High Current I/O Flash MCU

HT68F0036

Revision: V1.10 Date: November 04, 2019

Table of Contents

Features

CPU Features

- Operating voltage
  - f_{SYS}=8MHz: 1.8V~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 Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 2-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 1K×14
- RAM Data Memory: 64×8
- Emulated EEPROM Memory: 32×14
- Watchdog Timer function
- 13 bidirectional I/O lines
- High driving current outputs
- One pin-shared external interrupt
- One 8-bit programmable Timer/Event Counter with overflow interrupt and prescaler
- A pair of 8-bit PWM complementary outputs shared with I/O line
- Single Time-Base function for generation of fixed time interrupt signals
- Low voltage reset function
- Package type: 16-pin NSOP

General Description

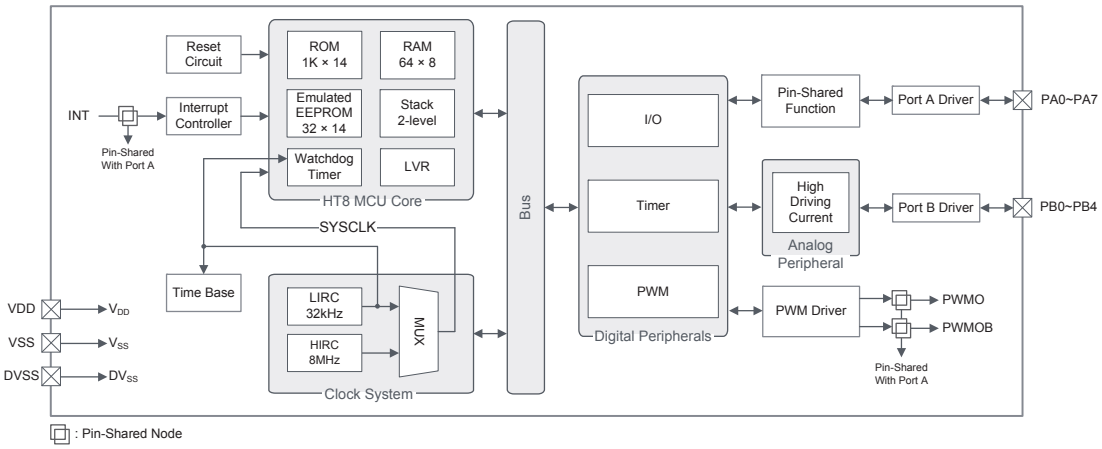
The device is a Flash Memory 8-bit high performance RISC architecture microcontroller. Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of Emulated EEPROM memory for storage of non-volatile data such as serial numbers, calibration data, etc.

A single extremely flexible Timer/Event Counter provides timing, event counting and pulse wide capture functions. A Pulse Width Modulator provides a pair of 8-bit PWM complementary outputs. An internal Watchdog Timer coupled with low voltage reset function, excellent noise immunity and ESD protection ensures that reliable operation is maintained in hostile electrical environments.

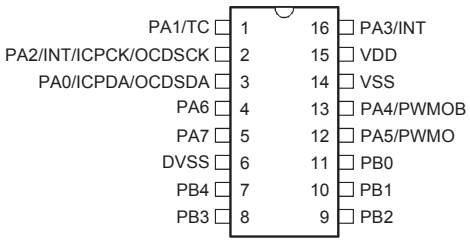
A full choice of internal high and low speed oscillators are provided and the two fully integrated system oscillators require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimize microcontroller operation and minimize power consumption.

The inclusion of flexible I/O programming features, Time-Base funciton, high driving current outputs along with other features ensure that the device will find excellent use in electronic metering, environmental monitoring, handheld instruments, household appliances, electronically controlled tools, motor driving in addition to many others.

Block Diagram



Pin Assignment



HT68F0036/HT68V0036
16 NSOP-A

Note: 1. The desired pin-shared function is determined by the corresponding pin-shared control bits.

2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the HT68V0036 device which is the OCDS EV chip for the HT68F0036 device.

Pin Description

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.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/ICPDA/ OCDSDA	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	ICPDA	—	ST	CMOS	ICP Data/Address pin
	OCDSDA	—	ST	CMOS	OCDS Data/Address pin, for EV chip only
PA1/TC	PA1	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TC	—	ST	—	Timer/Event Counter clock input
PA2/INT/ICPCK/ OCDSCK	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	IFS INTEG INTC	ST	—	External Interrupt
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/INT	PA3	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	IFS INTEG INTC	ST	—	External Interrupt
PA4/PWMOB	PA4	PAPU PAWU PASR	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PWMOB	PASR	—	CMOS	PWM signal inverting output
PA5/PWMO	PA5	PAPU PAWU PASR	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PWMO	PASR	—	CMOS	PWM signal output
PA6~PA7	PA6~PA7	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
PB0~PB4	PB0~PB4	—	ST	NMOS	General purpose high current driver I/O. The max. input voltage is 8V.

Pin Name	Function	OPT	I/T	O/T	Description
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground
DVSS	DVSS	—	PWR	—	High current driving current output negative power supply

Legend: I/T: Input type;
O/T: Output type;
OPT: Optional by register option;
PWR: Power;
ST: Schmitt Trigger input;
CMOS: CMOS output;
NMOS: NMOS output

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}−0.3V to 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_{OH} Total..... -80mA

I_{OL} Total (Port A or single PBn pin)..... 750mA

Total Power Dissipation (Ta=25°C)..... 1200mW

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

D.C. Characteristics

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency, pin load conditions, temperature and program instruction type, etc., can all exert an influence on the measured values.

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	$f_{SYS}=f_{HIRC}$=8MHz	1.8	—	5.5	V
	Operating Voltage – LIRC	$f_{SYS}=f_{LIRC}$=32kHz	1.8	—	5.5	V

Standby Current Characteristics

Ta=25°C, unless otherwise specify

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	1.8V	WDT off	—	0.2	0.6	0.7	μA
		3V		—	0.2	0.8	1.0	
		5V		—	0.5	1.0	1.2	
		1.8V	WDT on	—	1.0	2.0	2.4	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3	5	6	
	IDLE0 Mode – LIRC	1.8V	f_{SUB} on	—	1.5	2.5	3.0	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE1 Mode – HIRC	1.8V	f_{SUB} on, f_{SYS}=8MHz	—	240	350	420	μA
		3V		—	360	500	600	
		5V		—	600	800	960	

Note: When using the characteristic table data, the following notes should be taken into consideration:

1. Any digital inputs are set in a non-floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

Operating Current Characteristics

Ta=25°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	1.8V	f_{SYS}=32kHz	—	5	10	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	1.8V	f_{SYS}=8MHz	—	0.4	0.8	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	

Note: When using the characteristic table data, the following notes should be taken into consideration:

1. Any digital inputs are set in a non-floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Operating Current values are measured using a continuous NOP instruction program loop.

A.C. Characteristics

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency and temperature, etc., can all exert an influence on the measured values.

High Speed Internal Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at a user selected HIRC frequency and user selected voltage of either 3V or 5V.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temperature				
f_{HIRC}	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	
		1.8V~5.5V	25°C	-5%	8	+3%	
			-40°C~85°C	-10%	8	+5%	

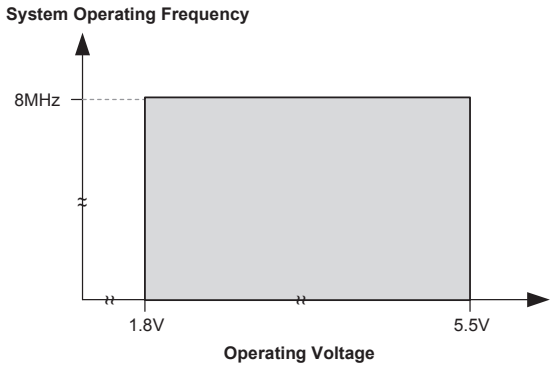
Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 1.8V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

Low Speed Internal Oscillator – LIRC – Frequency Accuracy

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	3V/5V	25°C	-1%	32	+1%	kHz
		1.8V~3.6V (trim @3V)	-10°C ~ 50°C	-4%	32	+4%	
			-40°C ~ 85°C	-6%	32	+6%	
		3.3V~5.5V (trim @5V)	-10°C ~ 50°C	-4%	32	+4%	
			-40°C ~ 85°C	-6%	32	+6%	
		2.2V~5.5V (trim @3V)	-40°C ~ 85°C	-6%	32	+6%	
		1.8V~3.6V (trim @3V)	-40°C ~ 85°C	-7%	32	+7%	
t_{START}	LIRC Start up Time	—	-40°C ~ 85°C	—	—	100	μs

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time Wake-up from Condition where f_{SYS} is off	—	f_{SYS}= f_H~f_H/64, f_H=f_{HIRC}	—	16	—	t_{HIRC}
		—	f_{SYS}=f_{SUB}=f_{LIRC}	—	2	—	t_{LIRC}
	System Start-up Time Wake-up from Condition where f_{SYS} is on	—	f_{SYS}=f_H~f_H/64, f_H=f_{HIRC}	—	2	3	t_H
		—	f_{SYS}=f_{SUB}=f_{LIRC}	—	2	3	t_{SUB}
	System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode	—	f_{HIRC} switches from off to on	14	16	18	t_{HIRC}
t_{RSTD}	System Reset Delay Time Reset Source from Power-on Reset or LVR Hardware Reset	—	RR_{POR}=5V/ms	42	48	54	ms
	System Reset Delay Time Reset Source from LVRC/WDTC Software Reset	—	—				
	System Reset Delay Time Reset Source from WDT Overflow	—	—	14	16	18	ms
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

Note: 1. For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.

2. The time units, shown by the symbols, t_{HIRC}, etc., are the inverse of the corresponding frequency values as provided in the frequency tables. For example t_{HIRC}=1/f_{HIRC}, t_{LIRC}=1/f_{LIRC}, etc.

3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.

