HT46R01/HT46R02/HT46R03

10-Pin MSOP A/D Type 8-Bit OTP MCU

Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0003E Communicating between the HT48 & HT46 Series MCUs and the HT93LC46 EEPROM
  - HA0016E Writing and Reading to the HT24 EEPROM with the HT48 MCU Series
  - HA0018E Controlling the HT1621 LCD Controller with the HT48 MCU Series
  - HA0049E Read and Write Control of the HT1380
  - 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}= 12MHz: 4.5V~5.5V
- 7 bidirectional I/O lines and 1 input line
- External interrupt input shared with an I/O line
- One or two 8-bit programmable Timer/Event Counters with overflow interrupt and 7-stage prescaler
- External crystal system oscillator
- Fully integrated internal RC oscillator available with three frequencies: 4MHz, 8MHz or 12MHz
- Watchdog Timer function
- PFD for audio frequency generation
- Power down and wake-up functions to reduce power consumption
- Up to 0.33μs instruction cycle with 12MHz system clock at V_{DD}= 5V
- 4, 6 or 8-level subroutine nesting
- 4-channel 8, 9 or 12-bit resolution A/D converter
- 8-bit PWM output shared with I/O line
- Bit manipulation instruction
- Table read instructions
- 63 powerful instructions
- All instructions executed in one or two machine cycles
- Low voltage reset function
- 10-pin ultra-small MSOP package

General Description

The HT46R01/HT46R02/HT46R03 are 8-bit high performance, RISC architecture microcontroller devices specifically designed for a wide range of applications. In addition to providing the usual microcontroller I/O control functions, the fully integrated A/D converter allows the interfacing and processing of external analog signals such as those from sensors. The PWM function also allows the driving of external analog signals for applications such as motor driving while the PFD function provides a means of generating a fixed frequency reference. The usual Holtek microcontroller features of low power consumption, I/O flexibility, timer functions, oscillator options, power down and wake-up functions, watchdog timer and low voltage reset, combine to provide devices with a huge range of functional options while still maintaining cost effectiveness. Their fully integrated system oscillator, with three frequency selections, and being supplied in an extremely small outline MSOP 10-pin package opens up a huge range of new application possibilities for these devices, some of which include may include industrial control, consumer products, household appliances subsystem controllers, etc.

Selection Table

Part No.	VDD	Program Memory	Data Memory	I/O	Timer	A/D	PWM	PFD	Stack	Package Types
HT46R01	2.2V~5.5V	1K×14	64×8	8	8-bit×1	8-bit×4	8-bit×1	√	4	10MSOP
HT46R02	2.2V~5.5V	2K×14	96×8	8	8-bit×2	9-bit×4	8-bit×1	√	6	10MSOP
HT46R03	2.2V~5.5V	4K×15	160×8	8	8-bit×2	12-bit×4	8-bit×1	√	8	10MSOP

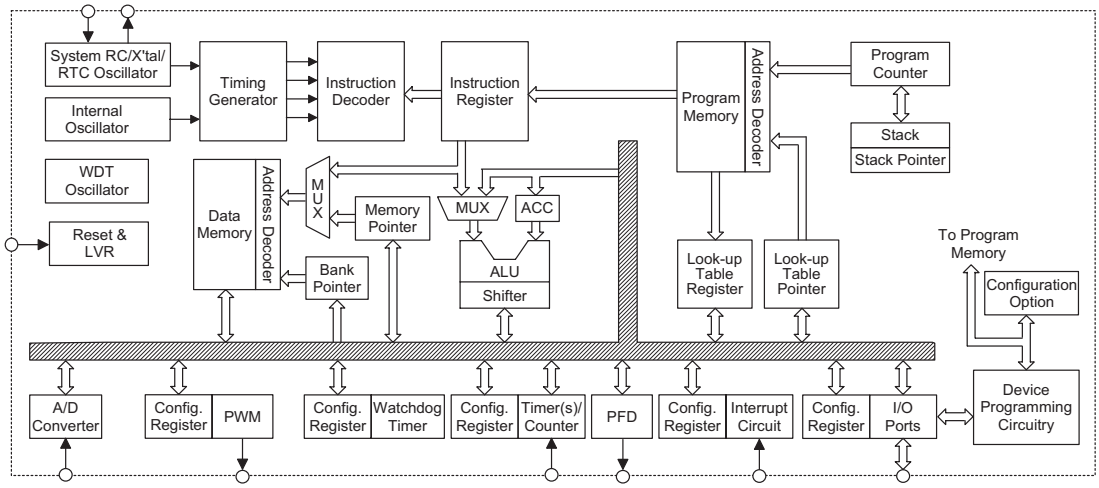
Device Markings

A suffix with the number 1, 2 or 3 will be added to each of the part numbers and printed on the package. The suffix number denotes the fixed oscillation frequency of the internal RC oscillator as shown.

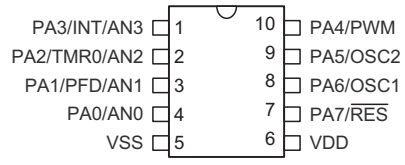
Part No.	Device Marking	Internal RC Oscillator Frequency
HT46R01	46R01-1 46R01-2 46R01-3	4MHz 8MHz 12MHz
HT46R02	46R02-1 46R02-2 46R02-3	4MHz 8MHz 12MHz
HT46R03	46R03-1 46R03-2 46R03-3	4MHz 8MHz 12MHz

Block Diagram

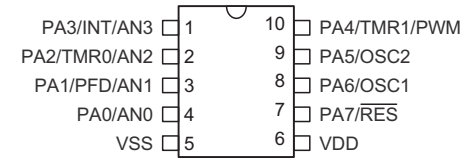
The following block diagram illustrates the main functional blocks.



Pin Assignment



HT46R01
— 10 MSOP-A



HT46R02/HT46R03
— 10 MSOP-A

Pin Description

Pin Name	I/O	Configuration Options	Description
PA0/AN0	I/O	—	Bidirectional single line I/O. The pin can be setup as a wake-up input using the wake-up register. Software instructions determine if the pin is a CMOS output or a Schmitt trigger input. A pull-high resistor can be connected using the pull-high register. PA0 is pin-shared with the AN0 input pin. The A/D input function is selected via software instructions. If selected as an A/D input, the I/O function and pull-high resistor functions are disabled automatically.
PA1/PFD/AN1	I/O	—	Bidirectional single line I/O. The pin can be setup as a wake-up input using the wake-up register. Software instructions determine if the pin is a CMOS output or a Schmitt trigger input. A pull-high resistor can be connected using the pull-high register. PA1 is pin-shared with the PFD output and the AN1 input pin. The A/D input function is selected via software instructions. If selected as an A/D input, the I/O function, PFD output and pull-high resistor functions are disabled automatically. If the A/D function is not selected the PFD output or I/O function selection is chosen via a bit in the CRTL0 register.
PA2/TMR0/AN2	I/O	—	Bidirectional single line I/O. The pin can be setup as a wake-up input using the wake-up register. Software instructions determine if the pin is a CMOS output or a Schmitt trigger input. A pull-high resistor can be connected using the pull-high register. PA2 is pin-shared with the TMR0 input and the AN2 input pin. The A/D input function is selected via software instructions. If selected as an A/D input, the I/O function, timer input and pull-high resistor functions are disabled automatically.
PA3/INT/AN3	I/O	—	Bidirectional single line I/O. The pin can be setup as a wake-up input using the wake-up register. Software instructions determine if the pin is a CMOS output or a Schmitt trigger input. A pull-high resistor can be connected using the pull-high register. PA3 is pin-shared with the INT input and the AN3 input pin. The A/D input function is selected via software instructions. If selected as an A/D input, the I/O function, external interrupt input and pull-high resistor functions are disabled automatically.
PA4/TMR1/PWM	I/O	—	Bidirectional single line I/O. The pin can be setup as a wake-up input using the wake-up register. Software instructions determine if the pin is a CMOS output or a Schmitt trigger input. A pull-high resistor can be connected using the pull-high register. On the HT46R01 devices, PA4 is pin-shared with the PWM output pin, while on the HT46R02 and HT46R03 devices, the pin is shared with both the TMR1 input and the PWM output pin. The PWM output or I/O function selection is chosen via a bit in CTRL0 register.
PA6/OSC1 PA5/OSC2	I/O	RC, Crystal, RTC or I/O	Bidirectional 2-line I/O. The pins can be setup as wake-up inputs using the wake-up register. Software instructions determine if the pins are CMOS outputs or Schmitt trigger inputs. Pull-high resistors can be connected using the pull-high register. Configuration options determine if the pins are to be used as oscillator pins or I/O pins. Configuration options also determine which oscillator mode is selected. The four oscillator modes are: 1. Internal RC OSC: both pins configured as I/Os 2. External crystal OSC: both pins configured as OSC1/OSC2 3. Internal RC + RTC OSC: both pins configured as OSC2, OSC1. 4. External RC OSC+PA5: PA6 configured as OSC1 pin, PA5 configured as I/O If the internal RC OSC is selected, the frequency will be fixed at either 4MHz, 8MHz or 12MHz, dependent upon which device is chosen.
PA7/$\overline{\text{RES}}$	I	PA7 or $\overline{\text{RES}}$	Active low Schmitt trigger reset input or PA7 input. A configuration option determines which function is selected.
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground

Note: 1. Each pin on PA except PA7 can be programmed through a configuration option to have a wake-up function.
2. Each pin on PA except PA7 can be selected to have a pull-high resistor.

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}=12MHz	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}=12MHz	—	6	12	mA
I_{DD4}	Operating Current (Internal RC+RTC OSC, Normal Mode)	3V	No load, f_{SYS}=4MHz	—	1	2	mA
		5V		—	2.5	5	mA
I_{DD5}	Operating Current (Internal RC+RTC OSC, Normal Mode)	3V	No load, f_{SYS}=8MHz	—	2	4	mA
		5V		—	4	8	mA
I_{DD6}	Operating Current (Internal RC+RTC OSC, Normal Mode)	5V	No load, f_{SYS}=12MHz	—	6	12	mA
I_{DD7}	Operating Current (Internal RC+RTC OSC, Slow Mode)	3V	No load, f_{SYS}=32768Hz	—	20	30	μA
		5V		—	40	60	μA
I_{STB1}	Standby Current (WDT Enabled, RTC Off)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	Standby Current (WDT Disabled, RTC Off)	3V	No load, system HALT	—	—	1	μA
		5V		—	—	2	μA
I_{STB3}	Standby Current (WDT Disabled, RTC On)	3V	No load, system HALT QOSC=1	—	—	5	μA
		5V		—	—	10	μA
V_{IL1}	Input Low Voltage for PA0~PA6, TMR0, TMR1 and INT	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage forPA0~PA6, TMR0, TMR1 and INT	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage (PA7/$\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage (PA7/$\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
V_{LVR1}	Low Voltage Reset 1	—	Configuration option: 4.2V	3.98	4.2	4.42	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: 2.1V	1.98	2.1	2.22	V
I_{OL}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V		10	20	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
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_{AD}	A/D Input Voltage	—	—	0	—	V_{DD}	V
E_{AD}	ADC Conversion Error	—	8, 9 bit ADC	—	±0.5	±1	LSB
D_{NL}	ADC Differential Non-linear	—	12 bit ADC, t_{AD}=1μs	—	—	±2	LSB
I_{NL}	ADC Integral Non-linear	—	12 bit ADC, t_{AD}=1μs	—	±2.5	±4	LSB
I_{ADC}	ADC Power Consumption	3V	No load, t_{AD}=1μs	—	0.5	1	mA
		5V		—	1.5	3	mA

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS1}	System Clock (Crystal OSC, RC OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
		—	4.5V~5.5V	400	—	12000	kHz
f_{SYS2}	System Clock (Internal RC OSC) (±5%)	4.5V~5.5V	12MHz, Ta=25°C	11400	12000	12600	kHz
		3.3V~5.5V	8MHz, Ta=25°C	7600	8000	8400	kHz
		2.7V~5.5V	4MHz, Ta=25°C	3800	4000	4200	kHz
f_{SYS3}	System Clock (32768 Crystal)	—	—	—	32768	—	Hz
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	—	12000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	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
V_{POR}	VDD Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
R_{POR}	VDD Rise Time to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{AD}	ADC Clock Period	—	—	1	—	—	μs
t_{ADC}	ADC Conversion Time	—	8 bit ADC	—	64	—	t_{AD}
			9 bit ADC	—	76	—	t_{AD}
			12 bit ADC	—	80	—	t_{AD}
t_{ADS}	ADC Sampling Time	—	—	—	32	—	t_{AD}

Note: t_{SYS}=1/f_{SYS1}, 1/f_{SYS2} or 1/f_{SYS3}

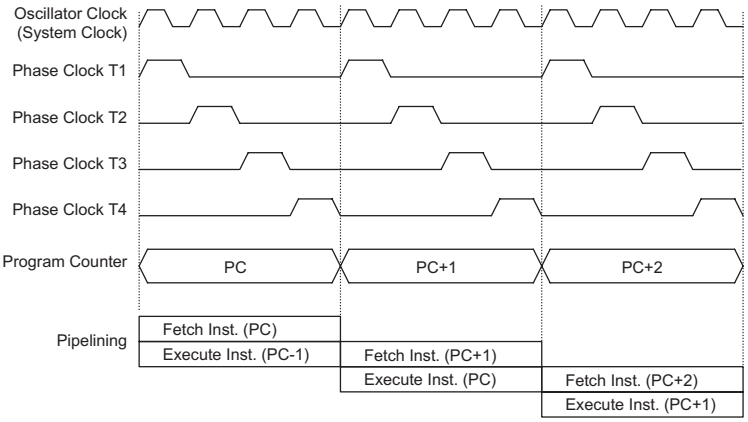
System Architecture

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 and A/D control system with maximum reliability and flexibility. This makes these devices suitable for low-cost, high-volume production for controller applications requiring from 1K up to 4K words of Program Memory and 64 to 128 bytes of Data Memory storage.

