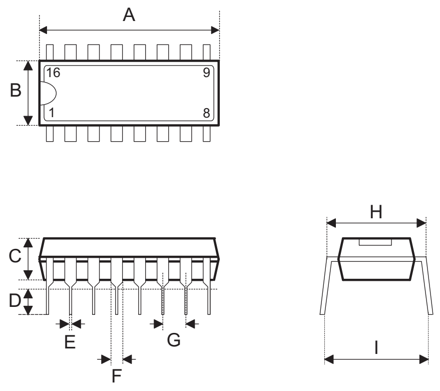


Fig2. 1/2 Lead Packages

• MS-001d (see fig1)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	780	—	880
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

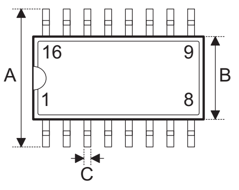
• MS-001d (see fig2)

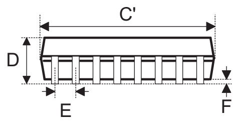
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	735	—	775
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

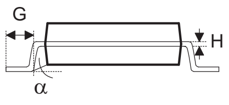
• MO-095a (see fig2)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	745	—	785
B	275	—	295
C	120	—	150
D	110	—	150
E	14	—	22
F	45	—	60
G	—	100	—
H	300	—	325
I	—	—	430

16-pin NSOP (150mil) Outline Dimensions



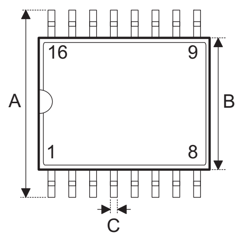


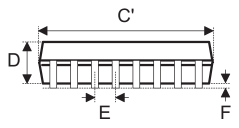


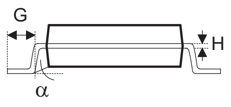
• MS-012

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	228	—	244
B	150	—	157
C	12	—	20
C′	386	—	394
D	—	—	69
E	—	50	—
F	4	—	10
G	16	—	50
H	7	—	10
α	0°	—	8°

16-pin SSOP (150mil) Outline Dimensions







Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	228	—	244
B	150	—	157
C	8	—	12
C′	189	—	197
D	54	—	60
E	—	25	—
F	4	—	10
G	22	—	28
H	7	—	10
α	0°	—	8°

20-pin DIP (300mil) Outline Dimensions

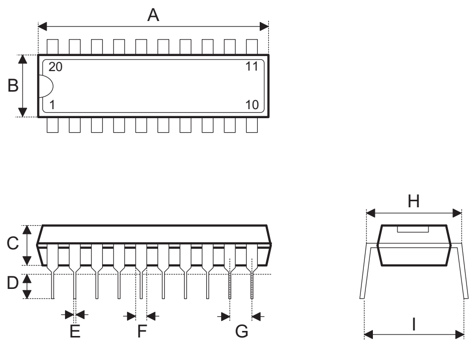


Fig1. Full Lead Packages

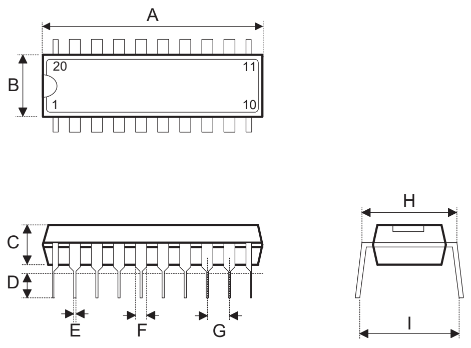


Fig2. 1/2 Lead Packages

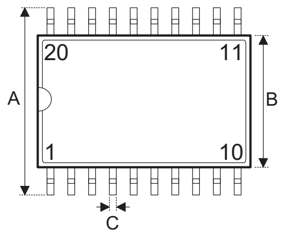
- MS-001d (see fig1)

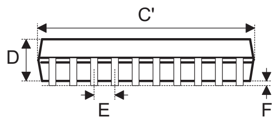
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	980	—	1060
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

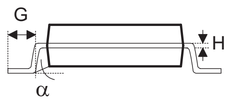
- MO-095a (see fig2)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	945	—	985
B	275	—	295
C	120	—	150
D	110	—	150
E	14	—	22
F	45	—	60
G	—	100	—
H	300	—	325
I	—	—	430

20-pin SSOP (150mil) Outline Dimensions

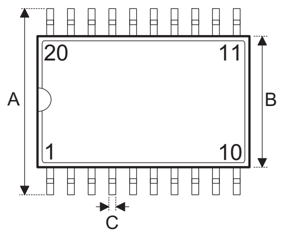


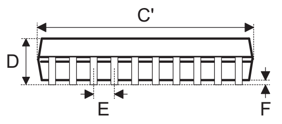


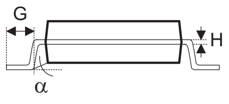


Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	228	—	244
B	150	—	158
C	8	—	12
C′	335	—	347
D	49	—	65
E	—	25	—
F	4	—	10
G	15	—	50
H	7	—	10
α	0°	—	8°

