Shock Detector Flash MCU

HT45F56

Revision: V1.11 Date: April 11, 2017

www.holtek.com

Table of Contents

Features

CPU Features

- Operating Voltage: f_{SYS}= 8MHz: 2.2V~5.5V
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator Types
  - Internal High Speed RC – HIRC
  - Internal 32kHz RC – LIRC
- Fully integrated internal 8MHz oscillator requires no external components
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one to three instruction cycles
- Table read instructions
- 63 powerful instructions
- 2-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Program Memory: 1K×14
- Data Memory: 32×8
- True EEPROM Memory: 32×8
- Watchdog Timer function
- 6 bidirectional I/O lines
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
- One comparator with hysteresis control and interrupt generation
- One operational amplifier with gain control
- One 6-bit D/A converter
- Debounce circuit for comparator output with interrupt generation
- Dual Time-Base functions for generation of fixed time interrupt signals
- Low voltage reset function
- Flash program memory can be re-programmed up to 100,000 times
- Flash program memory data retention > 10 years
- True EEPROM data memory can be re-programmed up to 1,000,000 times
- True EEPROM data memory data retention > 10 years
- Package types: 8-pin SOP

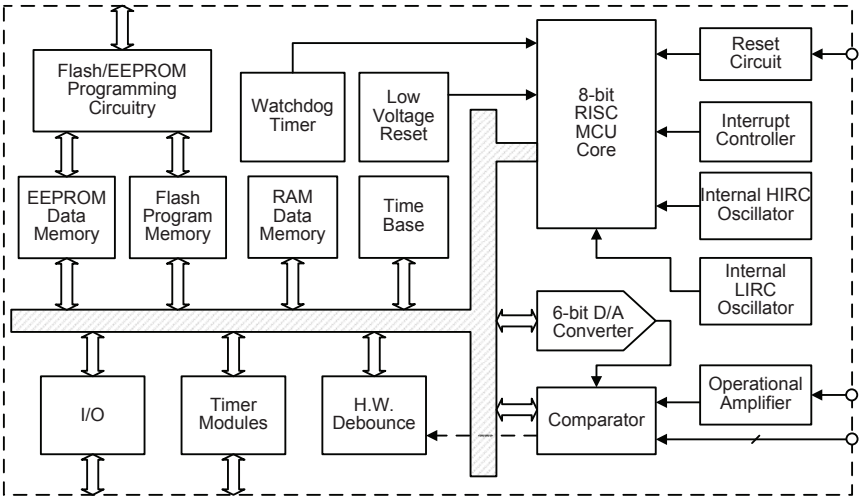
General Description

The device is a Flash Memory type 8-bit high performance RISC architecture microcontroller with fully integrated Shock Sensor which is designed for various product applications. Offering users the convenience of Flash memory multi-programming features, the dvice also includes a wide range of functions amd features. In addition to the flash program memory, other memory includes areas of RAM Data memory and true EEPROM Data memory.

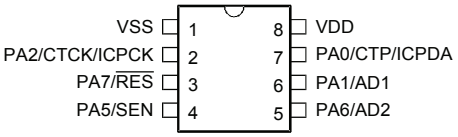
Analog features include a 6-bit DAC, a comparator and an Operational Amplifier. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. The inclusion of flexible I/O programming features, Time Base functions together with many other features further enhance device functionality. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of HIRC and LIRC oscillator functions are provided including a fully integrated system oscillator. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

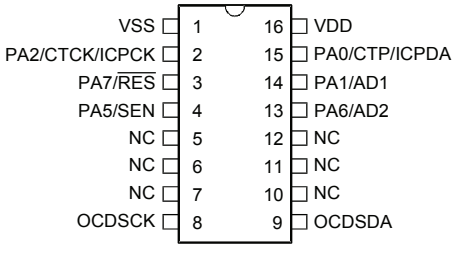
Block Diagram



Pin Assignment



HT45F56
8 SOP-A



HT45V56
16 NSOP-A

Note: The OCDSDA and OCDSCK pins are the OCDS dedicated pins and only available for the HT45V56 device which is the OCDS EV chip for the HT45F56 device.

Pin Descriptions

With the exception of the power pins and some relevant transformer control pins, all pins on these devices can be referenced by their Port name, e.g. PA.0, PA.1 etc, which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Analog to Digital Converter, Timer Module pins etc. The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pad Name	Function	OPT	I/T	O/T	Description
PA0/CTP/ICPDA	PA0	PAWU PAPU PSEL	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	CTP	PSEL	—	CMOS	CTM output
	ICPDA	—	ST	CMOS	ICP Data/Address pin
PA1/AD1	PA1	PAWU PAPU PSEL	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AD1	PSEL	AN	—	Threshold adjustment input
PA2/CTCK/ICPCK	PA2	PAWU PAPU PSEL	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	CTCK	—	ST	—	CTM clock input
	ICPCK	—	ST	CMOS	ICP Clock pin
PA5/SEN	PA5	PAWU PAPU PSEL	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SEN	PSEL	AN	—	Shock sensor analog input
PA6/AD2	PA6	PAWU PAPU PSEL	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AD2	PSEL	AN	—	Threshold adjustment input
PA7/$\overline{\text{RES}}$	PA7	PAPU PSEL	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{\text{RES}}$	RSTC	ST	—	External reset input
OCDSDA	OCDSDA	—	ST	CMOS	OCDS Data/Address pin, for EV chip only.
OCDSCK	OCDSCK	—	ST	—	OCDS Clock pin, for EV chip only.
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground.

Legend: I/T: Input type
O/T: Output type
OPT: Optional by configuration option (CO) or register option
ST: Schmitt Trigger input
CMOS: CMOS output
AN: Analog input
PWR: Power

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
Storage Temperature..... -50°C to 125°C
Operating Temperature..... -40°C to 85°C
I_{OL} Total 80mA
I_{OH} Total ..... -80mA
Total Power Dissipation 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to these devices. Functional operation of these devices at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect devices reliability.

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (HIRC)	—	f_{SYS}=8MHz	V_{LVR}	—	5.5	V
I_{DD}	Operating Current (HIRC)	3V	$f_{SYS}=f_H$=8MHz No load, all peripherals off	—	1.1	1.7	μA
		5V		—	2.9	4.4	mA
		3V	$f_{SYS}=f_H/2$ No load, all peripherals off	—	1.1	1.6	mA
		5V		—	1.8	3.0	mA
		3V	$f_{SYS}=f_H/4$ No load, all peripherals off	—	1.0	1.6	mA
		5V		—	1.7	2.8	mA
		3V	$f_{SYS}=f_H/8$ No load, all peripherals off	—	1.0	1.5	mA
		5V		—	1.5	2.5	mA
		3V	$f_{SYS}=f_H/16$ No load, all peripherals off	—	0.9	1.4	mA
		5V		—	1.4	2.2	mA
		3V	$f_{SYS}=f_H/32$ No load, all peripherals off	—	0.8	1.2	mA
		5V		—	1.3	2.0	mA
		3V	$f_{SYS}=f_H/64$ No load, all peripherals off	—	0.8	1.1	mA
		5V		—	1.1	1.7	mA
	Operating Current (LIRC)	3V	$f_{SYS}=f_{SUB}=f_{LIRC}$=32kHz No load, all peripherals off	—	40	60	μA
		5V		—	90	135	μA
I_{STB}	Standby Current (IDLE0 Mode)	3V	f_{SYS} off, f_{SUB} on No load, all peripherals off	—	3	5	μA
		5V		—	5	10	μA
	Standby Current (IDLE1 Mode)	3V	$f_{SYS}=f_{HIRC}$=8MHz on, f_{SUB} on No load, all peripherals off	—	0.8	1.6	mA
		5V		—	1.0	2.0	mA
	Standby Current (SLEEP0 Mode)	3V	LIRC off, WDT disable No load, all peripherals off	—	0.1	1.0	μA
		5V		—	0.3	2.0	μA
	Standby Current (SLEEP1 Mode)	3V	LIRC on, WDT enable No load, all peripherals off	—	1.3	5.0	μA
		5V		—	2.2	10.0	μA
V_{IL}	Input Low Voltage for I/O Ports or Input Pins except $\overline{RES}$ pin	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	V
	Input Low Voltage for $\overline{RES}$ pin	—	—	0	—	$0.4V_{DD}$	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IH}	Input High Voltage for I/O Ports or Input Pins except $\overline{RES}$ pin	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	V
	Input High Voltage for $\overline{RES}$ pin	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	Sink Current for I/O Port	3V	V_{OL} = $0.1V_{DD}$	8	16	—	mA
		5V		16	32	—	mA
I_{OH}	Source Current for I/O Port	3V	V_{OH} = $0.9V_{DD}$	-4.0	-8.0	—	mA
		5V		-8.0	-16.0	—	mA
R_{PH}	Pull-high Resistance for I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
f_{SYS}	System Clock (HIRC)	2.2~5.5V	$f_{SYS}=f_{HIRC}$=8MHz	—	8	—	MHz
	System Clock (LIRC)	2.2~5.5V	$f_{SYS}=f_{LIRC}$=32kHz	—	32	—	kHz
f_{HIRC}	High Speed Internal RC Oscillator (HIRC)	3V/5V	Ta=25°C	-2%	8	+2%	MHz
		3V/5V	Ta=0°C ~ 70°C	-5%	8	+5%	MHz
		2.2V~5.5V	Ta=0°C ~ 70°C	-8%	8	+8%	MHz
		2.2V~5.5V	Ta= -40°C to 85°C	-12%	8	+12%	MHz
f_{LIRC}	Low Speed Internal RC Oscillator (LIRC)	2.2V~5.5V	Ta= -40°C to 85°C	4	32	80	kHz
t_{TCK}	CTCK pin Minimum Input Pulse Width	—	—	0.3	—	—	μs
t_{RES}	External Reset Minimum Input Pulse Width	—	—	10	—	—	μs
t_{SST}	System Start-up Timer Period (Wake-up from Power Down Mode and f_{SYS} off)	—	$f_{SYS}=f_H=f_{HIRC}$	16	—	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	2	—	—	t_{LIRC}
	System Start-up Timer Period (Wake-up from Power Down Mode and f_{SYS} on)	—	$f_{SYS}=f_H=f_{HIRC}$	2	—	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	2	—	—	t_{SUB}
t_{RSTD}	System Reset Delay Time (Power-on Reset, LVR Hardware Reset, WDTC/RSTC Software Reset)	—	—	25	50	150	ms
	System Reset Delay Time ($\overline{RES}$ Reset / WDT Hardware Reset)	—	—	8.3	16.7	50	ms
t_{EERD}	EEPROM Read Time	—	—	—	—	4	t_{SYS}
t_{EEWR}	EEPROM Write Time	—	—	—	3	6	ms

Note: 1. t_{SYS}= 1/f_{SYS}.

2. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS and located as close to the device as possible.

LVR Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	- 5%	2.1	+ 5%	V
t_{BGS}	V_{BG} Turn on Stable Time	—	No load	—	—	150	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	625	μs

Comparator Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	4.5	—	5.5	V
V_{CM}	Common Mode Voltage	—	—	V_{SS}	—	V_{DD}-1.4	V
V_{OS}	Input Offset Voltage	—	—	-15	—	15	mV
I_{CMP}	Additional Current Consumption for Comparator Enabled	5V	—	—	100	130	μA
I_{PD}	Power Down Current	—	Comparator disabled	—	—	0.1	μA
t_{RP}	Response Time	5V	with 100mV overdrive, C_{LOAD}=3pF	—	—	2	μs

R-2R D/A Converter Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	4.5	—	5.5	V
I_{DAC}	Additional Current Consumption for D/A Converter Enabled	5V	—	—	200	250	μA
DNL	Differential Non-Linearity	5V	—	—	—	±0.5	LSB
INL	Integral Non-Linearity	5V	—	—	—	±1	LSB
R_O	R-2R Output Resistor	5V	—	—	20	—	kΩ

Shock Sensor Amplifier Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	4.5	—	5.5	V
V_{IN}	Inpuit Voltage	5V	—	—	—	±300	mV
R_S	Sample Rate	5V	—	—	—	12	kHz
A_V	DC Gain	5V	—	—	—	1.0	V/mV
f_{IN}	Input Frequency	5V	—	0	—	1.6	kHz
I_{OPA}	Additional Current Consumption for Amplifier Enabled	5V	—	—	—	500	μA
I_{PD}	Power Down Current	—	Amplifier disabled	—	—	2	μA

Debounce Circuit Electrical Characteristics

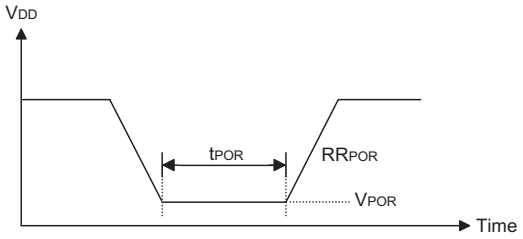
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	4.5	—	5.5	V
t_{DEB}	Debounce Time	5V	DSTAG[2:0]=001B	0.0575	0.125	0.255	ms
			DSTAG[2:0]=010B	0.12	0.25	0.505	ms
			DSTAG[2:0]=011B	0.245	0.5	1.005	ms
			DSTAG[2:0]=100B	0.495	1.0	2.005	ms
			DSTAG[2:0]=101B	0.995	2.0	4.005	ms
			DSTAG[2:0]=110B	1.995	4.0	8.005	ms
			DSTAG[2:0]=111B	1.995	4.0	8.005	ms

Power-on Reset Characteristics

Ta = 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{POR}	V_{DD} Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR_{POR}	V_{DD} Rising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for V_{DD} Stays at V_{POR} to Ensure Power-on Reset	—	—	1	—	—	ms



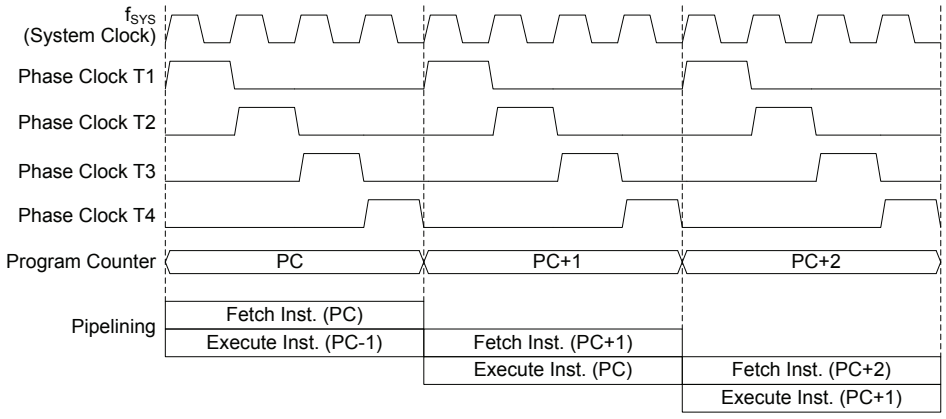
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their 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 instruction set operations, which 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.