4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

Ta=-40°C~85°C, unless otherwise specify

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports	—	—	0	—	0.3V_{DD}	V
V_{IH}	Input High Voltage for I/O Ports	—	—	0.7V_{DD}	—	V_{DD}	V
I_{OL}	Sink Current for PA0~PA3, PA6, PA7 Pins	3V	V_{OL}=0.1V_{DD}	5	10	—	mA
		5V		10	20	—	
	Sink Current for PA4/PWMOB and PA5/PWMO Pins	3V	V_{OL}=0.1V_{DD}	—	100	—	mA
		5V		—	200	—	
	Sink Current for PB Port	3V	V_{OL}=0.3V, CHPEN=0	—	500	—	mA
			V_{OL}=0.3V, CHPEN=1	—	600	—	
		5V	V_{OL}=0.5V, CHPEN=0	—	1000	—	
			V_{OL}=0.5V, CHPEN=1	—	1100	—	
I_{OH}	Source Current for PA0~PA3, PA6, PA7 Pins	3V	V_{OH}=0.9V_{DD}	-2.5	-5.0	—	mA
		5V		-5	-10	—	
	Source Current for PA4/PWMOB and PA5/PWMO Pins	3V	V_{OH}=0.9V_{DD}, Ta=25°C	—	-45	—	mA
		5V		—	-100	—	
R_{PH}	Pull-high Resistance for I/O Ports (Note)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I_{LEAK}	Input Leakage Current for I/O Ports	5V	V_{IN}=V_{DD} or V_{IN}=V_{SS}	—	—	±1	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{READYB}	PBn High Current Output Function Stable Time	—	READYB bit changes from 0 to 1 and then to 0 again	300	—	—	μs
t_{TC}	TC Clock Input Minimum Pulse Width	—	Ta=25°C	25	—	—	ns
t_{INT}	Interrupt Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs

Note: The R_{PH} internal pull-high resistance value is calculated by connecting to ground and enabling the input pin with a pull-high resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R_{PH} value.

Memory Characteristics

Ta=-40°C~85°C, unless otherwise specify

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage for Read/Erase/Write	—	—	V_{DDmin}	—	V_{DDmax}	V
Flash Program / Emulated EEPROM Memory							
t_{DEW}	Erase/Write Time – Flash Program Memory	5V	Ta=25°C	—	2	3	ms
	Erase/Write Time – Emulated EEPROM Memory	—	EWRTS[1:0]=00B	—	2	3	ms
		—	EWRTS[1:0]=01B	—	4	6	
		—	EWRTS[1:0]=10B	—	8	12	
		—	EWRTS[1:0]=11B	—	16	24	
E_P	Cell Endurance	—	—	10K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	Device in SLEEP Mode	1	—	—	V

Note: The Emulated EEPROM erase/write operation can only be executed when the f_{SYS} clock frequency is equal to or greater than 2MHz.

LVR Electrical Characteristics

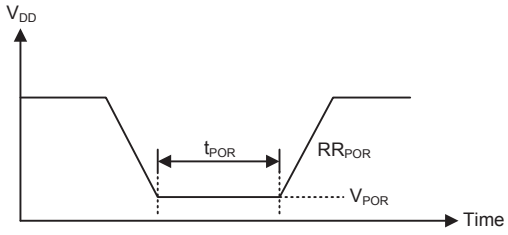
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 1.7V	-5%	1.7	+5%	V
I_{LVR}	Operating Current	3V	LVR enable, V_{LVR}=1.7V	—	—	15	μA
		5V		—	15	25	
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs

Power-on Reset Characteristics

Ta=-40°C~85°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



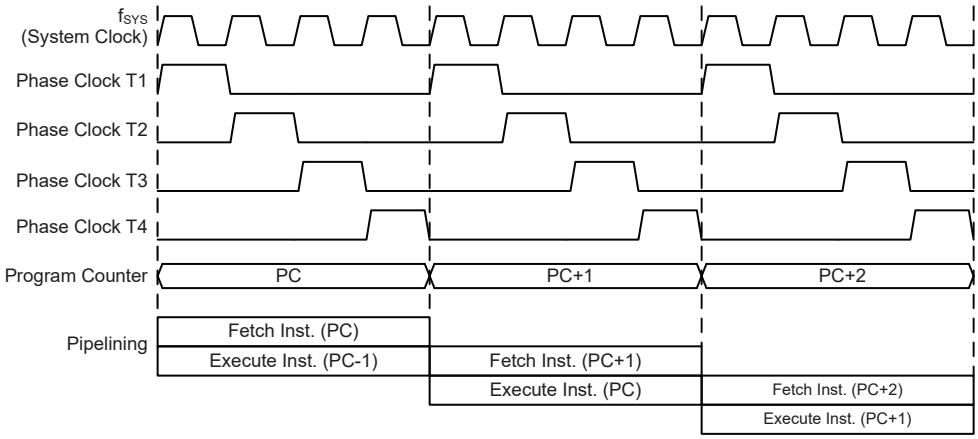
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes 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 which need one more cycle. 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 control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

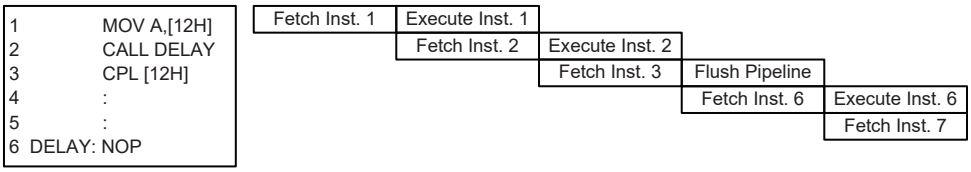
Clocking and Pipelining

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 Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demand a jump to a non-consecutive Program Memory address. 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	
Program Counter High Byte	**PCL Register**
PC9~PC8	PCL7~PCL0

Program Counter

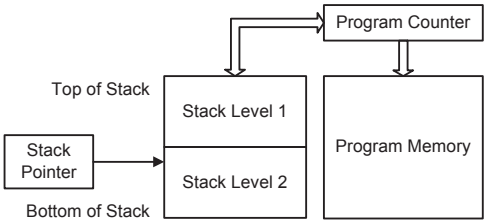
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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, organized into two levels, 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:

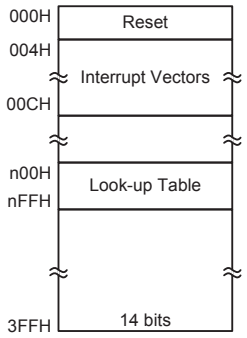
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement: INCA, INC, DECA, DEC
- Branch decision: JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, 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, the Flash device offers 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 set in any location within the Program Memory, is addressed by a separate table pointer register.



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 the device 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 configured by placing the address of the look up data to be retrieved in the table pointer registers, 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.

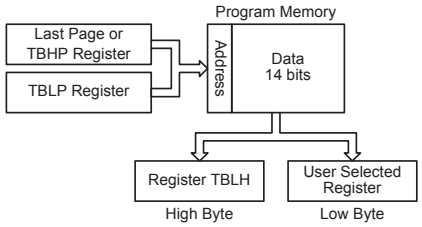


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory 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 words Program Memory of the device. The table pointer low byte register is set 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 specific address pointed by the TBLP and TBHP registers 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 the page that tbhp pointed
mov a,03h          ; initialise high table pointer
mov tbhp,a
     :
     :
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
                   ; the value "00H" will be transferred to the high byte register TBLH
     :
     :
org 300h           ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
```

In Circuit Programming – ICP

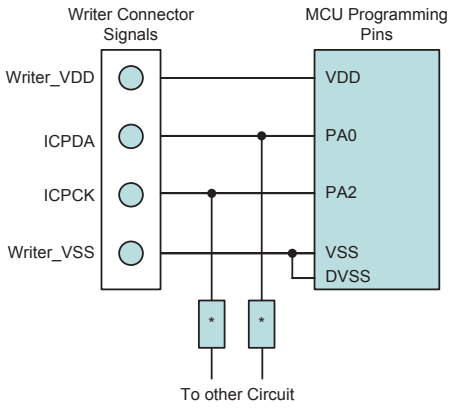
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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&DVSS	Ground

The Program 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 is 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 1kΩ or the capacitance of * must be less than 1nF.

On-Chip Debug Support – OCDS

There is an EV chip named HT68V0036 which is used to emulate the HT68F0036 device. The EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior 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. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. 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 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&DVSS	Ground

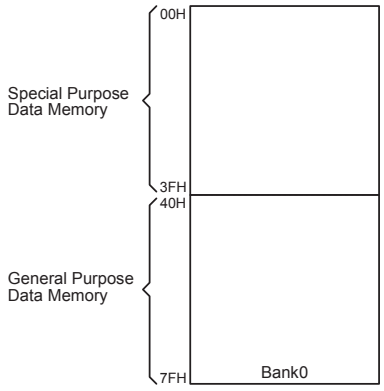
Data Memory

The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored.

Structure

Categorised into two types, the first of these is an area of RAM, known as the Special Function Data Memory. 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 known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The start address of the Data Memory for the device is 00H. The address range of the Special Purpose Data Memory for the device is from 00H to 3FH while the address range of the General Purpose Data Memory is from 40H to 7FH.



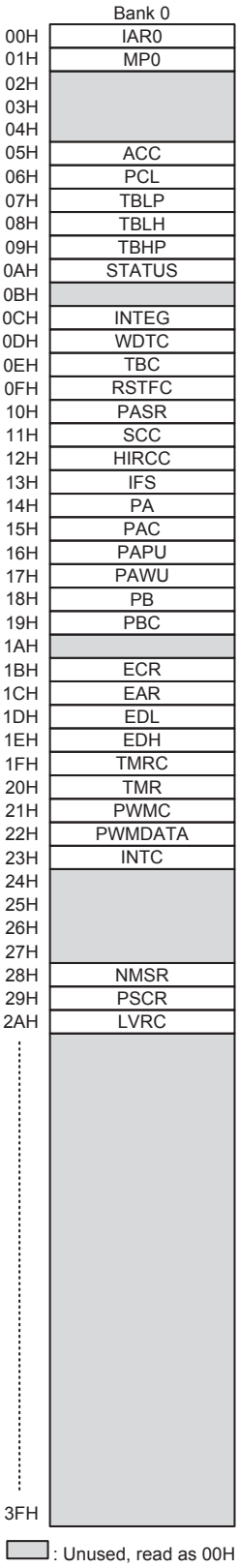
Data Memory Structure

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”.



Special Purpose Data Memory

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 Register – IAR0

The Indirect Addressing Register, IAR0, although having its location in normal RAM register space, does not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses this Indirect Addressing Register and Memory Pointer, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 register will result in no actual read or write operation to this register but rather to the memory location specified by the corresponding Memory Pointer, MP0. Acting as a pair, IAR0 and MP0 can together access data from Bank 0. As the Indirect Addressing Register is not physically implemented, reading the Indirect Addressing Register will return a result of "00H" and writing to the register will result in no operation.

Memory Pointer – MP0

One Memory Pointer, known as MP0 is provided. This Memory Pointer is physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the Indirect Addressing Register is carried out, the actual address that the microcontroller is directed to is the address specified by the Memory Pointer. MP0, together with the Indirect Addressing Register, IAR0, are used to access data from Bank 0.

The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.

Indirect Addressing Program Example

```
data .section ´data´
adres1   db ?
adres2   db ?
adres3   db ?
adres4   db ?
block    db ?
code .section at 0 ´code´
org 00h
start:
     mov a, 04h                  ; set size of block
     mov block, a
     mov a, offset adres1        ; Accumulator loaded with first RAM address
     mov MP0, a                  ; set memory pointer with first RAM address
loop:
     clr IAR0                    ; clear the data at address defined by MP0
     inc MP0                     ; increase 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 examples shown above, no reference is made to specific Data Memory addresses.