20-pin SOP (300mil) Outline Dimensions



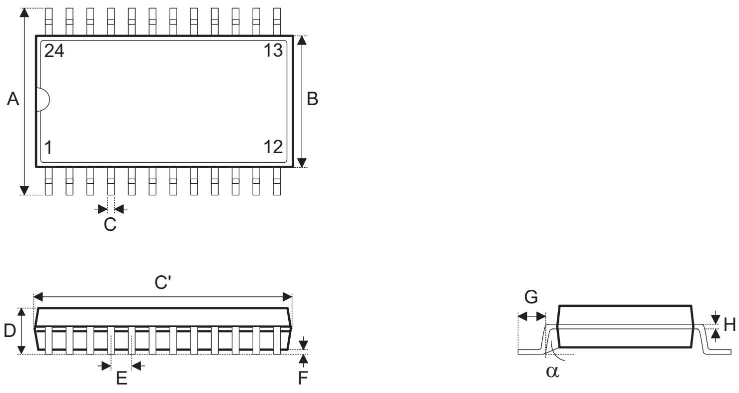




• MS-013

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	393	—	419
B	256	—	300
C	12	—	20
C′	496	—	512
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

24-pin SOP (300mil) Outline Dimensions

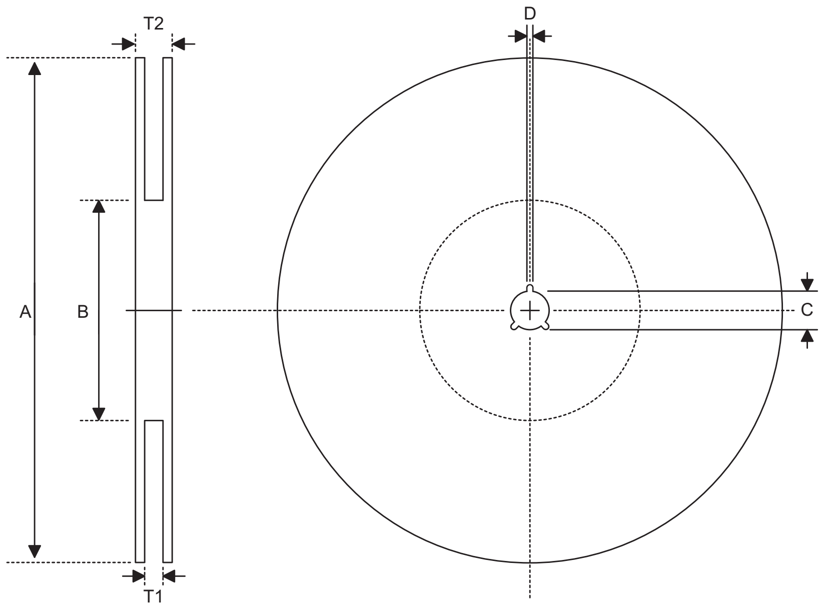


• MS-013

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	393	—	419
B	256	—	300
C	12	—	20
C′	598	—	613
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

Product Tape and Reel Specifications

Reel Dimensions



SOP 16N (150mil), SSOP 20S (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	$16.8^{+0.3/-0.2}$
T2	Reel Thickness	22.2±0.2

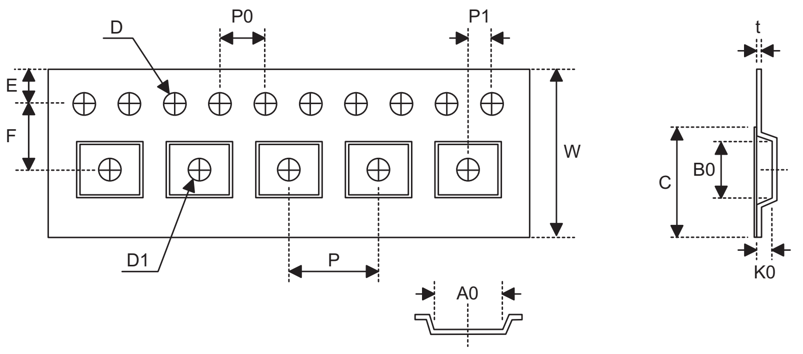
SSOP 16S

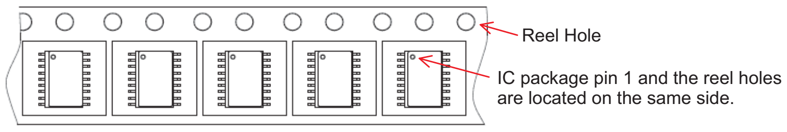
Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	$12.8^{+0.3/-0.2}$
T2	Reel Thickness	18.2±0.2

SOP 20W, SOP 24W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	$24.8^{+0.3/-0.2}$
T2	Reel Thickness	30.2±0.2

Carrier Tape Dimensions




Reel Hole
IC package pin 1 and the reel holes are located on the same side.

SOP 16N (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0±0.3
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	$1.55^{+0.1/-0.0}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.0}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	10.3±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

SSOP 16S

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	$12.0^{+0.3/-0.1}$
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	5.5±0.1
D	Perforation Diameter	1.55±0.10
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.4±0.1
B0	Cavity Width	5.2±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	9.3±0.1

SSOP 20S (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	$16.0^{+0.3/-0.1}$
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	$1.5^{+0.1/-0.0}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	9.0±0.1
K0	Cavity Depth	2.3±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

SOP 20W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	$24.0^{+0.3/-0.1}$
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	$1.5^{+0.1/-0.0}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.8±0.1
B0	Cavity Width	13.3±0.1
K0	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	21.3±0.1

SOP 24W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0±0.3
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	$1.55^{+0.10/-0.00}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.9±0.1
B0	Cavity Width	15.9±0.1
K0	Cavity Depth	3.1±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	21.3±0.1

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)
G Room, 3 Floor, No.1 Building, No.2016 Yi-Shan Road, Minhang District, Shanghai, China 201103
Tel: 86-21-5422-4590
Fax: 86-21-5422-4705
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5F, Unit A, Productivity Building, Gaoxin M 2nd, Middle Zone Of High-Tech Industrial Park, ShenZhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

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

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

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