Clocking and Pipelining

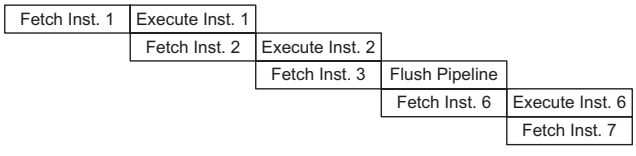
The main system clock, derived from either a Crystal/Resonator or RC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry 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  DELAY:  NOP
```



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 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.

Mode	Program Counter Bits											
	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0
External Interrupt	0	0	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 1 Overflow (HT46R02/HT46R03)	0	0	0	0	0	0	0	0	1	1	0	0
A/D Converter Interrupt (HT46R01)	0	0	0	0	0	0	0	0	1	1	0	0
A/D Converter Interrupt (HT46R02/HT46R03)	0	0	0	0	0	0	0	1	0	0	0	0
Skip	Program Counter + 2											
Loading PCL	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

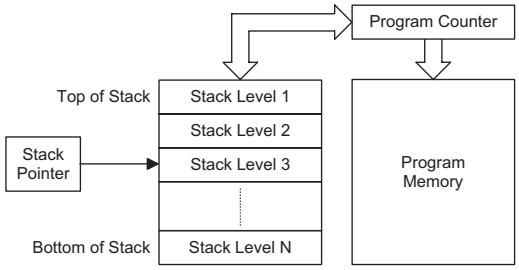
Program Counter

Note: PC11~PC8: Current Program Counter bits
@7~@0: PCL bits
#11~#0: Instruction code address bits
S11~S0: Stack register bits
For the HT46R01 devices, the Program Counter is 10 bits wide, i.e. from b9~b0.
For the HT46R02 devices, the Program Counter is 11 bits wide, i.e. from b10~b0.
For the HT46R03 devices, the Program Counter is 12 bits wide, i.e. from b11~b0.
Timer/Event Counter 1 does not exist on the HT46R01.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack can have either 4, 6 or 8 levels depending upon which device is selected and is neither part of the data nor part of the program space, and is neither readable nor writable. The activated level is indexed by the Stack Pointer, SP, and is 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 instruction 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.



Note: For the HT46R01, N=4, i.e. 4 levels of stack available.
For the HT46R02, N=6, i.e. 6 levels of stack available.
For the HT46R03, N=8, i.e. 8 levels of stack available.

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 calculation 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.

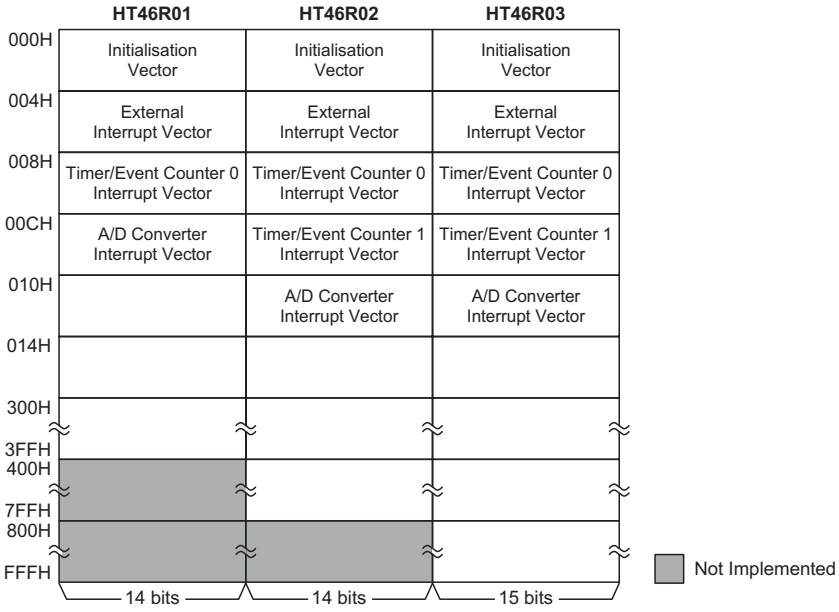
Structure

The Program Memory has a capacity of 1K by 14, 2K by 14 or 4K by 15 bits depending upon which device is selected. 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.

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 CTRL1 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.



Program Memory Structure

- Location 00CH
  For the HT46R01 devices, this internal vector is used by the A/D converter. When an A/D conversion cycle is complete, the program will jump to this location and begin execution if the A/D interrupt is enabled and the stack is not full. For the HT46R02 and HT46R03 devices, this vector is used by the Timer/Event Counter 1. If a Timer/Event Counter 1overflow occurs, the program will jump to this location and begin execution if the Timer/Event counter 1 interrupt is enabled and the stack is not full.
- Location 010H
  This internal vector is only used by the A/D interrupt converter in the HT46R02 and HT46R03 devices. When an A/D conversion cycle is complete, the program will jump to this location and begin execution if the A/D interrupt is enabled and the stack is not full.

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:

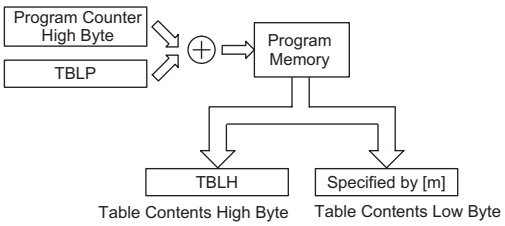


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the HT46R02 devices. 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 of the HT46R02 microcontrollers. 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 (for HT46R02)

dc          00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
            :
            :
```

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.

Instruction	Table Location Bits											
	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: PC11~PC8: Current Program Counter bits

@7~@0: Table Pointer TBLP bits

For the HT46R03 devices, the Table address location is 12 bits, i.e. from b11~b0.

For the HT46R02 devices, the Table address location is 11 bits, i.e. from b10~b0.

For the HT46R01 devices, the Table address location is 10 bits, i.e. from b9~b0.

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 but the length of each memory section is dictated by the type of microcontroller chosen. The start address of the Data Memory for all devices 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

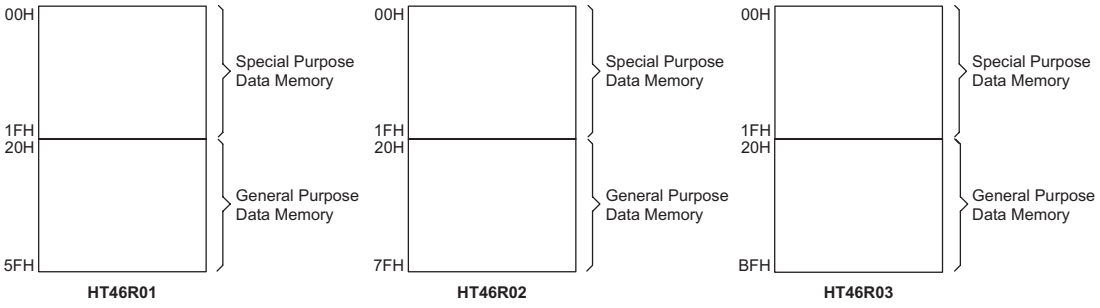
This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and writable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section.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

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 A/D converter 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.

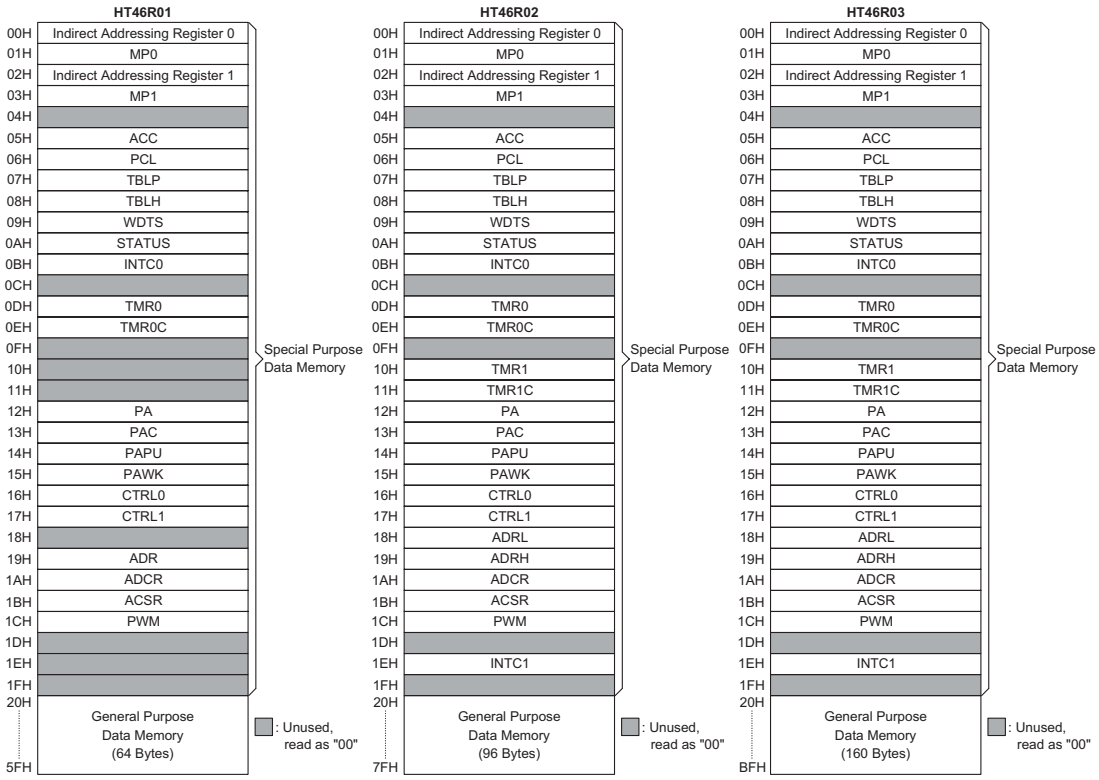
Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointer, MP0 or MP1. Acting as a pair, IAR0 with MP0 and IAR1 with MP1 can together access data from the Data Memory. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of ″00H″ and writing to the registers indirectly will result in no operation.