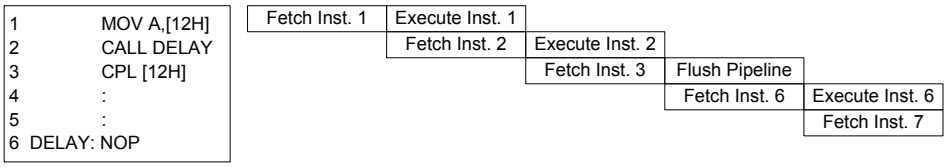
Clocking and Pipelining

The main system clock, derived from either a HIRC or LIRC 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



Instruction Featching

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. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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.

Program Counter	
High Byte	**Low Byte (PCL)**
PC9~PC8	PC7~PC0

Program Counter

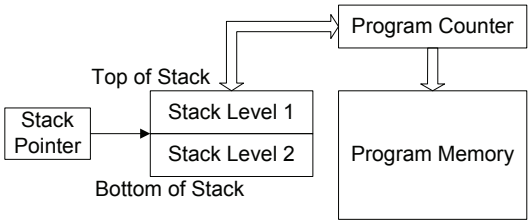
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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. Manipulating the PCL register may cause program branching, so an extra cycle is needed to pre-fetch.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack has multiple levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, 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.

If the stack is overflow, the first Program Counter save in the stack will be lost.



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:

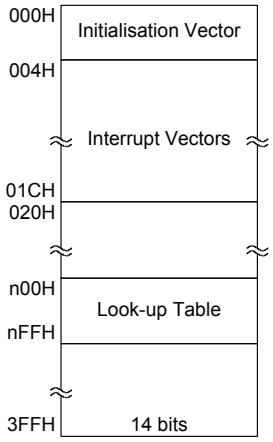
- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement:
  INCA, INC, DECA, DEC
- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA,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

Flash Program Memory

The Program Memory is the location where the user code or program is stored. For the device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, these Flash devices offer users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

Structure

The Program Memory has a capacity of 1K×14 bits. 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 a separate table pointer registers.



Program Memory Structure

Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by these devices reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

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 address of the look up data to be retrieved in the table pointer register, TBLP and TBHP. These registers define the total address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the Program Memory using the “TABRD [m]” or “TABRDL [m]” instructions respectively. When the instruction is 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 accompanying diagram illustrates the addressing data flow of the look-up table.

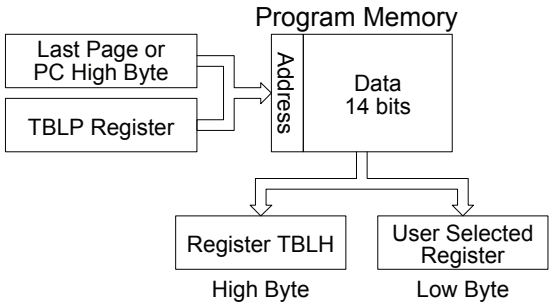


Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. 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 "300H" which refers to the start address of the last page within the 1K Program Memory of the device. The table pointer low byte register 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 "306H" 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 "TABRD [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 "TABRD [m] instruction is executed.

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 table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the 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.

Table Read Program Example

```
tempreg1 db ?          ; temporary register #1
tempreg2 db ?          ; temporary register #2
 :
mov a,06h              ; initialise low table pointer - note that this address is referenced
mov tblp,a             ; to the last page or current page
:
tabrd tempreg1         ; transfers value in table referenced by table pointer data at program
                       ; memory address "306H" transferred to tempreg1 and TBLH
dec tblp               ; reduce value of table pointer by one
tabrd tempreg2         ; transfers value in table referenced by table pointer data at program
                       ; memory address "305H" transferred to tempreg2 and TBLH in this
                       ; example the data "1AH" is transferred to tempreg1 and data "0FH" to
                       ; register tempreg2
:
org 300h               ; sets initial address of program memory
dc      00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
```

In Circuit Programming – ICP

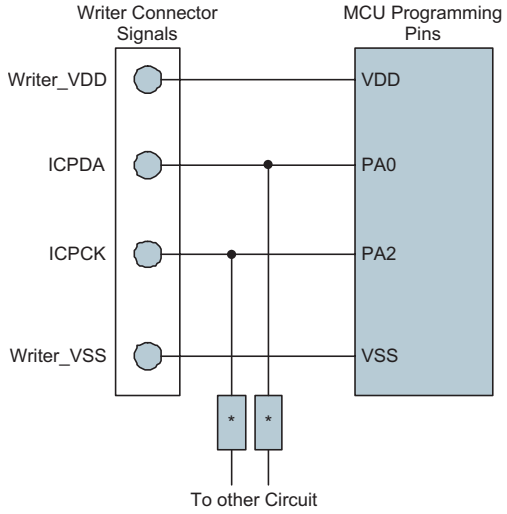
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VDD	VDD	Power Supply
VSS	VSS	Ground

The Program Memory and EEPROM data memory can be programmed serially in-circuit using this 4-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply. The technical details regarding the in-circuit programming of the device are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process, the user must take care of the ICPDA and ICPCK pins for data and clock programming purposes to ensure that no other outputs are connected to these two pins.



Note: * may be resistor or capacitor. The resistance of * must be greater than 1k or the capacitance of * must be less than 1nF.

On-Chip Debug Support – OCDS

There is an EV chip named HT45V56 which is used to emulate the real MCU device named HT45F56. The EV chip device also provides the "On-Chip Debug" function to debug the real MCU device during development process. The EV chip and real MCU devices, HT45V56 and HT45F56, are almost functional compatible except the "On-Chip Debug" function. Users can use the EV chip device to emulate the real MCU device behaviors by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

Holtek e-Link Pins	EV Chip OCDS Pins	Pin Description
OCDSDA	OCDSDA	On-Chip Debug Support Data/Address input/output
OCDSCK	OCDSCK	On-Chip Debug Support Clock input
VDD	VDD	Power Supply
VSS	VSS	Ground

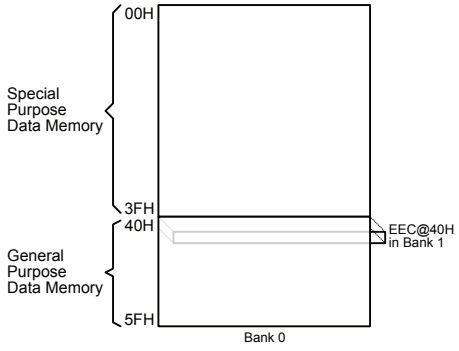
Data Memory

The Data Memory is an 8-bit wide RAM internal memory and is the location where temporary information is stored.

Structure

Divided into two types, the first of Data Memory 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.

The overall Data Memory is subdivided into two banks. The Special Purpose Data Memory registers are accessible in all banks, with the exception of the EEC register at address 40H, which is only accessible in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory for all devices is the address 00H.



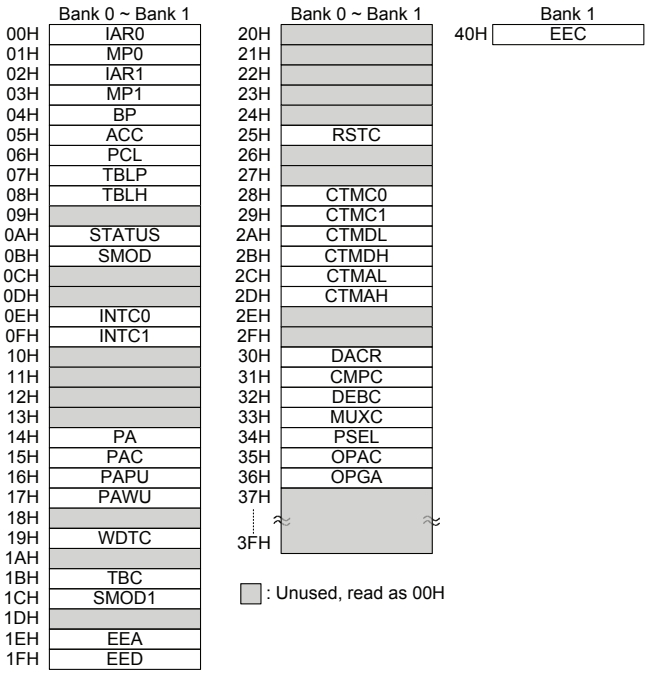
Data Memory Structure

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 programing for both reading and writing operations. By using the bit operation 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 writeable 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”.



Speciap Purpose Data Memory Structure

Special Function Register Description

Most of the Special Function Register details will be described in the relevant functional section. However, several registers require a separate description in this section.

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 Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data only from Bank 0 while the IAR1 register together with MP1 register can access data from any Data Memory bank. 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.

Memory Pointers – MP0, MP1

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 Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data from Bank 0 while the IAR1 and MP1 register pair can access data from any bank. 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.

