HOLTEK

8-Bit Flash MCU with Op Amps & Comparators

HT45F12

Revision: V1.00 Date: September 26, 2012

www.holtek.com

Features

CPU Features

- Operating voltage:
  - f_{SYS} = 910kHz: 2.2V~5.5V
  - f_{SYS} = 2MHz: 2.2V~5.5V
  - f_{SYS} = 4MHz: 2.2V~5.5V
  - f_{SYS} = 8MHz: 3.3V~5.5V
  - f_{SYS} = 12MHz: 4.5V~5.5V
- Tiny Power technology for low power operation
- Power down and wake-up functions to reduce power consumption
- Oscillator types:
  - External Crystal – HXT
  - External RC – ERC
  - Internal RC – HIRC
  - Internal 32kHz RC – LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- Fully integrated internal 32kHz, 910kHz, 2MHz, 4MHz and 8MHz oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory:1K×15
- RAM data memory:64×8
- EEPROM Memory:32×8
- Watchdog Timer function
- Up to 18 bidirectional I/O lines
- Multiple pin-shared external interrupts
- Single 8-bit programmable Timer/Event Counter with overflow interrupt
- Dual Time-Base functions
- Dual Comparator functions
- Dual Operational Amplifiers functions
- PFD/Buzzer for audio frequency generation
- Internal 2.4V/3.3V LDO
- Low voltage reset function
- Package: 16-pin NSOP, 20-pin SSOP

General Description

The device is Flash Memory Tinypower Type 8-bit high performance RISC architecture microcontroller.

Offering users the convenience of Flash Memory multi-programming features, this 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 EEPROM memory for storage of non-volatile data such as calibration data etc.

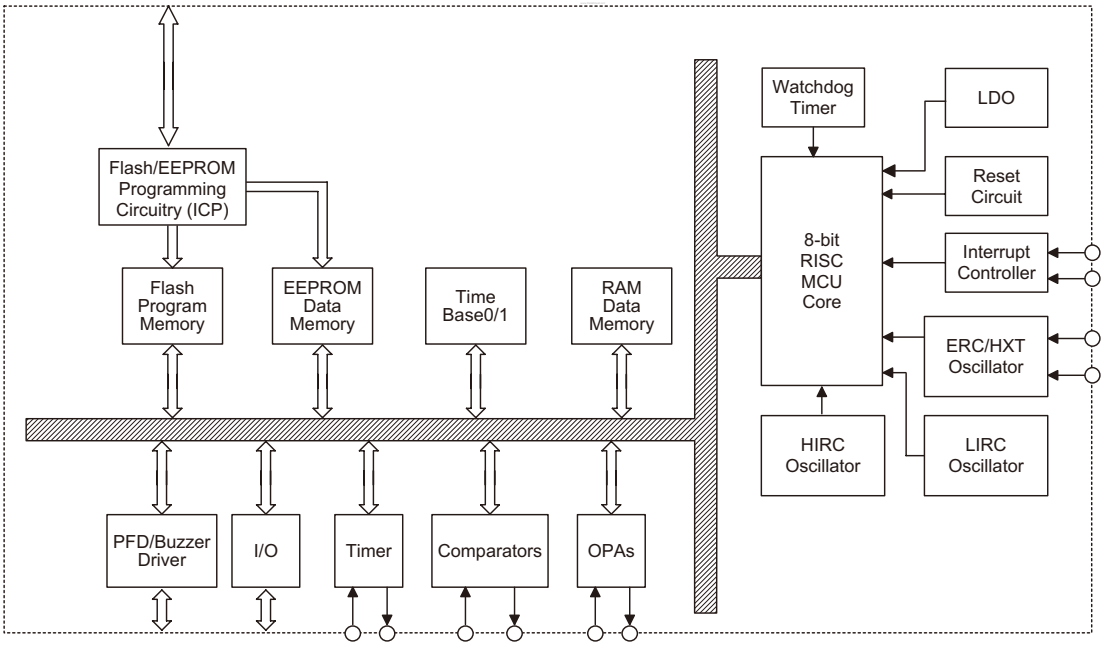
Analog features include dual Operational Amplifiers, dual Comparators and one internal 2.4V or 3.3V LDO (Low Drop Out) for voltage regulator. Protective features such as an internal Watchdog Timer and Low Voltage Reset ensure that reliable operation is maintained in hostile electrical environments.

A full choice of HXT, ERC, HIRC and LIRC oscillator functions are provided including a fully integrated system oscillator which requires no external components for its implementation. The unique Holtek TinyPower technology also gives the device extremely low current consumption characteristics, an extremely important consideration in the present trend for low power battery powered applications. The usual Holtek MCU features such as power down and wake-up functions, oscillator options, programmable frequency divider, etc. combine to ensure user applications require a minimum of external components.

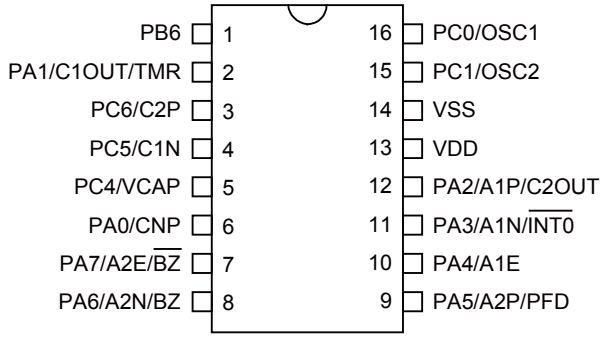
The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the device will find excellent use in applications such as electronic metering, environmental monitoring, handheld instruments, electronically controlled tools, motor driving and many others.

Block Diagram

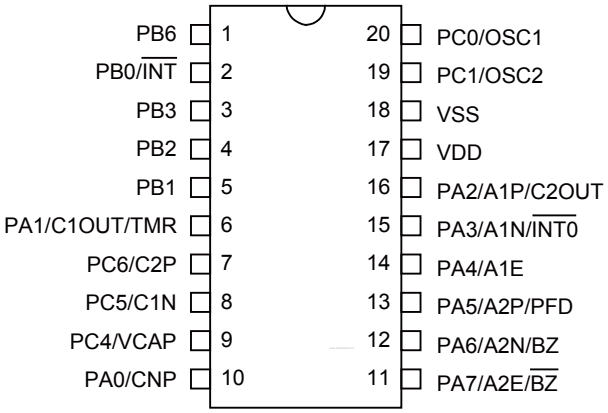
The following block diagram illustrates the main functional blocks.



Pin Assignment



HT45F12
16 NSOP-A



HT45F12
20 SSOP-A

Pin Description

Pin Name	Function	OPT	I/T	O/T	Description
PA0/CNP	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	CNP	CMP1C1	CMPI	CMOS	Comparator input pin
PA1/C1OUT/ TMR	PA1	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	C1OUT	CMP1C1	—	CMPO	Comparator 1 output pin
	TMR	—	ST	—	External Timer clock input
PA2/A1P/ C2OUT	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	A1P	OPA1C1	OPAI	—	OPA1 non-inverting input pin
	C2OUT	CMP2C1	—	CMPO	Comparator 2 output pin
PA3/A1N/$\overline{INT0}$	PA3	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	A1N	OPA1C1	OPAI	—	OPA1 inverting input pin
	$\overline{INT0}$	—	ST	—	External interrupt 0 input pin
PA4/A1E	PA4	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	A1E	OPA1C1	—	OPAO	OPA1 output pin
PA5/A2P/PFD	PA5	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	A2P	OPA2C1	OPAI	—	OPA2 non-inverting input pin
	PFD	MISC	—	CMOS	PFD output
PA6/A2N/BZ	PA6	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	A2N	OPA2C1	OPAI	—	OPA2 inverting input pin
	BZ	BPCTL	—	CMOS	Buzzer output
PA7/A2E/$\overline{BZ}$	PA7	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	A2E	OPA2C1	—	OPAO	OPA2 output pin
	$\overline{BZ}$	BPCTL	—	CMOS	Complementary buzzer output
PB0/$\overline{INT1}$	PB0	PBPU MISC	ST	CMOS NMOS	General purpose I/O. Register enabled pull-up and output NMOS structure.
	$\overline{INT1}$	—	ST	—	External interrupt 1 input pin
PB1	PB1	PBPU MISC	ST	CMOS NMOS	General purpose I/O. Register enabled pull-up and output NMOS structure.
PB2	PB2	PBPU MISC	ST	CMOS NMOS	General purpose I/O. Register enabled pull-up and output NMOS structure.
PB3	PB3	PBPU MISC	ST	CMOS NMOS	General purpose I/O. Register enabled pull-up and output NMOS structure.
PB6	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
PC0/OSC1	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	OSC1	CO	HXT	—	HXT/ERC pin
PC1/OSC2	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	OSC2	CO	—	HXT	HXT pin
PC4/VCAP	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	VCAP	LDOC	—	—	LDO output capacitor pin. Connect a 0.1μF capacitor.

Pin Name	Function	OPT	I/T	O/T	Description
PC5/C1N	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	C1N	CMP1C1	CMPI	—	Comparator 1 inverting input pin
PC6/C2P	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	C2P	CMP2C1	CMPI	—	Comparator 2 non-inverting input pin
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground

Note: I/T: Input type
O/T: Output type
OPT: Optional by configuration option (CO) or register option
PWR: Power
CO: Configuration option
ST: Schmitt Trigger input
CMOS: CMOS output
NMOS: NMOS output
HXT: High frequency crystal oscillator
OPAI: Operational Amplifier input
OPAO: Operational Amplifier output
CMPI: Comparator input
CMPO: Comparator output

As the Pin Description Summary table applies to the package type with the most pins, not all of the above listed pins may be present on package types with smaller numbers of pins.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}-0.3V to V_{SS}+6.0V

Input Voltage ..... V_{SS}-0.3V to V_{DD}+0.3V

I_{OL} Total 100mA

I_{OH} Total ..... -100mA

Total Power Dissipation 500mV

Storage Temperature ..... -50°C to 150°C

Operating Temperature ..... -40°C to 150°C

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

D.C. Characteristics

Ta= 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}= 910kHz, (HXT/ERC/HIRC)	2.2	—	5.5	V
			f_{SYS}= 2MHz, (HXT/ERC/ HIRC)	2.2	—	5.5	V
			f_{SYS}= 4MHz, (HXT/ERC/EC/HIRC)	2.2	—	5.5	V
			f_{SYS}= 8MHz, (HXT/ERC/EC/HIRC)	3.3	—	5.5	V
			f_{SYS}= 12MHz, (HXT/ERC/EC)	4.5	—	5.5	V
I_{DD1}	Operating Current (HXT, ERC)	3.3V	No load, f_{SYS}= f_M= 455kHz, LVR off, Comparator off, OPAs off	—	70	110	μA
			No load, f_{SYS}= f_M= 455kHz, LVR on, Comparator on, OPAs off	—	100	150	μA
I_{DD2}	Operating Current (ERC, HIRC)	3.3V	No load, f_M= 910kHz, f_{SYS}= f_{SLOW}= 455kHz, LVR off, Comparator off, OPAs off	—	90	135	μA
			No load, f_M= 910kHz, f_{SYS}= f_{SLOW}= 455kHz, LVR on, Comparator on, OPAs off	—	120	180	μA
I_{DD3}	Operating Current (ERC, HIRC)	3.3V	No load, f_{SYS}= f_M= 910kHz, LVR off, Comparator off, OPAs off	—	110	170	μA
			No load, f_{SYS}= f_M= 910kHz, LVR on, Comparator on, OPAs off	—	160	240	μA
I_{DD4}	Operating Current (ERC, HXT)	3.3V	No load, f_{SYS}= f_M= 1MHz, LVR off, Comparator off, OPAs off	—	120	180	μA
			No load, f_{SYS}= f_M=1MHz, LVR on, Comparator on, OPAs off	—	170	260	μA
I_{DD5}	Operating Current (HXT, ERC, HIRC)	3.3V	No load, f_{SYS}= f_M= 2MHz, LVR off, Comparator off, OPAs off	—	170	260	μA
			No load, f_{SYS}= f_M= 2MHz, LVR on, Comparator on, OPAs off	—	200	300	μA
I_{DD6}	Operating Current (HXT, ERC, HIRC)	3V	No load, f_{SYS}= f_M= 4MHz	—	420	630	μA
		5V		—	700	1000	μA
I_{DD7}	Operating Current (EC mode)	3V	No load, f_{SYS}= f_M= 4MHz	—	330	500	μA
		5V		—	550	820	μA
I_{DD8}	Operating Current (HXT, ERC, HIRC)	5V	No load, f_{SYS}= f_M= 8MHz	—	1.5	3.0	mA
I_{DD9}	Operating Current (HXT, ERC)	5V	No load, f_{SYS}= f_M=12MHz	—	2.5	5.0	mA
I_{DD10}	Operating Current (Slow Mode, f_M=4MHz) (HXT, ERC, HIRC)	3V	No load, f_{SYS}= f_{SLOW}= 1MHz	—	200	300	μA
		5V		—	400	600	μA
I_{DD11}	Operating Current (Slow Mode, f_M=4MHz) (HXT, ERC, HIRC)	3V	No load, f_{SYS}= f_{SLOW}= 2MHz	—	250	375	μA
		5V		—	560	840	μA
I_{DD12}	Operating Current (Slow Mode, f_M=8MHz) (HXT, ERC, HIRC)	3V	No load, f_{SYS}= f_{SLOW}= 2MHz	—	300	450	μA
		5V		—	680	1020	μA
I_{DD13}	Operating Current (Slow Mode, f_M=8MHz) (HXT, ERC, HIRC)	3V	No load, f_{SYS}= f_{SLOW}= 4MHz	—	450	800	μA
		5V		—	1000	1500	μA
I_{DD14}	Operating Current (f_{SYS}=f_{LIRC})	3V	No load, WDT off	—	10	20	μA
		5V		—	20	35	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB1}	Standby Current (Sleep) (f_{SYS}, f_{SUB}, f_S, f_{WDT}= off)	3V	No load, system HALT, WDT off	—	0.1	1.0	μA
		5V		—	0.2	2.0	μA
I_{STB2}	Standby Current (Sleep) (f_{SYS} Off, f_S On, f_{WDT}= f_{SUB}= f_{LIRC})	3V	No load, system HALT, WDT on	—	2	4	μA
		5V		—	4	6	μA
I_{STB3}	Standby Current (Idle) (f_{SYS} Off, f_{WDT} Off, f_S= f_{SUB}= f_{LIRC})	3V	No load, system HALT, WDT off	—	4	6	μA
		5V		—	6	9	μA
V_{IL1}	Input Low Voltage for I/O, TMR, $\overline{INT0}$ and $\overline{INT1}$	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O, TMR, $\overline{INT0}$ and $\overline{INT1}$	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL3}	Input Low Voltage (PB1~PB3)	5V	—	—	—	1	V
V_{IH3}	Input Low Voltage (PB1~PB3)	5V	—	2	—	—	V
V_{LVR1}	Low Voltage Reset	—	V_{LVR}= 2.10V	-5%× Typ.	2.10	-5%× Typ.	V
V_{LVR2}			V_{LVR}= 2.55V		2.55		
V_{LVR3}			V_{LVR}= 3.15V		3.15		
V_{LVR4}			V_{LVR}= 3.80V		3.80		
I_{OL}	I/O Port Sink Current	3V	V_{OL}= $0.1V_{DD}$	6	12	—	mA
		5V		10	25	—	mA
I_{OH}	I/O Port Source Current	3V	V_{OH}= $0.9V_{DD}$	-2	-4	—	mA
		5V		-5	-8	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
I_{LVR}	DC current when LVR turn on	—	LVR disable → LVR enable	—	10	20	μA
				—	15	30	μA
t_{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t_{SRESET}	Software Reset Width to Reset	—	—	45	90	120	μs