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

Memory Pointers – MP0, MP1

For all devices, two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, is the address specified by the related Memory Pointer. For the HT46R01 and HT46R02 devices, bit 7 of the Memory Pointers is not required to address the full memory space. It must be noted that when bit 7 of the Memory Pointers for these devices is read, a value of "1" will be returned. The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.

```
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.

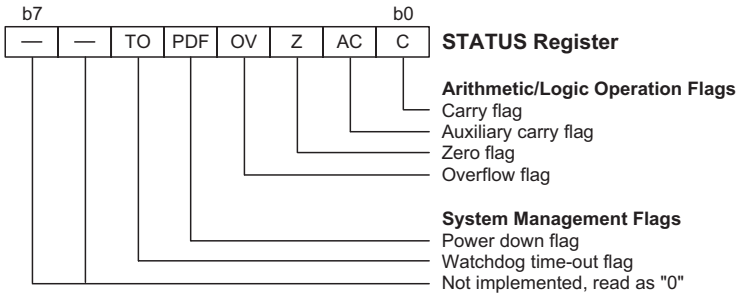
Status Register – STATUS

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.
- **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.



Status Register

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 both external, and internal timer and A/D 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. Note that the INTC1 register does not exist on the HT46R01 devices.

Timer/Event Counter Registers

Depending upon which device is selected, all devices contain one or two integrated 8-bit Timer/Event Counters. For the HT46R01 devices, which have a single 8-bit Timer/Event Counter, an associated register known as TMR0 is the location where the timer's 8-bit value is located. An associated control register, known as TMR0C, contains the setup information for this timer. As the HT46R02 and HT46R03 devices contain two 8-bit Timer/Event Counters and additional register pair exists, with the names TMR1 and TMR1C.

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.

Pulse Width Modulation Register – PWM

Each device contains a Pulse Width Modulator which has a corresponding control register, known as PWM. The 8-bit contents of this register defines the duty cycle value for the modulation cycle of the Pulse Width Modulator.

A/D Converter Registers – ADR, ADRL, ADRH, ADCR, ACSR

Each device contains a 4-channel 8-bit, 9-bit or 12-bit A/D converter. The correct operation of the A/D requires the use of one or two data registers, a control register and a clock source register. For the HT46R01 devices, which have an 8-bit A/D converter, there is a single data register, known as ADR. For the other devices, which contain higher resolution A/D converters, there are two data registers, a high byte data register known as ADRH, and a low byte data register known as ADRL. These are the register locations where the digital value is placed after the completion of an analog to digital conversion cycle. The channel selection and configuration of the A/D converter is setup via the control register ADCR while the A/D clock frequency is defined by the clock source register, ACSR.

System Control Register – CTRL0

This register is used to provide control over certain internal functions including certain system clock options, the PFD clock source and on/off control, the PWM mode and on/off control and an RTC Oscillator quick start up function.

System Control Register – CTRL1

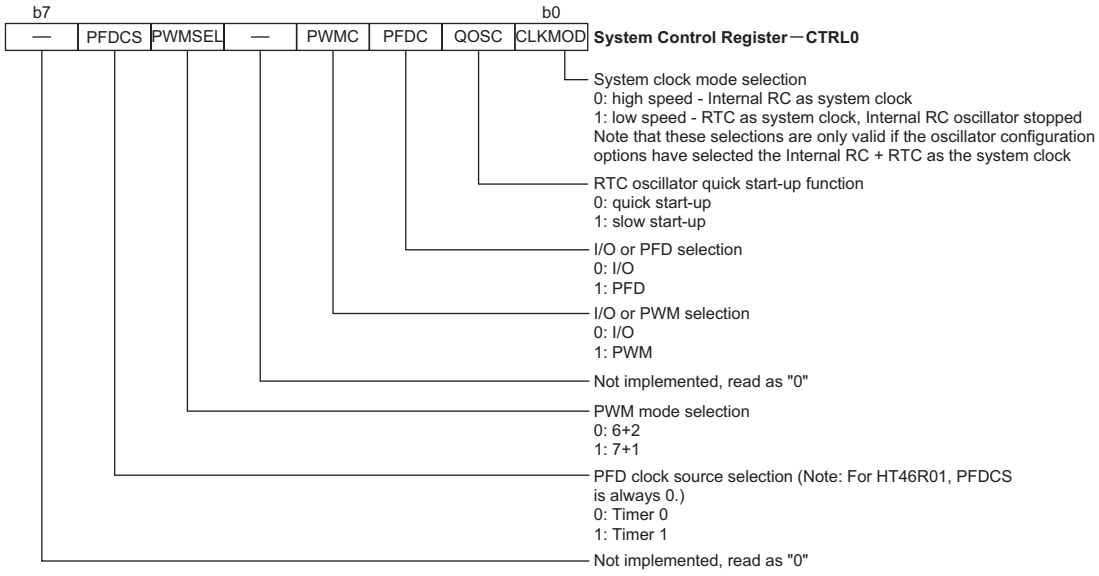
This register is used to provide control over certain internal functions including the External Interrupt edge trigger type and the Watchdog Timer control function.

Wake-up Function Register – PAWK

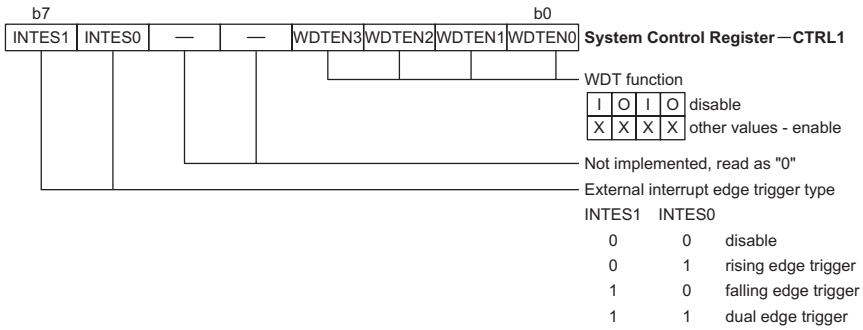
When the microcontroller enters the Power Down Mode, various methods exist to wake the device up and continue with normal operation. One method is to allow a low going edge on the I/O pins to have a wake-up function. This register is used to select which I/O pins are used to have this wake-up function.

Pull-high Register – PAPU

The I/O pins, if configured as inputs, can have internal pull-high resistors connected, which eliminates the need for external pull-high resistors. This register selects which I/O pins are connected to internal pull-high resistors.



System Control Register – CTRL0



System Control Register – CTRL1

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. There are seven 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~PA6, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selectable via a register known as PAPU, located in the Data Memory. 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~PA6 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 PA6 can be selected individually to have this wake-up feature using an internal register known as PAWK, located in the Data Memory.

I/O Port Control Registers

Port A has its own control register, known as PAC, which controls the input/output configuration. With this control register, each PA0~PA6 I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. Pins PA0 to PA6 port are directly mapped to a bit in its associated 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 PA3. To use the pin as an external interrupt input the correct bits in the INTCO register must be programmed. The pin must also be setup as an input by setting the appropriate bit in the Port Control Register. A pull-high resistor can also be selected via the appropriate port pull-high resistor register. Note that even if the pin is setup as an external interrupt input the I/O function still remains.

- External Timer/Event Counter Input
  Each device contains either one or two Timer/Event Counters depending upon which one is chosen. Each Timer/Event Counter has an external input pin, known as TMR0 or TMR1 which are pin-shared with I/O pins PA2 and PA4 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. Pull-high resistor options can also be selected via the appropriate port pull-high resistor register. Note that even if the pin is setup as an external timer input the I/O function still remains.

- PFD Output
  Each device contains a PFD function whose single output is pin-shared with PA1. The PFD output function of this pin along with the timer source is chosen via bits in the CTRL0 register. Note that the corresponding bit of the port control register, PAC.1, 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.

- PWM Output
  All devices contain a single PWM output pin shared with pins PA4. The PWM output function of this pin along with the mode type is chosen via bits in the CTRL0 register. Note that the corresponding bit or bits of the port control register, PAC.4, must setup the pin as an output to enable the PWM 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 resistor option, even if the PWM has been selected.

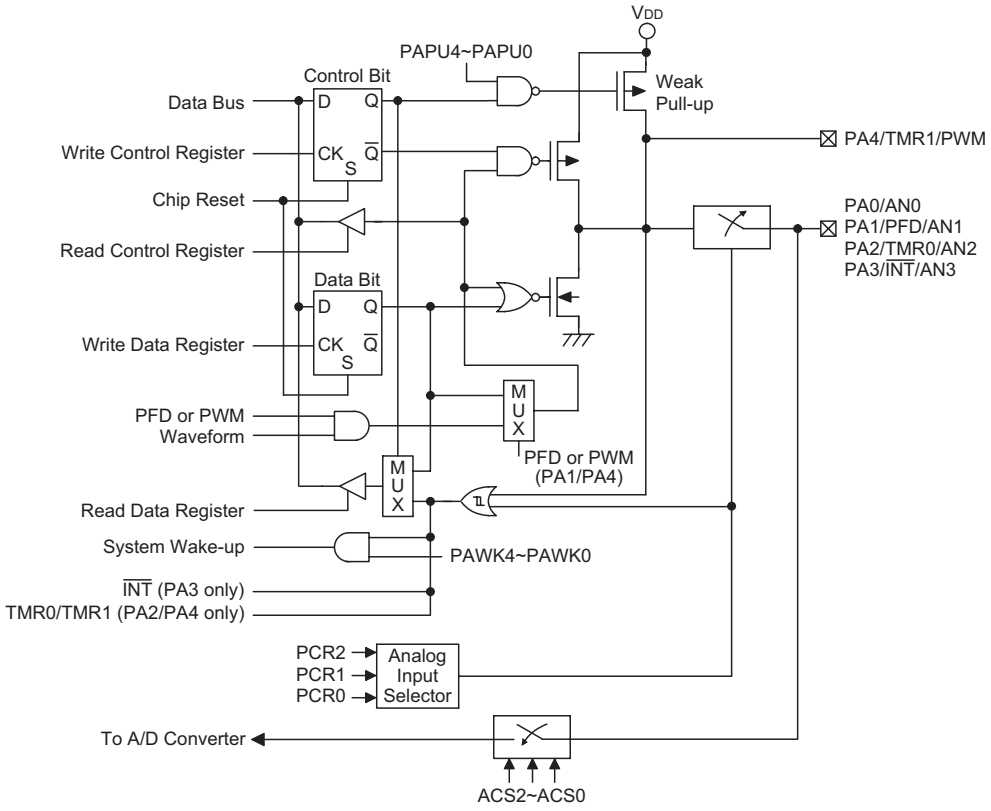
- A/D Inputs
  Each device has four A/D converter inputs. All of these analog inputs are pin-shared with PA0 to PA3. If these pins are to be used as A/D inputs and not as normal I/O pins then the corresponding bits in the A/D Converter Control Register, ADCR, must be properly set. There are no configuration options associated with the A/D function. If used as I/O pins, then full pull-high resistor selections remain, however if used as A/D inputs then any pull-high resistor selections associated with these pins will be automatically disconnected.
  Note that for A/D inputs that are shared with interrupt or timer pins, care should be taken when switching the pins to operate as an A/D input. In such cases it is recommended that the timer or interrupt input is first disabled as a false timer input signal or external interrupt signal may be generated when the A/D input is setup.

I/O Pin Structures

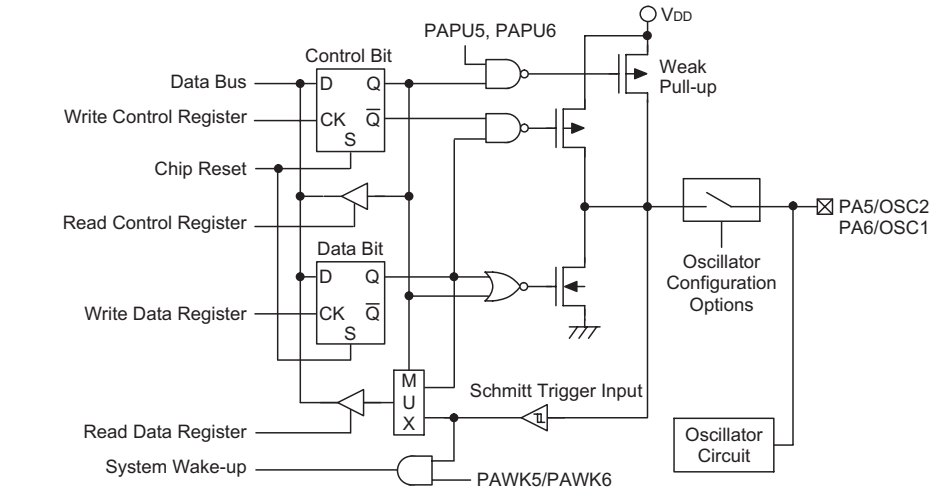
The diagrams illustrate the I/O pin internal structures. As the exact logical construction of the I/O pin may differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins.

Programming Considerations

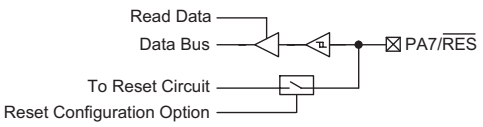
Within the user program, one of the first things to consider is port initialisation. After a reset, the PA data register and PAC port control register will be set high. This



PA0~PA4 Input/Output Ports

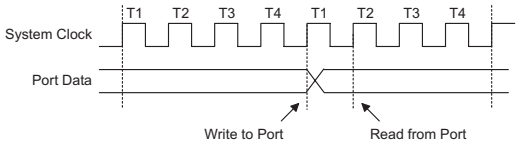


PA5~PA6 Input/Output Ports



PA7 Input Port

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 PAC port control register, is then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated PA 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 ports.



Read/Write Timing

Pins PA0 to PA6 each have a wake-up functions, selected via the PAWK register. 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.

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. The devices contain either one or two count-up timers each of 8-bit capacity depending upon which device is selected. As each timer has three different operating modes, they can be configured to operate as a general timer, an external event counter or as a pulse width measurement device. The provision of an internal prescaler to the clock circuitry of some 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 devices 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. The accompanying table lists the associated timer register names.

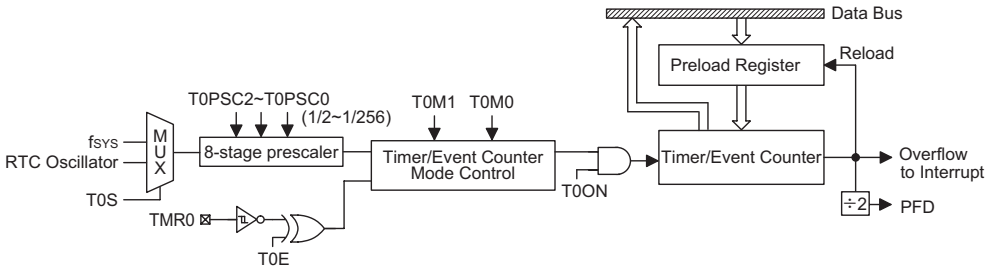
	HT46R01	HT46R02 HT46R03
No. of 8-bit Timers	1	2
Timer Register Name	TMR0	TMR0 TMR1
Timer Control Register	TMR0	TMR0C TMR1C

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 device 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.

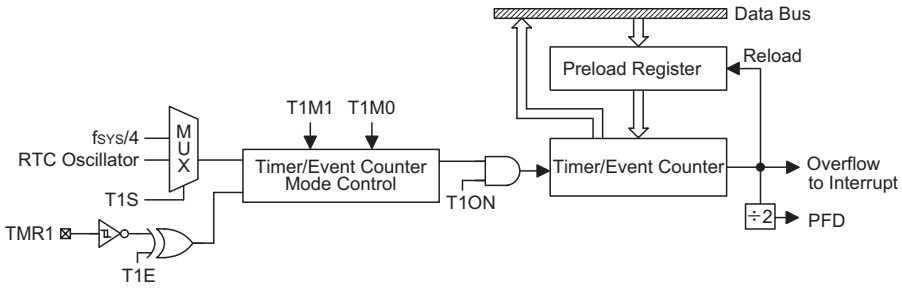
Configuring the Timer/Event Counter Input Clock Source

The internal timer's clock can originate from various sources, depending upon which device and 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 counting mode, the clock source being provided on an external timer pin TMR0 or TMR1, depending upon which device and which timer is used. Depending upon the condition of the T0E or T1E bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.



8-bit Timer/Event Counter 0 Structure – All Devices



8-bit Timer/Event Counter 1 Structure – HT46R02 and HT46R03

Timer Registers – TMR0, TMR1

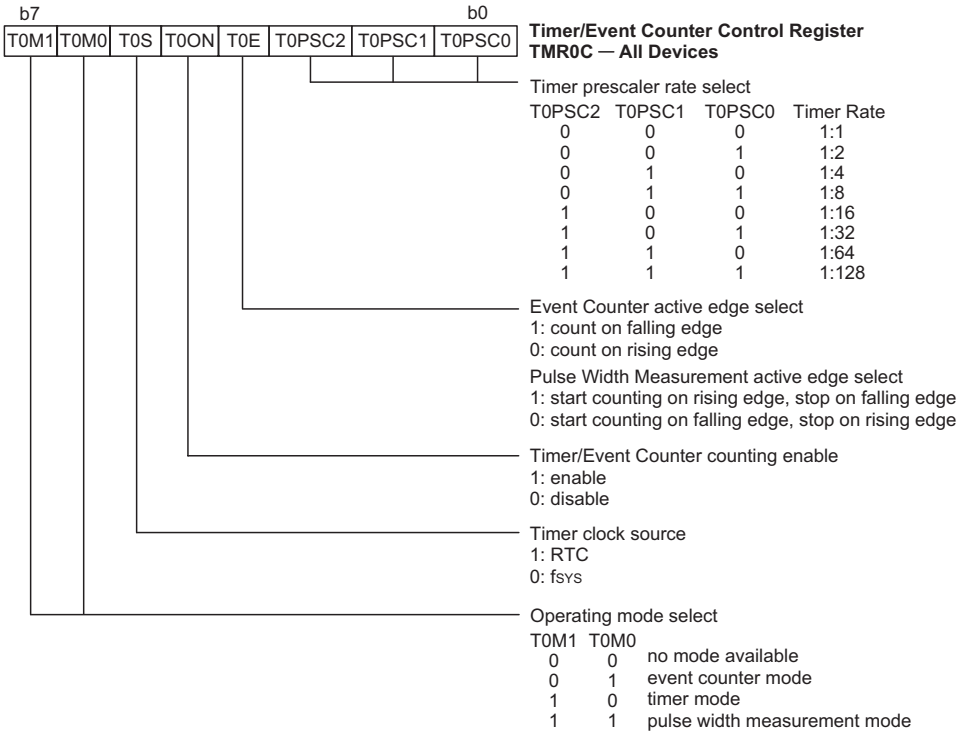
The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. These registers are known as TMR0 or TMR1, depending upon which device is used. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH 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.

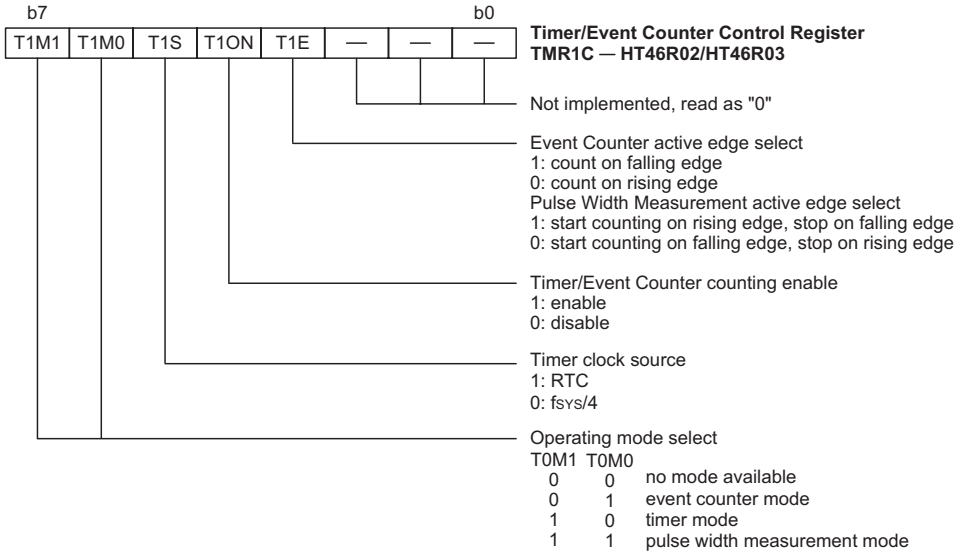
Note that to achieve a maximum full range count of FFH, the preload register must first be cleared to all zeros. It should be noted that after power-on, the preload registers will be in an unknown condition. Note that if the Timer/Event Counters are in an OFF condition and data is written to their preload 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.

Timer Control Registers – TMR0C, TMR1C

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

For devices with only one timer, the single Timer Control Register is known as TMR0C while for devices with more than one timer, there are two Timer Control Registers, known as TMR0C and TMR1C. It is the Timer Control Register together with its corresponding timer registers that control the full operation of the



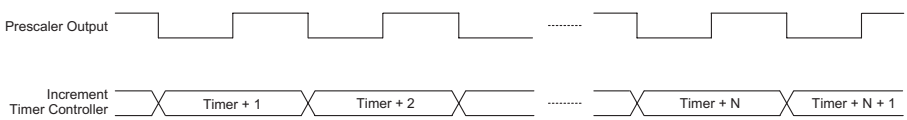


Timer/Event Counters. Before the timers can be used, it is essential that the appropriate Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

To choose which of the three modes the timer is to operate in, either in the timer mode, the event 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 timers that have prescalers, 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 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 utilized 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 either f_{SYS} or the RTC oscillator. 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. For Timer/Event Counter 1, the timer input clock source $f_{SYS}/4$ or the RTC oscillator. 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.



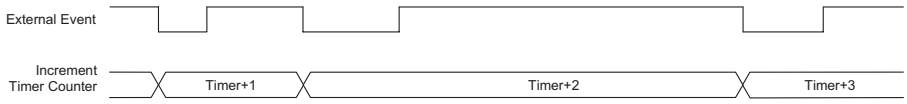
Timer Mode Timing Chart

Configuring the Event Counter Mode

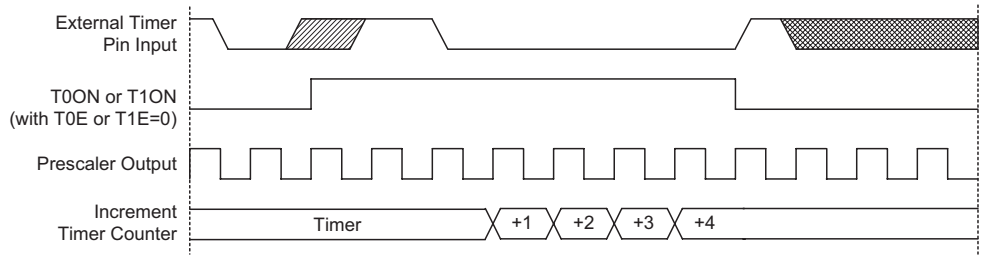
In this mode, a number of externally changing logic events, occurring on the external timer pin, can be recorded by the internal timer. For the timer to operate in the event counting mode, the bit pair, T0M1/T0M0 or T1M1/T1M0, depending upon which timer is used, must be set to 0 and 1 respectively. The 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 two 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 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 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.



Event Counter Mode Timing Chart



Prescaler Output is sampled at every falling edge of T1.

Pulse Width Measure Mode Timing Chart

Programmable Frequency Divider – PFD

The PFD output is pin-shared with the I/O pin PA1. The PFD on/off function and its timer source are selected via bits in the CTRL0 register, 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 1 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 PA1 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 PA1 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.

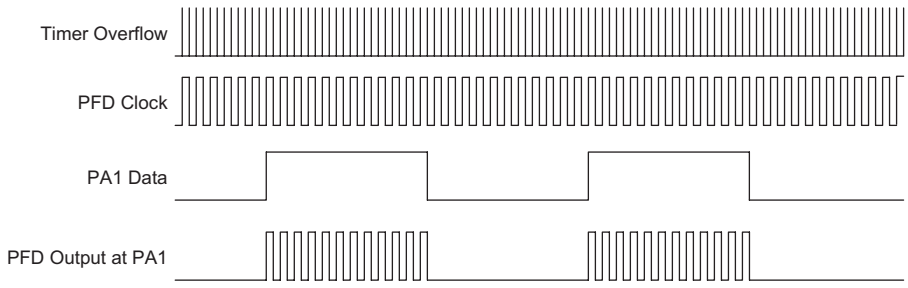
I/O Interfacing

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. 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



PFD Output Control

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 is based on the HT46R02 and HT46R03 devices, which contains two internal 8-bit Timer/Event Counters. 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 tmr1     ; 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
:                    :
```