Indirect Addressing Program Example

- **Example 1**

```
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 mp0,a             ; setup memory pointer with first RAM address
loop:
clr IAR0              ; clear the data at address defined by MP0
inc mp0               ; 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 RAM addresses.

Bank Pointer – BP

The Data Memory is divided into two banks, Bank 0 and Bank 1. Selecting the required Data Memory area is achieved using the Bank Pointer, BP. The Bank Pointer bit 0 is used to select Data Memory Bank 0 or Bank 1.

The Data Memory initialised to Bank 0 after a reset except for a WDT tome-out reset in the Power Down Mode in which case the Data Memory bank remains unaffected. It should be noted that the Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed irrespective of the value of the Bank Pointer. Accessing data from Bank 1 must be implemented using the Indirect Addressing.

BP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	DMBP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **DMBP0**: Data Memory Bank Point bit 0
0: Bank 0
1: Bank 1

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 three 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. Their value must be setup before any table read commands are executed. Their 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.

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 subroutine can corrupt the status register, precautions must be taken to correctly save it.

STATUS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R	R	R/W	R/W	R/W	R/W
POR	—	—	0	0	x	x	x	x

"x": unknown

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TO**: Watchdog Time-out flag
0: After power up ow executing the "CLR WDT" or "HALT" instruction
1: A watchdog time-out occurred

Bit 4 **PDF**: Power down flag
0: After power up ow executing the "CLR WDT" instruction
1: By executing the "HALT" instructin

Bit 3 **OV**: Overflow flag
0: No overflow
1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa

Bit 2 **Z**: Zero flag
0: The result of an arithmetic or logical operation is not zero
1: The result of an arithmetic or logical operation is zero

Bit 1 **AC**: Auxiliary flag
0: No auxiliary carry
1: 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 substraction

Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

The "C" flag is also affected by a rotate through carry instruction.

EEPROM Data Memory

These devices contain an area of internal EEPROM Data Memory. EEPROM, which stands for Electrically Erasable Programmable Read Only Memory, is by its nature a non-volatile form of re-programmable memory, with data retention even when its power supply is removed. By incorporating this kind of data memory, a whole new host of application possibilities are made available to the designer. The availability of EEPROM storage allows information such as product identification numbers, calibration values, specific user data, system setup data or other product information to be stored directly within the product microcontroller. The process of reading and writing data to the EEPROM memory has been reduced to a very trivial affair.

Capacity	Address
32 x 8	00H ~ 1FH

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 32×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and data register in Bank 0 and a single control register in Bank 1.

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in sector 0, they can be directly accessed in the same was as any other Special Function Register. The EEC register, however, being located in sector 1, can be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the Memory Pointer low byte register, MP1, must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEA	—	—	—	EEA4	EEA3	EEA2	EEA1	EEA0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM Registers List

EEA Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	EEA4	EEA3	EEA2	EEA1	EEA0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as 0.

Bit 4~0 **EEA4~EEA0**: Data EEPROM address bit 4 ~ bit0

EED Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**: Data EEPROM data bit 7~bit0

EEC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	WREN	WR	RDEN	RD
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as 0.

Bit 3 **WREN**: Data EEPROM write enable
0: Disable
1: Enable
This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2 **WR**: EEPROM write control
0: Write cycle has finished
1: Activate a write cycle
This is the Data EEPROM Write Control Bit and when set high by the application program will activate a write cycle. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.

Bit 1 **RDEN**: Data EEPROM read enable
0: Disable
1: Enable
This is the Data EEPROM Read Enable Bit which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.

Bit 0 **RD**: EEPROM read control
0: Read cycle has finished
1: Activate a read cycle
This is the Data EEPROM Read Control Bit and when set high by the application program will activate a read cycle. This bit will be automatically reset to zero by the hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

Note: The WREN, WR, RDEN and RD can not be set to "1" at the same time in one instruction. The WR and RD can not be set to "1" at the same time.

Reading Data from the EEPROM

To read data from the EEPROM, the read enable bit, RDEN, in the EEC register must first be set high to enable the read function. The EEPROM address of the data to be read must then be placed in the EEA register. If the RD bit in the EEC register is now set high, a read cycle will be initiated. Setting the RD bit high will not initiate a read operation if the RDEN bit has not been set. When the read cycle terminates, the RD bit will be automatically cleared to zero, after which the data can be read from the EED register. The data will remain in the EED register until another read or write operation is executed. The application program can poll the RD bit to determine when the data is valid for reading.

Writing Data to the EEPROM

To write data to the EEPROM, the EEPROM address of the data to be written must first be placed in the EEA register and the data placed in the EED register. Then the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed consecutively. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set again after the write cycle has started. Note that setting the WR bit high will not initiate a write cycle if the WREN bit has not been set. As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended.

Write Protection

Protection against inadvertent write operation is provided in several ways. After the device is powered on, the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on the Bank Pointer register, BP, will be reset to zero, which means that Data Memory Bank 0 will be selected. As the EEPROM control register is located in Bank 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

EEPROM Interrupt

The EEPROM write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. When an EEPROM write cycle ends, the DEF request flag will be set. If the global, EEPROM interrupts are enabled and the stack is not full, a subroutine call to the EEPROM Interrupt vector will take place. When the interrupt is serviced, the EEPROM interrupt flag DEF will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be Periodic by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Bank Pointer could be normally cleared to zero as this would inhibit access to Bank 1 where the EEPROM control register exist. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process. When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write cycle is executed and then re-enabled after the write cycle starts. Note that the device should not enter the IDLE or SLEEP mode until the EEPROM read or write operation is totally complete. Otherwise, the EEPROM read or write operation will fail.

Programming Example

- **Reading data from the EEPROM – polling method**

```
MOV  A, EEPROM_ADRES      ; user defined address
MOV  EEA, A
MOV  A, 040H              ; setup memory pointer MP1
MOV  MP1, A               ; MP1 points to EEC register
MOV  A, 01H               ; setup Bank Pointer BP
MOV  BP, A
SET  IAR1.1               ; set RDEN bit, enable read operations
SET  IAR1.0               ; start Read Cycle - set RD bit
BACK:
SZ   IAR1.0               ; check for read cycle end
JMP  BACK
CLR  IAR1                 ; disable EEPROM write
CLR  BP
MOV  A, EED               ; move read data to register
MOV  READ_DATA, A
```

- **Writing Data to the EEPROM – polling method**