Note: $t_{SUB} = 1/f_{SUB}$

A.C. Characteristics

Ta= 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
f_{SYS1}	System Clock (HXT, ERC, HIRC)	—	2.2~5.5V	400	—	4000	kHz
			3.3~5.5V	400	—	8000	kHz
			4.5~5.5V	400	—	12000	kHz
f_{SYS2}	8MHz HIRC	3.3V	Ta= 25°C	-2%	8	+2%	MHz
		3.3V	Ta= -40°C~85°C	-5%	8	+5%	MHz
		2.7~5.5V	Ta= -40°C~85°C	-10%	8	+10%	MHz
f_{SYS3}	4MHz HIRC	3.3V	Ta= 25°C	-2%	4	+2%	MHz
		3.3V	Ta= -40°C~85°C	-5%	4	+5%	MHz
		2.7~5.5V	Ta= -40°C~85°C	-10%	4	+10%	MHz
f_{SYS4}	2MHz HIRC	3.3V	Ta= 25°C	-2%	2	+2%	MHz
		3.3V	Ta= -40°C~85°C	-5%	2	+5%	MHz
		2.7~5.5V	Ta= -40°C~85°C	-10%	2	+10%	MHz
f_{SYS5}	910kHz HIRC	3.3V	Ta= 25°C	-2%	0.91	+2%	MHz
		3.3V	Ta= -40°C~85°C	-5%	0.91	+5%	MHz
		2.7~5.5V	Ta= -40°C~85°C	-10%	0.91	+10%	MHz
f_{LIRC}	System Clock (LIRC)	5V	Ta= 25°C	-10%	32	+10%	kHz
		2.2~5.5V	Ta= -40°C to 85°C	-50%	32	+60%	kHz
t_{INT}	Interrupt Minimum Pulse Width	—	—	1	3.3	5	μs
t_{RSTD}	System Reset Delay Time (Power On Reset)	—	—	25	50	100	ms
	System Reset Delay Time (Any Reset except Power On Reset)	—	—	8.3	16.7	33.3	ms
t_{SST}	System start-up timer period (wake-up from HALT, f_{SYS} off at HALT state)	—	f_{SYS}= HXT	128	—	—	t_{SYS}
			f_{SYS}= ERC or HIRC	16	—	—	
			f_{SYS}= LIRC	2	—	—	
	System Start-up Timer Period (Wake-up from HALT, f_{SYS} on at HALT state)	—	—	2	—	—	

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

OP Amplifier Electrical Characteristics

Ta= 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
D.C. Characteristic							
V_{DD}	Operating voltage	—	—	2.2	—	5.5	V
I_{DD}	Quiescent current	5V	No load	—	200	350	μA
V_{OPOS1}	Input offset voltage	5V	AxOF4~0= (10000)	-15	—	+15	mV
V_{OPOS2}	Input offset voltage	5V	By calibration	-4	—	+4	mV
I_{OPOS}	Input offset current	—	V_{DD}= 5V, V_{CM}= 1/2V_{DD}, Ta= -40°C~85°C	—	10	—	nA
V_{CM}	Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}-1.4	V
PSRR	Power Supply Rejection Ratio	—	—	58	80	—	dB
CMRR	Common Mode Rejection Ratio	—	V_{DD}= 5V V_{CM}= 0~(V_{DD}-1.4V)	58	80	—	dB
A.C. Characteristic							
A_{OL}	Open Loop Gain	—	—	60	80	—	dB
SR	Slew Rate +, Rate -	—	No load	—	0.1	—	V/μs
GBW	Gain Band Width	—	R_L= 1MΩ, C_L= 100pF	100k	2M	—	Hz

Comparator Electrical Characteristics

Ta= 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
V_{DDC}	Comparator Operating Voltage	—	—	2.2	—	5.5	mV
I_{DDC}	Comparator Operating Current	3.3V	—	—	20	40	μA
		5V	—	—	30	60	μA
V_{CPOS1}	Comparator Input Offset Voltage	—	CxOF4~0=(10000)	-10	—	+10	mV
V_{CPOS2}	Comparator Input Offset Voltage	—	By calibration	-4	—	+4	mV
V_{CM}	Comparator Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}-1.4V	V
AOL	Comparator Open Loop Gain	—	—	60	80	—	dB
t_{PD1}	Comparator Response Time	—	With 2mV overdrive	—	—	10	μs
t_{PD2}	Comparator Response Time	—	With 10mV overdrive	—	—	1.5	μs

LDO 2.4V Electrical Characteristics

Ta= 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DDIN}	Supply Voltage	—	—	3.0	3.3	3.6	V
V_{DDOUT}	Output Voltage	—	—	-5%	2.4	+5%	V
I_{DD}	Current Consumption	—	After startup, no load	—	30	—	μA

LDO 3.3V Electrical Characteristics

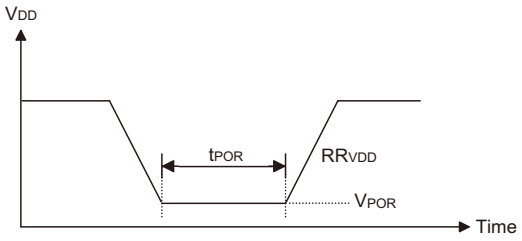
Ta= 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DDIN}	Supply Voltage	—	—	4.5	5	5.5	V
V_{DDOUT}	Output Voltage	—	—	-5%	3.3	+5%	V
I_{DD}	Current Consumption	—	After startup, no load	—	50	—	μA

Power-on Reset Characteristics

Ta= 25°C

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



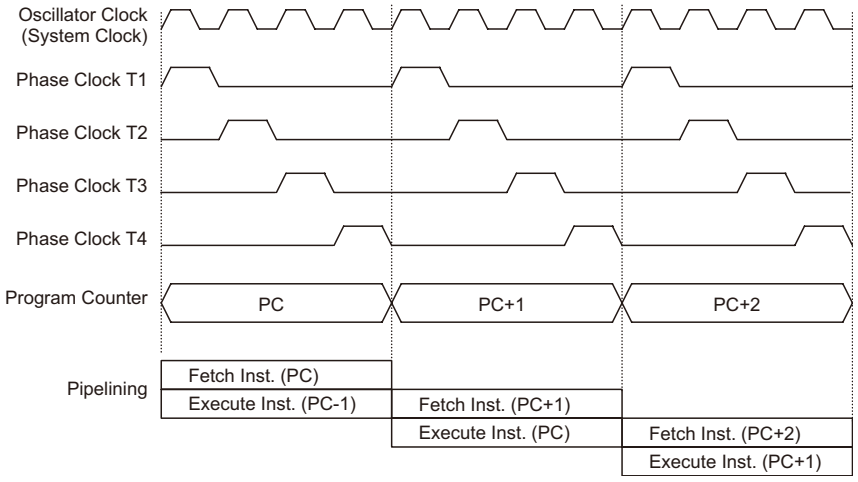
System Architecture

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

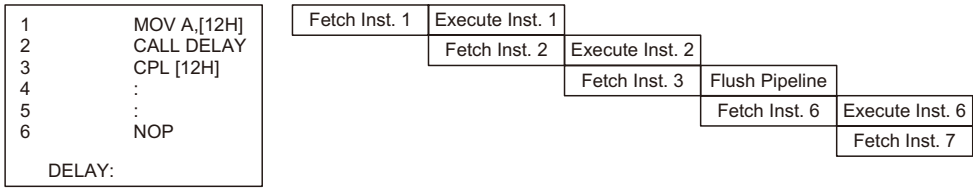
Clocking and Pipelining

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

For instructions involving branches, such as jump or call instructions, two instruction 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 – PC

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. It must be noted that only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by user.

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

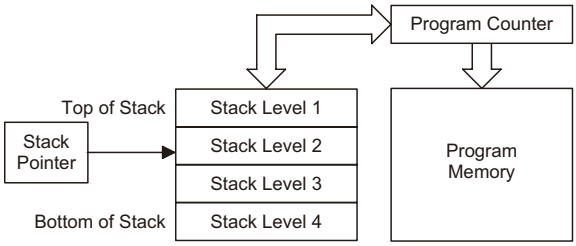
Program Counter	
Program Counter High Byte	**PCL Register**
PC9~PC8	PCL7~PCL0

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

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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 4 levels and 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 this 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.

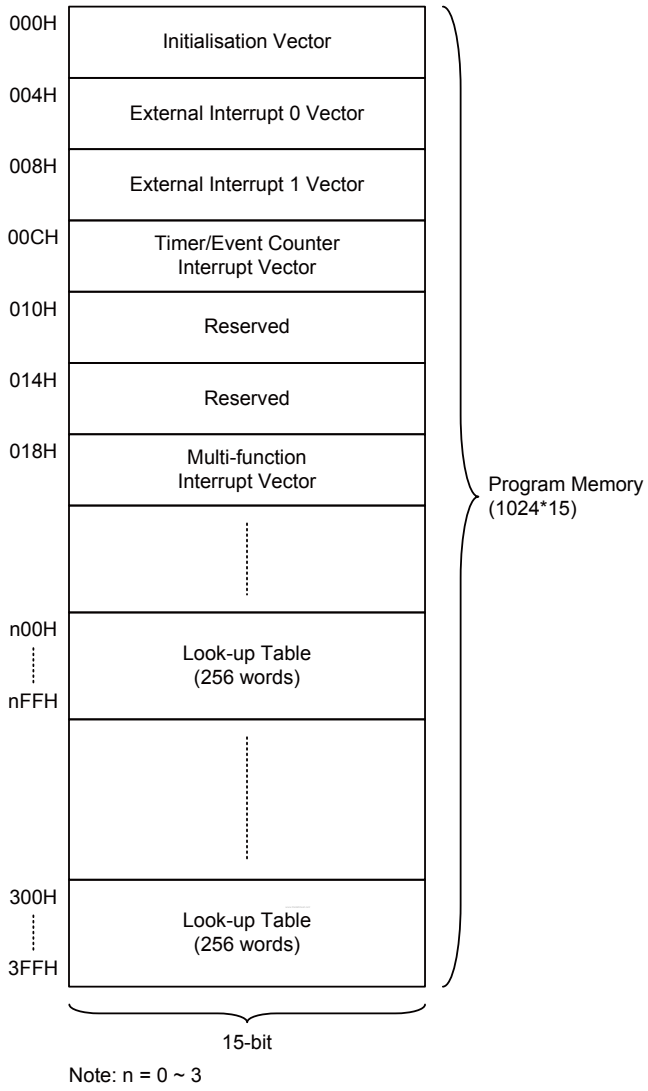
Structure

The Program Memory has a capacity of 1Kx15 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupts entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.

Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts.