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP and TBHP are the table pointers and indicate the location where the table data is located. Their value must be set 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 or executing the “CLR WDT” or “HALT” instruction
1: A watchdog time-out occurred

Bit 4 **PDF**: Power down flag
0: After power up or executing the “CLR WDT” instruction
1: By executing the “HALT” instruction

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 subtraction

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.

Emulated EEPROM Data Memory

The device contains an Emulated EEPROM Data Memory, which 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 the Emulated 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.

Emulated EEPROM Data Memory Structure

The Emulated EEPROM Data Memory capacity is 32×14 bits for the device. The Emulated EEPROM Erase operation is carried out in a page format while the Write and Read operations are carried out in a word format. The page size is assigned with a capacity of 16 words. Note that the Erase operation should be executed before the Write operation is executed.

Operations	Format
Erase	16 words/page
Write	1 word/time
Read	1 word/time
Note: Page size = 16 words.	

Emulated EEPROM Erase/Write/Read Format

Erase Page	EAR4	EAR [3:0]
0	0	xxxx
1	1	xxxx

"x": don't care

Erase Page Number and Selection

Emulated EEPROM Registers

Four registers control the overall operation of the Emulated EEPROM Data Memory. These are the address register, EAR, the data registers, EDL and EDH, and a single control register, ECR.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EAR	—	—	—	EAR4	EAR3	EAR2	EAR1	EAR0
EDL	D7	D6	D5	D4	D3	D2	D1	D0
EDH	—	—	D13	D12	D11	D10	D9	D8
ECR	EWRTS1	EWRTS0	EEREN	EER	EWREN	EWR	ERDEN	ERD

Emulated EEPROM Register List

- **EAR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	EAR4	EAR3	EAR2	EAR1	EAR0
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 **EAR4~EAR0**: Emulated EEPROM address bit 4 ~ bit 0

• **EDL 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**: Emulated EEPROM data bit 7 ~ bit 0

• **EDH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	D13	D12	D11	D10	D9	D8
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **D13~D8**: Emulated EEPROM data bit 13 ~ bit 8

• **ECR Register**

Bit	7	6	5	4	3	2	1	0
Name	EWRTS1	EWRTS0	EEREN	EER	EWREN	EWR	ERDEN	ERD
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 **EWRTS1~EWRTS0**: Emulated EEPROM Erase/Write time selection
00: 2ms
01: 4ms
10: 8ms
11: 16ms

Bit 5 **EEREN**: Emulated EEPROM Erase enable
0: Disable
1: Enable
This bit is used to enable the Emulated EEPROM erase function and must be set high before erase operations are carried out. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Clearing this bit to zero will inhibit the Emulated EEPROM erase operations.

Bit 4 **EER**: Emulated EEPROM Erase control
0: Erase cycle has finished
1: Activate an erase cycle
When this bit is set high by the application program, an erase cycle will be activated. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Setting this bit high will have no effect if the EEREN has not first been set high.

Bit 3 **EWREN**: Emulated EEPROM Write enable
0: Disable
1: Enable
This bit is used to enable the Emulated EEPROM write function and must be set high before write operations are carried out. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Clearing this bit to zero will inhibit the Emulated EEPROM write operations.

Bit 2 **EWR**: Emulated EEPROM Write control
0: Write cycle has finished
1: Activate a write cycle
When this bit is set high by the application program, a write cycle will be activated. 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 EWREN has not first been set high.

Bit 1 **ERDEN**: Emulated EEPROM Read enable
0: Disable
1: Enable
This bit is used to enable the Emulated EEPROM read function and must be set high before read operations are carried out. Clearing this bit to zero will inhibit the Emulated EEPROM read operations.

Bit 0 **ERD**: Emulated EEPROM Read control
0: Read cycle has finished
1: Activate a read cycle
When this bit is set high by the application program, a read cycle will be activated. 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 ERDEN has not first been set high.

Note: 1. The EEREN, EER, EWREN, EWR, ERDEN and ERD cannot be set to "1" at the same time in one instruction.
2. Note that the CPU will be stopped when a read, write or erase operation is successfully activated.
3. Ensure that the f_{SYS} clock frequency is equal to or greater than 2MHz and the f_{SUB} clock is stable before executing the erase or write operation.
4. Ensure that the read, write or erase operation is totally complete before executing other operations.

Erasing the Emulated EEPROM

For Emulated EEPROM erase operation the desired erase page address should first be placed in the EAR register. The number of the page erase operation is 16 words per page, therefore, the available page erase address is only specified by the EAR4 bit in the EAR register and the content of EAR3~EAR0 in the EAR register is not used to specify the page address. To erase the Emulated EEPROM page, the EEREN bit in the ECR register must first be set high to enable the erase function. After this the EER bit in the ECR register must be immediately set high to initiate an erase cycle. These two instructions must be executed in two consecutive instruction cycles to activate an erase operation successfully. The global interrupt bit EMI should also first be cleared before implementing any erase operations, and then set high again after the a valid erase activation procedure has completed. Note that the CPU will be stopped when an erase operation is successfully activated. When the erase cycle terminates, the CPU will resume executing the application program. And the EER bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been erased. The Emulated EEPROM erased page content will all be zero after an erase operation.

Writing Data to the Emulated EEPROM

For Emulated EEPROM write operation the data and desired write address should first be placed in the EDH/EDL and EAR registers respectively. To write data to the Emulated EEPROM, the EWREN bit in the ECR register must first be set high to enable the write function. After this the EWR bit in the ECR register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that the CPU will be stopped when a write operation is successfully activated. When the write cycle terminates, the CPU will resume executing the application program. And the EWR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the Emulated EEPROM.

Reading Data from the Emulated EEPROM

For Emulated EEPROM read operation the desired read address should first be placed in the EAR register. To read data from the Emulated EEPROM, the ERDEN bit in the ECR register must first be set high to enable the read function. After this a read cycle will be initiated if the ERD bit in the ECR register is now set high. Note that the CPU will be stopped when the read operation is successfully activated. When the read cycle terminates, the CPU will resume executing the application program. And the ERD bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been read from the Emulated EEPROM. Then the data can be read from the EDH/EDL data register pair by application program. The data will remain in the data register pair until another read, write or erase operation is executed.

Programming Considerations

Care must be taken that data is not inadvertently written to the Emulated EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. 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 or erasing data the EWR or EER bit must be set high immediately after the EWREN or EEREN bit has been set high, to ensure the write or erase cycle executes correctly. The global interrupt bit EMI should also be cleared before a write or erase cycle is executed and then set again after a valid write or erase activation procedure has completed. Note that the device should not enter the IDLE or SLEEP mode until Emulated EEPROM read, write or erase operation is totally complete. Otherwise, Emulated EEPROM read, write or erase operation will fail.

Programming Examples

Erase a Data Page of the Emulated EEPROM – polling method

```
MOV  A, EEPROM_ADRES          ; user defined page
MOV  EAR, A
MOV  A, 00H                   ; Erase time=2ms (40H for 4ms, 80H for 8ms, C0H for 16ms)
MOV  ECR, A
CLR  EMI
SET  EEREN                    ; set EEREN bit, enable erase operation
SET  EER                      ; start Erase Cycle - set EER bit - executed immediately
                              ; after setting EEREN bit
SET  EMI
BACK:
SZ   EER                      ; check for erase cycle end
JMP  BACK
:
```

Writing Data to the Emulated EEPROM – polling method

```
MOV  A, EEPROM_ADRES          ; user defined address
MOV  EAR, A
MOV  A, EEPROM_DATA_L         ; user defined data
MOV  EDL, A
MOV  A, EEPROM_DATA_H
MOV  EDH, A
MOV  A, 00H                   ; Write time=2ms (40H for 4ms, 80H for 8ms, C0H for 16ms)
MOV  ECR, A
CLR  EMI
SET  EWREN                    ; set EWREN bit, enable write operation
SET  EWR                      ; start Write Cycle - set EWR bit - executed immediately
                              ; after set EWREN bit
SET  EMI
BACK:
SZ   EWR                      ; check for write cycle end
JMP  BACK
:
```

Reading Data from the Emulated EEPROM – polling method

```
MOV  A, EEPROM_ADRES          ; user defined address
MOV  EAR, A
SET  ERDEN                    ; set ERDEN bit, enable read operation
SET  ERD                      ; start Read Cycle - set ERD bit
BACK:
SZ   ERD                      ; check for read cycle end
JMP  BACK
CLR  ECR                      ; disable Emulated EEPROM read if no more read operations
                              ; are required
MOV  A, EDL                   ; move read data to register
MOV  READ_DATA_L, A
MOV  A, EDH
MOV  READ_DATA_H, A
```

Note: For each read operation, the address register should be re-specified followed by setting the ERD bit high to activate a read cycle even if the target address is consecutive.

Oscillators

Various oscillator options 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 operations are selected through the relevant control registers.

Oscillator Overview

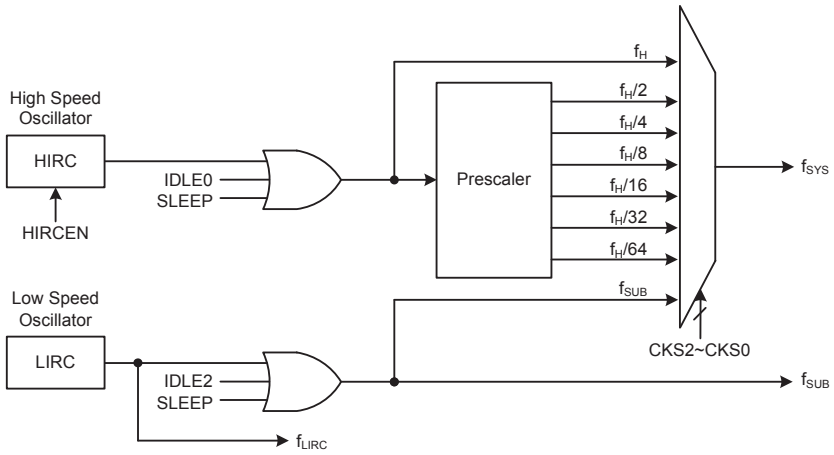
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupt. The fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carry 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	8MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two oscillator sources, one high speed oscillator and one low speed oscillator. The high speed system clock is sourced from the internal 8MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and the system clock can be dynamically selected.



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 one fixed frequency of 8MHz. 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. Note that this internal system clock option requires no external pins for its operation.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is a fully integrated low frequency RC oscillator with a typical frequency of 32kHz, 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.