```
MOV  A, EEPROM_ADRES      ; user defined address
MOV  EEA, A
MOV  A, EEPROM_DATA       ; user defined data
MOV  EED, A
MOV  A, 040H              ; setup memory pointer MP1
MOV  MP1, A               ; MP1L points to EEC register
MOV  A, 01H               ; setup Bank Pointer BP
MOV  BP, A
CLR  EMI
SET  IAR1.3               ; set WREN bit, enable write operations
SET  IAR1.2               ; start Write Cycle - set WR bit
SET  EMI
BACK:
SZ   IAR1.2               ; check for write cycle end
JMP  BACK
CLR  IAR1                 ; disable EEPROM write
CLR  BP
```

Oscillator

Various oscillator types offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving. Oscillator selections and operation are selected through a combination of application program and relevant control registers.

Oscillator Overview

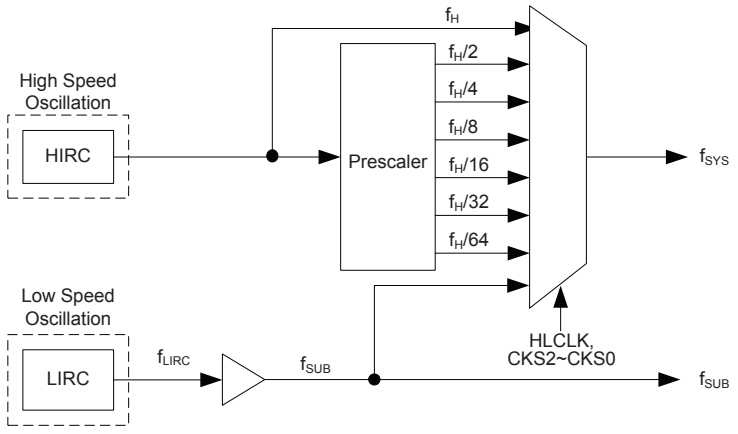
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carries with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

Type	Name	Frequency
Internal High Speed RC	HIRC	8 MHz
Internal Low Speed RC	LIRC	32 kHz

Oscillator Types

System Clock Configurations

There are two methods of generating the system clock, one high speed oscillator and one low speed oscillator. The high speed oscillator is the internal 8 MHz RC oscillator, HIRC. The low speed oscillator is the internal 32 kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK and CKS2~CKS0 bits in the SMOD register and as the system clock can be dynamically selected. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.



System Clock Configurations

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 8 MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 8MHz will have a tolerance within 2%.

Internal 32kHz Oscillator – LIRC

The Internal 32 kHz System Oscillator is the low frequency oscillator choices and fully integrated with a typical frequency of 32 kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Supplementary Oscillators

The low speed oscillator, in addition to providing a system clock source is also used to provide a clock source to two other device functions. These are the Watchdog Timer and the Time Base Interrupts.

Operating Modes and System Clocks

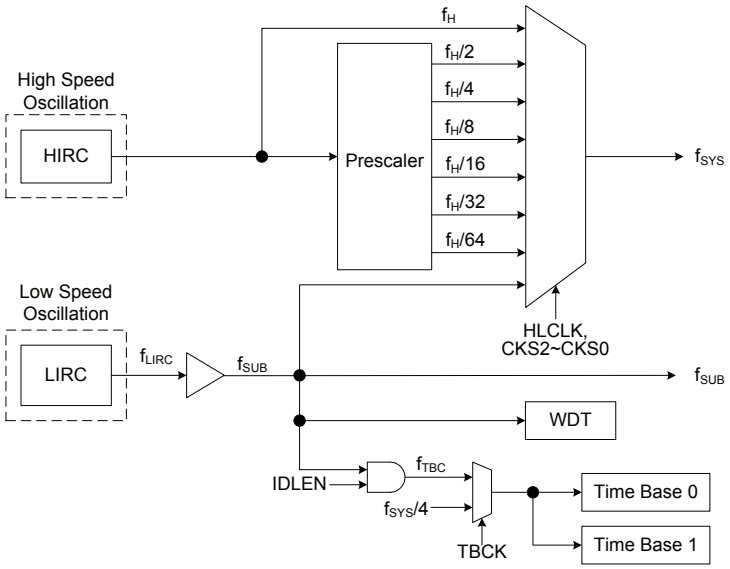
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice-versa, lower speed clocks reduce current consumption. As Holtek has provided these devices with both high and low speed clock sources and the means to switch between them dynamically, the user can optimise the operation of their microcontroller to achieve the best performance/power ratio.

System Clocks

The device has different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock selections using register programming, a clock system can be configured to obtain maximum application performance.

The main system clock can come from either a high frequency f_H or low frequency f_{SUB} source and is selected using the HLCLK and CKS2~CKS0 bits in the SMOD register. The high speed system clock is sourced from the HIRC oscillator while the low speed system clock source is sourced from the internal clock f_{SUB}. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.

There is one additional internal clock for the peripheral circuits, the Time Base clock, f_{TBC}. The f_{TBC} clock is sourced from the LIRC oscillator and used as a source for the Time Base interrupt functions and for the TM.



Device Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillation can be stopped to conserve the power. Therefore, there is no f_H~f_H/64 clock source for peripheral circuit to use.

System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the NORMAL Mode and SLOW Mode. The remaining four modes, the SLEEP0, SPEEL1, IDLE0 and IDLE1 Modes are used when the microcontroller CPU is switched off to conserve power.

Operation Mode	CPU	f_{SYS}	f_{SUB}	f_{TBC}
NORMAL	On	f_H~f_H/64	On	On
SLOW	On	f_{SUB}	On	On
IDLE0	Off	Off	On	On
IDLE1	Off	On	On	On
SLEEP0	Off	Off	Off	Off
SLEEP1	Off	Off	On	Off

NORMAL Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from the high speed oscillators, HIRC. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 and HLCLK bits in the SMOD register.Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from the LIRC oscillator. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW mode, the f_H is switched off.

SLEEP0 Mode

The SLEEP0 Mode is entered when an HALT instruction is executed and when the IDLEN bit is low. In the SLEEP0 mode the CPU will be stopped and the f_{SUB} clock will also be stopped as the Watchdog Timer function is disabled.

SLEEP1 Mode

The SLEEP1 Mode is entered when an HALT instruction is executed and when the IDLEN bit is low. In the SLEEP1 mode the CPU will be stopped. However, the f_{SUB} clock will continue to run as the Watchdog Timer function is enabled.

IDLE0 Mode

The IDLE0 Mode is entered when an HALT instruction is executed and when the IDLEN bit is high and the FSYSON bit is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU but some peripheral functions will remain operational such as the Watchdot Timer and TMs. In the IDLE0 mode, the system oscillator will be stopped.

IDLE1 Mode

The IDLE1 Mode is entered when an HALT instruction is executed and when the IDLEN bit is high and the FSYSON bit is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU but may contimue to provide a clock source to keep some peripheral functions operational such as the Watchdot Timer and TMs. In the IDLE1 mode, the system oscillator will continue to run and this system oscillator mayt be high or low speed system oscillator. In IELD1 mode, the Watchdog Timer clock, f_{SUB}, will be switched on.

Control Registers

The register, SMOD, is used for overall control of the internal clocks within the device.

SMOD Register

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	—	R	R	R/W	R/W
POR	0	0	0	—	0	0	1	1

Bit 7~5 **CKS2~CKS0**: System clock selection

000: f_{SUB}
001: f_{SUB}
010: $f_H/64$
011: $f_H/32$
100: $f_H/16$
101: $f_H/8$
110: $f_H/4$
111: $f_H/2$

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source, which can be the f_{SUB} clock source, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as 0.

Bit 3 **LTO**: Low speed system oscillator ready flag
0: Not ready
1: Ready
This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP0 mode, but after a wake-up has occurred the flag will change to a high level after 1~2 cycles as the LIRC oscillator is used.

Bit 2 **HTO**: High speed system oscillator ready flag
0: Not ready
1: Ready
This is the high speed system oscillator ready flag which indicates when the high speed system oscillator is stable. This flag is cleared to "0" by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable. Therefore this flag will always be read as "1" by the application program after device power-on.

Bit 1 **IDLEN**: IDLE mode control
0: Disable
1: Enable
This is the IDLE mode control bit and determines what happens when the HALT instruction is executed. If this bit is high, when a HALT instruction is executed the device will enter the IDLE Mode. In the IDLE1 Mode, the CPU will stop running but the system clock will continue to keep the peripheral functions operational, as the FSYSON bit is high. If the FSYSON bit is low, the CPU and the system clock will all stop in the IDLE0 mode. If this bit is low, the device will enter the SLEEP Mode when a HALT instruction is executed.

Bit 0 **HLCLK**: System clock selection
0: $f_H/2 \sim f_H/64$ or f_{SUB}
1: f_H
This bit is used to select if the f_H clock, the $f_H/2 \sim f_H/64$ clock or the f_{SUB} clock is used as the system clock. When this bit is high, the f_H clock will be selected and if low the $f_H/2 \sim f_H/64$ or f_{SUB} clock will be selected. When the system clock switches from the f_H clock to the f_{SUB} clock and the f_H clock will be automatically switched off to conserve power.

SMOD1 Register

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	RSTF	LVRF	—	WRF
R/W	R/W	—	—	—	R/W	R/W	—	R/W
POR	0	—	—	—	0	x	—	0

"x": unknown

Bit 7 **FSYSON**: f_{SYS} control in IDLE mode
0: Disable
1: Enable

Bit 6~4 Unimplemented, read as 0.

Bit 3 **RSTF**: Reset control register software reset flag
Described elsewhere.

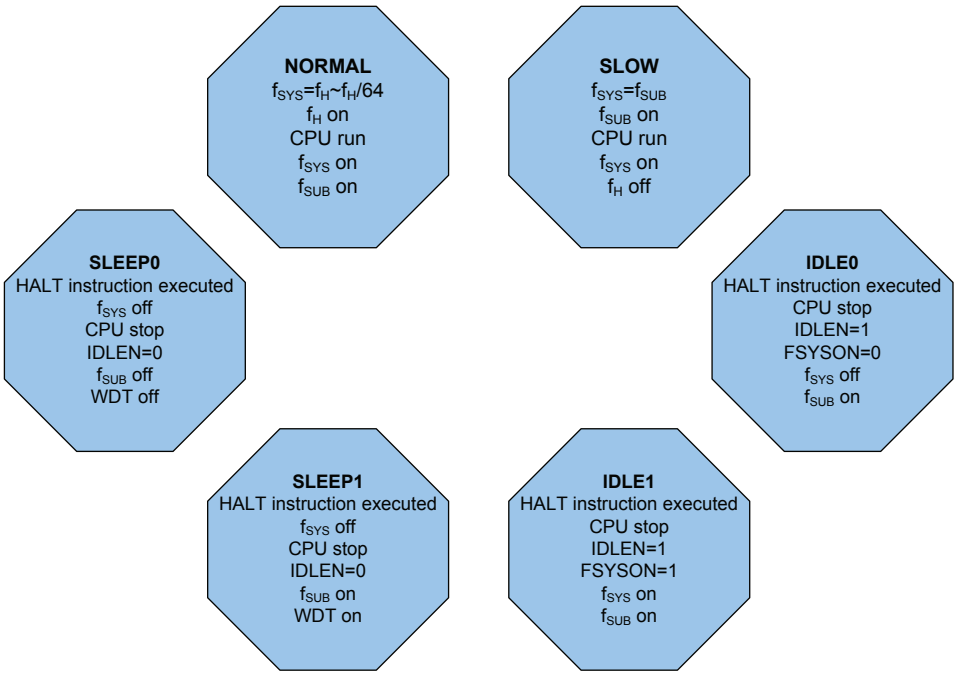
Bit 2 **LVRF**: LVR function reset flag
Described elsewhere.

Bit 1 Unimplemented, read as 0.

Bit 0 **WRF**: WDT control register software reset flag
Described elsewhere.

Operating Mode Switching

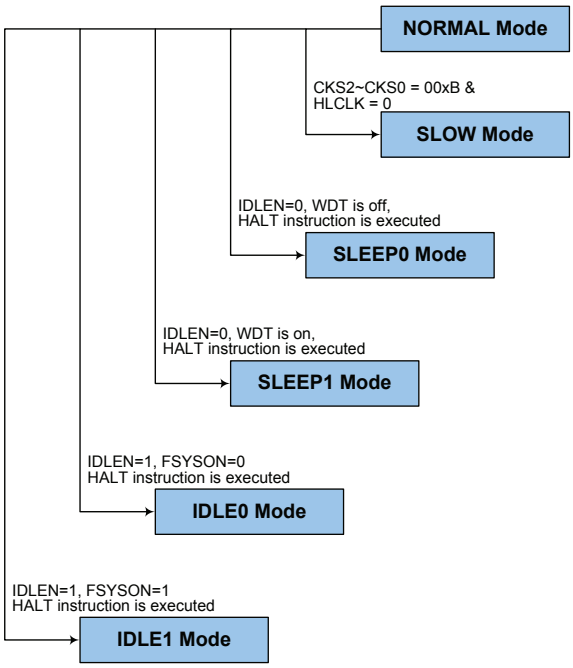
The devices can switch between operating modes dynamically allowing the user to select the best performance/power ratio for the present task in hand. In this way microcontroller operations that do not require high performance can be executed using slower clocks thus requiring less operating current and prolonging battery life in portable applications. In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/ SLOW Modes to the SLEEP/ IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SMOD register and FSYSON in the SMOD1 register. When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, fH, to the clock source, $f_H/2$~ $f_H/64$ or f_{SUB}. If the clock is from the f_{SUB}, the high speed clock source will stop running to conserve power. When this happens it must be noted that the $f_H/16$ and $f_H/64$ internal clock sources will also stop running, which may affect the operation of other internal functions such as the TMs.



NORMAL Mode to SLOW Mode Switching

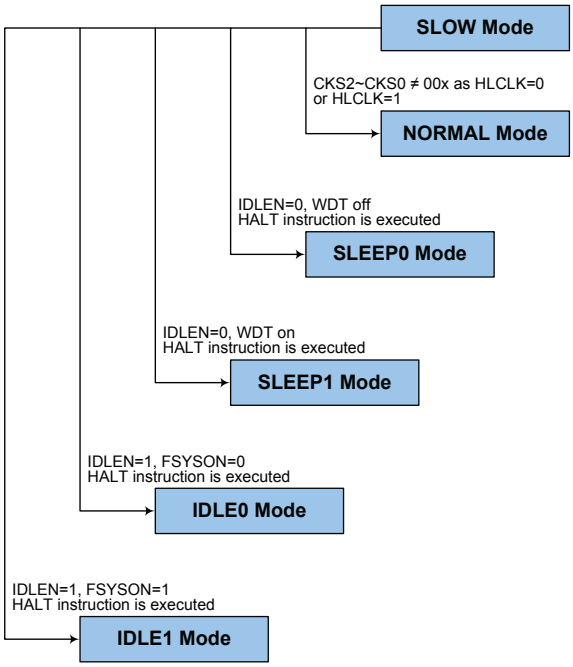
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by setting the HLCLK bit to "0" and setting the CKS2~CKS0 bits to "000" or "001" in the SMOD register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs. This is monitored using the LTO bit in the SMOD register.



SLOW Mode to NORMAL Mode Switching

In SLOW mode the system clock is derived from f_{SUB}. To switch back to the NORMAL Mode, where the high speed system oscillator is used, the HLCLK bit should be set to "1" or HLCLK bit is "0" but the CKS2~CKS0 field is set to "01x" or "1xx". As a certain amount of time will be required for the high frequency clock to stablise, the status of the HTO bit is checked.



Entering the SLEEP0 Mode

There is only one way for the device to enter the SLEEP0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in the SMOD register equal to "0" and the WDT is off. When this instruction is executed under the conditions described above, the following will occur:

- The system clock, WDT clock and Time Base clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and stopped as the WDT is disabled.

Entering the SLEEP1 Mode

There is only one way for the device to enter the SLEEP1 Mode and that is to execute the “HALT” instruction in the application program with the IDLEN bit in the SMOD register equal to “0” and the WDT is on. When this instruction is executed under the conditions described above, the following will occur:

- The system clock and Time Base clock will be stopped and the application program will stop at the "HALT" instruction. However, the WDT clock will continue to run.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting as the WDT is enabled.

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the “HALT” instruction in the application program with the IDLEN bit in the SMODE register equal to “1” and the FSYSON bit in the SMOD1 register equal to “0”. When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the “HALT” instruction, but the Time Base clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled.