- Location 000H
  This vector is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.
- Location 004H
  This vector is used by the external interrupt 0. If the external interrupt pin receives an active edge, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.
- Location 008H
  This vector is used by the external interrupt 1. If the external interrupt pin receives an active edge, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.
- Location 00CH
  This internal vector is used by the Timer/Event Counter. If a Timer/Event Counter overflow occurs, the program will jump to this location and begin execution if the timer/event counter interrupt is enabled and the stack is not full.
- Location 018H
  This internal vector is used by the Multi-function Interrupt. When the Time Base overflows, a Comparator output interrupt, an EEPROM Write or Read cycle ends interrupt, the program will jump to this location and begin execution if the relevant interrupt is enabled and the stack is not full.

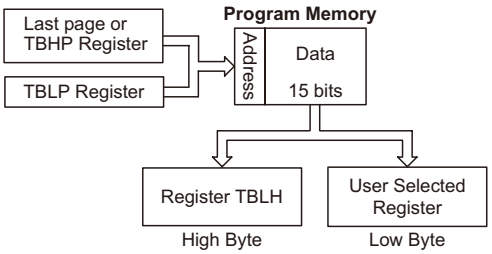


Program Memory Structure

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the address of the look up data to be retrieved in the table pointer register, TBLP and TBHP. These registers define the total address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the Program Memory using the "TABRDC[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.



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

Table Location

Note: b9~b0: Table location bits

PC9~PC8: Current Program Counter bits

@7~@0: Table Pointer TBLP bits

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 is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "306H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRDC [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRDL [m]" instruction is executed.

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
mov a,07h          ; initialise high table pointer
mov tbhp,a
:
:
tabrdc 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
tabrdc tempreg2    ; transfers value in table referenced by table pointer data at program
                   ; memory address "305H" transferred to tempreg2 and TBLH in this
                   ; example the data "1AH" is transferred to tempreg1 and data "0FH" to
                   ; register tempreg2
:
:
org 300h           ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

In Circuit Programming

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 5-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.

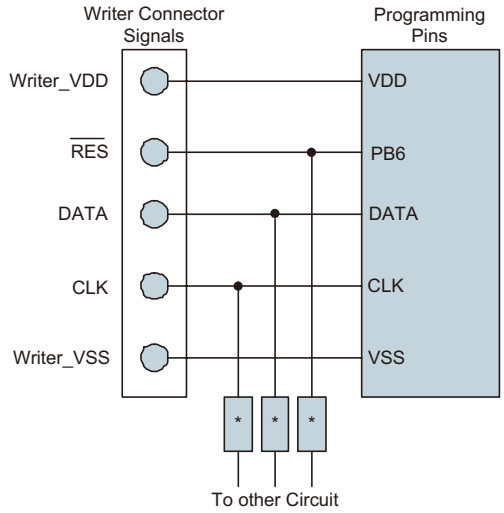
MCU Programming Pins	Function
DATA	Serial Data Input/Output
CLK	Serial Clock
$\overline{\text{RES}}$	Device Reset
VDD	Power Supply
VSS	Ground

The Program Memory and EEPROM data memory can both be programmed serially in-circuit using this 5-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 and one line for the reset. The technical details regarding the in-circuit programming of the device are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process the $\overline{\text{RES}}$ pin will be held low by the programmer disabling the normal operation of the microcontroller and taking control of the PA0 and PA2 I/O pins for data and clock programming purposes. The user must take care to ensure that no other outputs are connected to these two pins.

Programmer Pin	MCU Pins
$\overline{RES}$	PB6
DATA	PA0
CLK	PA2

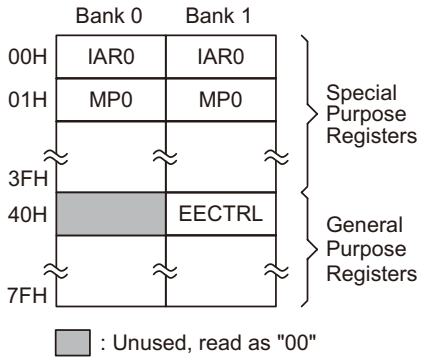
Programmer and MCU Pins



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

RAM 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.



Data Memory Structure

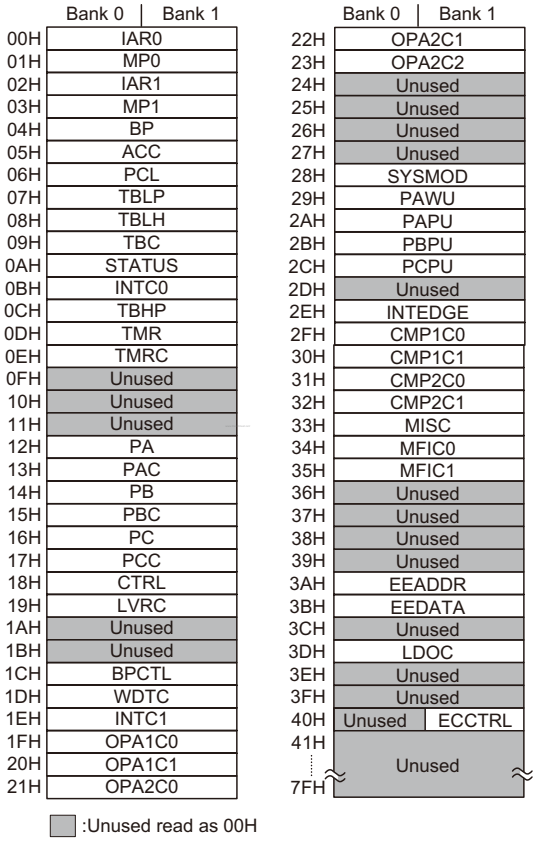
Note: Most of the Data Memory bits can be directly manipulated using the SET [m].i and CLR [m].i with the exception of a few dedicated bits. The Data Memory can also be accessed through the memory pointer registers.

Structure

Divided into two sections, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which 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 RAM Data Memory is subdivided into 2 banks, known as bank 0 and bank1, The Special Purpose Data Memory registers are accessible in all banks, with the exception of the EECTRL register at address 40H, which is only accessible in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory is the address 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 Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data from Bank0 while the IAR1 and MP1 register pair can access data from any bank. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of "00H" and writing to the registers indirectly will result in no operation.

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank 0, while MP1 and IAR1 are used to access data from all banks according to BP register. Direct Addressing can only be used with Bank 0, all other Banks must be addressed indirectly using MP1 and IAR1.

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´
org00h
start:
     mov a,04h            ; setup size of block
     mov block,a
     mov a,offset adres1  ; Accumulator loaded with first RAM address
     mov mp0,a            ; setup memory pointer with first RAM address
loop:
     clr IAR0             ; clear the data at address defined by mp0
     inc mp0              ; increment memory pointer
     sdz block            ; check if last memory location has been cleared
     jmp loop
continue:
```

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

Bank Pointer – BP

The Data Memory is divided into two Banks, known as Bank 0 and Bank 1. A Bank Pointer, which is bit 0 of the Bank Pointer register is used to select the required Data Memory bank. Only data in Bank 0 can be directly addressed, as data in Bank 1 must be indirectly addressed using Memory Pointer MP1 and Indirect Addressing Register IAR1. Using Memory Pointer MP0 and Indirect Addressing Register IAR0 will always access data from Bank 0, irrespective of the value of the Bank Pointer. Memory Pointer MP1 and Indirect Addressing Register IAR1 can indirectly address data in either Bank 0 or Bank 1 depending upon the value of the Bank Pointer.

The Data Memory is initialised to Bank 0 after a reset, except for the WDT time-out reset in the Idle/ Sleep Mode, in which case, the Data Memory bank remains unaffected. It should be noted that Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within either Bank 0 or Bank 1. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer.

BP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	BP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **BP0**: Select Data Memory Banks
0: Bank 0
1: Bank 1

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, 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 pointer and indicates the location where the table data is located. Their value must be setup before any table read commands are executed. Their value can be changed, for example using the “INC” or “DEC” instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller. With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition, operations related to the status register may give different results due to the different instruction operations. The TO flag can be affected only by a system power-up, a WDT time-out or by executing the “CLR WDT” or “HALT” instruction. The PDF flag is affected only by executing the “HALT” or “CLR WDT” instruction or during a system power-up.

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

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- PDF is cleared by a system power-up or executing the “CLR WDT” instruction. PDF is set by executing the “HALT” instruction.
- TO is cleared by a system power-up or executing the “CLR WDT” or “HALT” instruction. TO is set by a WDT time-out.

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

STATUS Register

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

"x" unknown

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 **TO**: Watchdog Time-Out flag
0: After power up 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

C is also affected by a rotate through carry instruction.

System Control Register – CTRL

This register is used to provide control over various internal functions. Some of these include the certain system clock options, the LVR control, Watchdog Timer function.

CTRL Register

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

"x" unknown

Bit 7 **FSYSON**: f_{SYS} Control in IDLE Mode
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2 **LVRF**: LVR function reset flag
0: Not occur
1: Occurred
This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

bit 1 **LRF**: LVR Control register software reset flag
0: Not occur
1: Occurred
This bit is set to 1 if the LVRC register contains any non defined LVR voltage register values. This in effect acts like a software-reset function. This bit can only be cleared to 0 by the application program.

bit 0 **WRF**: WDT Control register software reset flag
0: Not occur
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.

EEPROM Data Memory

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

EEPROM Data Memory Structure

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

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEADDR, the data register, EEDATA and a single control register, EECTRL. As both the EEADDR and EEDATA registers are located in Bank 0, they can be directly accessed in the same way as any other Special Function Register. The EECTRL register however, being located in Bank1, cannot be directly addressed directly and can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EECTRL control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EECTRL register are executed.

EEPROM Control Registers List

Name	Bit							
	7	6	5	4	3	2	1	0
EEADDR	—	—	—	D4	D3	D2	D1	D0
EEDATA	D7	D6	D5	D4	D3	D2	D1	D0
EECTRL	—	—	—	—	WTEN	WT	RDEN	RD

EEADDR Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	D4	D3	D2	D1	D0
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 Data EEPROM address

Data EEPROM address bit 4 ~ bit 0

EECTRL Register

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

Bit 7~4 Unimplemented, read as "0"

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

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

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

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

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

EEDATA 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 Data EEPROM data
Data EEPROM data bit 7 ~ bit 0

Reading Data from the EEPROM

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

Writing Data to the EEPROM

To write data to the EEPROM, the write enable bit, WTEN, in the EECTRL register must first be set high to enable the write function. The EEPROM address of the data to be written must then be placed in the EEADDR register and the data placed in the EEDATA register. If the WT bit in the EECTRL register is now set high, an internal write cycle will then be initiated. Setting the WT bit high will not initiate a write cycle if the WTEN bit has not been set. As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WT bit in the EECTRL register or by using the EEPROM interrupt. When the write cycle terminates, the WT bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WT bit to determine when the write cycle has ended.

Write Protection

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

EEPROM Interrupt

The EEPROM write or read interrupt is generated when an EEPROM write or read cycle has ended. The EEPROM interrupt must first be enabled by setting the E2I bit in the relevant interrupt register. However as the EEPROM is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM write cycle ends, the E2F request flag and its associated multi-function interrupt request flag will both be set. If the global, EEPROM and Multi-function interrupts are enabled and the stack is not full, a jump to the associated Multi-function Interrupt vector will take place. When the interrupt is serviced only the Multi-function interrupt flag will be automatically reset, the EEPROM interrupt flag must be manually reset by the application program. More details can be obtained in the Interrupt section.

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Bank Pointer could be normally cleared to zero as this would inhibit access to Bank 1 where the EEPROM control register exist. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process. The WT bit in the EECTRL register should be set immediately after the WTEN bit is set, otherwise the EEPROM write cycle will not be executed.

Programming Examples

- Reading data from the EEPROM – polling method

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

- Writing Data to the EEPROM - polling method