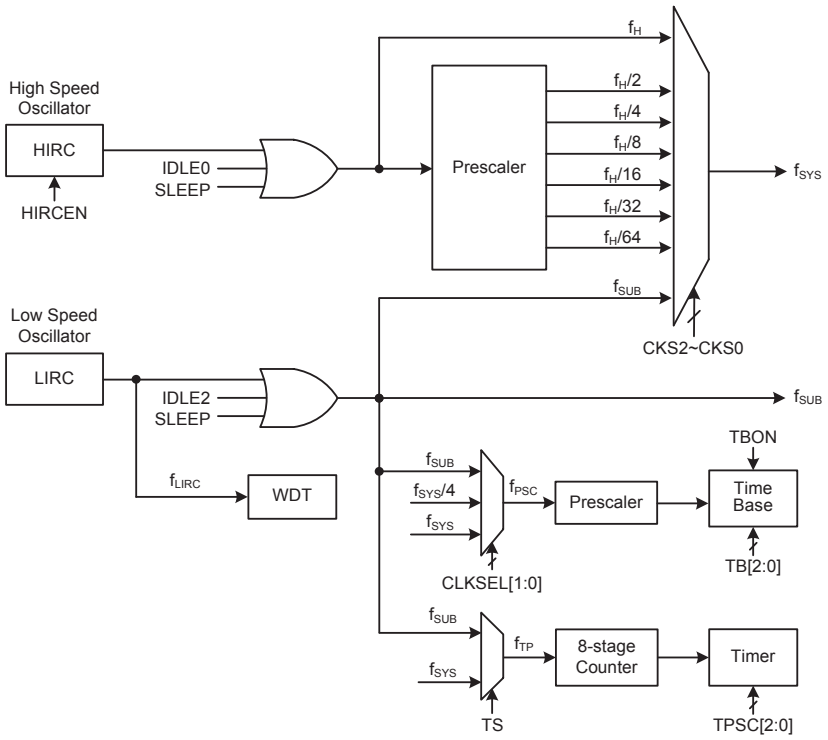
Operating Modes and System Clocks

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 the device 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 many different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock options 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 CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.



Device Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillator will stop to conserve the power or continue to oscillate to provide the clock source, f_H~$f_H/64$, for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

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 FAST Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Mode are used when the microcontroller CPU is switched off to conserve power.

Operation Mode	CPU	Register Setting			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
FAST	On	x	x	000~110	f_H~f_H/64	On	On	On
SLOW	On	x	x	111	f_{SUB}	On/Off (1)	On	On
IDLE0	Off	0	1	000~110	Off	Off	On	On
				111	On			
IDLE1	Off	1	1	xxx	On	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off	On
				111	Off			
SLEEP	Off	0	0	xxx	Off	Off	Off	On/Off (2)

"x": Don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.
2. The f_{LIRC} clock can be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

FAST Mode

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 oscillator. This mode allows the microcontroller to operate normally with a clock source which will come from the high speed oscillator, 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 bits in the SCC 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}, which is derived from the LIRC oscillator.

SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bit are low. In the SLEEP mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral function will also be stopped, too. However the f_{LIRC} clock can continues to operate if the WDT function is enabled.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is low and the FSIDEN bit in the SCC register is high. In the IDLE0 Mode the CPU will be switched off but the low speed oscillator will be turned on to drive some peripheral functions.

IDLE1 Mode

The IDLE1 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is high. In the IDLE1 Mode the CPU will be switched off but both the high and low speed oscillators will be turned on to provide a clock source to keep some peripheral functions operational.

IDLE2 Mode

The IDLE2 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is low. In the IDLE2 Mode the CPU will be switched off but the high speed oscillator will be turned on to provide a clock source to keep some peripheral functions operational.

Control Registers

The SCC and HIRCC registers are used to control the system clock and the corresponding oscillator configurations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
HIRCC	—	—	—	—	—	—	HIRCF	HIRCEN

System Operating Mode Control Register List

- **SCC Register**

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	0	0	0	—	—	—	0	0

Bit 7~5 **CKS2~CKS0**: System clock selection
000: f_H
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: f_{SUB}
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from f_H or f_{SUB}, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4~2 Unimplemented, read as "0"

Bit 1 **FHIDEN**: High Frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing a "HALT" instruction.

Bit 0 **FSIDEN**: Low Frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing a "HALT" instruction.

• **HIRCC Register**

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

Bit 7~2 Unimplemented, read as "0"

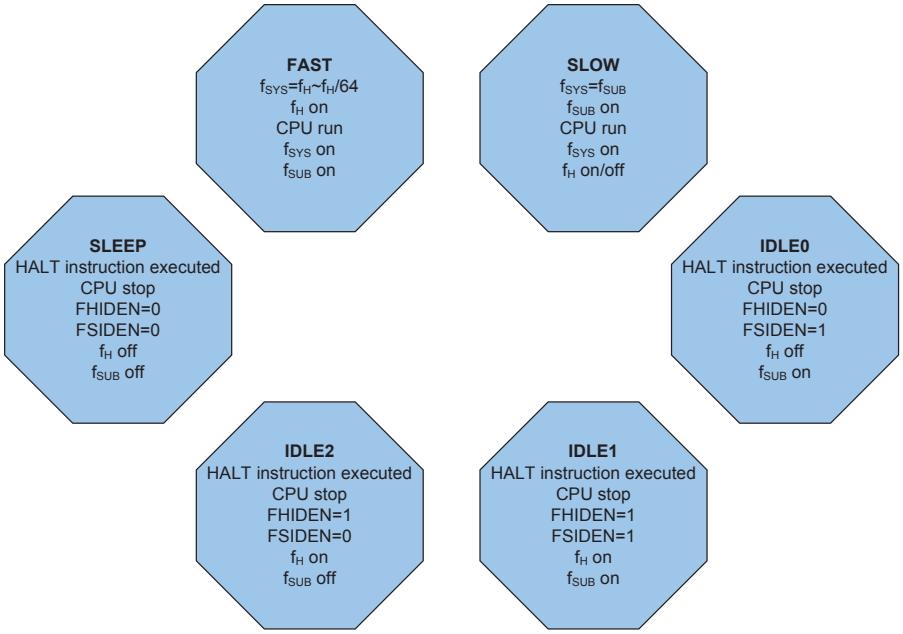
Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

Bit 0 **HIRCEN**: HIRC oscillator enable control
0: Disable
1: Enable

Operating Mode Switching

The device 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.

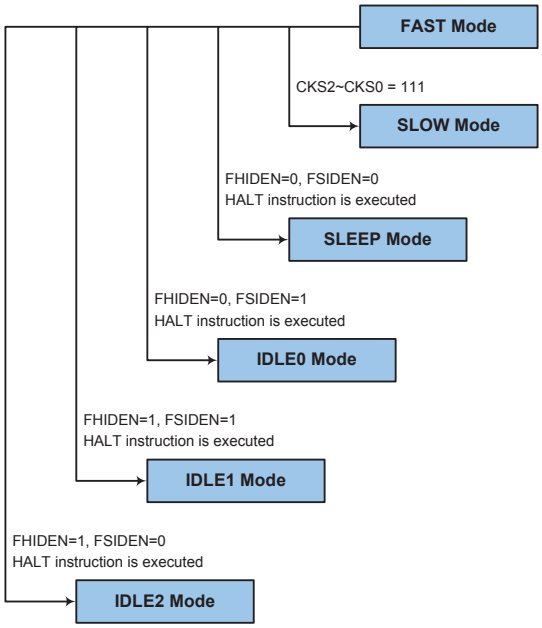
In simple terms, mode switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the FHIDEN and FSIDEN bits in the SCC register.



FAST Mode to SLOW Mode Switching

When running in the FAST 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 set the CKS2~CKS0 bits to "111" in the SCC 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.

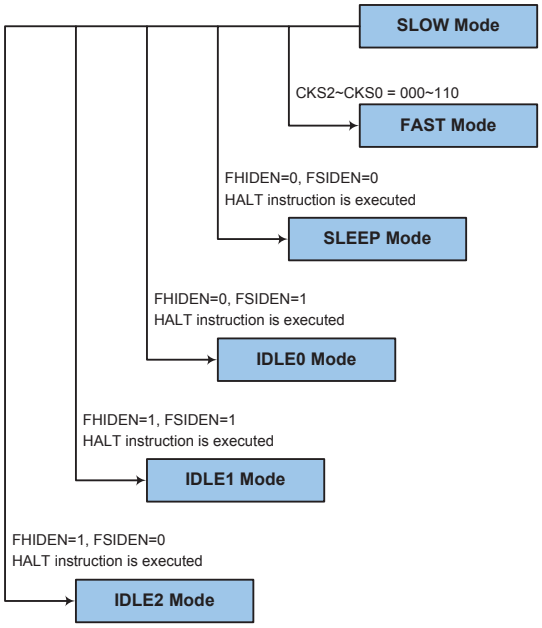
The SLOW Mode system clock is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

In the SLOW mode the system clock is derived from f_{SUB}. When system clock is switched back to the FAST mode from f_{SUB}, the CKS2~CKS0 bits should be set to "000"~"110" and then the system clock will respectively be switched to f_H~f_H/64.

However, if f_H is not used in the SLOW mode and thus switched off, it will take some time to re-oscillate and stabilise when switching to the FAST mode from the SLOW Mode. This is monitored using the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilisation is specified in the System Start Up Time Characteristics.



Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "0". In this mode all the clocks and functions will be switched off except the WDT function. 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.
- 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 the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

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 FHIDEN bit in the SCC register equal to "0" and the FSIDEN bit in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The f_H clock will be stopped and the application program will stop at the "HALT" instruction, but the f_{SUB} 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 the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

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 both the FHIDEN and FSIDEN bits in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The f_H and f_{SUB} clocks 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 the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE2 Mode

There is only one way for the device to enter the IDLE2 Mode and that is to execute the "HALT" instruction in the application program with the FHIDEN bit in the SCC register equal to "1" and the FSIDEN bit in the SCC register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

- The f_H clock will be on but the f_{SUB} clock will be off 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 the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

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 and IDLE2 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 the device which has different package types, as there may be unbonded pins. These must either be set as outputs or if set 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 set 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. Also note that additional standby current will also be required if the LIRC oscillator has enabled.

In the IDLE1 and IDLE2 Mode the high speed oscillator is on, if the peripheral function clock source is derived from the high speed 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, stabilise 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 falling edge on Port A
- A system interrupt
- A WDT overflow

When the device executes the “HALT” instruction, it will enter the IDLE or SLEEP mode and the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction. If the system is woken up by a WDT overflow, a Watchdog Timer reset will be initiated and the TO flag will be set to 1. 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, other flags remain in their original status.

Each pin on Port A can be set using the PAWU register to permit a negative transition on the pin to wake-up the system. When 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 clock, f_{LIRC} which is sourced from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32kHz 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 time-out period as well as the enable/disable operation.