Entering the IDLE1 Mode

There is only one way for the device to enter the IDLE1 Mode and that is to execute the “HALT” instruction in the application program with IDLEN bit in the SMODE register equal to “1” and the FSYSON bit in the SMOD1 register equal to “1”. When this instruction is executed under the conditions described above, the following will occur:

- The system clock and Time Base clock will be on but the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 Mode, 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. This also applies to devices which have different package types, as there may be unbonbed pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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. In the IDLE1 Mode the systen oscillator is on, if the peripheral function clock source is derived from the high speed system oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

To minimise power consumption the device can enter the SLEEP or any IDLE Mode, where the CPU will be switched off. However, when the device is woken up again, it will take a considerable time for the original system oscillator to restart, stablise and allow normal operation to resume.

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

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

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

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake up the system. When a Port A pin wake-up occurs, the program will resume execution at the instruction following the “HALT” instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the “HALT” instruction. In this situation, the interrupt which woke up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

Watchdog Timer

The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise.

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal f_{SUB} clock derived from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32 kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{15} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. This register controls the overall operation of the Watchdog Timer. The WRF software reset flag is used to indicate whether the WDT control register software reset occurs or not.

WDTC Register

Bit	7	6	5	4	3	2	1	0
Name	WE4	WE3	WE2	WE1	WE0	WS2	WS1	WS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	0	1	1

Bit 7~3 **WE4~WE0**: WDT function enable control
10101: Disabled
01010: Enabled
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the SMOD1 register will be set to 1.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^8/f_{SUB}$
001: $2^9/f_{SUB}$
010: $2^{10}/f_{SUB}$
011: $2^{11}/f_{SUB}$
100: $2^{12}/f_{SUB}$
101: $2^{13}/f_{SUB}$
110: $2^{14}/f_{SUB}$
111: $2^{15}/f_{SUB}$
These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period.

SMOD1 Register

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	RSTF	LVRF	—	WRF
R/W	R/W	—	—	—	R/W	R/W	—	R/W
POR	0	—	—	—	0	x	—	0

"x": unknown

Bit 7 **FSYSON**: f_{SYS} control in IDLE mode
Described elsewhere.

Bit 6~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Described elsewhere.

Bit 2 **LVRF**: LVR function reset flag
Described elsewhere.

Bit 1 Unimplemented, read as "0"

Bit 0 **WRF**: WDT control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the WDT control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control and reset control of the Watchdog Timer. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B while the WDT function will be enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after 2~3 f_{LIRC} clock cycles. After power on these bits will have a value of 01010B.

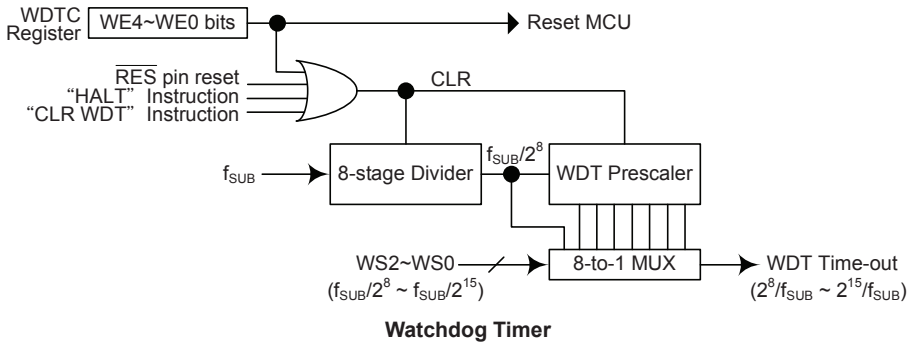
WE4 ~ WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

Watchdog Timer Enable/Disable Control

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 SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Four methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 field, the second is using the Watchdog Timer software clear instruction, the third is using a HALT instruction and the fourth is an external hardware reset.

There is only one method of using software instruction to clear the Watchdog Timer. That is to use the single "CLR WDT" instruction to clear the WDT contents.

The maximum time out period is when the 2^{15} division ratio is selected. As an example, with a 32 kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 1 second for the 2^{15} division ratio and a minimum timeout of 7.8ms for the 2^8 division ration.



Watchdog Timer

Reset and Initialisation

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

In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high.

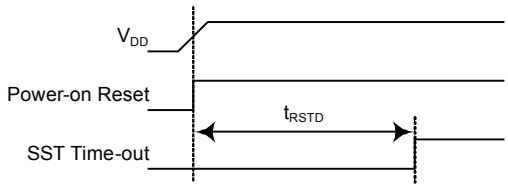
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply 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.



Note: t_{RSTD} is power-on delay with typical time = 50 ms

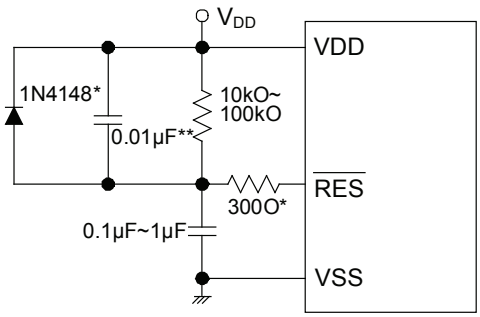
Power-On Reset Timing Chart

$\overline{RES}$ Pin Reset

Although the microcontroller has an internal RC reset function, if the V_{DD} 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.

For most applications a resistor connected between V_{DD} 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 minimize any stray noise interference. For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.

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

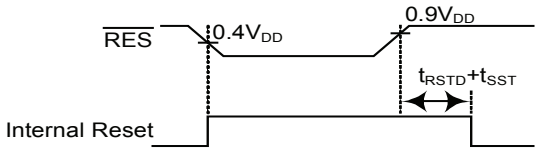


Note: "*" It is recommended that this component is added for added ESD protection.

"**" It is recommended that this component is added in environments where power line noise is significant.

External $\overline{RES}$ Circuit

Pulling the $\overline{RES}$ Pin low using external hardware will also execute a device reset. In this case, as in the case of other resets, the Program Counter will reset to zero and program execution initiated from this point.



Note: t_{RSTD} is power-on delay with typical time = 16.7 ms

$\overline{RES}$ Reset Timing Chart

There is an internal reset control register, RSTC, which is used to provide a reset when the device operates abnormally due to the environmental noise interference. If the content of the RSTC register is set to any value other than 01010101B or 10101010B, it will reset the device after 2~3 f_{LIRC} clock cycles. After power on the register will have a value of 01010101B.

RSTC7 ~ RSTC0 Bits	Reset Function
01010101B	I/O or other functions
10101010B	$\overline{RES}$ function
Any other value	Reset MCU

Reset Function Control

- **RSTC Register**

Bit	7	6	5	4	3	2	1	0
Name	RSTC7	RSTC6	RSTC5	RSTC4	RSTC3	RSTC2	RSTC1	RSTC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7~0 **RSTC7~RSTC0**: Reset function control
01010101: I/O or other functions
10101010: $\overline{RES}$ function
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after 2~3 LIRC clock cycles and the RSTF bit in the SMOD1 register will be set to 1. All reset will reset this register as POR value except the WDT time-out reset.

- **SMOD1 Register**

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	RSTF	LVRF	—	WRF
R/W	R/W	—	—	—	R/W	R/W	—	R/W
POR	0	—	—	—	0	x	—	0

"x": unknown

Bit 7 **FSYSON**: f_{SYS} control in IDLE mode
Described elsewhere.

Bit 6~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the RSTC control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

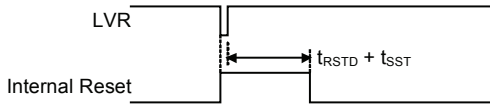
Bit 2 **LVRF**: LVR function reset flag
Described elsewhere.

Bit 1 Unimplemented, read as "0"

Bit 0 **WRF**: WDT control register software reset flag
Described elsewhere.

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 always enabled with a specific LVR voltage, V_{LVR}. 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 and the LVRF bit in the SMOD1 register will also be set to 1. 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 LVR characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value is 2.1V and the LVR circuit will reset the device when the supply voltage is less than 2.1V for more than the t_{LVR} time. Note that the LVR function will be automatically disabled when the device enters the power down mode.



Note: t_{RSTD} is power-on delay with typical time = 50 ms

Low Voltage Reset Timing Chart

- **SMOD1 Register**

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	RSTF	LVRF	—	WRF
R/W	R/W	—	—	—	R/W	R/W	—	R/W
POR	0	—	—	—	0	x	—	0

"x": unknown

Bit 7 **FSYSON**: f_{SYS} control in IDLE mode
Described elsewhere.

Bit 6~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Described elsewhere.

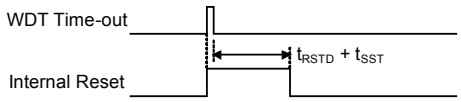
Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set to 1 when a specific low voltage reset condition occurs. Note that this bit can only be cleared to 0 by the application program.

Bit 1 Unimplemented, read as "0"

Bit 0 **WRF**: WDT control register software reset flag
Described elsewhere.

Watchdog Time-out Reset during Normal Operation

The Watchdog time-out Reset during normal operation is the same as the hardware Low Voltage Reset except that the Watchdog time-out flag TO will be set to "1".

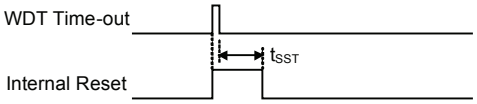


Note: t_{RSTD} is power-on delay with typical time = 16.7 ms

WDT Time-out Reset during NORMAL Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

The Watchdog time-out Reset during SLEEP or IDLE Mode 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 SLEEP or IDLE Mode Timing Chart

Reset Initial Conditions

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

TO	PDF	Reset Function
0	0	Power-on reset
u	u	$\overline{\text{RES}}$ LVR reset during NORMAL or SLOW Mode operation
1	u	WDT time-out reset during NORMAL or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

"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	Reset Function
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Base	Clear after reset, WDT begins counting
Timer Modules	Timer Modules will be turned off
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 the microcontroller internal registers.

Register	Reset (Power On)	$\overline{\text{RES}}$ Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE or SLEEP)*
IAR0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP0	1000 0000	1000 0000	1000 0000	1000 0000	1uuu uuuu
IAR1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	1000 0000	1000 0000	1000 0000	1000 0000	1uuu uuuu
BP	---- ---0	---- ---0	---- ---0	---- ---0	---- ---0
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--uu uuuu	--uu uuuu	--1u uuuu	--11 uuuu
SMOD	000- 0011	000- 0011	000- 0011	000- 0011	uuu- uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
PA	111- -111	111- -111	111- -111	111- -111	uuu- -uuu
PAC	111- -111	111- -111	111- -111	111- -111	uuu- -uuu
PAPU	000- -000	000- -000	000- -000	000- -000	uuu- -uuu
PAWU	000- -000	000- -000	000- -000	000- -000	uuu- -uuu
WDTC	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	0011 -111	0011 -111	0011 -111	0011 -111	uuuu -uuu
SMOD1	0--- 0x-0	0--- uu-u	0--- u1-u	0--- uu-u	u--- uu-u
EEA	---0 0000	---0 0000	---0 0000	---0 0000	---u uuuu
EED	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
CTMC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDH	---- --00	---- --00	---- --00	---- --00	---- --uu
CTMAL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMAH	---- --00	---- --00	---- --00	---- --00	---- --uu