```
MOV A, EEPROM_ADRES    ; user defined address
MOV EEADDR, A
MOV A, EEPROM_DATA     ; user defined data
MOV EEDATA, A
MOV A, 040H            ; setup memory pointer MP1
MOV MP1, A             ; MP1 points to EECTRL register
MOV A, 01H             ; setup Bank Pointer
MOV BP, A
SET IAR1.3             ; set WTEN bit, enable write operations
SET IAR1.2             ; start Write Cycle - set WT bit
BACK:
SZ IAR1.2              ; check for write cycle end
JMP BACK
CLR IAR1               ; disable EEPROM read/write
CLR BP
```

Oscillator

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 selections and operation are selected through registers.

Oscillator Overview

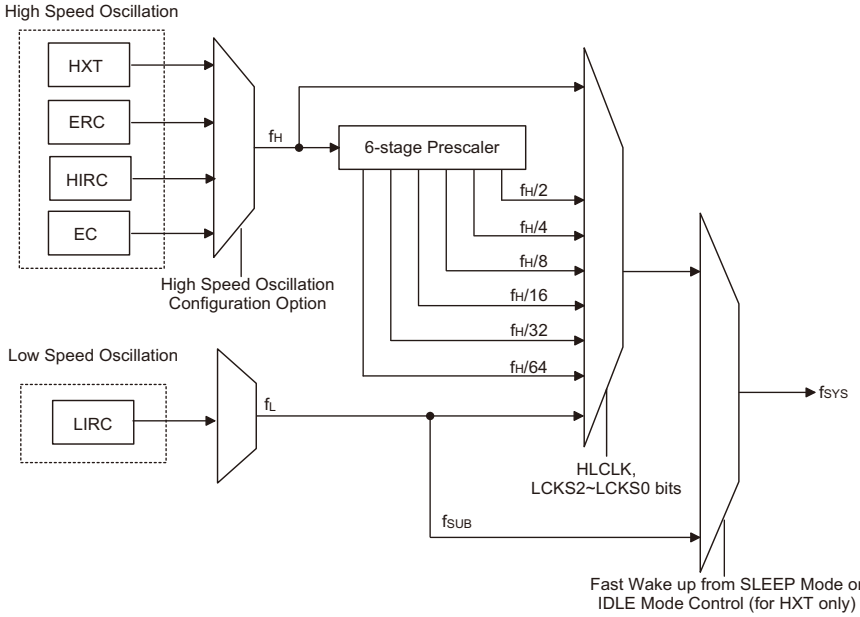
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base functions. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options are selected through the configuration options. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. 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	Freq.	Pins
External Crystal	HXT	400kHz~12MHz	OSC1/OSC2
External RC	ERC	910kHz, 2MHz, 4MHz	OSC1
External Clock	EC	400kHz~12MHz	OSC1
Internal High Speed RC	HIRC	910kHz, 2/4/8MHz	—
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

System Clock Configurations

There are five methods of generating the system clock, four high speed oscillators and one low speed oscillators. The high speed oscillators are the external crystal/ceramic oscillator, external RC network oscillator, external clock and the internal 910kHz, 2MHz, 4MHz or 8MHz RC oscillator. The one low speed oscillators is the internal 32kHz RC oscillator.

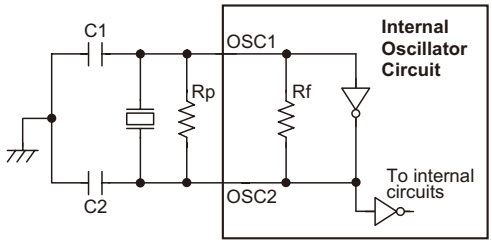


System Clock Configurations

Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and LCKS2~LCKS0 bits in the SYSMOD register and as the system clock can be dynamically selected. The actual source clock used for each of the high speed and low speed oscillators is chosen via configuration options. The frequency of the slow speed or high speed system clock is also determined using the HLCLK bit and LCKS2~LCKS0 bits in the SYSMOD register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.

External Crystal/Ceramic Oscillator – HXT

The External Crystal/Ceramic System Oscillator is one of the high frequency oscillator choices, which is selected via configuration option. For most crystal oscillator configurations, the simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, without requiring external capacitors. However, for some crystal types and frequencies, to ensure oscillation, it may be necessary to add two small value capacitors, C1 and C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.



Note: 1. Rp is normally not required. C1 and C2 are required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

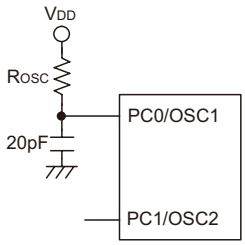
Crystal/Resonator Oscillator – HXT

Crystal Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
12MHz	0pF	0pF
8MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

External RC Oscillator – ERC

Using the ERC oscillator only requires that a resistor, with a value between 56kΩ and 2.4MΩ, is connected between OSC1 and VDD, and a capacitor is connected between OSC1 and ground, providing a low cost oscillator configuration. It is only the external resistor that determines the oscillation frequency; the external capacitor has no influence over the frequency and is connected for stability purposes only. 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. Here only the OSC1 pin is used, which is shared with I/O pin PC0, leaving pin PC1 free for use as a normal I/O pin.



External RC Oscillator – ERC

External Oscillator – EC

The system clock can also be supplied by an externally supplied clock giving users a method of synchronising their external hardware to the microcontroller operation. This is selected using a configuration option and supplying the clock on pin OSC1. Pin OSC2 should be left floating if the external oscillator is used. The internal oscillator circuit contains a filter circuit to reduce the possibility of erratic operation due to noise on the oscillator pin, however as the filter circuit consumes a certain amount of power, a configuration option exists to turn this filter off. Not using the internal filter should be considered in power sensitive applications and where the externally supplied clock is of a high integrity and supplied by a low impedance source.

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has four fixed frequencies of either 910kHz, 2MHz, 4MHz or 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 if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PC0 and PC1 are free for use as normal I/O pins.

Internal 32kHz Oscillator – LIRC

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

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.

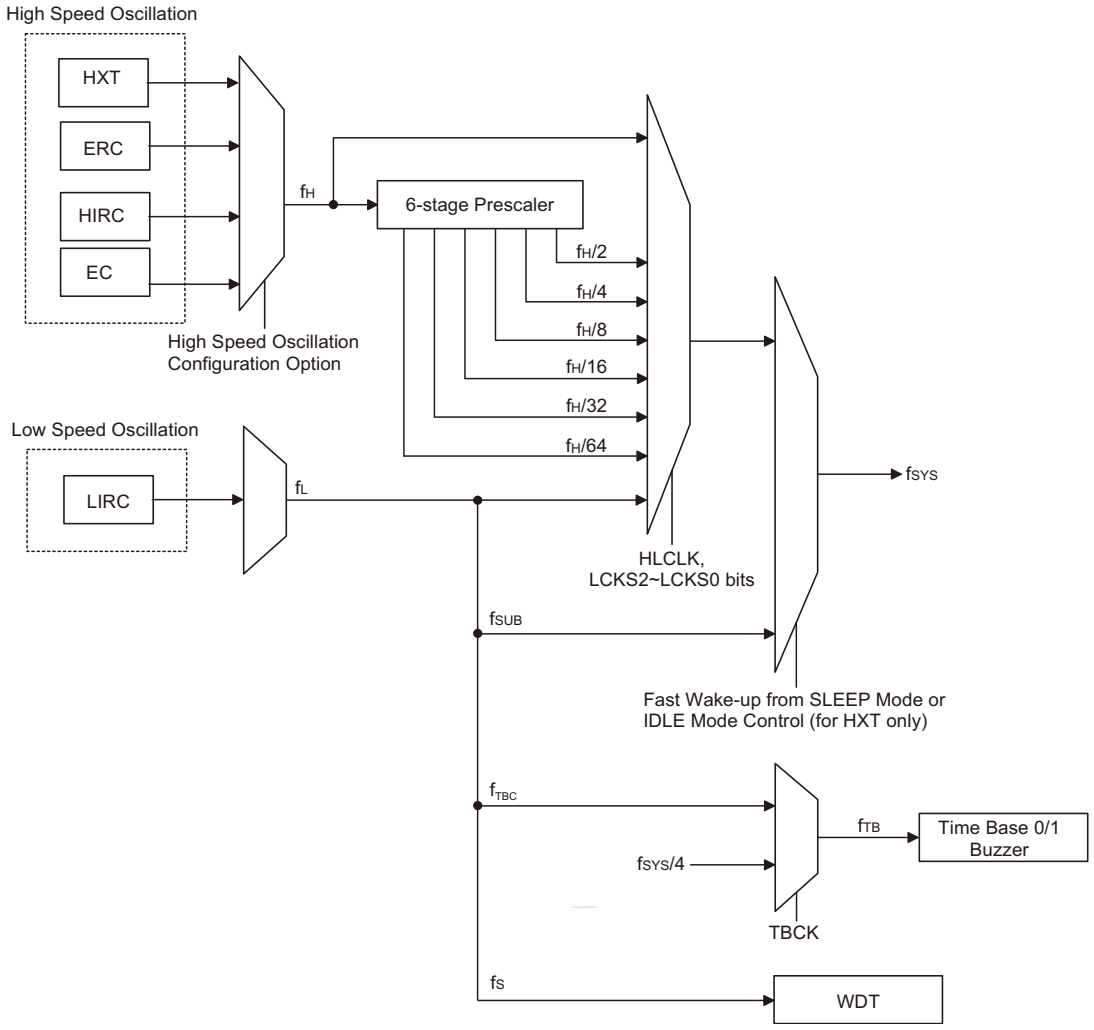
System Clocks

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 configuration options and 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_L, source, and is selected using the HLCLK bit and LCKS2~LCKS0 bits in the SYSMOD register. The high speed system clock can be sourced from either an HXT, ERC, EC or HIRC oscillator, selected via a configuration option. The low speed system clock source can be sourced from internal clock LIRC. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.

System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application.There are two modes allowing normal operation of the microcontroller, the NORMAL Mode and SLOW Mode.The remaining four modes, the SLEEP0, SLEEP1, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.



System Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_L from f_H, the high speed oscillation will stop to conserve the power. Thus there is no f_H~f_H/64 for peripheral circuit to use.

Operation Mode	Descriptions				
	CPU	f_{SYS}	f_{SUB}	f_S	f_{TBC}
Normal mode	On	f_H~ f_H/64	On	On	On
Slow mode	On	f_L	On	On	On
IDLE0 mode	Off	Off	On	On	On
IDLE1 mode	Off	On	On	On	On
SLEEP0 mode	Off	Off	Off	Off	Off
SLEEP1 mode	Off	Off	On	On	Off

NORMAL Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from one of the high speed oscillators, either the HXT, ERC, EC or HIRC oscillators. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the LCKS2~LCKS0 and HLCLK bits in the SYSMOD 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 the low speed oscillator, the LIRC. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW Mode, the f_H is off.

SLEEP0 Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SYSMOD register is low. In the SLEEP0 mode the CPU will be stopped, and the f_{SUB} and f_S clocks will be stopped too, and the Watchdog Timer function is disabled.

SLEEP1 Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SYSMOD register is low. In the SLEEP1 mode the CPU will be stopped. However the f_{SUB} and f_S clocks will continue to operate if the Watchdog Timer function is enabled.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SYSMOD register is high and the FSYSON bit in the CTRL register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU but some peripheral functions will remain operational such as the Watchdog Timer and Time Base. In the IDLE0 Mode, the Watchdog Timer clock f_S will be on.

IDLE1 Mode

The IDLE1 Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SYSMOD register is high and the FSYSON bit in the CTRL register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU but may continue to provide a clock source to keep some peripheral functions operational such as the Watchdog Timer and Time Base. In the IDLE1 Mode, the system oscillator will continue to run, and this system oscillator may be high speed or low speed system oscillator. In the IDLE1 Mode the Watchdog Timer clock f_S will be on.

Control Register

A single register, SYSMOD, is used for overall control of the internal clocks within the device.

SYSMOD Register

Bit	7	6	5	4	3	2	1	0
Name	LCKS2	LCKS1	LCKS0	FSTEN	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	0	0	0	0	0	0	1	1

bit 7~5 **LCKS2 ~ LCKS0**: The system clock selection when HLCLK is "0"
000: f_L (f_{LIRC})
001: f_L (f_{LIRC})
010: $f_H/64$
011: $f_H/32$
100: $f_H/16$
101: $f_H/8$
110: $f_H/4$
111: $f_H/2$

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

Bit 4 **FSTEN:** Fast Wake-up Control (only for HXT)
0:Disable
1:Enable

This is the Fast Wake-up Control bit which determines if the f_{SUB} clock source is initially used after the device wakes up. When the bit is high, the f_{SUB} clock source can be used as a temporary system clock to provide a faster wake up time as the f_{SUB} clock is available.

bit 3 **LTO**: LIRC System OSC SST ready flag
0: Not ready
1: Ready

This is the low speed system oscillator SST ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP0 Mode but after a wake-up has occurred, the flag will change to a high level after 1~2 clock cycles if the LIRC oscillator is used.

bit 2 **HTO**: HIRC System OSC SST ready flag
0: Not ready
1: Ready

This is the high speed system oscillator SST ready flag which indicates when the high speed system oscillator is stable after a wake-up has occurred. The flag will be low when in the SLEEP or IDLE0 Mode but after power on reset or a wake-up has occurred, the flag will change to a high level after1024 clock cycles if the HXT oscillator is used and after 15~16 clock cycles if the ERC or HIRC oscillator is used.

bit 1 **IDLEN**: IDLE Mode Control
0: Disable
1: Enable

This is the IDLE Mode Control bit and determines what happens when the HALT instruction is executed. If this bit is high, when a HALT instruction is executed the device will enter the IDLE Mode. In the IDLE1 Mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if FSYSON bit is high. If FSYSON bit is low, the CPU and the system clock will all stop in IDLE0 mode. If the bit is low the device will enter the SLEEP Mode when a HALT instruction is executed.

bit 0 **HLCLK**: System Clock Selection
0: $f_H/2 \sim f_H/64$ or f_L
1: f_H
This bit is used to select if the f_H clock or the $f_H/2 \sim f_H/64$ or f_L clock is used as the system clock. When the bit is high the f_H clock will be selected and if low the $f_H/2 \sim f_H/64$ or f_L clock will be selected. When system clock switches from the f_H clock to the f_L clock and the f_H clock will be automatically switched off to conserve power.