Pulse Width Modulator

Each microcontroller is provided with a single Pulse Width Modulation, PWM, output. Useful for such applications such as motor speed control, the PWM function provides outputs with a fixed frequency but with a duty cycle that can be varied by setting particular values into the corresponding PWM register.

A single register, known as PWM and located in the Data Memory is assigned to the Pulse Width Modulator. It is here that the 8-bit value, which represents the overall duty cycle of one modulation cycle of the output waveform, should be placed. To increase the PWM modulation frequency, each modulation cycle is subdivided into two or four individual modulation subsections, known as the 7+1 mode or 6+2 mode respectively. The required mode and the on/off control for the PWM is selected using the CTRL0 register. Note that when using the PWM, it is only necessary to write the required value into the PWM register and select the required mode setup and on/off control using the CTRL0 register, the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware. For all devices, the PWM clock source is the system clock f_{SYS}.

This method of dividing the original modulation cycle into a further 2 or 4 sub-cycles enable the generation of higher PWM frequencies which allow a wider range of applications to be served. As long as the periods of the generated PWM pulses are less than the time constants of the load, the PWM output will be suitable as such long time constant loads will average out the pulses of the PWM output. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood. As the PWM clock is the system clock, f_{SYS}, and as the PWM value is 8-bits wide, the overall PWM cycle frequency is $f_{SYS}/256$. However, when in the 7+1 mode of operation the PWM modulation frequency will be $f_{SYS}/128$, while the PWM modulation frequency for the 6+2 mode of operation will be $f_{SYS}/64$.