EEC	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
DACR	0-00 0000	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
CMPC	x-00 ---1	x-00 ---1	x-00 ---1	x-00 ---1	u-uu ---u
DEBC	---- -000	---- -000	---- -000	---- -000	---- -uuu
MUXC	--00 --00	--00 --00	--00 --00	--00 --00	--uu --uu
PSEL	--00 0111	--00 0111	--00 0111	--00 0111	--uu uuuu
OPAC	-1-- --00	-1-- --00	-1-- --00	-1-- --00	-u-- --uu
OPGA	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu

Note: “u” stands for unchanged
“x” stands for “unknown”
“-“ stands for unimplemented

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. With the input or output designation of every pin fully under user program control, pull-high selections for all ports and wake-up selections on certain pins, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

The device providea bidirectional input/output lines labeled with port name PA. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. 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.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	—	—	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	—	—	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	—	—	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	—	—	PAWU2	PAWU1	PAWU0

I/O Registers List

"—": Unimplemented, read as "0".

PAn: Port A Data bit

0: Data 0

1: Data 1

PACn: Port A Pin type selection

0: Output

1: Input

PAPUn: Port A Pin pull-high function control

0: Disable

1: Enable

PAWUn: Port A Pin wake-up function control

0: Disable

1: Enable

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, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the relevant pull-high control registers and are implemented using weak PMOS transistors.

Port A Wake-up

The HALT instruction forces the microcontroller into the SLEEP or IDLE Mode which preserves power, 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 Port A pins from high to low. This function is especially suitable for applications that can be woken up via external switches. Each pin on Port A can be selected individually to have this wake-up feature using the PAWU register.

I/O Port Control Register

Each Port has its own control register, known as PAC, which controls the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. 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 these pins, the desired function of the multi-function I/O pins is selected by a register via the application program control.

Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. This device includes pin-shared function selection register, labeled as PSEL, which can select the desired functions of the multi-function pin-shared pins.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. To select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

- **PSEL Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PSEL5	PSEL4	PSEL3	PSEL2	PSEL1	PSEL0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	1	1	1

Bit 7~6 Unimplemented, read as 0

Bit 5 **PSEL5**: PA7 output type
0: CMOS type
1: Open drain type

Bit 4 **PSEL4**: PA2 output type
0: CMOS type
1: Open drain type

Bit 3 **PSEL3**: PA0 pin function selection
0: PA0
1: CTP

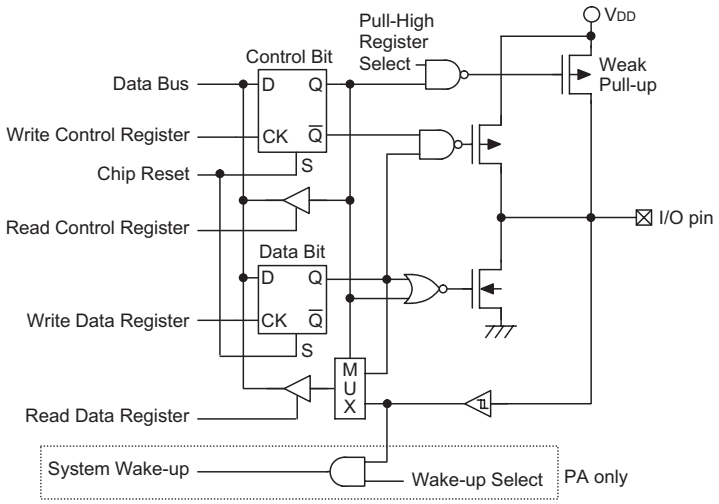
Bit 2 **PSEL2**: PA6 pin function selection
0: PA6
1: AD2

Bit 1 **PSEL1**: PA5 pin function selection
0: PA5
1: SEN

Bit 0 **PSEL0**: PA1 pin function selection
0: PA1
1: AD1

I/O Pin Structures

The accompanying diagrams illustrate the internal structures of some generic I/O pin types. As the exact logical construction of the I/O pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins. The wide range of pin-shared structures does not permit all types to be shown.



Generic Input/Output Structure

Programming Considerations

Within the user program, one of the things first to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set to high. This means that all I/O pins will be defaulted to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate 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.

Port A has the additional capability of providing wake-up functions. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the Port A pins. Single or multiple pins on Port A can be setup to have this function.

Compact Type Timer Module – CTM

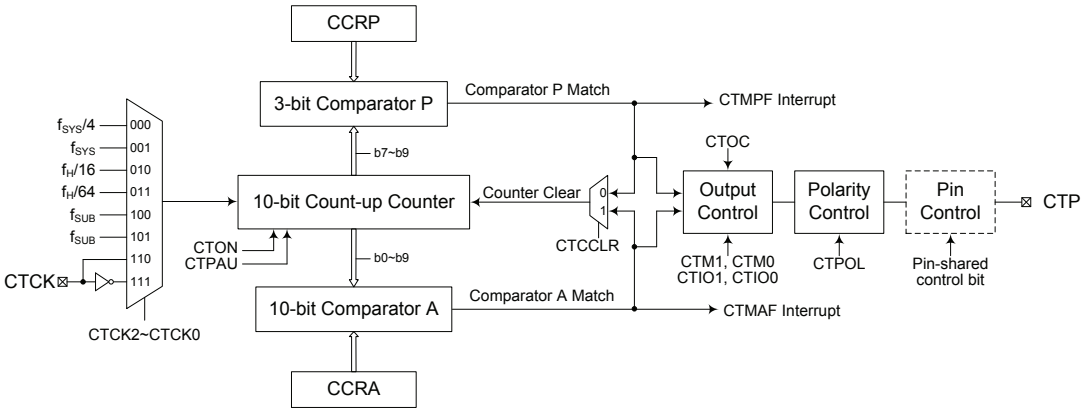
One of the most fundamental functions in any microcontroller devices is the ability to control and measure time. To implement time related functions the device includes one Timer Module, generally abbreviated to the name TM. This TM is the simplest type of the TMs which contains a multi-purpose timing unit. Although the simplest form of the TM types, the Compact TM type still contains three operating modes, which are Compare Match Output, Timer/Event Counter and PWM Output modes. The Compact TM can also be controlled with an external input pin and can drive one external output pin.

The key to understanding how the TM operates is to see it in terms of a free running count-up counter whose value is then compared with the value of pre-programmed internal comparators. When the free running count-up counter has the same value as the pre-programmed comparator, known as a compare match situation, a TM interrupt signal will be generated which can clear the counter and perhaps also change the condition of the TM output pin. The CTM has two interrupts, one for each of the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs. When a TM interrupt is generated, it can be used to clear the counter and also to change the state of the TM output pin.

The internal TM counter is driven by a user selectable clock source, which can be an internal clock or an external pin. The selection of the required clock source is implemented using the CTCK2~CTCK0 bits in the CTM control registers. The clock source can be a ratio of the system clock, f_{SYS}, or the internal high clock, f_H, the f_{SUB} clock source or the external CTCK pin. The CTCK pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

The TM output pin can be selected using the corresponding pin-shared function selection bits described in the Pin-shared Function section. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external CTP output pin is also the pin where the TM generates the PWM output waveform. As the TM output pins are pin-shared with other functions, the TM output function must first be setup using relevant pin-shared function selection register.

CTM Core	CTM Input Pin	CTM Output Pin
10-bit CTM	CTCK	CTP



Compact Type TM Block Diagram

Compact TM Operation

The Compact TM core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is three-bit wide whose value is compared with the highest three bits in the counter while the CCRA is ten-bit wide and therefore compares with all counter bits.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the CTON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a TM interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control an output pin. All operating setup conditions are selected using relevant internal registers.

Compact Type TM Register Description

Overall operation of the Compact TM is controlled using a series of registers. A read only register pair exists to store the internal counter 16-bit value, while a read/write register pair exists to store the internal 10-bit CCRA value. The remaining two registers are control registers which setup the different operating and control modes and as well as the three CCRP bits.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CTMC0	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
CTMC1	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
CTMDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMDH	—	—	—	—	—	—	D9	D8
CTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMAH	—	—	—	—	—	—	D9	D8

10-bit Compact TM Registers List

CTMDL Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 CTM Counter Low Byte Register bit 7 ~ bit 0
CTM 10-bit Counter bit 7 ~ bit 0

CTMDH Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 CTM Counter High Byte Register bit 1 ~ bit 0
CTM 10-bit Counter bit 9 ~ bit 8

CTMAL Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 CTM CCRA Low Byte Register bit 7 ~ bit 0
CTM 10-bit CCRA bit 7 ~ bit 0

CTMAH Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"
Bit 1~0 CTM CCRA High Byte Register bit 1 ~ bit 0
CTM 10-bit CCRA bit 9 ~ bit 8

CTMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CTPAU**: CTM Counter Pause control
0: Run
1: Pause
The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the CTM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **CTCK2~CTCK0**: Select CTM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: CTCK rising edge clock
111: CTCK falling edge clock
These three bits are used to select the clock source for the CTM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **CTON**: CTM Counter On/Off control
0: Off
1: On
This bit controls the overall on/off function of the CTM. Setting the bit high enables the counter to run while clearing the bit disables the CTM. Clearing this bit to zero will stop the counter from counting and turn off the CTM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the CTM is in the Compare Match Output Mode then the CTM output pin will be reset to its initial condition, as specified by the CTOC bit, when the CTON bit changes from low to high.

Bit 2~0 **CTRP2~CTRP0**: CTM CCRP 3-bit register, compared with the CTM Counter bit 9 ~ bit 7
000: 1024 CTM clocks
001: 128 CTM clocks
010: 256 CTM clocks
011: 384 CTM clocks
100: 512 CTM clocks
101: 640 CTM clocks
110: 768 CTM clocks
111: 896 CTM clocks

These three bits are used to setup the value on the internal CCRP 3-bit register, which are then compared with the internal counter's highest three bits. The result of this comparison can be selected to clear the internal counter if the CTCCLR bit is set to zero. Setting the CTCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest three counter bits, the compare values exist in 128 clock cycle multiples. Clearing all three bits to zero is in effect allowing the counter to overflow at its maximum value.

CTMC1 Register

Bit	7	6	5	4	3	2	1	0
Name	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **CTM1~CTM0**: Select CTM Operating Mode
00: Compare Match Output Mode
01: Undefined
10: PWM Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the CTM. To ensure reliable operation the CTM should be switched off before any changes are made to the CTM1 and CTM0 bits. In the Timer/Counter Mode, the CTM output pin control will be disabled.

Bit 5~4 **CTIO1~CTIO0**: Select CTM external pin (CTP) function

Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

PWM Output Mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Undefined

Timer/Counter Mode
Unused

These two bits are used to determine how the CTM output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the CTM is running.

In the Compare Match Output Mode, the CTIO1 and CTIO0 bits determine how the CTM output pin changes state when a compare match occurs from the Comparator A. The CTM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the CTM output pin should be setup using the CTOC bit in the CTMC1 register. Note that the output level requested by the CTIO1 and CTIO0 bits must be different from the initial value setup using the CTOC bit otherwise no change will occur on the CTM output pin when a compare match occurs. After the CTM output pin changes state, it can be reset to its initial level by changing the level of the CTON bit from low to high.