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	1	1	1

Bit 7~3 **WE4~WE0**: WDT function software control
10101: Disable
01010: Enable
Other values: Reset MCU
When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set high.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $[(2^8-2^0)\sim2^8]/f_{LIRC}$
001: $[(2^9-2^1)\sim2^9]/f_{LIRC}$
010: $[(2^{10}-2^2)\sim2^{10}]/f_{LIRC}$
011: $[(2^{11}-2^3)\sim2^{11}]/f_{LIRC}$
100: $[(2^{12}-2^4)\sim2^{12}]/f_{LIRC}$
101: $[(2^{13}-2^5)\sim2^{13}]/f_{LIRC}$
110: $[(2^{14}-2^6)\sim2^{14}]/f_{LIRC}$
111: $[(2^{15}-2^7)\sim2^{15}]/f_{LIRC}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

• RSTFC Register

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

"x": unknown

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.

Bit 1 **LRF**: LVR control register software reset flag
Refer to the Low Voltage Reset section.

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

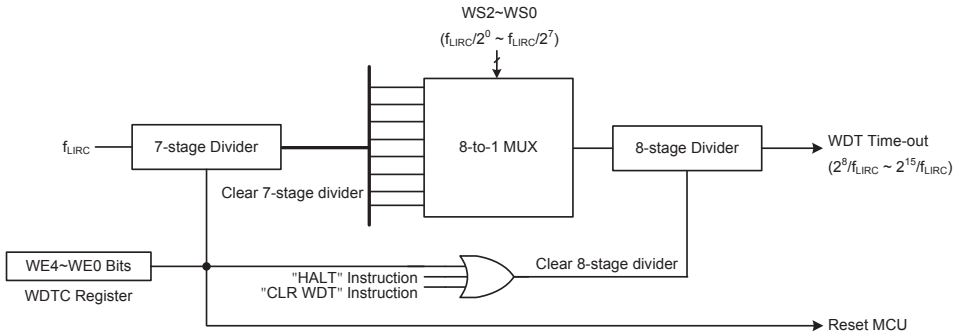
The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using 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. 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 a delay time, t_{SRESET}. 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. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDTC software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bits, the second is using the Watchdog Timer software clear instruction, the third is via a HALT instruction. 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.

The maximum time-out period is when the 2^{15} division ratio is selected. As an example, with a 32kHz 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 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 device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontroller. In this case, internal circuitry will ensure that the microcontroller, after a short delay, will be in a well-defined state and ready to execute the first program instruction. After this power-on reset, certain important internal registers will be set to defined states before the program commences. One of these registers is the Program Counter, which will be reset to zero forcing the microcontroller to begin program execution from the lowest Program Memory address.

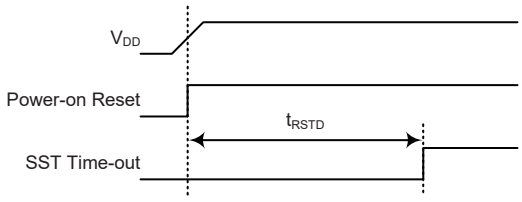
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

Reset Functions

There are several ways in which a microcontroller reset can occur through events occurring internally.

Power-on Reset

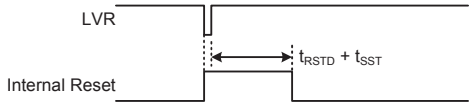
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. 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.



Power-on Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level. 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 in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. 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. If the LVS7~LVS0 bits are set to 01011010B, the LVR function is enabled with a fixed LVR voltage of 1.7V. If the LVS7~LVS0 bits are set to 10100101B, the LVR function is disabled. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01011010B. Note that the LVR function will be automatically disabled when the device enters the IDLE or SLEEP mode.



Low Voltage Reset Timing Chart

- **LVRC Register**

Bit	7	6	5	4	3	2	1	0
Name	LVS7	LVS6	LVS5	LVS4	LVS3	LVS2	LVS1	LVS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	1	0	1	0

Bit 7~0 **LVS7~LVS0**: LVR voltage select control
01011010: 1.7V
10100101: disable
Any other value: Generates MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps more than a t_{LVR} time. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than 01011010B and 10100101B, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time, t_{SRESET}. However in this situation the register contents will be reset to the POR value.

- **RSTFC Register**

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

"x": unknown

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred

This bit is set high when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

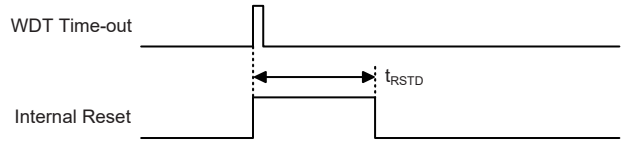
Bit 1 **LRF**: LVR control register software reset flag
0: Not occurred
1: Occurred

This bit is set high if the LVRC register contains any non-defined LVRC register values. This in effect acts like a software-reset function. This bit can only be cleared to zero by the application program.

Bit 0 **WRF**: WDT control register software reset flag
Refer to the Watchdog Timer Control Register section.

Watchdog Time-out Reset during Normal Operation

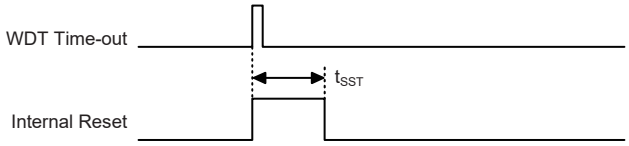
When the Watchdog time-out Reset during normal operations in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during 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 System Start Up Time Characteristics for t_{SST} details.



WDT Time-out Reset during SLEEP or IDLE 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 Conditions
0	0	Power-on reset
u	u	LVR reset during FAST or SLOW Mode operation
1	u	WDT time-out reset during FAST 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	Condition After Reset
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Base	Cleared after reset, WDT begins counting
Timer/Event Counter	Timer/Event Counter will be turned off
Input/Output Ports	I/O ports will be set as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

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

Register	Reset (Power On)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	xxxx xxxx	uuuu uuuu	uuuu uuuu
MP0	xxxx xxxx	xxxx xxxx	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu
TBHP	---- --xx	---- --uu	---- --uu
STATUS	--00 xxxx	--1u uuuu	--11 uuuu
INTEG	---- --00	---- --00	---- --uu
WDTC	0101 0111	0101 0111	uuuu uuuu
TBC	0--- -000	0--- -000	u--- -uuu
RSTFC	---- -x00	---- -uuu	---- -uuu
PASR	---- --00	---- --00	---- --uu
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- --01	---- --01	---- --uu
IFS	---- ---0	---- ---0	---- ---u
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PB	---1 1111	---1 1111	---u uuuu
PBC	---1 1111	---1 1111	---u uuuu
ECR	0000 0000	0000 0000	uuuu uuuu
EAR	---0 0000	---0 0000	---u uuuu
EDL	0000 0000	0000 0000	uuuu uuuu
EDH	--00 0000	--00 0000	--uu uuuu
TMRC	0000 1000	0000 1000	uuuu uuuu
TMR	0000 0000	0000 0000	uuuu uuuu
PWMC	0000 ---0	0000 ---0	uuuu ---u
PWMDATA	0000 0000	0000 0000	uuuu uuuu
INTC	-000 0000	-000 0000	-uuu uuuu
NMSR	---- --00	---- --00	---- --uu
PSCR	---- -000	---- -000	---- -uuu
LVRC	0101 1010	0101 1010	uuuu 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 provides bidirectional input/output lines labeled with port names PA~PB. 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	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	—	PB4	PB3	PB2	PB1	PB0
PBC	—	—	—	PBC4	PBC3	PBC2	PBC1	PBC0
NMSR	—	—	—	—	—	—	CHPEN	READYB

"—": Unimplemented, read as "0"

I/O Logic Function Register List

Port A 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 on Port A, when configured as a digital 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.

Note that the pull-high resistor can be controlled by the relevant pull-high control register only when the pin-shared functional pin is selected as a digital input or NMOS output. Otherwise, the pull-high resistors cannot be enabled.

• PAPU Register

Bit	7	6	5	4	3	2	1	0
Name	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
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 **PAPU7~PAPU0**: PA7~PA0 pull-high function control
0: Disable
1: Enable

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.

Note that the wake-up function can be controlled by the wake-up control register only when the pin is selected as a general purpose input and the MCU enters the IDLE or SLEEP mode.

• **PAWU Register**

Bit	7	6	5	4	3	2	1	0
Name	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
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 **PAWU7~PAWU0**: PA7~PA0 wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

Each I/O port has its own control register known as PAC~PBC, to control the input/output configuration. With this control register, each CMOS output, NMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be set as a CMOS output for Port A or a NMOS output for Port B. If the pin is currently set 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.

• **PxC Register**

Bit	7	6	5	4	3	2	1	0
Name	PxC7	PxC6	PxC5	PxC4	PxC3	PxC2	PxC1	PxC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

PxCn: I/O Port x Pin type selection
0: Output
1: Input
The PxCn bit is used to control the pin type selection. Here the "x" can be A and B. However, the actual available bits for each I/O Port may be different.

Port B High Current Output Function Control

After the PBn pin has been set as an NMOS output by clearing the PBCn bit to zero, the PBn internal charge pump circuit can be switched on by setting the CHPEN bit in the NMSR register. The READYB bit in the same register can be used to check whether the NMOS output driver is ready to function.

- **NMSR Register**

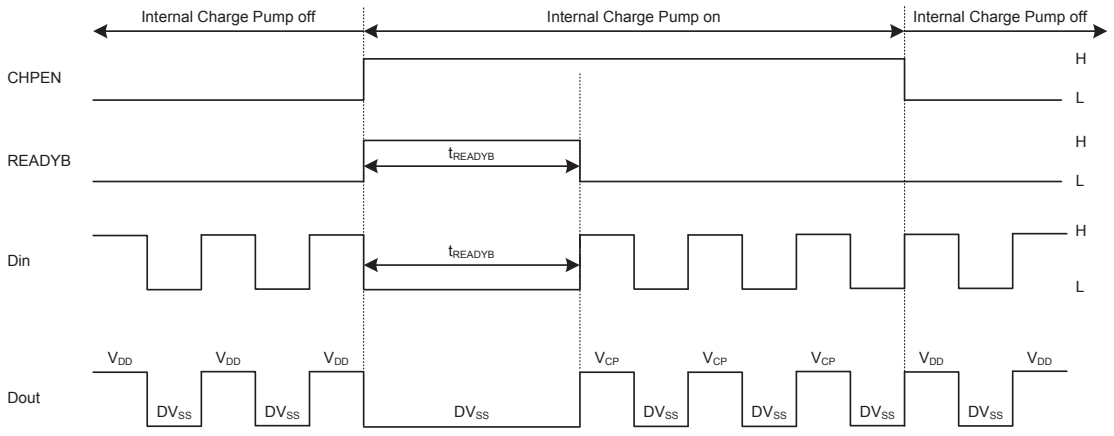
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CHPEN	READYB
R/W	—	—	—	—	—	—	R/W	R
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

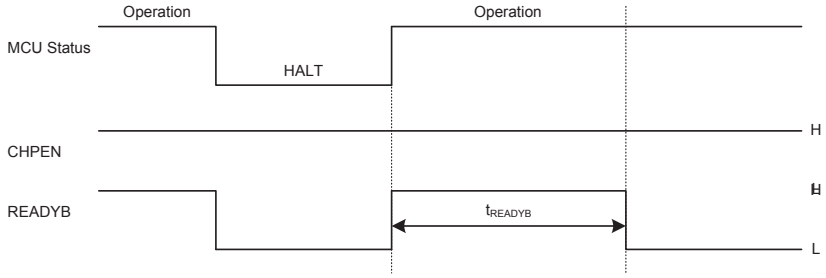
Bit 1 **CHPEN**: Internal charge pump switch on/off control
0: Disable
1: Enable
This bit is used to enable the internal charge pump circuit and is only available when the PBn pin is configured as an NMOS type output by clearing the PBCn bit to zero. When a HALT instruction is executed, the charge pump circuit will be turned off automatically until the MCU is woken up.