Fast Wake-up

To minimise power consumption the device can enter the SLEEP or IDLE0 Mode, where the system clock source to the device will be stopped. However when the device is woken up again, it can take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume. To ensure the device is up and running as fast as possible a Fast Wake-up function is provided, which allows f_{SUB}, namely LIRC oscillator, to act as a temporary clock to first drive the system until the original system oscillator has stabilised. As the clock source for the Fast Wake-up function is f_{SUB}, the Fast Wake-up function is only available in the SLEEP1 and IDLE0 modes. When the device is woken up from the SLEEP0 mode, the Fast Wake-up function has no effect because the f_{SUB} clock is stopped. The Fast Wake-up enable/disable function is controlled using the FSTEN bit in the SYSMOD register.

If the HXT oscillator is selected as the NORMAL Mode system clock, and if the Fast Wake-up function is enabled, then it will take one to two t_{SUB} clock cycles of the LIRC oscillator for the system to wake-up. The system will then initially run under the f_{SUB} clock source until 1024 HXT clock cycles have elapsed, at which point the HTO flag will switch high and the system will switch over to operating from the HXT oscillator.

If the ERC or HIRC oscillators or LIRC oscillator is used as the system oscillator then it will take 15~16 clock cycles of the ERC or HIRC or 1~2 cycles of the LIRC to wake up the system from the SLEEP or IDLE0 Mode. The Fast Wake-up bit, FSTEN will have no effect in these cases.

System Oscillator	FSTEN Bit	Wake-up Time (SLEEP0 Mode)	Wake-up Time (SLEEP1 Mode)	Wake-up Time (IDLE0 Mode)	Wake-up Time (IDLE1 Mode)
HXT	0	1024 HXT cycles	1024 HXT cycles		1~2 HXT cycles
	1	1024 HXT cycles	1~2 f_{SUB} cycles (System runs with f_{SUB} first for 1024HXT cycles and then switches over to run with the HXT clock)		1~2 HXT cycles
ERC	×	15~16 ERC cycles	15~16 ERC cycles		1~2 ERC cycles
EC	×	15~16 EC cycles	15~16 RC cycles		1~2 EC cycles
HIRC	×	15~16 HIRC cycles	15~16 HIRC cycles		1~2 HIRC cycles
LIRC	×	1~2 LIRC cycles	1~2 LIRC cycles		1~2 LIRC cycles

Wake-Up Times

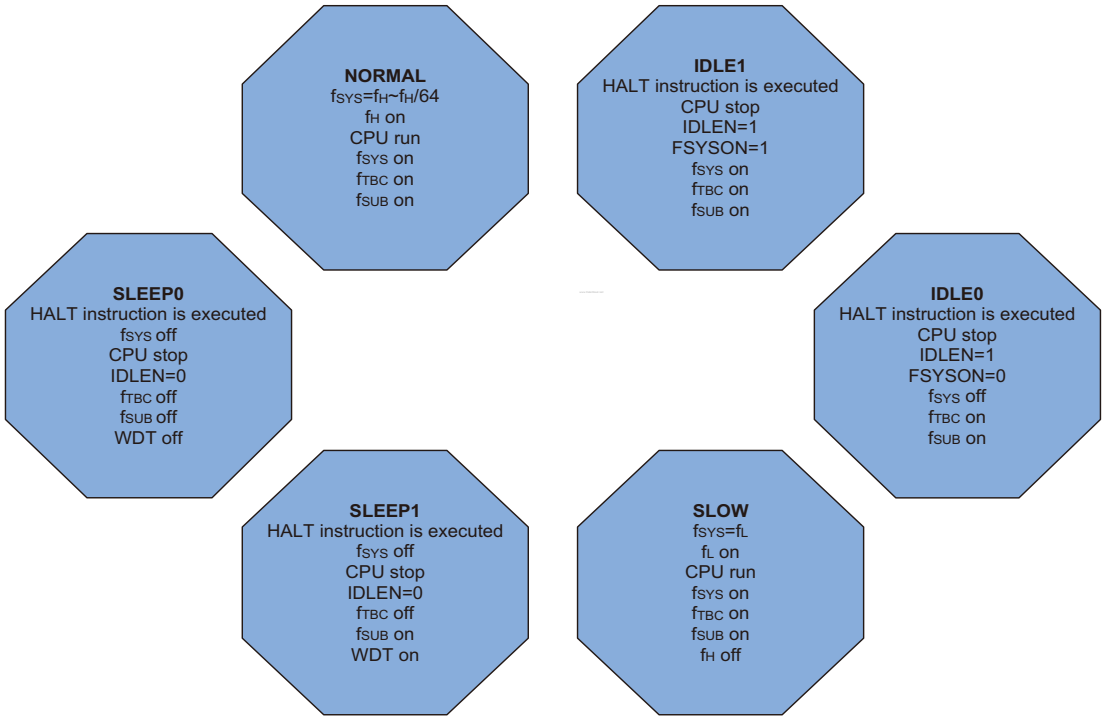
Note that there is no fast start-up function when the device is in SLEEP mode with the WDT is disabled. We turn off the regulator in this case. It will take 500μs at most to become stable for regulator waked up from SLEEP mode.

Operating Mode Switching and Wake-up

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 NORMAL Mode and SLOW Mode is executed using the HLCLK bit and LCKS2~LCKS0 bits in the SYSMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Mode is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SYSMOD register and FSYSON in the CTRL register.

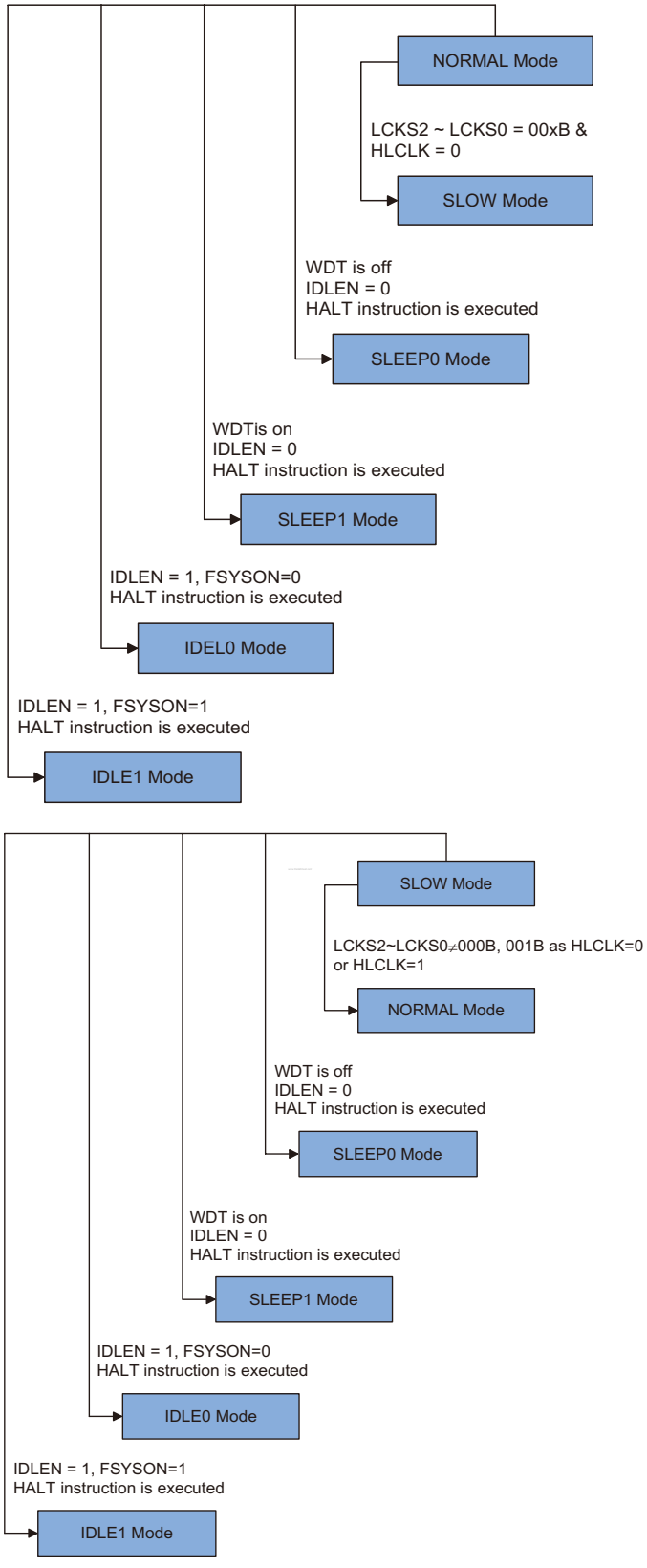
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~$f_H/64$ or f_L. If the clock is from the f_L, the high speed clock source will stop running to conserve power. When this happens it must be noted that the $f_H/16$ and $f_H/64$ internal clock sources will also stop running, which may affect the operation of other internal functions. The accompanying flowchart shows what happens when the device moves between the various operating modes.



NORMAL Mode to SLOW Mode Switching

When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to "0" and set the LCKS2~LCKS0 bits to "000" or "001" in the SYSMOD register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

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


NORMAL Mode
LCKS2 ~ LCKS0 = 00xB &
HLCLK = 0
SLOW Mode
WDT is off
IDLEN = 0
HALT instruction is executed
SLEEP0 Mode
WDTis on
IDLEN = 0
HALT instruction is executed
SLEEP1 Mode
IDLEN = 1, FSYSON=0
HALT instruction is executed
IDEL0 Mode
IDLEN = 1, FSYSON=1
HALT instruction is executed
IDLE1 Mode
SLOW Mode
LCKS2~LCKS0≠000B, 001B as HLCLK=0
or HLCLK=1
NORMAL Mode
WDT is off
IDLEN = 0
HALT instruction is executed
SLEEP0 Mode
WDT is on
IDLEN = 0
HALT instruction is executed
SLEEP1 Mode
IDLEN = 1, FSYSON=0
HALT instruction is executed
IDLE0 Mode
IDLEN = 1, FSYSON=1
HALT instruction is executed
IDLE1 Mode

SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system uses the LIRC low speed system oscillator. To switch back to the NORMAL Mode, where the high speed system oscillator is used, the HLCLK bit should be set to "1" or HLCLK bit is "0", but LCKS2~LCKS0 is set to "010", "011", "100", "101", "110" or "111". As a certain amount of time will be required for the high frequency clock to stabilise, the status of the HTO bit is checked. The amount of time required for high speed system oscillator stabilization depends upon which high speed system oscillator type is used.

Entering the SLEEP0 Mode

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

The system clock, WDT clock and Time Base clock will be stopped and the application program will stop at the "HALT" instruction.

- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and stopped
- 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.

Entering the SLEEP1 Mode

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

- The system clock and Time Base clock will be stopped and the application program will stop at the "HALT" instruction, but the WDT will remain with the clock source coming from the f_L clock.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting as the WDT is enabled.
- 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.

Entering the IDLE0 Mode

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

- The system clock will be stopped and the application program will stop at the "HALT" instruction, but the Time Base clock and f_{SUB} clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT is enabled.
- 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.

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 the IDLEN bit in SYSMOD register equal to "1" and the FSYSON bit in CTRL register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The system clock and Time Base clock and f_{SUB} clock will be on and the application program will stop at the "HALT" instruction
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT is enabled regardless of the WDT clock source which originates from the f_L clock
- 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.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to device which has different package types, as there may be unbonded pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the configuration options have enabled the LIRC oscillator.

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

Wake-up

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

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

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

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_S, which is in turn supplied by the LIRC oscillator. The LIRC internal oscillator has an approximate period of 32 kHz at a supply voltage of 5V. However, it should be noted that 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^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. This register together with several configuration options control the overall operation of the Watchdog Timer.

WDTC Register

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

Bit 7 ~ 3 **WE4 ~ WE0**: WDT enable bit
If the WDT configuration option is selected as "Always Enabled":
10101 or 01010: Enabled
Other: Reset MCU
If the WDT configuration option is selected as "Application Program Enabled":
10101: Disabled
01010: Enabled
Other Values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the CTRL register will be set to 1.

Bit 2 ~ 0 **WS2 ~ WS0**: Select WDT Timeout Period
000: $2^8/f_S$
001: $2^{10}/f_S$
010: $2^{12}/f_S$
011: $2^{14}/f_S$
100: $2^{15}/f_S$
101: $2^{16}/f_S$
110: $2^{17}/f_S$
111: $2^{18}/f_S$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

CTRL Register

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

Bit 7 **FSYSON**: f_{SYS} Control in IDLE Mode
Describe elsewhere.

Bit 6~3 Unimplemented, read as "0"

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

Bit 1 **LRF**: LVR Control register software reset flag
Describe elsewhere.

Bit 0 **WRF**: WDT Control register software reset flag
0: Not occur
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 instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. A Watchdog Timer configuration option determines if the Watchdog Timer is always enabled or if it is enabled using the application program. With regard to the Watchdog Timer enable/disable function, there are also five bits, WE4~WE0, in the WDTC register to offer additional enable/disable and reset control of the Watchdog Timer. If the WDT configuration option has selected that the WDT function is always enabled, then WE4~WE0 bits still have effect on the WDT function. When the WE4~WE0 bits value are equal to 01010B or 10101B, the WDT function is enabled. However, if the WE4~WE0 bits are changed to any other values except 01010B and 10101B, which could be caused by adverse environmental conditions such as noise, it will reset the microcontroller after 2~3 LIRC clock cycles. If the WDT configuration option has selected that the WDT function is controlled using the application program, then the WDT control register bits, WE4~WE0, are used to enable or disable the Watchdog Timer. In this case the WDT function will be disabled when the WE4~WE0 bits are

equal to 10101B and enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after 2~3 LIRC clock cycles. After power on these bits will have a value of 01010B.

WDT Configuration Option	WE4~WE0 Bits	WDT Function
Always Enabled	01010B or 10101B	Enable
	Any other value	Reset MCU
Application Program Enabled	10101B	Disable
	01010B	Enable
	Any other value	Reset MCU

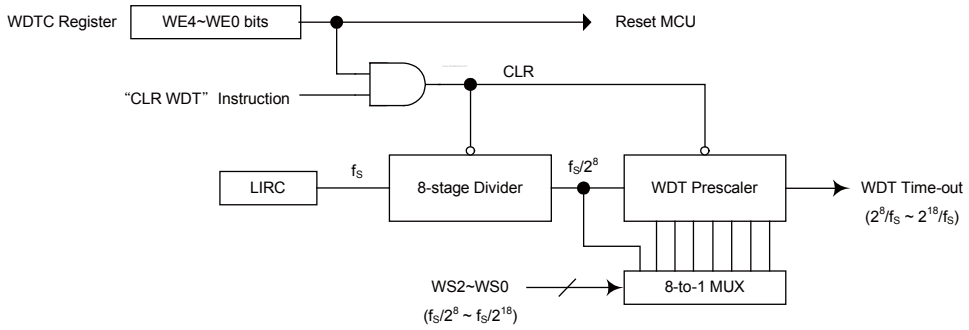
"x": don't care.

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. For some low power consumption applications, it is recommended to disable the WDT function before entering the Power down Mode. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is an external hardware reset, the second is using the Watchdog Timer software clear instruction and 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 contents.