In the PWM Mode, the CTIO1 and CTIO0 bits determine how the CTM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the CTIO1 and CTIO0 bits only after the CTM has been switched off. Unpredictable PWM outputs will occur if the CTIO1 and CTIO0 bits are changed when the CTM is running.

Bit 3 **CTOC**: CTP Output control

Compare Match Output Mode
0: Initial low
1: Initial high

PWM Output Mode
0: Active low
1: Active high

This is the output control bit for the CTM output pin. Its operation depends upon whether CTM is being used in the Compare Match Output Mode or in the PWM Mode. It has no effect if the CTM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the CTM output pin before a compare match occurs. In the PWM Mode it determines if the PWM signal is active high or active low.

Bit 2 **CTPOL**: CTP Output polarity control
0: Non-inverted
1: Inverted

This bit controls the polarity of the CTP output pin. When the bit is set high the CTM output pin will be inverted and not inverted when the bit is zero. It has no effect if the CTM is in the Timer/Counter Mode.

Bit 1 **CTDPX**: CTM PWM duty/period control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

This bit determines which of the CCRA and CCRP registers are used for period and duty control of the PWM waveform.

Bit 0 **CTCCLR**: CTM Counter Clear condition selection
0: CTMn Comparator P match
1: CTMn Comparator A match

This bit is used to select the method which clears the counter. Remember that the Compact TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the CTCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The CTCCLR bit is not used in the PWM Mode.

Compact Type TM Operation Modes

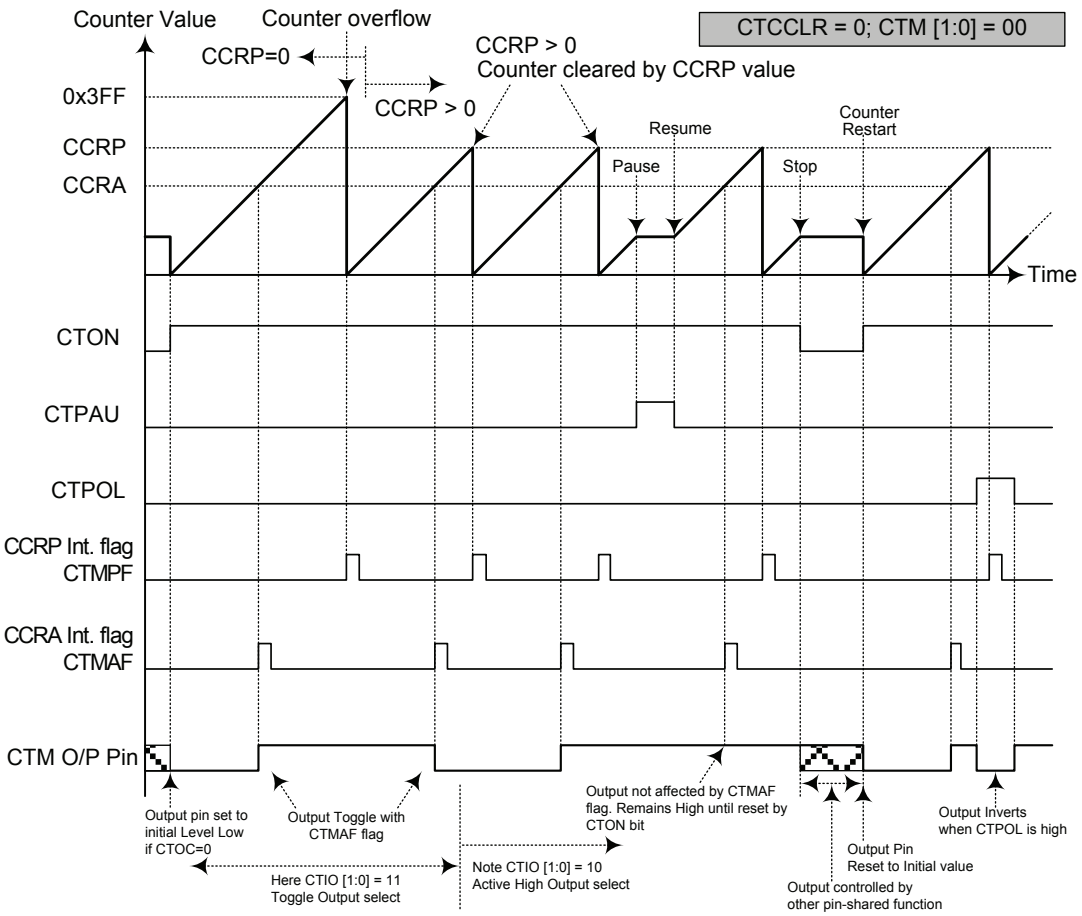
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Mode or Timer/Counter Mode. The operating mode is selected using the CTM1 and CTM0 bits in the CTMC1 register.

Compare Match Output Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 register, should be set to "00" respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match occurs from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMAF and CTMPF interrupt request flags for the Comparator A and Comparator P respectively, will both be generated.

If the CTCCLR bit in the CTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTCCLR is high no CTMPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the CTMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the CTM output pin will change state. The CTM output pin condition however only changes state when a CTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTM output pin. The way in which the CTM output pin changes state are determined by the condition of the CTIO1 and CTIO0 bits in the CTMC1 register. The CTM output pin can be selected using the CTIO1 and CTIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the CTM output pin, which is setup after the CTON bit changes from low to high, is setup using the CTOC bit. Note that if the CTIO1 and CTIO0 bits are zero then no pin change will take place.

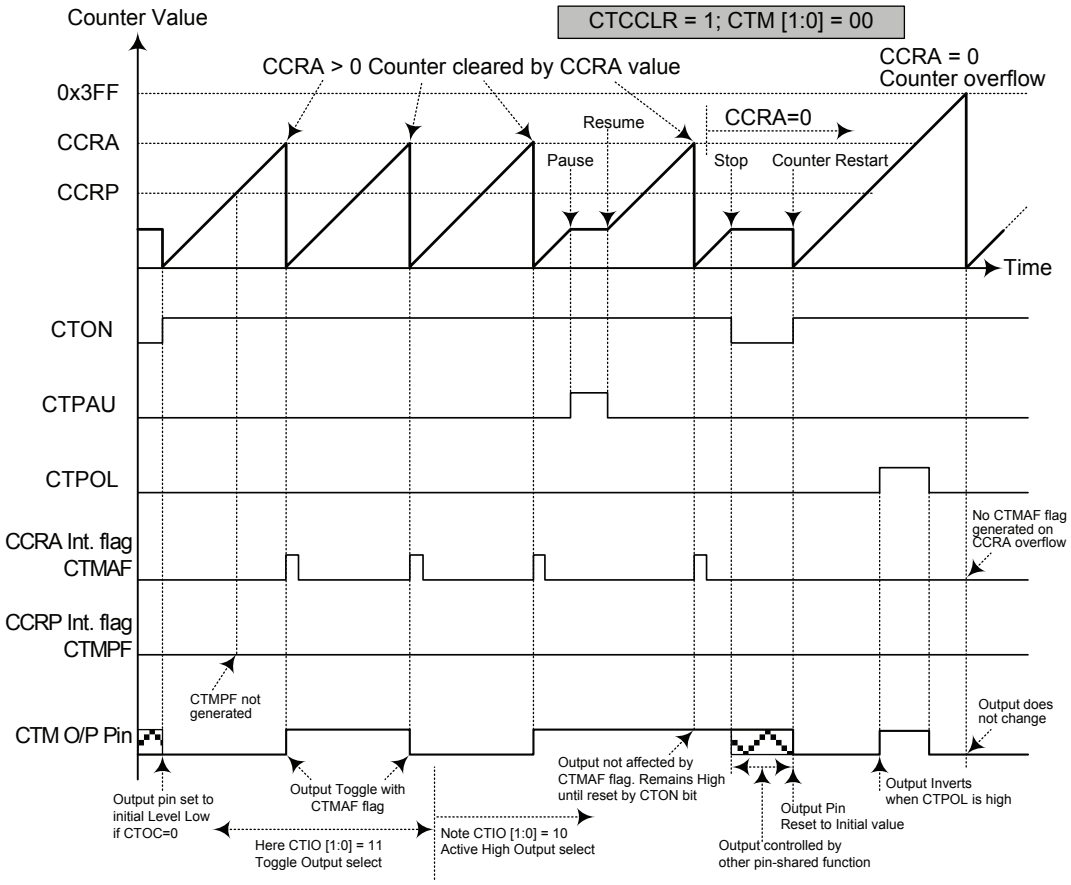


Compare Match Output Mode – CTCCLR = 0

Note: 1. With CTCCLR = 0, a Comparator P match will clear the counter

2. The CTM output pin controlled only by CTMAF flag

3. The output pin is reset to its initial state by CTON bit rising edge



Compare Match Output Mode – CTCCLR = 1

Note: 1. With CTCCLR = 1, a Comparator A match will clear the counter

2. The CTM output pin is controlled only by CTMAF flag

3. The CTM output pin is reset to initial state by CTON rising edge

4. The CTMPF flags is not generated when CTCCLR = 1

Timer/Counter Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the CTM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the CTM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 register should be set to 10 respectively. The PWM function within the CTM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the CTM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM mode, the CTCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTDPX bit in the CTMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The CTOC bit in the CTMC1 register is used to select the required polarity of the PWM waveform while the two CTIO1 and CTIO0 bits are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The CTPOL bit is used to reverse the polarity of the PWM output waveform.

- **10-bit CTM, PWM Mode, Edge-aligned Mode, CTDPX=0**

CCRP	001b	011b	011b	100b	101b	110b	111b	000b
Period	128	256	384	512	640	768	896	1024
Duty	CCRA							

If f_{SYS} = 8MHz, CTM clock source is $f_{SYS}/4$, CCRP = 2 and CCRA = 128,

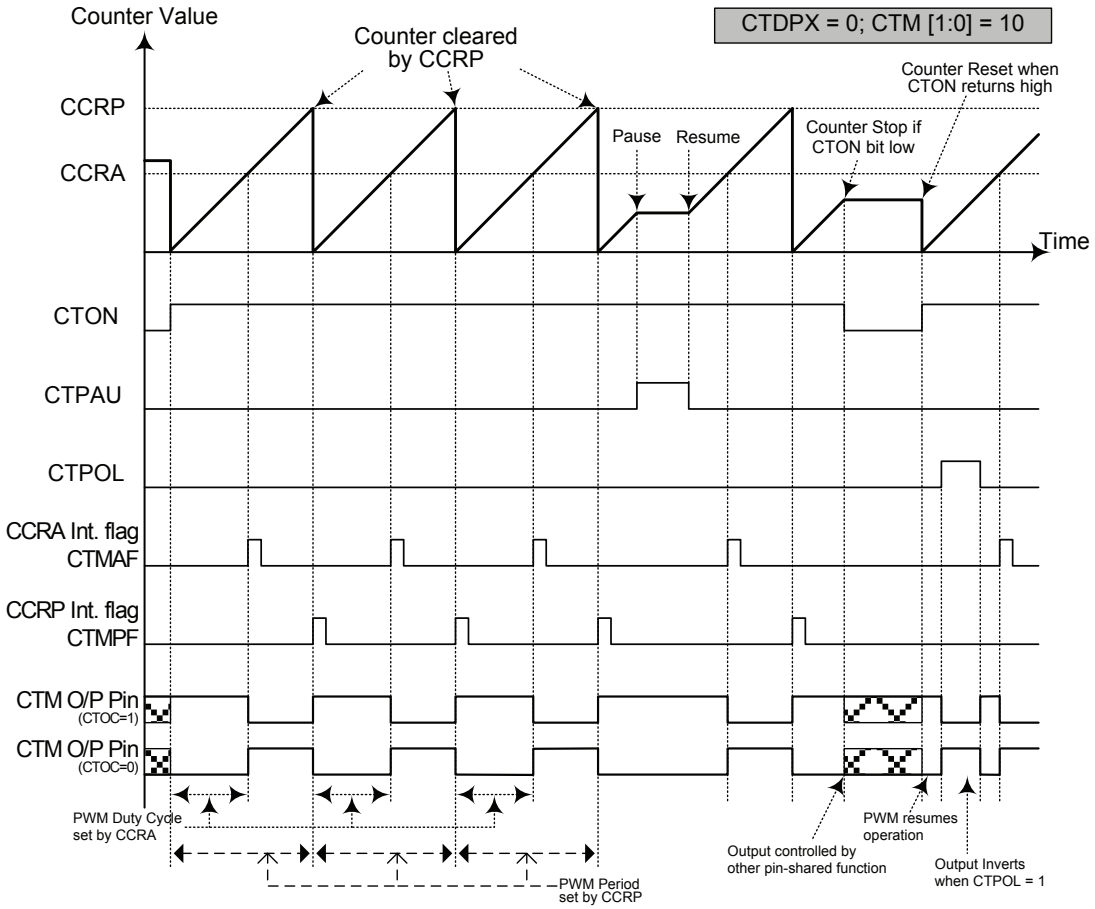
The CTM PWM output frequency = ($f_{SYS}/4$) / (2x256) = $f_{SYS}/2048$ = 4 kHz, duty = 128/(2x256)= 25%.

If the Duty value defined by the CCRA register is equal to or greater than the Period value, then the PWM output duty is 100%.

- **10-bit CTM, PWM Mode, Edge-aligned Mode, CTDPX=1**

CCRP	001b	011b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

The PWM output period is determined by the CCRA register value together with the CTM clock while the PWM duty cycle is defined by the CCRP register value.



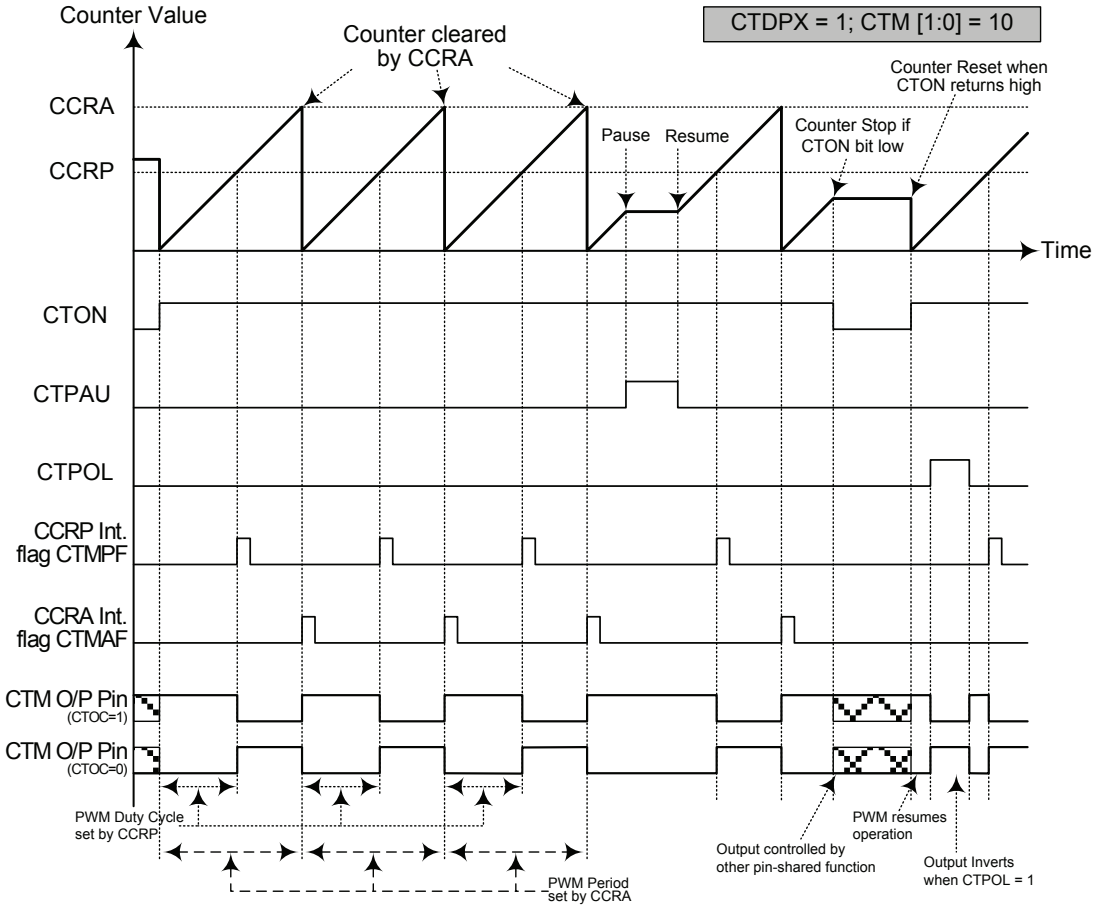
PWM Output Mode – CTDXP = 0

Note: 1. Here CTDPX = 0 – Counter cleared by CCRP

2. A counter clear sets PWM Period

3. The internal PWM function continues even when CTIO1, CTIO0 = 00 or 01

4. The CTCCLR bit has no influence on PWM operation



PWM Output Mode – CTDXP = 1

Note: 1. Here CTDPX = 1 – Counter cleared by CCRA

2. A counter clear sets PWM Period

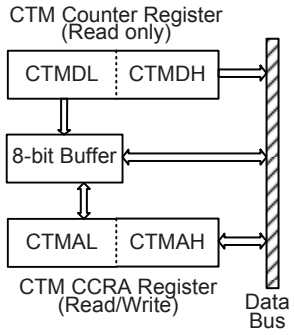
3. The internal PWM function continues even when CTIO [1:0] = 00 or 01

4. The CTCCLR bit has no influence on PWM operation

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA registers, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed.

As the CCRA registers is implemented in the way shown in the following diagram and accessing these register pairs is carried out in a specific way as described above, it is recommended to use the "MOV" instruction to access the CCRA low byte registers, named CTMAL, using the following access procedures. Accessing the CCRA low byte registers without following these access procedures will result in unpredictable values.



The following steps show the read and write procedures:

- Writing Data to CCRA
  - Step 1. Write data to Low Byte CTMAL
    – note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte CTMAH
    – here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA
  - Step 1. Read data from the High Byte CTMDH or CTMAH
    – here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - Step 2. Read data from the Low Byte CTMDL or CTMAL
    – this step reads data from the 8-bit buffer.

Digital to Analog Converter

There is a 6-bit R-2R Digital to Analog converter integrated in this device. The digital data to be converted is stored in the DACR register. The D/A converter output can be used as the reference voltage applied on the negative input of the comparator. The DACEN bit is used to control the D/A converter function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
DACR	DACEN	—	DA5	DA4	DA3	DA2	DA1	DA0
OPAC	—	OPAEN	—	—	—	—	CKS1	CKS0

Digital to Analog Converter Registers List

DACR Register

Bit	7	6	5	4	3	2	1	0
Name	DACEN	—	DA5	DA4	DA3	DA2	DA1	DA0
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

Bit 7 **DACEN**: D/A converter Enable control
0: Disable
1: Enable

Bit 6 Unimplemented, read as "0"

Bit 5~0 **DA5~DA0**: D/A converter digital data bit 5 ~ bit 0

$$\text{D/A output voltage} = \frac{[\text{DA5 ~ DA0}] \times V_{DD}}{64}$$

OPAC Register

Bit	7	6	5	4	3	2	1	0
Name	—	OPAEN	—	—	—	—	CKS1	CKS0
R/W	—	R/W	—	—	—	—	R/W	R/W
POR	—	1	—	—	—	—	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **OPAEN**: Operational Amplifier Enable control
Described elsewhere.

Bit 5~2 Unimplemented, read as "0"

Bit 1~0 **CKS1~CKS0**: D/A converter clock source selection
00: $f_{LIRC}/4$
01~11: Reserved, can not be used.

Operational Amplifier

This device contains an operational amplifier with programmable gain selections, which is used to amplify the small analog input signal. After amplification the output signal can be sent to the comparator positive input to compare with the negative input signal. The overall function is controlled by the OPAC and OPGA registers.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OPAC	—	OPAEN	—	—	—	—	CKS1	CKS0
OPGA	—	—	—	—	GAS3	GAS2	GAS1	GAS0

Operational Amplifier Registers List

OPAC Register

Bit	7	6	5	4	3	2	1	0
Name	—	OPAEN	—	—	—	—	CKS1	CKS0
R/W	—	R/W	—	—	—	—	R/W	R/W
POR	—	1	—	—	—	—	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **OPAEN**: Operational Amplifier Enable control
0: Disable
1: Enable

Bit 5~2 Unimplemented, read as "0"

Bit 1~0 **CKS1~CKS0**: D/A converter clock source selection
Described elsewhere.

OPGA Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	GAS3	GAS2	GAS1	GAS0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

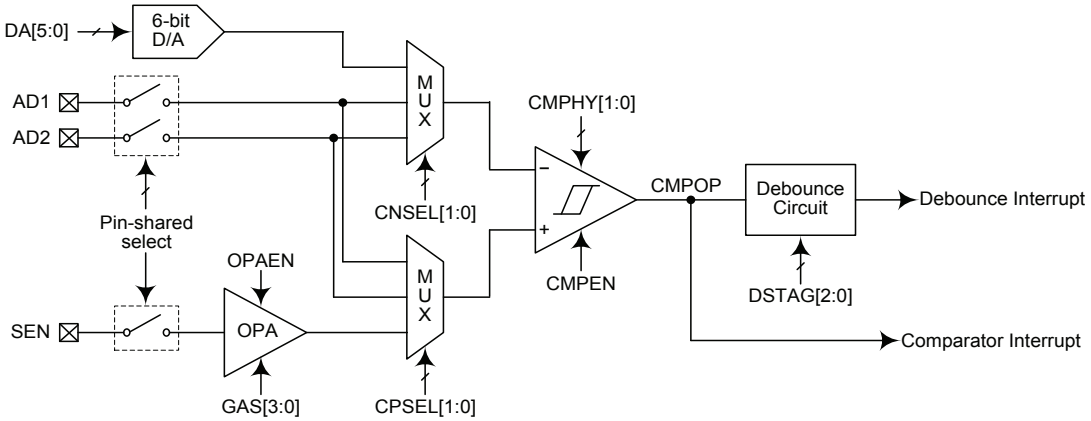
Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **GAS3~GAS0**: Operational Amplifier gain selection

0000: Gain = 1	0100: Gain = 10	1000: Gain = 15	1100: Gain = 25
0001: Gain = 10	0101: Gain = 100	1001: Gain = 150	1101: Gain = 250
0010: Gain = 30	0110: Gain = 300	1010: Gain = 450	1110: Gain = 750
0011: Gain = 40	0111: Gain = 400	1011: Gain = 600	1111: Gain = 1000

Comparators

The device contains an analog comparator which operates with the 6-bit R-2R D/A converter and operational amplifier in applications. These functions offer flexibility via the corresponding registers controlled features such as power-down, hysteresis, etc. In sharing their pins with normal I/O pins the comparator does not waste precious I/O pins if their functions are otherwise unused.



Comparator Block Diagram

Comparator Operation

The comparator functions are used to compare two analog voltages and provide an output based on their input difference. Any pull-high resistors connected to the shared comparator input pins will be automatically disconnected when the corresponding comparator functional pins are selected. As the comparator inputs approach their switching level, some spurious output signals may be generated on the comparator output due to the slow rising or falling nature of the input signals. This can be minimised by the hysteresis function. Ideally the comparator should switch at the point where the positive and negative inputs signals are at the same voltage level. However, unavoidable input offsets introduce some uncertainties here. The hysteresis function, if enabled, will also increase the switching offset value. The hysteresis window will be changed for different selections.

An interrupt will be generated when the comparator output changes state from low to high. Another interrupt will be generated when the comparator output rising edge transition occurs and keeps more than a certain debounce time determined by the DSTAG2~DSTAG0 bits. If the comparator output rising edge transition occurs but does not keep more than the specific debounce time, the debounce circuit interrupt will not be generated.

Comparator Registers