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

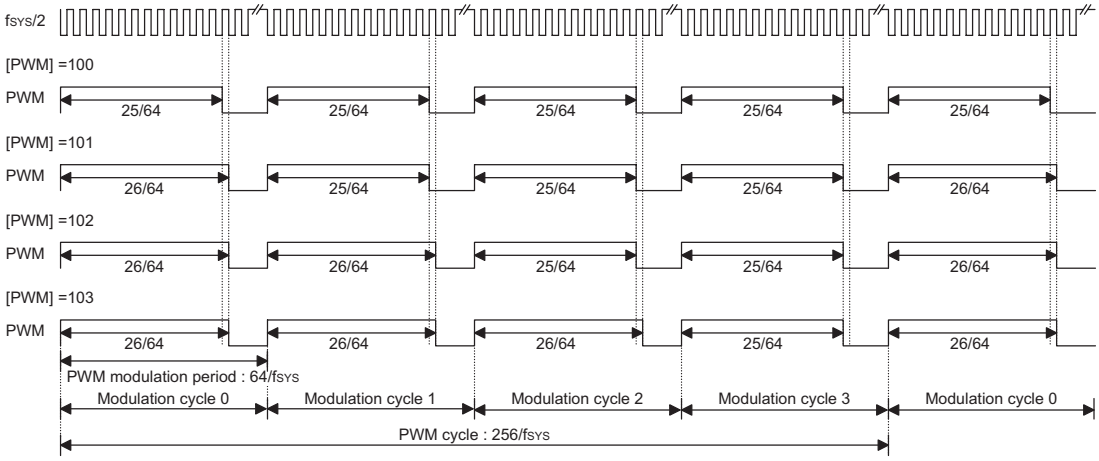
6+2 PWM Mode

Each full PWM cycle, as it is controlled by an 8-bit PWM register, has 256 clock periods. However, in the 6+2 PWM mode, each PWM cycle is subdivided into four individual sub-cycles known as modulation cycle 0 ~ modulation cycle 3, denoted as i in the table. Each one of these four sub-cycles contains 64 clock cycles. In this mode, a modulation frequency increase of four is achieved. The 8-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit2~bit7 is denoted here as the DC value. The second group which consists of bit0~bit1 is known as the AC value. In the 6+2 PWM mode, the duty cycle value of each of the four modulation sub-cycles is shown in the following table.

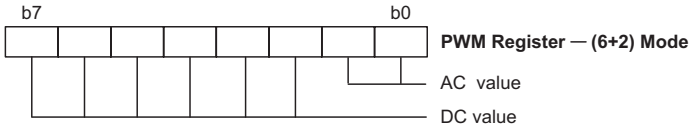
Parameter	AC (0~3)	DC (Duty Cycle)
Modulation cycle i (i=0~3)	i<AC	$\frac{DC+1}{64}$
	i≥AC	$\frac{DC}{64}$

6+2 Mode Modulation Cycle Values

The following diagram illustrates the waveforms associated with the 6+2 mode of PWM operation. It is important to note how the single PWM cycle is subdivided into 4 individual modulation cycles, numbered from 0~3 and how the AC value is related to the PWM value.



6+2 PWM Mode



PWM Register for 6+2 Mode

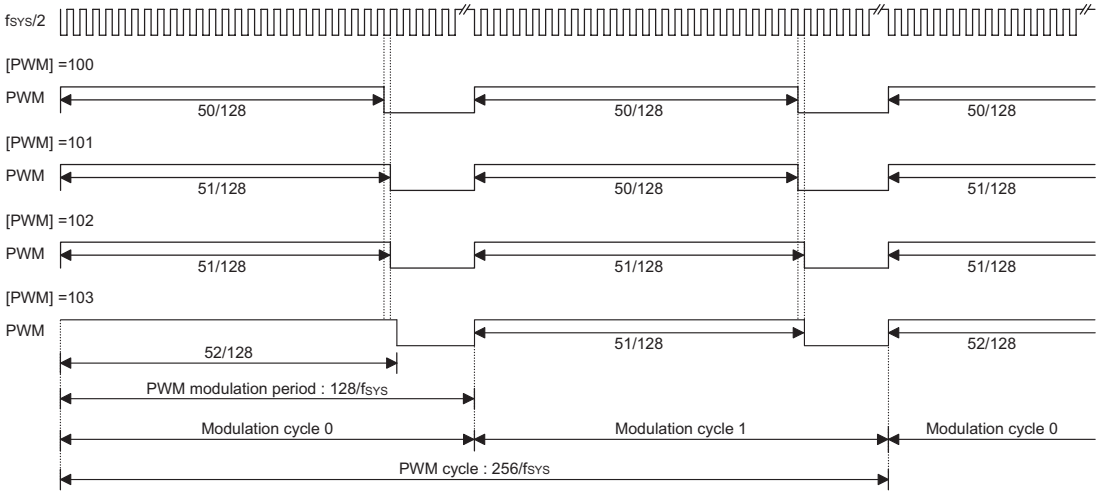
7+1 PWM Mode

Each full PWM cycle, as it is controlled by an 8-bit PWM register, has 256 clock periods. However, in the 7+1 PWM mode, each PWM cycle is subdivided into two individual sub-cycles known as modulation cycle 0 ~ modulation cycle 1, denoted as i in the table. Each one of these two sub-cycles contains 128 clock cycles. In this mode, a modulation frequency increase of two is achieved. The 8-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit1~bit7 is denoted here as the DC value. The second group which consists of bit0 is known as the AC value. In the 7+1 PWM mode, the duty cycle value of each of the two modulation sub-cycles is shown in the following table.

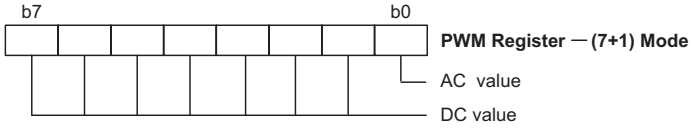
Parameter	AC (0~1)	DC (Duty Cycle)
Modulation cycle i (i=0~1)	i<AC	$\frac{DC+1}{128}$
	i≥AC	$\frac{DC}{128}$

7+1 Mode Modulation Cycle Values

The following diagram illustrates the waveforms associated with the 7+1 mode PWM operation. It is important to note how the single PWM cycle is subdivided into 2 individual modulation cycles, numbered 0 and 1 and how the AC value is related to the PWM value.



7+1 PWM Mode



PWM Register for 7+1 Mode

PWM Output Control

The PWM output is pin-shared with the I/O pin PA4. To operate as a PWM output and not as an I/O pin, the correct bits must be set in the CTRL0 register. A zero value must also be written to the corresponding bit in the I/O port control register PAC.4 to ensure that the PWM output pin is setup as an output. After these two initial steps have been carried out, and of course after the required PWM value has been written into the PWM register, writing a high value to the corresponding bit in the output data register PA.4 will enable the PWM data to appear on the pin. Writing a zero value will disable the PWM output function and force the output low. In this way, the Port A data output register can be used as an on/off control for the PWM function. Note that if the CTRL0 register has selected the PWM function, but a high value has been written to its corresponding bit in the PAC control register to configure the pin as an input, then the pin can still function as a normal input line, with pull-high resistor options.

PWM Programming Example

The following sample program shows how the PWM output is setup and controlled.

```
mov a,64h    ; setup PWM value of 100
             ; decimal which is 64H
mov pwm,a
set ctrl.5   ; select the 7+1 PWM mode
set ctrl.3   ; select pin PA4 to have a PWM
             ; function
clr pac.4    ; setup pin PA4 as an output
set pa.4     ; PD.0=1; enable the PWM
             ; output
:                  :
clr pa.4     ; disable the PWM output – pin
             ; PD4 forced low
```

Analog to Digital Converter

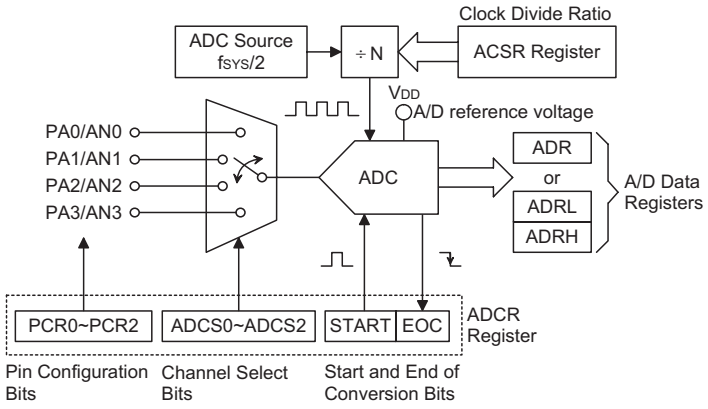
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the microcontroller, the need for external components is reduced significantly with the corresponding follow-on benefits of lower costs and reduced component space requirements.

A/D Overview

Each of the devices contains a 4-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into either an 8-bit, 9-bit or 12-bit digital value, depending upon which device is used.

Device	Input Channels	Conversion Bits	Input Pins
HT46R01	4	8	PA0~PA3
HT46R02	4	9	PA0~PA3
HT46R03	4	12	PA0~PA3

The following diagram shows the overall internal structure of the A/D converter, together with its associated registers.



A/D Converter Structure

A/D Converter Data Registers – ADR, ADRL, ADRH

For the HT46R01 devices, which have an 8-bit A/D converter, a single register, known as ADR, is used to store the 8-bit analog to digital conversion value. For the remaining devices, which have a 9 or 12-bit A/D converter, two registers are required, a high byte register, known as ADRH, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. For devices which use two A/D Converter Data Registers, note that only the high byte register ADRH utilises its full 8-bit contents. The low byte register utilises only some of its 8-bit contents as it contains only the lowest bits of the converted value.

In the following tables, D0~D8 are the A/D conversion data result bits.

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	D7	D6	D5	D4	D3	D2	D1	D0

A/D Data Register – HT46R01

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADRL	D0	—	—	—	—	—	—	—
ADRH	D8	D7	D6	D5	D4	D3	D2	D1

A/D Data Register – HT46R02

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADRL	D3	D2	D1	D0	—	—	—	—
ADRH	D11	D10	D9	D8	D7	D6	D5	D4

A/D Data Register – HT46R03

A/D Converter Control Register – ADCR

To control the function and operation of the A/D converter, a control register known as ADCR is provided. This 8-bit register defines functions such as the selection of which analog channel is connected to the internal A/D converter, which pins are used as analog inputs and which are used as normal I/Os as well as controlling the start function and monitoring the A/D converter end of conversion status.

One section of this register contains the bits ACS2~ACS0 which define the channel number. As each of the devices contains only one actual analog to digital converter circuit, each of the individual 4 analog inputs must be routed to the converter. It is the function of the ACS2~ACS0 bits in the ADCR register to determine which analog channel is actually connected to the internal A/D converter. Note that the ACS2 bit must always be assigned a zero value. The ADCR control register also contains the PCR2~PCR0 bits which determine which pins on Port A are used as analog inputs for the A/D converter and which pins are to be used as normal I/O pins. If the 3-bit address on PCR2~PCR0 has a value of "100" or higher, then all four pins, namely AN0, AN1, AN2 and AN3 will all be set as analog inputs. Note that if the PCR2~PCR0 bits are all set to zero, then all the Port B pins will be setup as normal I/Os and the internal A/D converter circuitry will be powered off to reduce the power consumption.

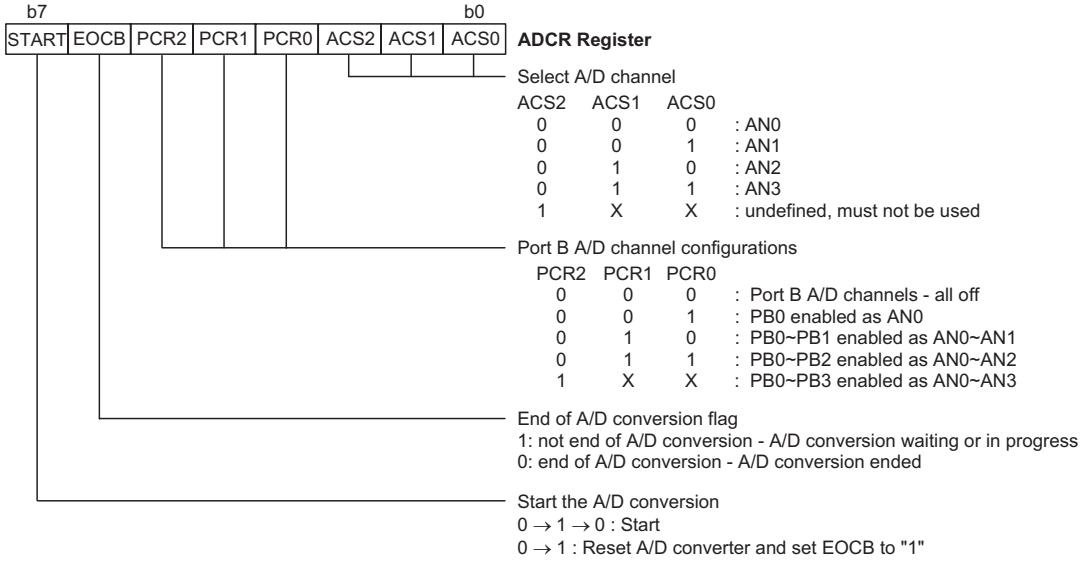
The START bit in the ADCR register is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the EOCB bit in the ADCR register will be set high and the analog to digital converter will be reset. It is the START bit that is used to control the overall on/off operation of the internal analog to digital converter.