Bit 0 **READYB**: PB pins high current output function ready flag
0: Ready
1: Not ready
This bit is used to indicate that whether the NMOS output driver is ready to deliver the data signal or not. This bit is only available when the CHPEN bit has been set high and the output function is NMOS type. When the high current output function is enabled, including a wake-up from the IDLE/SLEEP mode, a certain period delay is necessary for the NMOS output driver to become stable before the data signal is sent to the PB output pins. Ensure that the NMOS output driver is ready by polling the READYB bit before delivering the data signal.

When the CHPEN bit changes from "0" to "1", the READYB bit will first be set to "1", after a certain period delay, known as t_{READYB}, it will be cleared to "0" indiating that the high current output function is ready to deliver data signals. More details regarding the relationship between the relevant register settings and the internal charge pump control will be described in the I/O Pin Structures section.



When in the NMOS output mode and the charge pump has been activated, if a HALT instruction is executed, the charge pump circuit will be turned off automatically with the CHPEN bit remaining high. After the MCU is woken up, the charge pump will be re-enabled automatically without requiring CHPEN bit re-configuration. However, a period delay t_{READYB} is still required before the NMOS output driver returns stable.



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 series of registers 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. The device includes a Port A Output Function Selection register, labeled as PASR, and Input Function Selection register, labeled as IFS, 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. For most pin-shared functions, 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. However, a special point must be noted for some digital input pins, which may share the same pin-shared control configuration with its corresponding general purpose I/O function when setting the relevant pin-shared control bit. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be set as an input by setting the corresponding bit in the I/O port control register. 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.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PASR	—	—	—	—	—	—	PASR1	PASR0
IFS	—	—	—	—	—	—	—	INTPS

Pin-shared Function Selection Register List

• **PASR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PASR1	PASR0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1 **PASR1**: PA4 Pin-Shared function selection
0: PA4
1: PWMOB

Bit 0 **PASR0**: PA5 Pin-Shared function selection
0: PA5
1: PWMO

• **IFS Register**

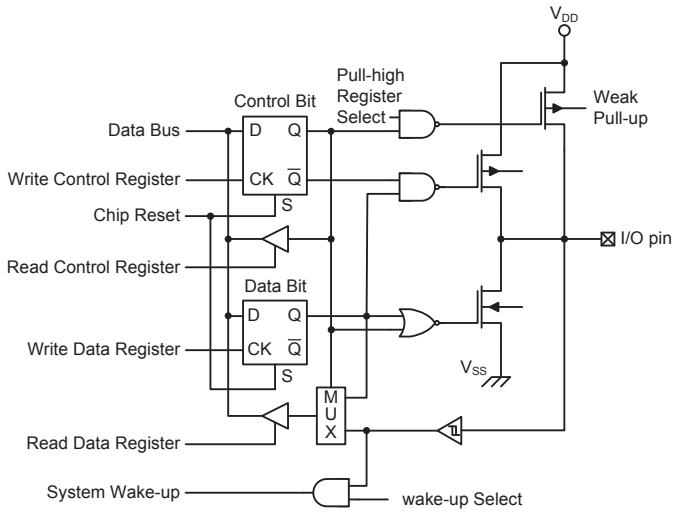
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	INTPS
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

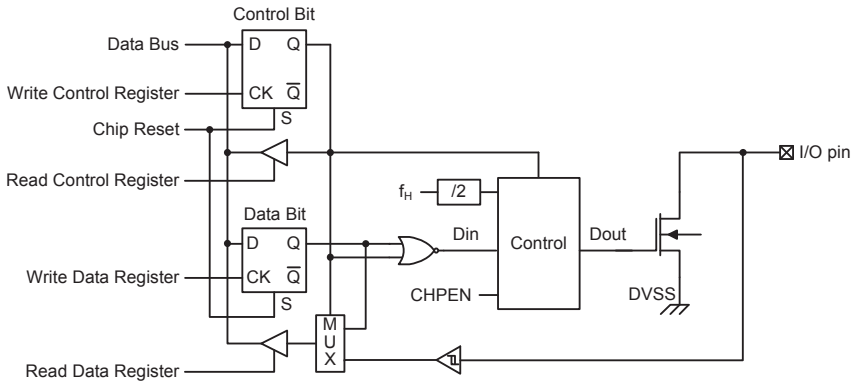
Bit 0 **INTPS**: INT input source pin selection
0: PA3
1: PA2

I/O Pin Structures

The accompanying diagram illustrates the internal structure of the I/O logic function. As the exact logical construction of the I/O pin will differ from this drawing, it is supplied as a guide only to assist with the functional understanding of the I/O logic function. The wide range of pin-shared structures does not permit all types to be shown.



Logic Function Input/Output Structure – PA Port



Logic Function Input/Output Structure – PB Port

Status	Register Setting		Internal Charge Pump Circuit	Dout Voltage	Dout Output Source
	PBCn	CHPEN			
Output	0	0	Off	V_{DD}/DV_{SS}	Din
	0	1	On	V_{CP}/DV_{SS}	
Input	1	x	Off	DV_{SS}	0

"x": Don't care

Note: V_{CP} is the charge pump voltage.

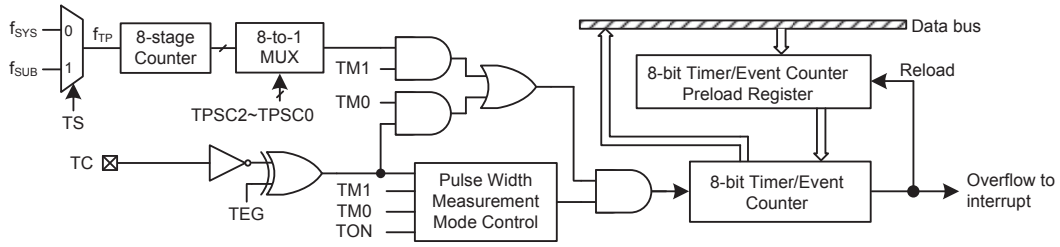
Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to set 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 function. 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 set to have this function.

Timer/Event Counter

The provision of the Timer/Event Counter forms an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contains an 8-bit Timer/Event Counter, which contains an 8-bit programmable count-up counter and the clock may come from an external or internal clock source. As the timer has three different operating modes, it can be configured to operate as a general timer, an external event counter or a pulse width capture device.



8-bit Timer/Event Counter

Timer/Event Counter Input Clock Source

The Timer/Event Counter clock source can originate from various sources, an internal clock or an external pin. The internal clock source is used when the timer is in the Timer Mode and Pulse Width Capture Mode. For the Timer/Event Counter, this internal clock source can be configured by the TS bit in the TMRC Timer Control Register to be derived from the f_{SYS} or f_{SUB} clock, the division ratio of which is selected by the TPSC2~TPSC0 bits in the TMRC Timer/Event Control Register.

An external clock source is used when the Timer/Event Counter is in the Event Counter Mode, the clock source is provided on the external TC pin. Depending upon the condition of the TEG bit, each high to low or low to high transition on the external timer pin will increase the counter by one.

Timer/Event Counter Registers

There are two registers related to the Timer/Event Counter. The first is the TMR register that contains the actual value of the timer and into which an initial value can be preloaded. Writing to the TMR register will transfer the specified data to the Timer/Event Counter. Reading the TMR register will read the contents of the Timer/Event Counter. The second is the TMRC control register, which is used to define the operating mode, select the internal clock source, control the counting enable or disable and select the active edge.

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMRC	TM1	TM0	TS	TON	TEG	TPSC2	TPSC1	TPSC0
TMR	D7	D6	D5	D4	D3	D2	D1	D0

Timer/Event Counter Register List

Timer Register – TMR

The timer register TMR is the place where the actual timer value is stored. The value in the timer register increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit Timer/Event Counter, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be loaded with the preload register value and continue counting.

Note that to achieve a maximum full range count of FFH, the preload register must first be cleared. Note that if the Timer/Event Counter is in an off condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter until an overflow occurs.

• **TMR 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**: Timer preload register byte

Timer Control Register – TMRC

The flexible features of the Holtek microcontroller Timer/Event Counter are implemented by operating in three different modes, the options of which are determined by the contents of control register bits.

The Timer Control Register is known as TMRC. It is the Timer Control Register together with its corresponding timer register that controls the full operation of the Timer/Event Counter. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

To select which of the three modes the timer is to operate in, namely the Timer Mode, the Event Counter Mode or the Pulse Width Capture Mode, the TM1~TM0 bits in the Timer Control Register must be set to the required logic levels. The timer-on bit TON provides the basic on/off control of the respective timer. Setting the bit to high allows the counter to run. Clearing the bit stops the counter. When the internal clock source is used, it can be sourced from the f_{SYS} or f_{SUB} clock selected by setting the TS bit. Bits TPSC2~TPSC0 determine the division ratio of the selected clock source. The internal clock selection will have no effect if an external clock source is used. If the timer is in the Event Counter or Pulse Width Capture Mode, the active transition edge type is selected by the logic level of the TEG bit in the Timer Control Register.

• **TMRC Register**

Bit	7	6	5	4	3	2	1	0
Name	TM1	TM0	TS	TON	TEG	TPSC2	TPSC1	TPSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7~6 **TM1~TM0**: Timer/Event Counter operating mode selection
00: Unused
01: Event Counter Mode
10: Timer Mode
11: Pulse Width Capture Mode

Bit 5 **TS**: Timer f_{TP} clock source selection
0: f_{SYS}
1: f_{SUB}

Bit 4 **TON**: Timer/Event Counter counting enable
0: Disable
1: Enable

Bit 3 **TEG**: Timer/Event Counter active edge selection
Event Counter Mode
0: Count on rising edge
1: Count on falling edge
Pulse Width Capture Mode
0: Start counting on falling edge, stop on rising edge
1: Start counting on rising edge, stop on falling edge

Bit 2~0 **TPSC2~TPSC0**: Timer internal clock selection
000: f_{TP}
001: $f_{TP}/2$
010: $f_{TP}/4$
011: $f_{TP}/8$
100: $f_{TP}/16$
101: $f_{TP}/32$
110: $f_{TP}/64$
111: $f_{TP}/128$

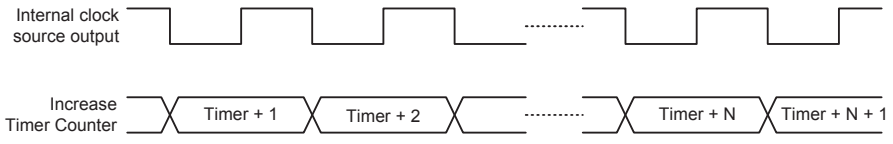
Timer/Event Counter Operating Modes

The Timer/Event Counter can operate in one of three operating modes, Timer Mode, Event Counter Mode or Pulse Width Capture Mode. The operating mode is selected using the TM1 and TM0 bits in the TMRC register.