The maximum time out period is when the 2^{18} division ratio is selected. As an example, with a 32 kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 second for the 2^{18} division ratio, and a minimum timeout of 7.8ms for the 2^8 division ration.



Watchdog Timer

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the 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.

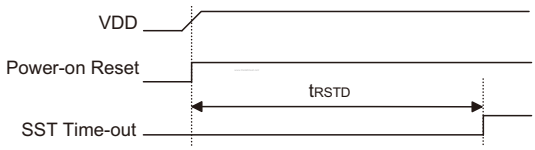
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, is implemented in situations where the power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

Reset Functions

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

Power-on Reset

The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.



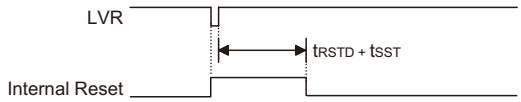
Note: t_{RSTD} is power-on delay, typical time=50ms

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. The LVR function has a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the CTRL register will also be set to 1. The LVR includes the following specifications: For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for greater than the value t_{LVR} specified in the A.C. characteristics. If the low voltage state does not exceed t_{LVR}, the LVR will ignore it and will not perform a reset function. One of a range of specified voltage values for V_{LVR} can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits have any other value, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the device after 2~3 LIRC clock cycles. When this happens, the LRF bit in the CTRL register will be set to 1. After power on the register will have the value of 01010101B.

Note that the LVR function will be automatically disabled when the device enters the power down mode.



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

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	0	1	0	1

Bit 7~0 **LVS7 ~ LVS0**: LVR voltage select
01010101: 2.1V (default)
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Other values: Generates MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage values above, an MCU reset will be generated. The reset operation will be activated after 2~3 LIRC clock cycles. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than the four defined register values above, will also result in the generation of an MCU reset. The reset operation will be activated after 2~3 LIRC clock cycles. However in this situation the register contents will be reset to the POR value.

- **CTRL Register**

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

Bit 7 **FSYSON**: f_{SYS} Control in IDLE Mode
Describe elsewhere.

Bit 6~3 Unimplemented, read as "0"

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

This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

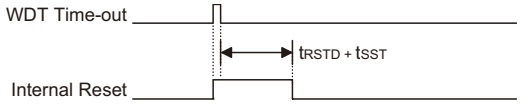
Bit 1 **LRF**: LVR Control register software reset flag
0: Not occurred
1: Occurred

This bit is set to 1 if the LVRC register contains any non defined LVR voltage register values. This in effect acts like a software reset function. This bit can only be cleared to 0 by the application program.

Bit 0 **WRF**: WDT Control register software reset flag
Describe elsewhere.

Watchdog Time-out Reset during Normal Operation

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

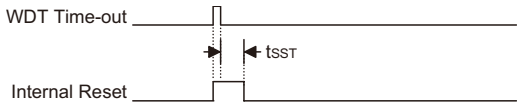


Note: t_{RSTD} is power-on delay, typical time=16.7ms

WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



Note: The t_{SST} is 16 clock cycles if the system clock source is provided by ERC or HIRC. The t_{SST} is 128 clock for HXT. The t_{SST} is 1~2 clock for LIRC.

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 NORMAL or SLOW Mode operation
1	u	WDT time-out reset during NORMAL or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

Note: "u" stands for unchanged

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

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

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects the microcontroller internal registers. Note that where more than one package type exists the table will reflect the situation for the larger package type.

Register	Power-on Reset	WDT Time-out (Normal Operation)	LVR Reset (Normal Operation)	LVR Reset (HALT)	WDT Time-out (HALT)
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
TBHP	---- --xx	---- --uu	---- --uu	---- --uu	---- --uu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
MFIC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFIC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	-1-- 1111	-1-- 1111	-1-- 1111	-1-- 1111	-u-- uuuu
PBC	-1-- 1111	-1-- 1111	-1-- 1111	-1-- 1111	-u-- uuuu
PC	-111 --11	-111 --11	-111 --111	-111 --11	-uuu --uu
PCC	-111 --11	-111 --11	-111 --11	-111 --11	-uuu --uu
PAWU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBPU	-0-- 0000	-0-- 0000	-0-- 0000	-0-- 0000	-u-- uuuu
PCPU	-000 --00	-000 --00	-000 --00	-000 --00	-uuu --uu
SYSMOD	0000 0x11	0000 0x11	0000 0x11	0000 0x11	uuuu uuuu
INTEDGE	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
MISC	0000 ---0	0000 ---0	0000 ---0	0000 ---0	uuuu ---u
LDOC	---0 0000	---0 0000	---0 0000	---0 0000	---u uuuu
LVRC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
CMP1C0	0001 0000	0001 0000	0001 0000	0001 0000	uuuu uuuu
CMP1C1	1--- 0010	1--- 0010	1--- 0010	1--- 0010	u--- uuuu
CMP2C0	0001 0000	0001 0000	0001 0000	0001 0000	uuuu uuuu
CMP2C1	00-- 0010	00-- 0010	00-- 0010	00-- 0010	uu-- uuuu
OPA1C0	0001 0000	0001 0000	0001 0000	0001 0000	uuuu uuuu
OPA1C1	0000 1100	0000 1100	0000 1100	0000 1100	uuuu uuuu
OPA2C0	0001 0000	0001 0000	0001 0000	0001 0000	uuuu uuuu
OPA2C1	0000 1100	0000 1100	0000 1100	0000 1100	uuuu uuuu
OPA2C2	00-- 0000	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
CTRL	0--- -x00	0--- -x00	0--- -x00	0--- -x00	u--- -uuu
WDTC	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	0011 -111	0011 -111	0011 -111	0011 -111	uuuu -uuu
BPCTL	---0 0000	---0 0000	---0 0000	---0 0000	---u uuuu

Note: "-" not implement
"u" stands for "unchanged"
"x" stands for "unknown"

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~PC. 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
PAWU	D7	D6	D5	D4	D3	D2	D1	D0
PAPU	D7	D6	D5	D4	D3	D2	D1	D0
PA	D7	D6	D5	D4	D3	D2	D1	D0
PAC	D7	D6	D5	D4	D3	D2	D1	D0
PBPU	—	D6	—	—	D3	D2	D1	D0
PB	—	D6	—	—	D3	D2	D1	D0
PBC	—	D6	—	—	D3	D2	D1	D0
PCPU	—	D6	D5	D4	—	—	D1	D0
PC	—	D6	D5	D4	—	—	D1	D0
PCC	—	D6	D5	D4	—	—	D1	D0

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, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using registers PAPU~PCPU, and are implemented using weak PMOS transistors.

PAPU 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 Port A bit7~ bit 0 Pull-High Control
0: Disable
1: Enable

PBPU Register

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

Bit 7 Unimplemented, read as "0"

Bit 6 Port B bit 6 Pull-High Control
0: Disable
1: Enable

Bit 5~4 Unimplemented, read as "0"

Bit 3~0 Port B bit 3~ bit 0 Pull-High Control
0: Disable
1: Enable

PCPU Register

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

Bit 7 Unimplemented, read as "0"

Bit 6~4 Port C bit 6~4 Pull-High Control
0: Disable
1: Enable

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 Port C bit 1~0 Pull-High 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.

PAWU 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 **PAWU**: Port A bit 7 ~ bit 0 Wake-Up Control
0: Disable
1: Enable

I/O Port Control Registers

Each I/O port has its own control register known as PAC~PCC, to control the input/output configuration. With this control register, each CMOS 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 setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register.

However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

PAC 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	1	1	1	1	1	1	1	1

Bit 7~0 **PAC**: Port A bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

PBC Register

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

Bit 7 Unimplemented, read as "0"

Bit 6 **PBC**: Port B bit 6 Input/Output Control
0: Output
1: Input

Bit 5~4 Unimplemented, read as "0"

Bit 3~0 **PBC**: Port B bit 3~0 Input/Output Control
0: Output
1: Input

PCC Register

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

Bit 7 Unimplemented, read as "0"

Bit 6~4 **PCC**: Port C bit 6~4 Input/Output Control
0: Output
1: Input

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **PCC**: Port C bit 1~0 Input/Output Control
0: Output
1: Input

MISC Register

Bit	7	6	5	4	3	2	1	0
Name	ODE3	ODE2	ODE1	ODE0	—	—	—	PFDEN
R/W	R/W	R/W	R/W	R/W	—	—	—	R/W
POR	0	0	0	0	—	—	—	0

Bit 7 **ODE3**: PB3 Open Drain Control
0: Disable
1: Enable

Bit 6 **ODE2**: PB2 Open Drain Control
0: Disable
1: Enable

Bit 5 **ODE1**: PB1 Open Drain Control
0: Disable
1: Enable

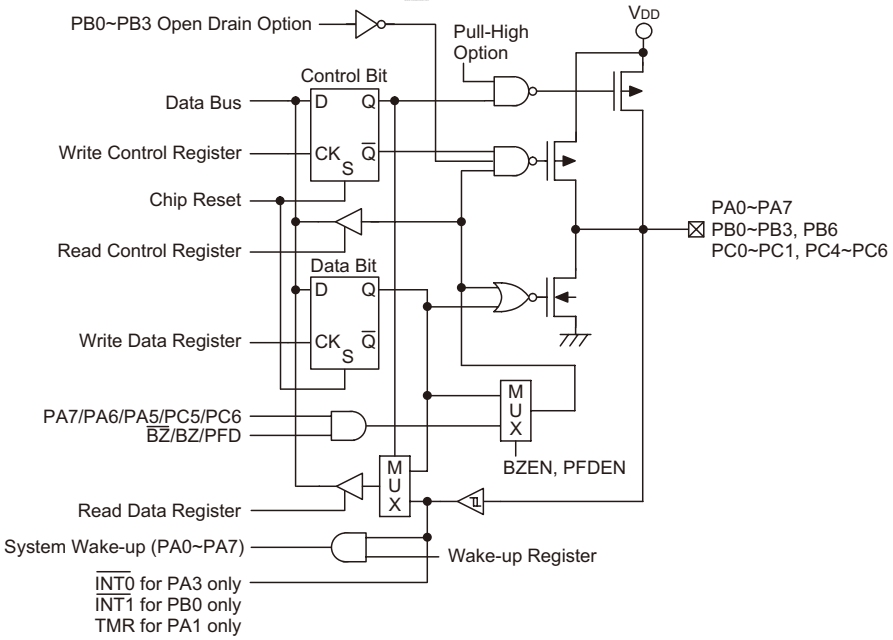
Bit 4 **ODE0**: PB0 Open Drain Control
0: Disable
1: Enable

Bit 3~1 Unimplemented, read as "0"

Bit 0 **PFDEN**: PFD related control - described elsewhere

I/O Pin Structures

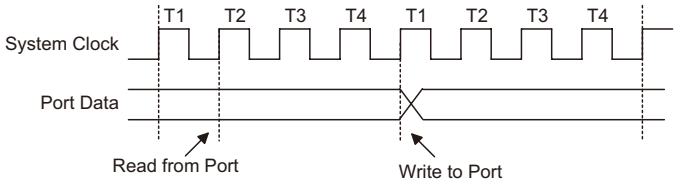
The accompanying diagrams illustrate the internal structures of some generic I/O pin types. As the exact logical construction of the I/O pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins. The wide range of pin-shared structures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



Generic Input/Output Structure

Programming Considerations

Within the user program, one of the 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, PAC~PCC, are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers, PA~PC, 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.



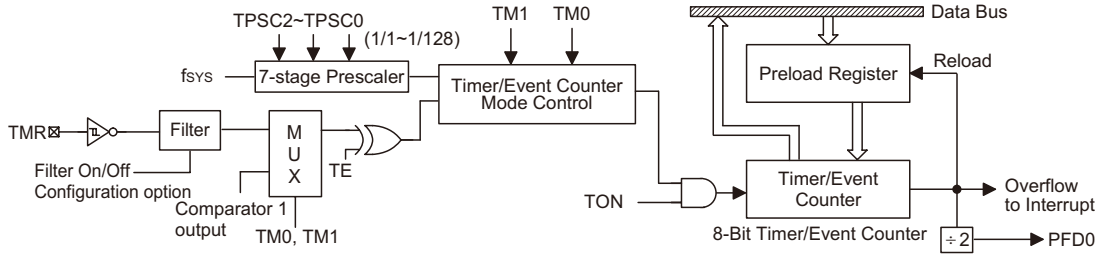
Read Modify Write Timing

Port A has the additional capability of providing wake-up functions. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the Port A pins. Single or multiple pins on Port A can be setup to have this function. In addition, the Port B pins also provide Open Drain I/O structure options which can be controlled by the specific register.

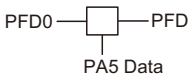
Timer/Event Counter