The EOCB bit in the ADCR register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically cleared to zero by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the EOCB bit in the ADCR register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

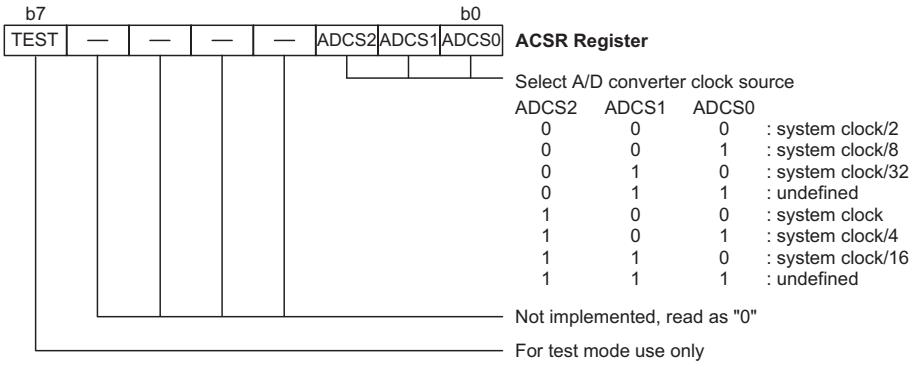
A/D Converter Clock Source Register – ACSR

The clock source for the A/D converter, which originates from the system clock f_{SYS}, is first divided by a division ratio, the value of which is determined by the bits ADCS0 to ADCS2 in the ACSR register.

Although the A/D clock source is determined by the system clock f_{SYS}, and by bits ADCS0 to ADCS2, there are some limitations on the maximum A/D clock source speed that can be selected. As the minimum value of permissible A/D clock period, t_{AD}, is 1us for all devices, care must be taken for system clock speeds in excess of 1MHz. For system clock speeds in excess of 1MHz, the ADCS0 to ADCS2 bits should not be set to give an A/D clock period less than the specified minimum A/D clock period which may result in inaccurate A/D conversion values. Refer to the table for examples, where values marked with an asterisk * show where special care must be taken, as the values are less than the specified minimum A/D Clock Period.



ADCR Register



ACSR Register

f_{SYS}	A/D Clock Period (t_{AD})/ADCS0~ADCS2					
	f_{SYS} 100	**$f_{SYS}/2$ 000**	**$f_{SYS}/4$ 101**	**$f_{SYS}/8$ 001**	**$f_{SYS}/16$ 110**	**$f_{SYS}/32$ 010**
1MHz	1μs	2μs	4μs	8μs	16μs	32μs
2MHz	500ns*	1μs	2μs	4μs	8μs	16μs
4MHz	250ns*	500ns*	1μs	2μs	4μs	8μs
8MHz	125ns*	250ns*	500ns*	1μs	2μs	4μs
12MHz	62.5ns*	0.125ns*	250ns*	500ns*	1μs	2μs

A/D Clock Period Examples

A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins on Port A. Bits PCR0~PCR2 in the ADCR register, not configuration options, determine whether the input pins are setup as normal Port A input/output pins or whether they are setup as analog inputs. In this way, pins can be changed under program control to change their function from normal I/O operation to analog inputs and vice versa. Pull-high resistors, which are setup via the PAPU register, apply to the input pins only when they are used as normal I/O pins, if setup as A/D inputs the pull-high resistors will be automatically disconnected. Note that it is not necessary to first setup the A/D pin as an input in the PAC port control register to enable the A/D input, when the PCR2~PCR0 bits enable an A/D input, the status of the port control register will be overridden. The VDD power supply pin is used as the A/D converter reference voltage, and as such analog inputs must not be allowed to exceed this value. Appropriate measures should also be taken to ensure that the VDD pin remains as stable and noise free as possible.

Initialising the A/D Converter

The internal A/D converter must be initialised in a special way. Each time the A/D channel selection bits are modified by the program, the A/D converter must be re-initialised. If the A/D converter is not initialised after the channel selection bits are changed, the EOCB flag may have an undefined value, which may produce a false end of conversion signal. To initialise the A/D converter after the channel selection bits have changed, then, within a time frame of one to ten instruction cycles, the START bit in the ADCR register must first be set high and then immediately cleared to zero. This will ensure that the EOCB flag is correctly set to a high condition.

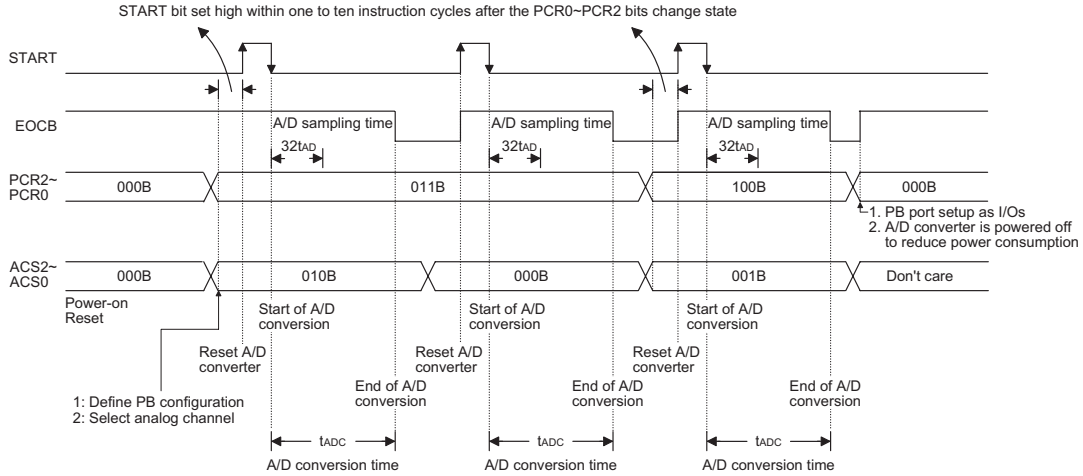
Summary of A/D Conversion Steps

The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCS0 to ADCS20 in the ACSR register.
- Step 2
  Select which pins on Port A are to be used as A/D inputs and configure them as A/D input pins by correctly programming the PCR0~PCR2 bits in the ADCR register.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS0~ACS2 bits which are also contained in the ADCR register. Note that this step can be combined with Step 2 into a single ADCR register programming operation.
- Step 4
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, in the interrupt control register must be set high and the A/D converter interrupt bit, EADI, in the interrupt control register must also be set high.
- Step 5
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR register from low to high and then low again. Note that this bit should have been originally set to a zero value.
- Step 6
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers be read to obtain the conversion value. As an alternative method if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

Note: When checking for the end of the conversion process, if the method of polling the EOCB bit in the ADCR register is used, the interrupt enable step above can be omitted.

The following timing diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing.



Note: A/D clock must be $f_{SYS}/2$, $f_{SYS}/8$ or $f_{SYS}/32$

A/D Conversion Timing

The setting up and operation of the A/D converter function is fully under the control of the application program as there are no configuration options associated with the A/D converter. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is dependent upon the device chosen and is a function of the A/D clock period t_{AD} as shown in the table.

Device	A/D Conversion Time
HT46R01	$64t_{AD}$
HT46R02	$76t_{AD}$
HT46R03	$80t_{AD}$

A/D Conversion Time

Programming Considerations

When programming, special attention must be given to the A/D channel selection bits in the ADCR register. If these bits are all cleared to zero no external pins will be selected for use as A/D input pins allowing the pins to be used as normal I/O pins. When this happens the power supplied to the internal A/D circuitry will be reduced resulting in a reduction of supply current. This ability to reduce power by turning off the internal A/D function by clearing the A/D channel selection bits may be an important consideration in battery powered applications.

Another important programming consideration is that when the A/D channel selection bits change value the A/D converter must be re-initialised. This is achieved by pulsing the START bit in the ADCR register immediately after the channel selection bits have changed state. The exception to this is where the channel selection bits are all cleared, in which case the A/D converter is not required to be re-initialised.

A/D Programming Example

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the EOCB bit in the ADCR register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

Example: using an EOCB polling method to detect the end of conversion for the HT46R01 devices.

```
clr     EADI                ; disable ADC interrupt
mov     a,00000001B
mov     ACSR,a              ; setup the ACSR register to select fSYS/8 as
                            ; the A/D clock
mov     a,00100000B         ; setup ADCR register to configure Port PA0~PA3
                            ; as A/D inputs
mov     ADCR,a              ; and select AN0 to be connected to the A/D
                            ; converter
        :
```

```
                :                   ; As the Port A channel bits have changed the
                                    ; following START
                                    ; signal (0-1-0) must be issued within 10
                                    ; instruction cycles
                :
Start_conversion:
            clr     START
            set     START           ; reset A/D
            clr     START           ; start A/D
Polling_EOC:
            sz      EOCB            ; poll the ADCR register EOCB bit to detect end
                                    ; of A/D conversion
            jmp     polling_EOC     ; continue polling
            mov     a,ADR           ; read conversion result value from the ADR
                                    ; register
            mov     adr_buffer,a    ; save result to user defined memory
                    :
                    :
            jmp     start_conversion ; start next A/D conversion
```

Example: using an interrupt method to detect the end of conversion for the HT46R01 devices

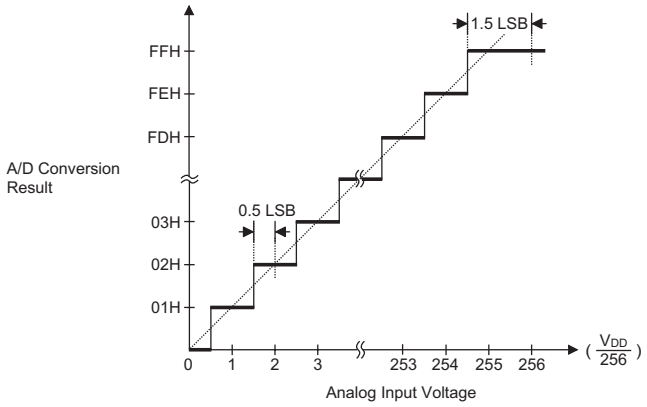
```
            clr     EADI            ; disable ADC interrupt
            mov     a,00000001B
            mov     ACSR,a          ; setup the ACSR register to select fSYS/8 as
                                    ; the A/D clock

            mov     a,00100000B     ; setup ADCR register to configure Port PA0~PA3
                                    ; as A/D inputs
            mov     ADCR,a          ; and select AN0 to be connected to the A/D
                                    ; converter
                    :
                                    ; As the Port A channel bits have changed the
                                    ; following START
                                    ; signal (0-1-0) must be issued within 10
                                    ; instruction cycles
                    :
Start_conversion:
            clr     START
            set     START           ; reset A/D
            clr     START           ; start A/D
            clr     ADF             ; clear ADC interrupt request flag
            set     EADI            ; enable ADC interrupt
            set     EMI             ; enable global interrupt
                    :
                    :
                    :
; ADC interrupt service routine
ADC_ISR:
            mov     acc_stack,a     ; save ACC to user defined memory
            mov     a,STATUS
            mov     status_stack,a  ; save STATUS to user defined memory
                    :
                    :
            mov     a,ADR           ; read conversion result value from the ADR
                                    ; register
            mov     adr_buffer,a    ; save result to user defined register
                    :
                    :
EXIT_INT_ISR:
            mov     a,status_stack
            mov     STATUS,a        ; restore STATUS from user defined memory
            mov     a,acc_stack     ; restore ACC from user defined memory
            reti
```

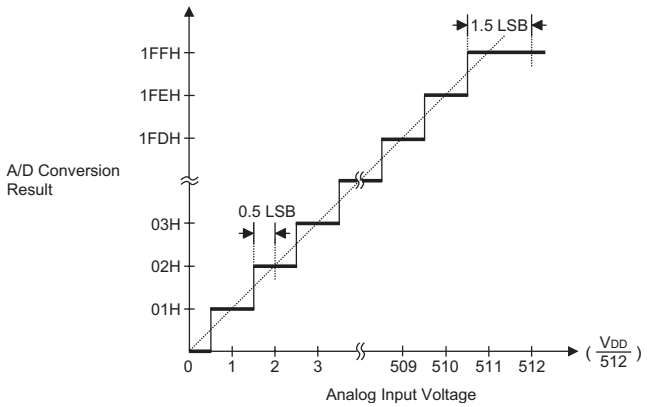
A/D Transfer Function

As the HT46R01 devices contain an 8-bit A/D converter, their full-scale converted digitised value is equal to 0FFH. Since the full-scale analog input value is equal to the V_{DD} voltage, this gives a single bit analog input value of $V_{DD}/256$. For the HT46R02 devices, which contains a 9-bit A/D converter, their full-scale converted digitised value is equal to 1FFH, giving a single bit analog input value of $V_{DD}/512$, while for the HT46R03 devices, which contains a 12-bit A/D converter, their full-scale converted digitised value is equal to FFFH, giving a single bit analog input value of $V_{DD}/4096$. The following graphs show the ideal transfer function between the analog input value and the digitised output value for the A/D converters.