Timer Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "10" respectively. In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows.

When operating in this mode the internal clock f_{TP} is used as the timer clock, which can be selected to be devirved from f_{SYS} or f_{SUB} by setting the TS bit in the TMRC register. The division of the f_{TP} clock is selected by the TPSC2~TPSC0 bits in the same register. The timer-on bit TON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increases by one. When the timer reaches its maximum 8-bit, FFH Hex, value and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. It should be noted that in the Timer mode, even if the device is in the IDLE/SLEEP mode, if the selected internal clock is still activated and a timer overflow occurs, it will generate a timer interrupt and corresponding wake-up source.



Timer Mode Timing Chart

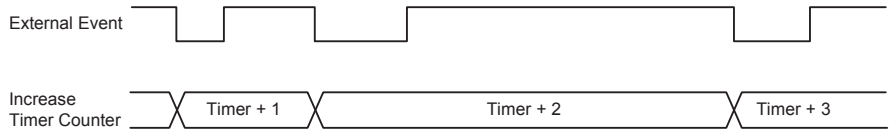
Event Counter Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "01" respectively. In this mode, a number of externally changing logic events, occurring on the external timer TC pin, can be recorded by the Timer/Event Counter.

When operating in this mode, the external timer pin, TC, is used as the Timer/Event Counter clock source. After the other bits in the Timer Control Register have been properly configured, the enable bit TON, can be set high to enable the Timer/Event Counter. If the Active Edge Selection bit, TEG, is low, the Timer/Event Counter will increase each time the TC pin receives a low to high transition. If the TEG bit is high, the counter will increase each time the TC pin receives a high to

low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting.

As the external timer pin TC is pin-shared with other pin functions, the TC pin function must first be selected using relevant pin-shared function selection bits. The pin must also be set as an input by setting the corresponding bit in the port control register. It should be noted that in the Event Counter mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (TEG=1)

Pulse Width Capture Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "11" respectively. In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin.

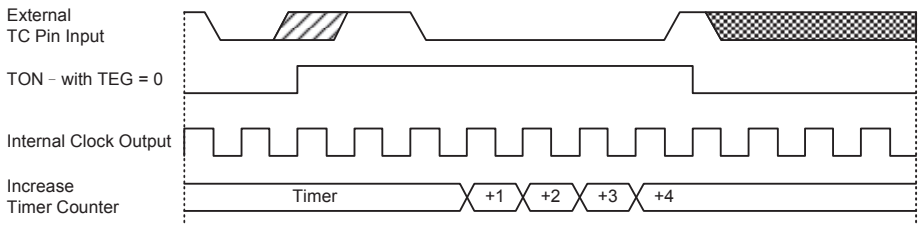
When operating in this mode the internal clock f_{TP} is used as the timer clock, which can be selected to be devirved from f_{SYS} or f_{SUB} by setting the TS bit in the TMRC register. The division of the f_{TP} clock is selected by the TPSC2~TPSC0 bits in the same register. After the other bits in the Timer Control Register have been properly configured, the enable bit TON, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the TC pin.

If the active Edge Selection bit TEG is low, once a high to low transition has been received on the TC pin, the Timer/Event Counter will start counting based on the internal selected clock source until the TC pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. If the Active Edge Selection bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the TC pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will then be automatically reset to zero. It is important to note that in the pulse width capture Mode, the enable bit is automatically reset to zero when the external control signal on the TC pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under application program control.

The residual value in the Timer/Event Counter, which can now be read by the program, therefore represents the length of the pulse received on the TC pin. As the enable bit has now been reset, any further transitions on the external timer pin will be ignored. The timer cannot begin further pulse width capture until the enable bit is set high again by the application program. In this way, single shot pulse measurements can be easily made. It should be noted that in this mode the Timer/Event Counter is controlled by logical transitions on the external timer pin and not by the logic level. When the Timer/Event Counter is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting.

As the external timer pin TC is pin-shared with other pin functions, the TC pin function must first be selected using relevant pin-shared function selection bits. The pin must also be set as an input by setting the corresponding bit in the port control register. It should be noted that in the Pulse Width Capture mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC pin if the internal clock source is still

activated and the external signal continues to change state. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Note: Internal Clock Output is sampled at every falling edge of T1.

Pulse Width Capture Mode Timing Chart (TEG=0)

Programming Considerations

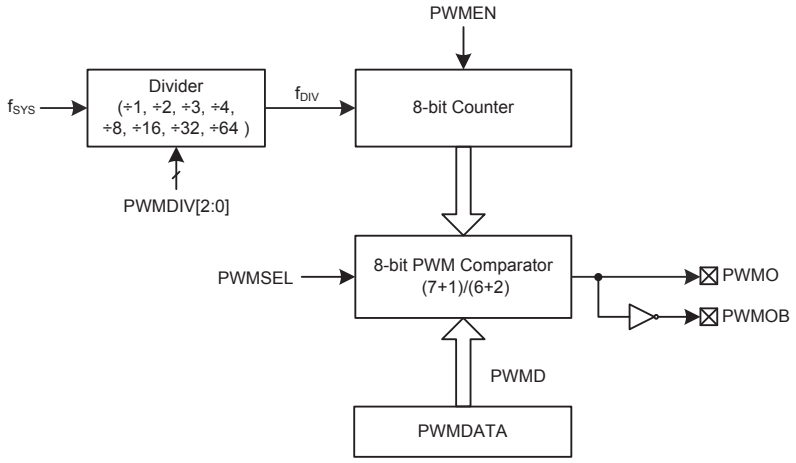
When running in the Timer Mode, the internal timer clock is used as the timer clock source and is therefore synchronised with the overall operation of the microcontroller. In this mode when the timer register is full, the microcontroller will generate an internal interrupt signal directing the program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

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

When the Timer/Event Counter overflows, its corresponding interrupt request flag in the interrupt control register will be set to generate an interrupt signal. If the Timer/Event Counter interrupt is enabled this will in turn allow program branch to its interrupt vector. However irrespective of whether the interrupt is enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in the IDLE/SLEEP mode. This situation may occur if the Timer/Event Counter internal clock source is still activated or if the external signal continues to change state. In such cases, the Timer/Event Counter will continue to count and if an overflow occurs the device will be woken up. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the “HALT” instruction to enter the IDLE/SLEEP mode.

Pulse Width Modulator

The device contains an 8-bit pulse width modulation function. Useful for applications such as buzzer control, the PWM function provides a pair of complementary outputs with a fixed frequency but with a duty cycle that can be varied by setting particular values into the PWMDATA register.



PWM Block Diagram

PWM Registers Description

There two registers control the overall operation of the Pulse Width Modulator channel. These are the data register, PWMDATA and a single control register, PWMC.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PWMC	PWMEN	PWMDIV2	PWMDIV1	PWMDIV0	—	—	—	PWMSEL
PWMDATA	D7	D6	D5	D4	D3	D2	D1	D0

PWM Register List

• PWMC Register

Bit	7	6	5	4	3	2	1	0
Name	PWMEN	PWMDIV2	PWMDIV1	PWMDIV0	—	—	—	PWMSEL
R/W	R/W	R/W	R/W	R/W	—	—	—	R/W
POR	0	0	0	0	—	—	—	0

Bit 7 **PWMEN**: PWM enable control
0: Disable
1: Enable

Note: 1. When the PWMEN bit is cleared to zero to disable the PWM function, the internal PWMO signal will be pulled low. However, the external PWMO pin status will be floating when the multi-functional pin is selected as a PWMO output and the PWMEN bit is cleared to zero.

2. After the PWMEN bit is set high, the first PWM modulation cycle period and duty may not match the expected waveform. The PWM output will be normal after the first PWM cycle.

Bit 6~4 **PWMDIV2~PWMDIV0**: f_{DIV} frequency selection
000: $f_{DIV}=f_{SYS}$
001: $f_{DIV}=f_{SYS}/2$
010: $f_{DIV}=f_{SYS}/3$
011: $f_{DIV}=f_{SYS}/4$
100: $f_{DIV}=f_{SYS}/8$
101: $f_{DIV}=f_{SYS}/16$
110: $f_{DIV}=f_{SYS}/32$
111: $f_{DIV}=f_{SYS}/64$

Bit 3~1 Unimplemented, read as "0"

Bit 0 **PWMSEL**: PWM type selection
0: (6+2) bits mode
1: (7+1) bits mode

- **PWMDATA 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**: PWM output duty cycle PWMD bit 7 ~ bit 0
Note that the duty cycle value PWMD, once being changed, will reflect on the PWMO output signal immediately. Therefore, a sudden PWM duty change will occur in the current PWM cycle, resulting in undesired waveform, which only lasts a PWM cycle. Starting from the next new PWM cycle, the PWM duty will be in accordance with the new PWMD value.

PWM Operation

The PWMC register and PWMDATA register are assigned to the Pulse Width Modulator channel. The PWM channel has a data register, PWMDATA, the content of which is an 8-bit data, abbreviated as PWMD, representing the overall duty cycle of one modulation cycle of the output waveform. To increase the PWM modulation frequency, each modulation cycle is subdivided into two or four individual modulation subsections, known as the (7+1) bits mode or (6+2) bits mode respectively. The PWM counter clock frequency, f_{DIV}, comes from f_{SYS} or its division. The required mode, clock source selection and the enable/disable control for the PWM channel are selected using the PWMC register. Note that when using the PWM, it is only necessary to write the required value into the PWMDATA register and select the required mode, clock source and enable/disable control using the PWMC register, the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware.

This method of dividing the original modulation cycle into a further 2 or 4 sub-cycles enables the generation of higher PWM frequencies which allow a wider range of applications to be served. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood. As the PWM clock is f_{DIV}, and as the PWM value is 8-bits wide, the overall PWM cycle frequency is $f_{DIV}/256$. However, when in the (7+1) mode the PWM modulation frequency will be $f_{DIV}/128$, while the PWM modulation frequency for the (6+2) mode will be $f_{DIV}/64$.