The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contains a sigle count-up timer of 8-bit capacity. As the timer has three different operating modes, it can be configured to operate as a general timer, an external event counter or as a pulse width capture device. The provision of an internal prescaler to the clock circuitry on giving added range to the timer.

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



8-bit Timer/Event Counter Structure



Note: The output is controlled by PA5 data.

Timer Control Register – TMRC

The flexible features of the Holtek microcontroller Timer/Event Counter enables it to operate in four different modes, the options of which are determined by the contents of its control register.

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

To choose which of the four modes the timer is to operate in, either in the timer mode, the external event counting mode, the internal event counter mode, or the pulse width measurement mode, bits 7 and 6 of the Timer Control Register, which is known as the bit pair TM1/TM0, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as TON, provides the basic on/off control of the timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. For the timer that has prescaler, bits 0~2 of the Timer Control Register determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or pulse width measurement mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as TE.

TMRC Register

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

Bit 7 ~ 6 **TM1, TM0**: Timer operation mode selection
- 00: Event counter mode, the input signal is from Comparator 1 output
- 01: Event counter mode, the input signal is from TMR pin
- 10: Timer mode
- 11: Pulse width capture mode

Bit 5 Unimplemented, read as "0"

Bit 4 **TON**: Timer/event counter counting enable
- 0: Disable
- 1: Enable

Bit 3 **TE**: Event counter active edge selection
- 1: Count on falling edge
- 0: Count on rising edge

Pulse Width Capture active edge selection
- 1: Start counting on rising edge, stop on falling edge
- 0: Start counting on falling edge, stop on rising edge

Bit 2~0 **TPSC2, TPSC1, TPSC0**: Timer prescaler rate selection

Timer internal clock
- 000: f_{SYS}
- 001: $f_{SYS}/2$
- 010: $f_{SYS}/4$
- 011: $f_{SYS}/8$
- 100: $f_{SYS}/16$
- 101: $f_{SYS}/32$
- 110: $f_{SYS}/64$
- 111: $f_{SYS}/128$

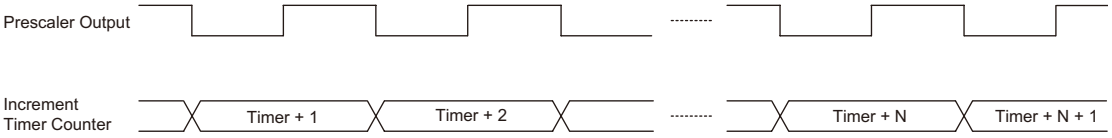
Configuring the Timer Mode

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. To operate in this mode, the Operating Mode Select bit pair, TM1/TM0, in the Timer Control Register must be set to the correct value as shown.

Bit7	Bit6
1	0

Control Register Operating Mode Select Bits for the Timer Mode

In this mode the internal clock is used as the timer clock. The timer input clock source is f_{SYS} oscillator. However, this timer clock source is further divided by a prescaler, the value of which is determined by the bits TPSC2~TPSC0 in the Timer Control 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 increments by one. When the timer is full and overflows, an interrupt sigal is generated and the timer will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt enable bit in the Interrupt Control Register, INTC0, is reset to zero.



Timer Mode Timing Chart

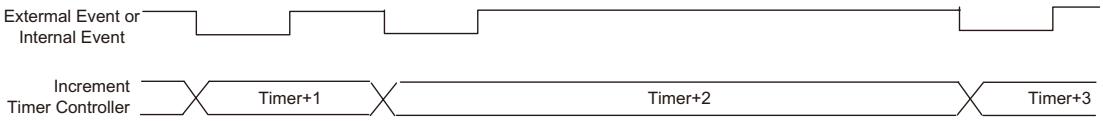
Configuring the Event Counter Mode

In this mode, a number of externally changing logic events, occurring on the internal comparators output, can be recorded by the Timer/Event Counter. To operate in this mode, the Operating Mode Select bit pair, TM1/TM0, in the Timer Control Register must be set to the correct value as shown.

Bit7	Bit6
0	0/1

Control Register Operating Mode Select Bits for the Event Counter Mode Mode.

In this mode, the comparator output, CMP1OP, is used as the Timer/Event Counter clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit TON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. If the Active Edge Select bit TE, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the Active Edge Select bit is high, the counter will increment each time the external timer 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. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the Interrupt Control Register, INTC0, is reset to zero. It should be noted that in the internal event counting mode, even if the microcontroller is in the Power Down Mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (TE= 1)

Pulse Width Capture Mode

In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin. To operate in this mode, the Operating Mode Select bit pair, TM1/TM0, in the Timer Control Register must be set to the correct value as shown.

Bit7	Bit6
1	1

Control Register Operating Mode Select Bits for the Pulse Width Measurement Mode

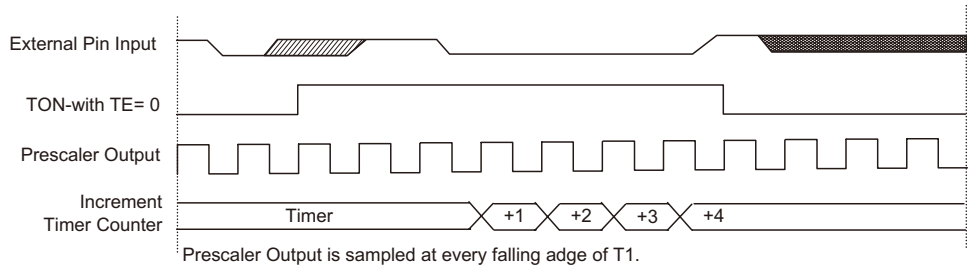
In this mode the internal clock, f_{SYS} is used as the internal clock for the 8-bit Timer/Event Counter. However, the clock source is further divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits TPSC2~TPSC0, which are bits 2~0 of the Timer Control Register. After other bits in the Timer Control Register have been setup, the enable bit TON, which is bit 4 of the Timer Control Register, 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 external timer pin.

If the Active Edge Select bit TE, which is bit 3 of the Timer Control Register, is low, once a high to low transition has been received on the external timer pin, the Timer/Event Counter will start counting until the external timer 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 Select bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. 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 external timer pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under 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 TMR 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 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. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register is reset to zero.

As the external pin is shared with an I/O pin, to ensure that the pin is configured to operate as a pulse width capture pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the pulse width capture mode, the second is to ensure that the port control register configures the pin as an input.



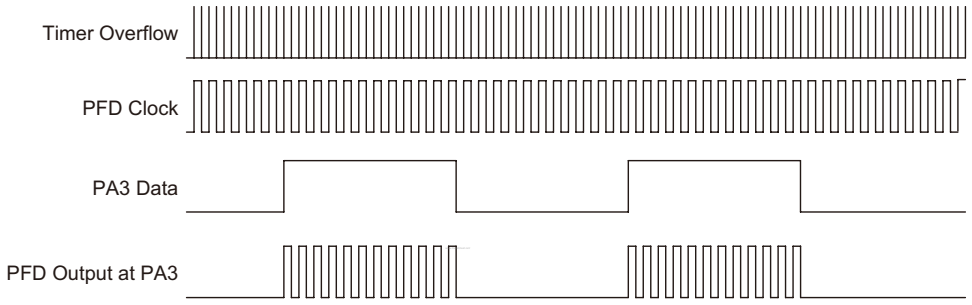
Pulse Width Capture Mode Timing Chart (TE= 0)

Programmable Frequency Divider – PFD

The Programmable Frequency Divider provides a means of producing a variable frequency output suitable for applications requiring a precise frequency generator.

The PFD output is pin-shared with the I/O pin PA5. The PFD function is enabled via PFDEN bit in MISC register, however, if not enabled, the pin can operate as a normal I/O pin.

The output frequency is controlled by loading the required values into the timer registers and prescaler registers to give the required division ratio. The timer will begin to count-up from this preload register value until full, at which point an overflow signal is generated, causing the PFD output to change state. The timer will then be automatically reloaded with the preload register value and continue counting-up.



PFD Output Control

MISC Register

Bit	7	6	5	4	3	2	1	0
Name	ODE3	ODE2	ODE1	ODE0	—	—	—	PFDEN
R/W	R/W	R/W	R/W	R/W	—	—	—	R/W
POR	0	0	0	0	—	—	—	0

Bit 7 **ODE3**: PB3 Open Drain Control
0: Disable
1: Enable

Bit 6 **ODE2**: PB2 Open Drain Control
0: Disable
1: Enable

Bit 5 **ODE1**: PB1 Open Drain Control
0: Disable
1: Enable

Bit 4 **ODE0**: PB0 Open Drain Control
0: Disable
1: Enable

Bit 3 ~ 1 Unimplemented, read as "0"

Bit 0 **PFDEN**: PFD function control
0: Disable
1: Enable

Prescaler

Bits TPSC0~TPSC2 of the TMRC register can be used to define the pre-scaling stages of the internal clock sources of the Timer/Event Counter. The Timer/Event Counter overflow signal can be used to generate signals for the PFD and Timer Interrupt.

I/O Interfacing

The Timer/Event Counter, when configured to run in the event counter or pulse width measurement mode, requires the use of the external pin for correct operation. As this pin is a shared pin it must be configured correctly to ensure it is setup for use as a Timer/Event Counter input and not as a normal I/O pin. This is implemented by ensuring that the mode select bits in the Timer/Event Counter control register, select either the event counter or pulse width measurement mode. Additionally the Port Control Register must be set high to ensure that the pin is setup as an input. Any pull-high resistor on this pin will remain valid even if the pin is used as a Timer/Event Counter input.

Timer/Event Counter Pin Internal Filter

The external Timer/Event Counter pin is connected to an internal filter to reduce the possibility of unwanted event counting events or inaccurate pulse width measurements due to adverse noise or spikes on the external Timer/Event Counter input signal. As this internal filter circuit will consume a limited amount of power, a configuration option is provided to switch off the filter function, an option which may be beneficial in power sensitive applications, but in which the integrity of the input signal is high. Care must be taken when using the filter on/off configuration option as it will be applied not only to the external Timer/Event Counter pin but also to the external interrupt input pins. Individual Timer/Event Counter or external interrupt pins cannot be selected to have a filter on/off function.

Programming Considerations

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

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

the timer mode bits have been properly setup. Setting the timer enable bit high together with a mode bit modification, may lead to improper timer operation if executed as a single timer control register byte write instruction.

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

Timer Program Example

This program example shows how the Timer/Event Counter register is setup, along with how the interrupt is enabled and managed. Note how the Timer/Event Counter is turned on, by setting bit 4 of the Timer Control Register. The Timer/Event Counter can be turned off in a similar way by clearing the same bit. This example program sets the Timer/Event Counter to be in the timer mode, which uses the internal system clock as the clock source.