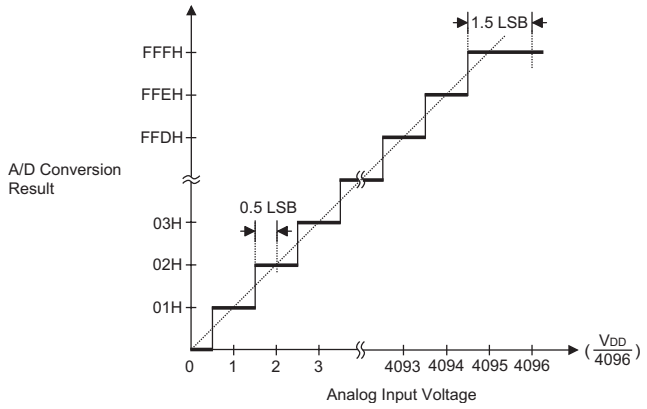
Note that to reduce the quantisation error, a 0.5 LSB offset is added to the A/D Converter input. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the V_{DD} level.



Ideal A/D Transfer Function – HT46R01



Ideal A/D Transfer Function – HT46R02



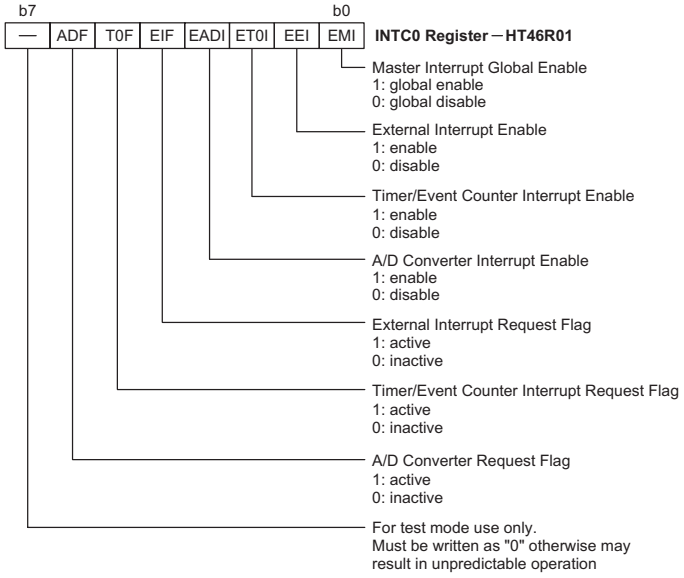
Ideal A/D Transfer Function – HT46R03

Interrupts

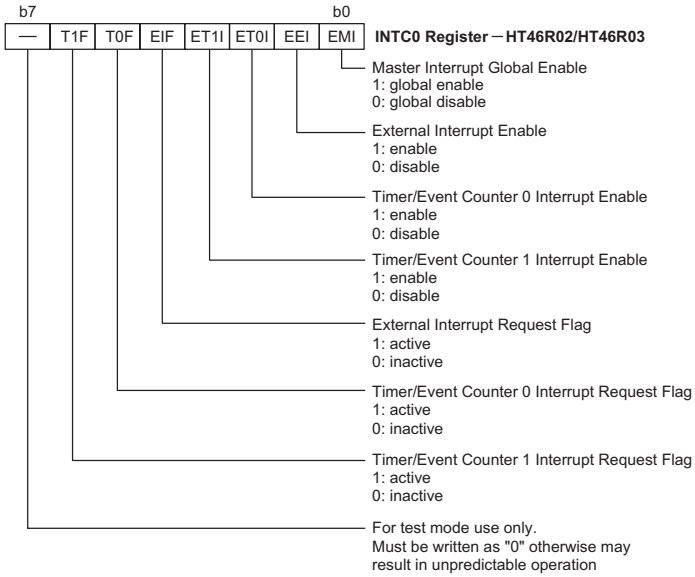
Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller 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 A/D converter interrupt.

Interrupt Register

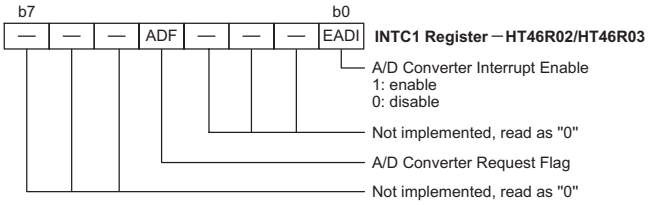
Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by a single INTC0 register or dual INTC0 and INTC1 registers, which are located in the Data Memory. By controlling the appropriate enable bits in these registers each individual interrupt can be enabled 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.



INTC0 Register – HT46R01



INTC0 Register – HT46R02/HT46R03



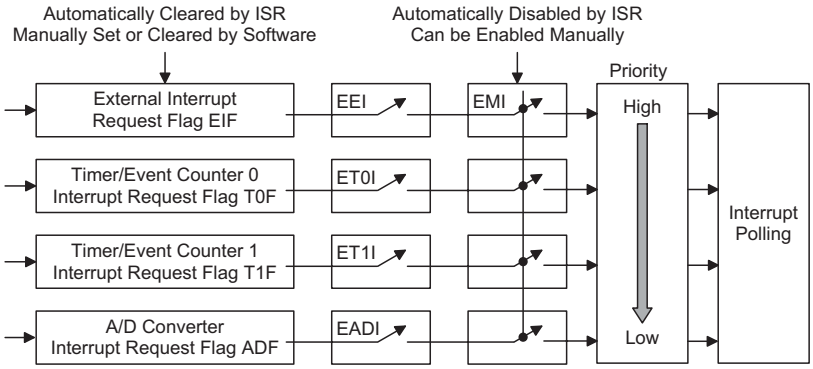
INTC1 Register – HT46R02/HT46R03

Interrupt Operation

A Timer/Event Counter overflow, an end of A/D conversion or an active edge on the external interrupt pin 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 following 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 Scheme

Note: In the figure, the T1F interrupt request flag and the ET1I interrupt enable bit only exist for the HT46R02/HT46R03 devices, which have two timers.

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	HT46R01	HT46R02 HT46R03
External Interrupt	1	1
Timer/Event Counter 0 Overflow	2	2
Timer/Event Counter 1 Overflow	N/A	3
A/D Converter Interrupt	3	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, EEI, 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 externalINTline. 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 CTRL1 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

The external interrupt pin is pin-shared with the I/O pin PA3 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 CTRL1 register. The pin must also be setup as an input by setting the corresponding PAC.3 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 Interrupt

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. For those devices with two timers, the Timer/Event Counter 1 overflow has an interrupt vector at location 0CH. 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.

A/D Interrupt

For an A/D interrupt to occur, the global interrupt enable bit, EMI, and the corresponding interrupt enable bit, EADI, must be first set. An actual A/D interrupt will take place when the A/D converter request flag, ADF, is set, a situation that will occur when an A/D conversion process has completed. When the interrupt is enabled, the stack is not full and an A/D conversion process finishes execution, a subroutine call to the A/D interrupt vector at location 0CH for the HT46R01 devices or 10H for the other devices, will take place. When the interrupt is serviced, the A/D interrupt request flag, ADF, 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.

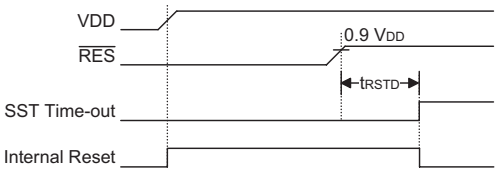
In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{RES}$ line is forcefully pulled 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 $\overline{RES}$ reset is implemented in situations where the power supply voltage falls below a certain threshold.

Reset Functions

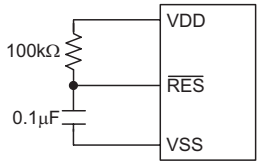
There are five ways in which a microcontroller reset can occur, through events occurring both internally and externally:

- 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 proper reset operation. For this reason 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.



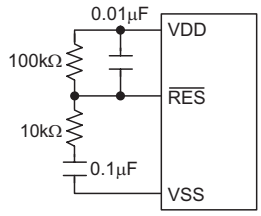
Power-On Reset Timing Chart

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

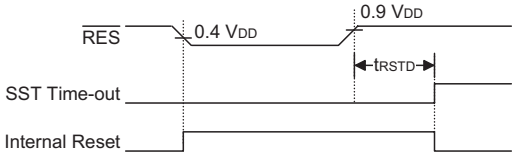
For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



Enhanced Reset Circuit

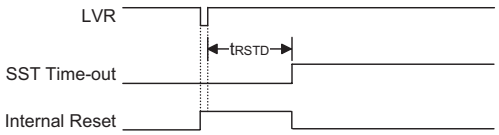
More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

- $\overline{RES}$ 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.



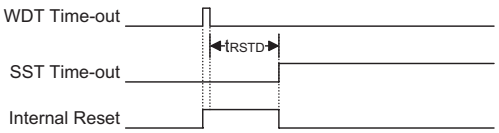
$\overline{RES}$ Reset Timing Chart

- 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 is selected via a configuration option. 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 can be selected via configuration options.



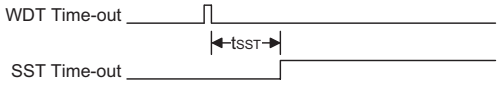
Low Voltage Reset Timing Chart

- Watchdog Time-out Reset during Normal Operation
  The Watchdog time-out Reset during normal operation is the same as a hardware $\overline{RES}$ 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

Note: That the SST can be enable (1024 clocks) or disable (only 2 clock needed) by configuration option if system clock is not from external crystal.

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	$\overline{RES}$ reset during power-on
u	u	$\overline{RES}$ 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.

HT46R01

Register	Reset (Power-on)	WDT time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
Program Counter	000H	000H	000H	000H	000H
MP0	1xxx xxxx	1uuu uuuu	uuuu uuuu	uuuu uuuu	1uuu uuuu
MP1	1xxx xxxx	1uuu uuuu	uuuu uuuu	uuuu 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	---- -111	---- -111	---- -111	---- -111	---- -uuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
PAWK	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
CTRL0	-00- 0000	-00- 0000	-00- 0000	-00- 0000	-uu- uuuu
CTRL1	10-- 1010	10-- 1010	10-- 1010	10-- 1010	uu-- uuuu
PWM	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- -000	1--- -000	1--- -000	1--- -000	u--- -uuu

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

HT46R02

Register	Reset (Power-on)	WDT time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
Program Counter	000H	000H	000H	000H	000H
MP0	1xxx xxxx	1uuu uuuu	uuuu uuuu	uuuu uuuu	1uuu uuuu
MP1	1xxx xxxx	1uuu uuuu	uuuu uuuu	uuuu 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	---- -111	---- -111	---- -111	---- -111	---- -uuu
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	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu
TMR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1---	0000 1---	0000 1---	0000 1---	uuuu u---
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
PAWK	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
CTRL0	-00- 0000	-00- 0000	-00- 0000	-00- 0000	-uu- uuuu
CTRL1	10-- 1010	10-- 1010	10-- 1010	10-- 1010	uu-- uuuu
PWM	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	x--- ----	x--- ----	x--- ----	x--- ----	u--- ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- -000	1--- -000	1--- -000	1--- -000	u--- -uuu

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

HT46R03

Register	Reset (Power-on)	WDT time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
Program Counter	000H	000H	000H	000H	000H
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
WDTS	---- -111	---- -111	---- -111	---- -111	---- -uuu
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	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu
TMR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1---	0000 1---	0000 1---	0000 1---	uuuu u---
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
PAWK	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
CTRL0	-00- 0000	-00- 0000	-00- 0000	-00- 0000	-uu- uuuu
CTRL1	10-- 1010	10-- 1010	10-- 1010	10-- 1010	uu-- uuuu
PWM	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	xxxx ----	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- -000	1--- -000	1--- -000	1--- -000	u--- -uuu

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

Oscillator

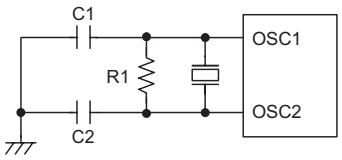
Various oscillator options offer the user a wide range of functions according to their various application requirements. Four types of system clocks can be selected while various clock source options for the Watchdog Timer are provided for maximum flexibility. All oscillator options are selected through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

System Clock Configurations

There are four methods of generating the system clock, using an external crystal/ceramic oscillator, an external RC network, an internal RC clock source and a combined internal RC clock source and Real Time Clock. One of these four methods must be selected using the configuration options.