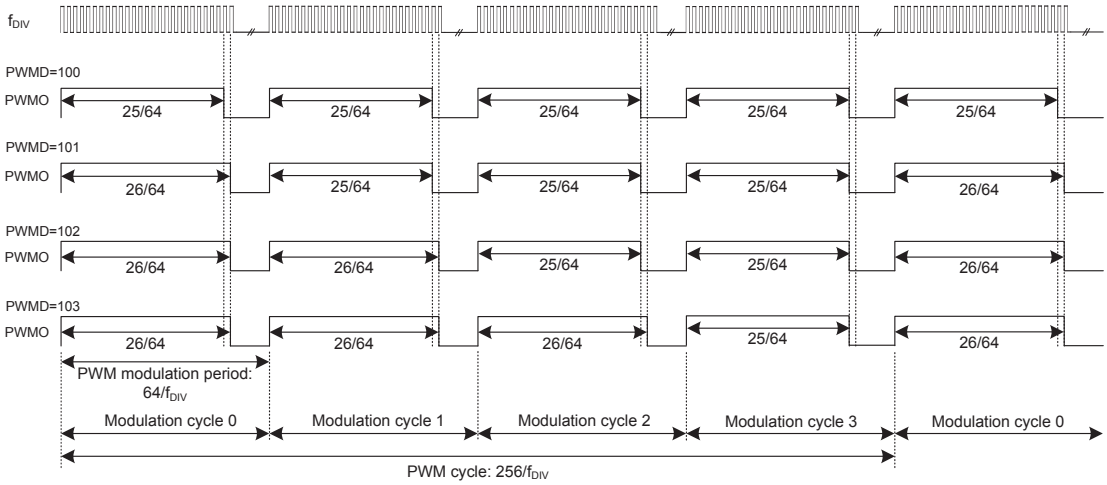
(6+2) Bits PWM Mode Modulation

A (6+2) bits mode PWM cycle is divided into four modulation cycles, which are named as Modulation cycle 0 ~ Modulation cycle 3. Each modulation cycle has 64 PWM input clock periods. In the (6+2) bits PWM mode, the PWMD is divided into two groups. Group 1 is denoted by DC which is the value of PWMD bit 7 ~ bit 2. Group 2 is denoted by AC which is the value of PWMD bit 1 ~ bit 0. The modulation frequency, modulation cycle duty, PWM cycle frequency and PWM cycle duty of the (6+2) bits mode PWM output signals are summarized in the following table.

Modulation Frequency	Modulation Cycle i	Modulation Cycle Duty		PWM Cycle Frequency	Duty Cycle
$f_{DIV}/64$	i=0~3	i<AC	(DC+1)/64	$f_{DIV}/256$	PWMD/256
		i≥AC	DC/64		

(6+2) Bits PWM Mode Summary

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



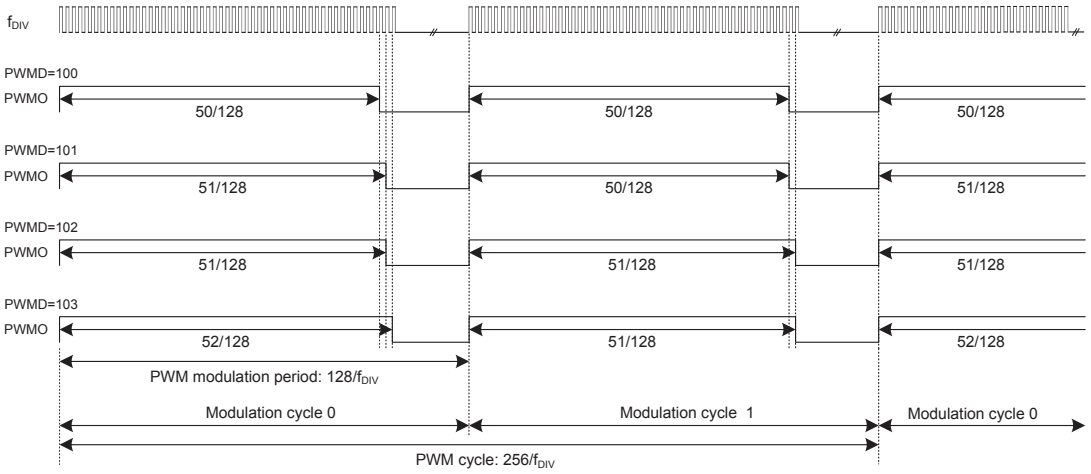
(6+2) Bits PWM Mode Modulation Waveform

(7+1) Bits PWM Mode Modulation

A (7+1) bits mode PWM cycle is divided into two modulation cycles, which is named as Modulation cycle 0 ~ Modulation cycle 1. Each modulation cycle has 128 PWM input clock periods. In the (7+1) bits PWM mode, the PWMD is divided into two groups. Group 1 is denoted by DC which is the value of PWMD bit 7 ~ bit 1. Group 2 is denoted by AC which is the value of PWMD bit 0. The modulation frequency, modulation cycle duty, PWM cycle frequency and PWM cycle duty of the (7+1) bits mode PWM output signals are summarized in the following table.

Modulation Frequency	Modulation Cycle i	Modulation Cycle Duty		PWM Cycle Frequency	PWM Cycle Duty
$f_{DIV}/128$	i=0~1	i<AC	(DC+1)/128	$f_{DIV}/256$	PWMD/256
		i≥AC	DC/128		

(7+1) Bits PWM Mode Summary



(7+1) Bits PWM Mode Modulation Waveform

Interrupts

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

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. The number of registers falls into two categories. The first is the INTC register which sets the primary interrupts, the second is the INTEG register to set the external interrupt trigger edge type.

The interrupt register contains a number of enable bits to enable or disable individual registers 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
Global	EMI	—
INT Pin	INTE	INTF
Time Base	TBE	TBF
Timer/Event Counter	TE	TF

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC	—	TF	TBF	INTF	TE	TBE	INTE	EMI

Interrupt Register List

• **INTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INTS1	INTS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **INTS1~INTS0**: Interrupt edge control for INT pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

• **INTC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TF	TBF	INTF	TE	TBE	INTE	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 **TF**: Timer/Event Counter interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TBF**: Time Base interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INTF**: INT interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TE**: Timer/Event Counter interrupt control
0: Disable
1: Enable

Bit 2 **TBE**: Time Base interrupt control
0: Disable
1: Enable

Bit 1 **INTE**: INT interrupt control
0: Disable
1: Enable

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

Interrupt Operation

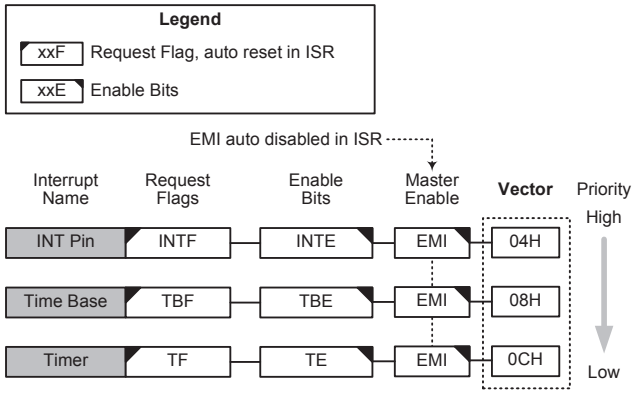
When the conditions for an interrupt event occur, such as a timer overflow, the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set 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 diagram with their order of priority. All interrupt sources have their own individual vector. Once an interrupt subroutine is serviced, all the 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

External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the external interrupt enable bit, INTE, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if its external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be set as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTF, will be automatically

reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selection on the external interrupt pin will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

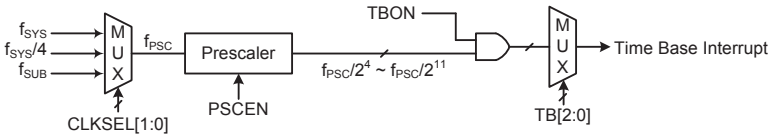
Timer/Event Counter Interrupt

An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TF, is set, which occurs when the Timer/Event Counter overflows. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Timer/Event Counter Interrupt enable bit, TE, must first be set. When the interrupt is enabled, the stack is not full and the Timer/Event Counter overflows, a subroutine call to its interrupt vector, will take place. When the interrupt is serviced, the Timer/Event Counter Interrupt flag, TF, 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 signals from the timer function. When this happens its interrupt request flag TBF will be set. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI and Time Base enable bit, TBE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to its interrupt vector location will take place. When the interrupt is serviced, the interrupt request flag, TBF, 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_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} 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 CLKSEL[1:0] bits in the PSCR register.



Time Base Interrupt

- **PSCR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PSCEN	CLKSEL1	CLKSEL0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **PSCEN**: Prescaler control
0: Disable
1: Enable

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source f_{PSC} selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

• **TBC Register**

Bit	7	6	5	4	3	2	1	0
Name	TBON	—	—	—	—	TB2	TB1	TB0
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB0ON**: Time Base control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Time Base Time-out period selection
000: $2^4/f_{PSC}$
001: $2^5/f_{PSC}$
010: $2^6/f_{PSC}$
011: $2^7/f_{PSC}$
100: $2^8/f_{PSC}$
101: $2^9/f_{PSC}$
110: $2^{10}/f_{PSC}$
111: $2^{11}/f_{PSC}$

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 the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pin 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.

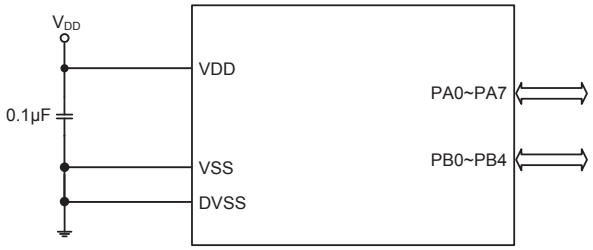
It is recommended that programs do not use the "CALL" instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged 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



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 or 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 up to 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 is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

Operation: ACC ← ACC − [m] − $\overline{C}$

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

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

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

Operation: [m] ← ACC − [m] − $\overline{C}$

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

SDZ [m] Skip if decrement Data Memory is 0

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

Operation:
[m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

Description The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no TBHP) 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

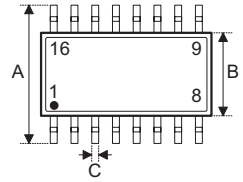
Package Information

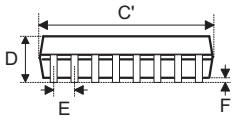
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consul

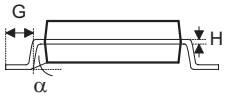
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- Carton information

16-pin NSOP (150mil) Outline Dimensions







Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C'	—	0.390 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	—	6.000 BSC	—
B	—	3.900 BSC	—
C	0.31	—	0.51
C'	—	9.900 BSC	—
D	—	—	1.75
E	—	1.270 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°