Full control over the internal comparator is provided via the control registers, CMPC, MUXC and DEBC. The comparator output is recorded via a bit in the respective control register. The comparator power down control bit, CMPEN, is used to control the overall comparator function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CMPC	CMPOP	—	CMPHY1	CMPHY0	—	—	—	CMPEN
MUXC	—	—	CNSEL1	CNSEL0	—	—	CPSEL1	CPSEL0
DEBC	—	—	—	—	—	DSTAG2	DSTAG1	DSTAG0

Comparator Registers List

CMPC Register

Bit	7	6	5	4	3	2	1	0
Name	CMPOP	—	CMPHY1	CMPHY0	—	—	—	CMPEN
R/W	R	—	R/W	R/W	—	—	—	R/W
POR	x	—	0	0	—	—	—	1

"x": unknown

Bit 7 **CMPOP**: Comparator output status
0: Positive input voltage < Negative input voltage
1: Positive input voltage > Negative input voltage
This bit is read-only and used to store compatarot output status.

Bit 6 Unimplemented, read as "0".

Bit 5~4 **CMPHY1~CMPHY0**: Comparator hysteresis selection
00: ±0mV
01: ±25mV
10: ±50mV
11: ±75mV

Bit 3~1 Unimplemented, read as "0"

Bit 0 **CMPEN**: Comparator Enable control
0: Disable
1: Enable

MUXC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	CNSEL1	CNSEL0	—	—	CPSEL1	CPSEL0
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

"x": unknown

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **CNSEL1~CNSEL0**: Comparator negative input selection
00: AD2
01: AD1
1x: DAC output

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **CPSEL1~CPSEL0**: Comparator positive input selection
00: OPA output
01: AD1
1x: AD2

DEBC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	DSTAG2	DSTAG1	DSTAG0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **DSTAG2~DSTAG0**: Debounce time selection
000: No debounce
001: 4 f_{LIRC} clock cycles
010: 8 f_{LIRC} clock cycles
011: 16 f_{LIRC} clock cycles
100: 32 f_{LIRC} clock cycles
101: 64 f_{LIRC} clock cycles
110: 128 f_{LIRC} clock cycles
111: 128 f_{LIRC} clock cycles

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module 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. These devices contain several external interrupt and internal interrupts functions. The external interrupts are generated by the action of the external INT0 and INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, EEPROM, Comparator and debounce circuit.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. These registers are the INTC0~INTC2 registers which setup the primary interrupts.

Each register contains a number of enable bits to enable or disable individual interrupts as well as interrupt flags to indicate the presence of an interrupt request. The naming convention of these follows a specific pattern. First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an "E" for enable/disable bit or "F" for request flag.

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
Comparator	CPE	CPF	—
Debounce	DEBE	DEBF	—
Time Base	TBnE	TBnF	n = 0 ~ 1
EEPROM write operation	DEE	DEF	—
CTM	CTMPE	CTMPF	—
	CTMAE	CTMAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTC0	—	TB0F	DEBF	CPF	TB0E	DEBE	CPE	EMI
INTC1	TB1F	DEF	CTMAF	CTMPF	TB1E	DEE	CTMAE	CTMPE

Interrupt Registers List

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	TB0F	DEBF	CPF	TB0E	DEBE	CPE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **TB0F**: Time Base 0 interrupt request flag
0: no request
1: interrupt request

Bit 5 **DEBF**: Debounce circuit interrupt request flag
0: no request
1: interrupt request

Bit 4 **CPF**: Comparator interrupt request flag
0: no request
1: interrupt request

Bit 3 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

Bit 2 **DEBE**: Debounce circuit interrupt control
0: Disable
1: Enable

Bit 1 **CPE**: Comparator interrupt control
0: Disable
1: Enable

Bit 0 **EMI**: Global interrupt control
0: Disable
1: Enable

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	TB1F	DEF	CTMAF	CTMPF	TB1E	DEE	CTMAE	CTMPE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **DEF**: Data EEPROM Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **CTMAF**: CTM Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTMPF**: CTM Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

Bit 2 **DEE**: Data EEPROM Interrupt control
0: disable
1: enable

Bit 1 **CTMAE**: CTM Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **CTMPE**: CTM Comparator P match Interrupt control
0: Disable
1: Enable

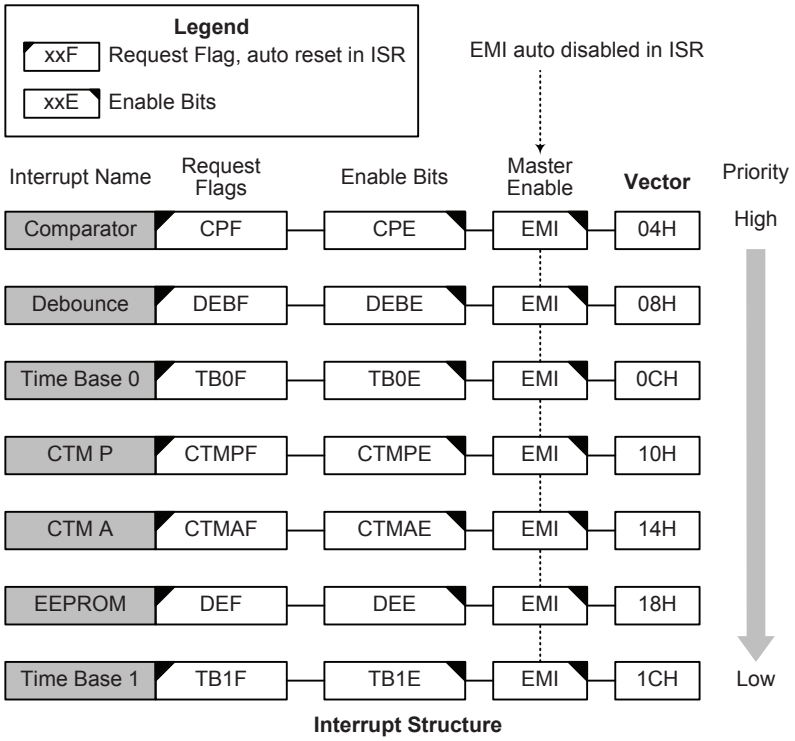
Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A or Comparator output transition, etc, the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, 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 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, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all other interrupts will be blocked, as the global interrupt enable bit, 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. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.



Interrupt Structure

Comparator Interrupt

The Comparator Interrupt is controlled by the Comparator output transition. A Comparator Interrupt request will take place when the Comparator Interrupt request flag, CPF, is set, which occurs when the Comparator output changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Comparator Interrupt enable bit, CPE, must first be set. When the interrupt is enabled, the stack is not full and the Comparator output transition occurs, a subroutine call to the Comparator Interrupt vector will take place. When the interrupt is serviced, the Comparator Interrupt flag, CPF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

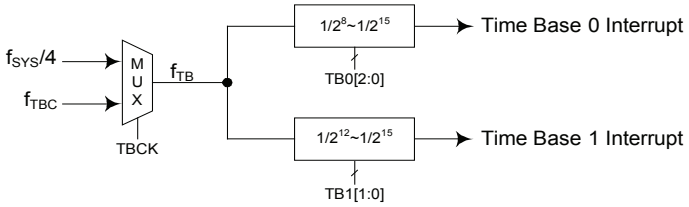
Debounce Interrupt

The Debounce Interrupt is controlled by the Debounce circuit output transition. A Debounce Interrupt request will take place when the Debounce Interrupt request flag, DEBF, is set, which occurs when the Debounce circuit output changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Comparator Interrupt enable bit, DEBE, must first be set. When the interrupt is enabled, the stack is not full and the Debounce circuit output transition occurs, a subroutine call to the Debounce Interrupt vector will take place. When the interrupt is serviced, the Debounce Interrupt flag, DEBF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Time Base Interrupt

The function of the Time Base Interrupt is to provide regular time signal in the form of an internal interrupt. It is controlled by the overflow signal from its internal timer. When this happens its interrupt request flag, TBnF, will be set. To allow the program to branch to its respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bit, TBnE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to its respective vector location will take place. When the interrupt is serviced, the interrupt request flag, TBnF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, f_{TB}, originates from the internal clock source $f_{SYS}/4$ or f_{TBC} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the TBCK bit in the TBC register.



Time Base Interrupts

TBC Register

Bit	7	6	5	4	3	2	1	0
Name	TBON	TBCK	TB11	TB10	—	TB02	TB01	TB00
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	0	0	1	1	—	1	1	1

Bit 7 **TBON**: Time Base Function Enable Control
0: Disable
1: Enable

Bit 6 **TBCK**: Time Base clock source f_{TB} selection
0: f_{TBC}
1: $f_{SYS}/4$

Bit 5~4 **TB11~TB10**: Time Base 1 time-out period selection
00: $2^{12}/f_{TB}$
01: $2^{13}/f_{TB}$
10: $2^{14}/f_{TB}$
11: $2^{15}/f_{TB}$

Bit 3 Uimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Time Base 0 time-out period selection
000: $2^{8}/f_{TB}$
001: $2^{9}/f_{TB}$
010: $2^{10}/f_{TB}$
011: $2^{11}/f_{TB}$
100: $2^{12}/f_{TB}$
101: $2^{13}/f_{TB}$
110: $2^{14}/f_{TB}$
111: $2^{15}/f_{TB}$

EEPROM Interrupt

The EEPROM Write Interrupt is controlled by the EEPROM Write operation completion. An EEPROM Write Interrupt request will take place when the EEPROM Write Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Write Interrupt enable bit, DEE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the respective EEPROM Write Interrupt vector will take place. When the EEPROM Write Interrupt is serviced, the Comparator Interrupt flag, DEF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

TM Interrupts

The Compact TMs have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective TM Interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant CTM Interrupt vector locations, will take place. When the TM interrupt is serviced, the corresponding CTM interrupt request flag will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though these devices are in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins, a low power supply voltage or comparator input change may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

By disabling the relevant 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 the application program.

It is recommended that programs do not use the "CALL" instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. 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.

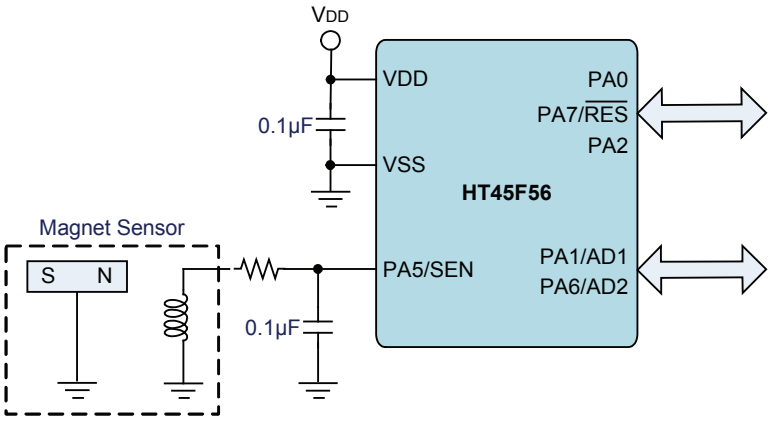
Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

Application Circuits

Sense Magnetic Sensor Module Signal by SEN Pin



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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 which 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 set 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
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

Mnemonic	Description	Cycles	Flag Affected
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 Operation			
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 Operation			
TABRD [m]	Read table (specific page) to TBLH and Data Memory	2Note	None
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 bit wise 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 $ACC \leftarrow \overline{[m]}$

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

Description This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

OR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

ORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

Description The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation Program Counter ← Stack

Affected flag(s) None

RET A,x Return from subroutine and load immediate data to ACC

Description The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

Description The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

RLA [m] Rotate Data Memory left with result in ACC

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − C

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

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

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

Operation [m] ← ACC − [m] − C

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

SDZ [m] Skip if decrement Data Memory is 0

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

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

TABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer pair (TBHP and 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

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

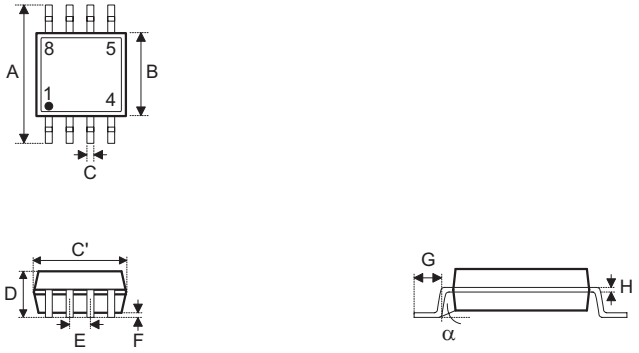
Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Further Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- Packing Meterials Information
- Carton information

8-pin SOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C’	—	0.193 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6 BSC	—
B	—	3.9 BSC	—
C	0.31	—	0.51
C’	—	4.9 BSC	—
D	—	—	1.75
E	—	1.27 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°