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 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.

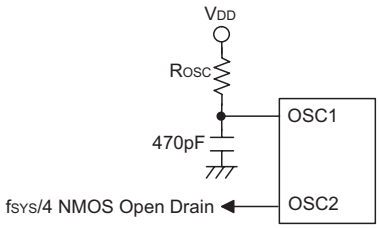


Crystal/Ceramic Oscillator

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

External System RC Oscillator

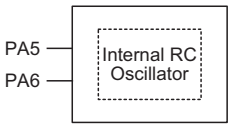
After selecting the correct configuration option, using the external system RC oscillator requires that a resistor, with a value between 24kΩ and 1.5MΩ, is connected between OSC1 and VDD, and a 470pF capacitor is connected to ground. Although this is a cost effective oscillator configuration, the oscillation 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. Note that only the OSC1 pin is used, which is shared with I/O pin PA6, leaving pin PA5 free for use as a normal I/O pin.



RC Oscillator

Internal System RC Oscillator

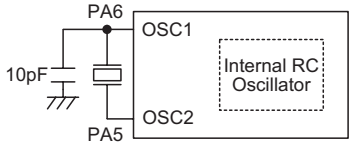
In addition to the external crystal/resonator or external RC system clock configurations, the devices also include an internal RC oscillator. The internal oscillator has three fixed frequencies of either 4MHz, 8MHz or 12MHz, the choice of which is indicated by the suffix marking next to the part number of the device used. This oscillator is fully integrated within the microcontroller and requires no external components. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PA5 and PA6 are free for use as normal I/O pins.



- Internal RC System Oscillator
- PA5/PA6 used as normal I/Os

Internal System RC Oscillator with RTC

When the microcontroller enters the Power Down Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the Power Down Mode. To do this, another clock, independent of the system clock, must be provided. To do this a configuration option exists to allow the Internal System RC Oscillator to be used in conjunction with an Real Time Clock, RTC, oscillator. Here the OSC1 and OSC2 pins, which are shared with I/O pins PA6 and PA5 should be connected to a 32768Hz crystal to implement this internal RTC oscillator. Additionally a 10pf capacitor should be connected between OSC1 and ground.



- Internal RC System Oscillator with External RTC Oscillator
- 32768Hz Crystal Connected to PA5/PA6

For applications using the RTC oscillator, the system clock can be chosen to be either the Internal System RC Oscillator or the RTC oscillator itself. This selection is made using the CLKMOD bit in the CTRL0 register. If this bit is set high then the 32768Hz external crystal will also provide the system clock source. If the bit is low then the system clock source will be the Internal RC Oscillator. When the system enters the Power Down Mode, the system clock, irrespective of whether the CLKMOD bit has selected the RTC or Internal RC Oscillator as its source, will always stop running. The following table shows the relationship between the CLKMOD bit and the various oscillators.

Operating Mode	CLKMOD Bit	Internal RC Osc.	RTC	System Clock
Normal Running	0	On	On	RC Oscillator
	1	Off	On	32768Hz
Power Down	X	Off	On	Stopped

During power up there is a time delay associated with the RTC oscillator waiting for it to start up. Bit 1 of the CTRL0 register, known as the QOSC bit, is provided to give a quick start-up function and can be used to minimize this delay. During a power up condition, this bit will be cleared to 0 which will initiate the RTC oscillator quick start-up function. However, as there is additional power consumption associated with this quick start-up function, to reduce power consumption after start up takes place, it is recommended that the application program should set the QOSC bit high about 2 seconds after power on. It should be noted that, no matter what condition the QOSC bit is set to, the RTC oscillator will always function normally, only there is more power consumption associated with the quick start-up function.

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.
- If the RTC oscillator configuration option is enabled then the RTC clock will keep running.
- 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 or RTC 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. The RTC, if configured for use, will also consume a limited amount of power, as it continues to run when the device enters the Power Down Mode. To keep the RTC power consumption to a minimum level the QOSC bit in the CTRL0 register, which controls the quick start up function, should be set high. If any I/O pins are configured as A/D analog inputs using the channel configuration bits in the ADCR register, then the A/D converter will be turned on and a certain amount of power will be consumed. It may be therefore desirable before entering the Power Down Mode to ensure that the A/D converter is powered down by ensuring that any A/D input pins are setup as normal logic inputs with pull-high resistors.

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 PA0 to PA6
- 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.

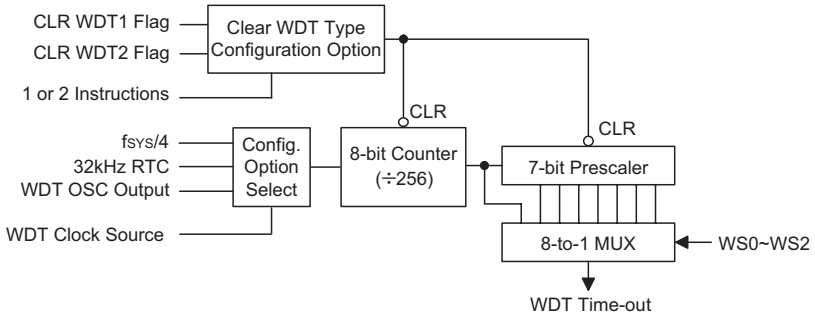
Pins PA0 to PA6 can be setup via an individual configuration option to permit a negative transition on the pin to wake-up the system. When a PA0 to PA6 pin wake-up occurs, the program will resume execution at the instruction following the ″HALT″ instruction.

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.



Watchdog Timer

Setting up the various Watchdog Timer options are controlled via the configuration options and two internal registers WDTS and CTRL1. Enabling the Watchdog Timer can be controlled by both a configuration option and the WDTEN bits in the CTRL1 internal register in the Data Memory.

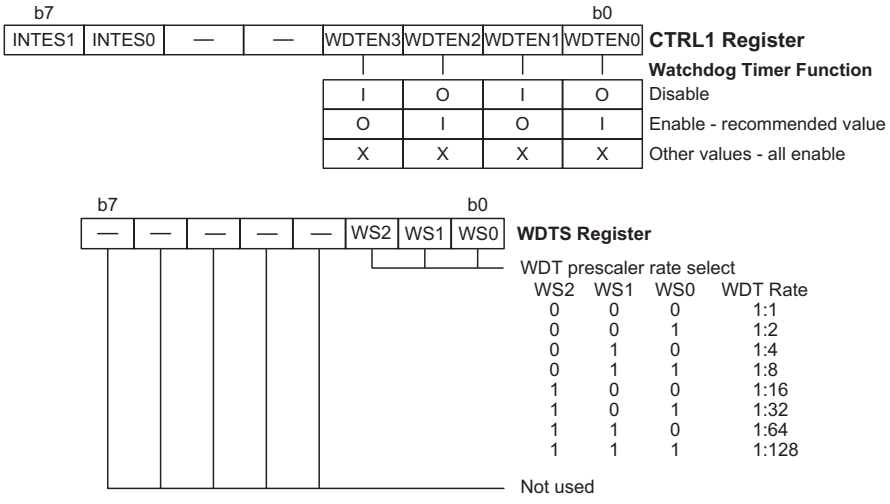
Configuration Option	CTRL1 Register	WDT Function
Disable	Disable	OFF
Enable	Disable	ON
Disable	Enable	ON
Enable	Enable	ON

Watchdog Timer On/Off Control

The Watchdog Timer will be disabled if bits WDTEN3~WDTEN0 in the CTRL1 register are written with the binary value 1010B. This will be the condition when the device is powered up. Although any other data written to WDTEN3~WDTEN0 will ensure that the Watchdog Timer is enabled, for maximum protection it is recommended that the value 0101B is written to these bits.

The Watchdog Timer clock can emanate from three different sources, selected by configuration option. These are its own fully integrated dedicated internal oscillator, the RTC 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 $f_{SYS}/4$ clock and the RTC. 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 or the RTC 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.

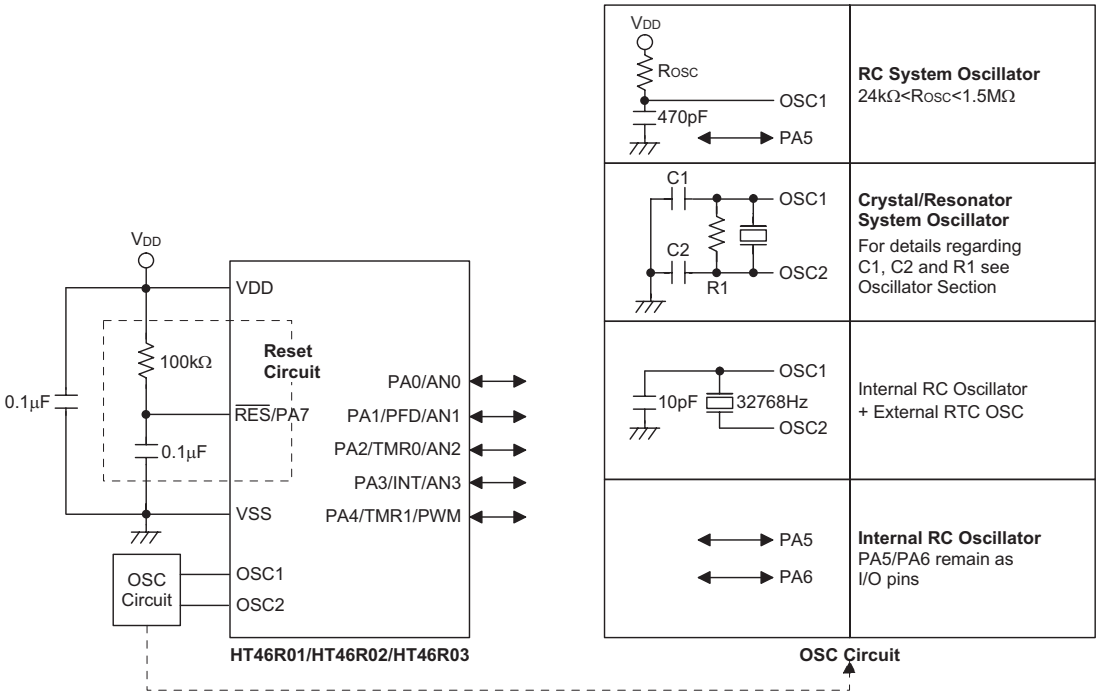


Configuration Options

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
1	Watchdog Timer: enable or disable
2	Watchdog Timer clock source: WDT internal oscillator, f_{SYS}/4 or RTC
3	CLRWDT instructions: 1 or 2 instructions
4	System oscillator: Internal RC, Internal RC with external RTC, External Crystal, External RC
5	LVR function: enable or disable
6	LVR voltage: 2.1V, 3.15V or 4.2V
7	RES or PA7
8	SST: enable or disable

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

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

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

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

Instruction Definition

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

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

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

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

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

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

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

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) C

RR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

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

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

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

Affected flag(s) C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

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

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

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

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

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

SDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation $[m] \leftarrow [m] - 1$
Skip if [m] = 0

Affected flag(s) None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation $ACC \leftarrow [m] - 1$
Skip if ACC = 0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation $[m].i \leftarrow 1$

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m] = 0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC = 0

Affected flag(s) None

SNZ [m].i Skip if bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i $\neq$ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC – [m]

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

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC – [m]

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

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC – x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

Description If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m] = 0

Affected flag(s) None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i = 0

Affected flag(s) None

TABRDC [m] Read table (current page) to TBLH and Data Memory

Description The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

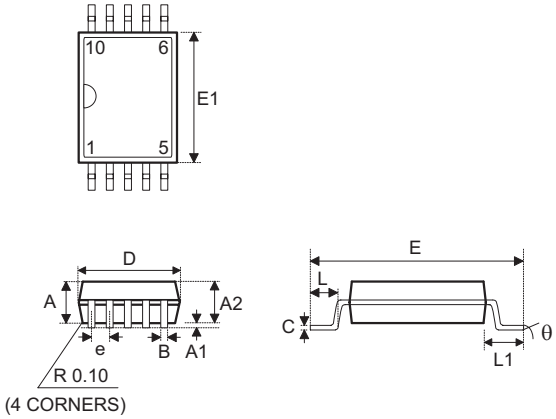
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

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

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

Package Information

10-pin MSOP Outline Dimensions



Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	—	1.1
A1	0	—	0.15
A2	0.75	—	0.95
B	0.17	—	0.27
C	—	—	0.25
D	—	3	—
E	—	4.9	—
E1	—	3	—
e	—	0.5	—
L	0.4	—	0.8
L1	—	0.95	—
θ	0°	—	8°

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

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

Holtek Semiconductor Inc. (Shanghai Sales Office)
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China 200233
Tel: 86-21-6485-5560
Fax: 86-21-6485-0313
http://www.holtek.com.cn

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

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

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

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