```
org 04h        ; external interrupt vector
org 0ch        ; Timer/Event Counter interrupt vector
jmp tmrint     ; jump here when the Timer/Event Counter overflows
:
org 20h        ; main program
               ; internal Timer/Event Counter interrupt routine
:
tmrint:
               ; Timer/Event Counter main program placed here
:
reti:
:
begin:
               ; setup Timer registers
mov a,09bh     ; setup Timer preload value
mov tmr,a
mov a,081h     ; setup Timer control register
mov tmrc,a     ; timer mode and prescaler set to /2
               ; setup interrupt register
mov a,009h     ; enable master interrupt and timer interrupt
mov intc0,a
set tmrc.4     ; start Timer/Event Counter - note mode bits must be previously setup
```

LDO function

The device contains a low power voltage regulator implemented in CMOS technology. Using CMOS technology ensures low voltage drop and low quiescent current. There are two fixed output voltages of 2.4V and 3.3V, which can be controlled by a specific register. The internal LDO output combined with various options by register can provide a fixed voltage for the OPA reference voltage, and as a fixed power supply for external device.

LDOC register

Bit	7	6	5	4	3	2	1	0
Name	—	—	RSEL	VLOE	REN1	VRES	VSEL	LDOEN
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 **RSEL**: Select resistor for R_{TOTAL}
0: total 500 kΩ
1: total 200 kΩ

Bit 4 **VLOE**: LDO output voltage control bit
0: Disable
1: Enable
If the VLOE and LDOEN are set to "1", the LDO will output 2.4V or 3.3V to pin and disable I/O function.

Bit 3 **REN1**: Bias voltage divided resistor control bit
0: Disable
1: Enable
If the REN1 is set to "1", that will turn on the resistor DC path, which will generate bias voltage for OPAs.

Bit 2 **VRES**: Divided resistor voltage supply selection bit
0: VDD
1: VLDO

Bit 1 **VSEL**: LDO output voltage selection bit
0: 2.4V
1: 3.3V

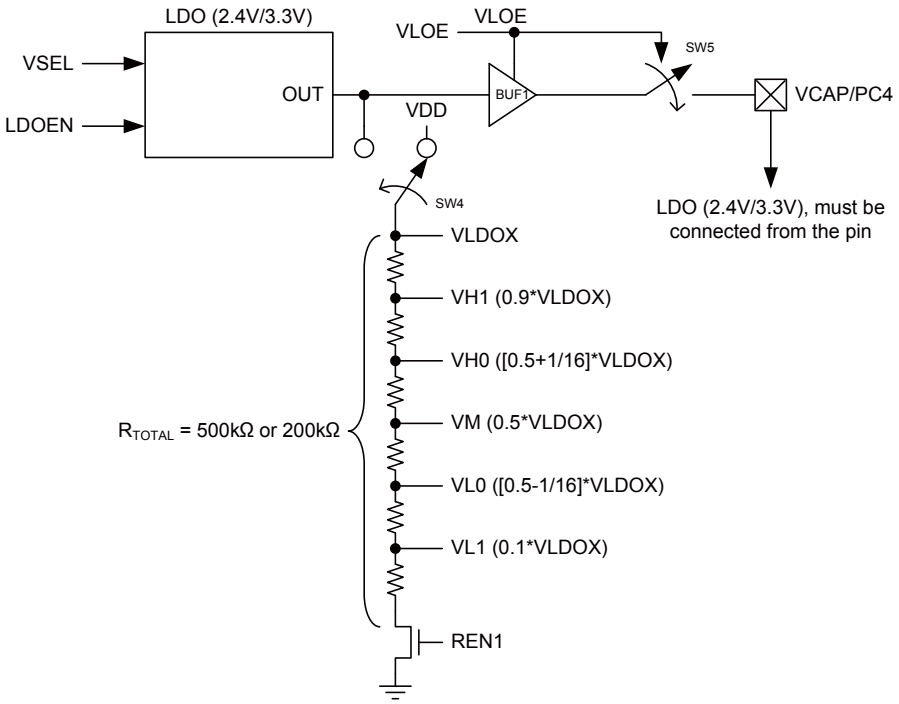
Bit 0 **LDOEN**: LDO control bit
0: Disable
1: Enable

Note: 1. The total resistance of the divided resistor, 500 kΩ or 200 kΩ, can be selected by the RSEL flag.

2. To disable the LDO function will turn off the BUF1 as well, no matter the LDO output voltage control bit, VLOE, is enabled or not.

3. If the LDO is disabled, LDOEN=0, then the SW4 will be turned to VDD, no matter VRES flag is "1" or not.

The following block diagram illustrates the functional structure for LDO and divided resister.



Operational Amplifiers

There are two fully integrated Operational Amplifiers in the device, OPA1 and OPA2. These OPAs can be used for signal amplification according to specific user requirements. The OPAs can be disabled or enabled entirely under software control using internal registers. With specific control registers, some OPA related applications can be more flexible and easier to be implemented, such as Unit Gain Buffer, Non-Inverting Amplifier, Inverting Amplifier and various kinds of filters, etc. In addition, the device provides the calibration function to adjust the OPA offset.

Operational Amplifier Registers

The internal Operational Amplifiers are fully under the control of internal registers, OPA1C0, OPA1C1, OPA2C0, OPA2C1 and OPA2C2. These registers control enable/disable function, input path selection, gain control, polarity and calibration function.

OPA1C0 register

Bit	7	6	5	4	3	2	1	0
Name	A1OP	A1OFM	A1RS	A1OF4	A1OF3	A1OF2	A1OF1	A1OF0
R/W	R	R/W	R/W	R	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **A1OP**: Operational amplifier output; positive logic. This bit is read only.

Bit 6 **A1OFM**: Operational amplifier mode or input offset voltage cancellation mode
0: Operational amplifier mode
1: Input offset voltage cancellation mode

Bit 5 **A1RS**: Operational amplifier input offset voltage cancellation reference selection bit
0: Select A1N as the reference input
1: Select A1P as the reference input

Bit 4~0 **A1OF4~A1OF0**: Operational amplifier input offset voltage cancellation control bits

OPA1C1 register

Bit	7	6	5	4	3	2	1	0
Name	A1O2CIN	A1O2N	A1PSEL1	A1PSEL0	A1PS	A1NS	A1OEN	A1EN
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	1	0	0

Bit 7 **A1O2CIN**: OPA1 output to comparator input control bit
0: Disable
1: Enable

Bit 6 **A1O2N**: OPA1 output to OPA1 Inverting input control bit
0: Disable
1: Enable

Bit 5~4 **A1PSEL1, A1PSEL0**: OPA1 Non-inverting input selection bit
00: No connection
01: From $0.9V_{LDO}$
10: From VM
11: From $0.1V_{LDO}$

Bit 3 **A1PS**: OPA1 positive input switch On/Off bit
0: Off
1: On

Bit 2 **A1NS**: OPA1 negative input switch On/Off bit
0: Off
1: On

Bit 1 **A1OEN**: OPA1 output enable or disable control bit
0: Disable
1: Enable

Bit 0 **A1EN**: OPA1 enable or disable control bit
0: Disable
1: Enable

OPA2C0 register

Bit	7	6	5	4	3	2	1	0
Name	A2OP	A2OFM	A2RS	A2OF4	A2OF3	A2OF2	A2OF1	A2OF0
R/W	R	R/W	R/W	R	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **A2OP**: Operational amplifier output; positive logic. This bit is read only.

Bit 6 **A2OFM**: Operational amplifier mode or input offset voltage cancellation mode
0: Operational amplifier mode
1: Input offset voltage cancellation mode

Bit 5 **A2RS**: Operational amplifier input offset voltage cancellation reference selection bit
0: Select A2N as the reference input
1: Select A2P as the reference input

Bit 4~0 **A2OF4~A2OF0**: Operational amplifier input offset voltage cancellation control bits

OPA2C1 register

Bit	7	6	5	4	3	2	1	0
Name	A2O2CIN	A2O2N	A2PSEL1	A2PSEL0	A2PS	A2NS	A2OEN	A2EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	1	0	0

Bit 7 **A2O2CIN**: OPA2 output to comparator input control bit
0: Disable
1: Enable

Bit 6 **A2O2N**: OPA2 output to OPA2 Inverting input control bit
0: Disable
1: Enable

Bit 5~4 **A2PSEL1, A2PSEL0**: OPA2 Non-inverting input selection bit
00: No connection
01: From $0.9V_{LDO}$
10: From VM
11: From $0.1V_{LDO}$

Bit 3 **A2PS**: OPA2 positive input switch On/Off bit
0: Off
1: On

Bit 2 **A2NS**: OPA2 negative input switch On/Off bit
0: Off
1: On

Bit 1 **A2OEN**: OPA2 output enable or disable control bit
0: Disable
1: Enable

Bit 0 **A2EN**: OPA2 enable or disable control bit
0: Disable
1: Enable

OPA2C2 register

Bit	7	6	5	4	3	2	1	0
Name	A1O2A2N	A1O2A2P	—	—	PGAEN	PGA2	PGA1	PGA0
R/W	R/W	R/W	—	—	R/W	R/W	R/W	R/W
POR	0	0	—	—	0	0	0	0

Bit 7 **A1O2A2N**: OPA1 output to OPA2 Inverting input control bit
0: Disable
1: Enable

Bit 6 **A1O2A2P**: OPA1 output to OPA2 Non-inverting input control bit
0: Disable
1: Enable

Bit 5~4 Unimplemented, read as "0"

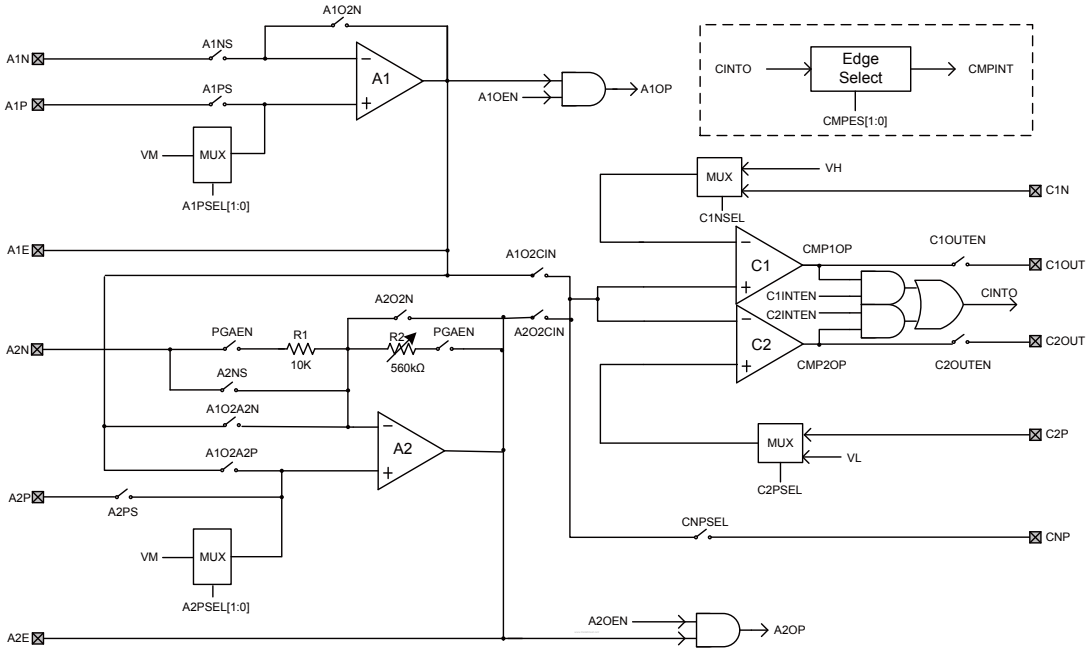
Bit 3 **PGAEN**: OPA2 PGA gain enable control bits
0: Off
1: On

Bit 2~0 **PGA2, PGA1, PGA0**: OPA2 Gain control bits
000: 1
001: 8
010: 16
011: 24
100: 32
101: 40
110: 48
111: 56

Operational Amplifier Operation

The advantages of multiple switches and input path options, various reference voltage selection, up to 8 kinds of internal software gain control, offset reference voltage calibration function and power down control for low power consumption enhance the flexibility of these two OPAs to suit a wide range of application possibilities.

The following block diagram illustrates the main functional blocks of the OPAs and Comparator in this device.



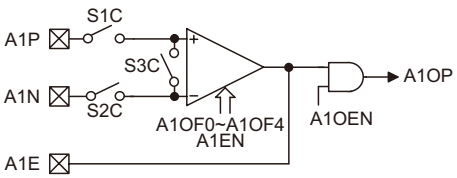
Operational Amplifier Functions

The OPAs are connected together internally in a specific way and the output of OPAs can also be connected to the internal comparators as shown in the block diagram. Each of the OPAs has its own control register, with the name OPA1C0, OPA1C1, OPA2C0, OPA2C1 and OPA2C2 which are used to control the enable/disable function, the calibration procedure and the programmable gain function of OPA2.

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

The calibration steps are as following:

- Set A1OFM=1 to setup the offset cancellation mode, here S3C is closed.
- Set A1RS to select which input pin is to be used as the reference voltage - S1C or S2C is closed
- Adjust A1OF0~A1OF4 until the output status changes
- Set A1OFM = 0 to restore the normal OPA mode
- Repeat the same procedure from steps 1 to 4 for OPA2



A1OFM	A1RS	S1C	S2C	S3C
0	X	ON	ON	OFF
1	0	OFF	ON	ON
0	1	ON	OFF	ON

"X": don't care

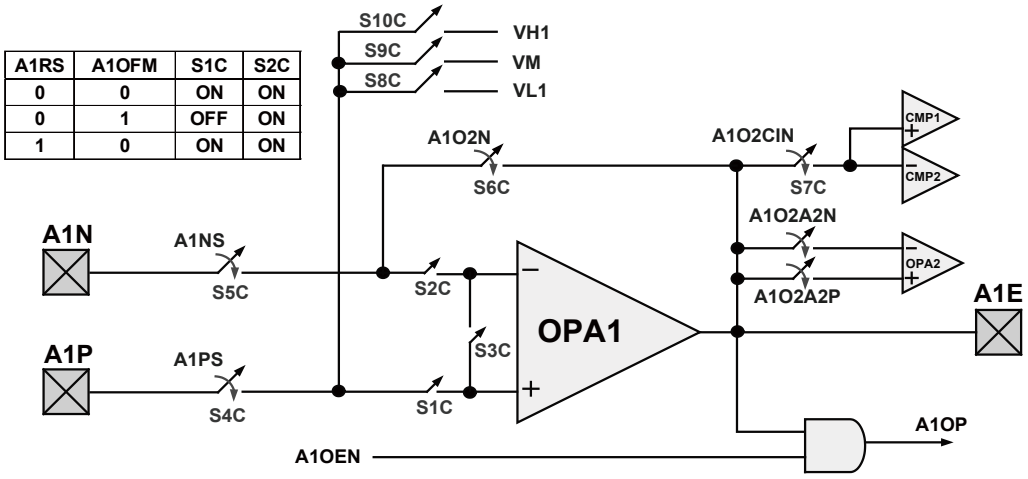
OPA1 Switch Control

The following diagram and table illustrate the OPA1 switch control setting and the corresponding connections. Note that some switch control selections will force some switches to be controlled by hardware automatically.

For example:

- The S7C is closed when A1O2CIN= 1 and the S7C is opened when A1O2CIN = 0.
- The A1PS= 1 will force A1PSEL1, 0 = (00), i.e. S10C, S9C, S8C will be opened.
- When the A1EN= 0, S6C switch are opened by hardware, then the related I/O pins can be used as the other functions.

A1RS	A1OFM	S1C	S2C
0	0	ON	ON
0	1	OFF	ON
1	0	ON	ON



A1PS=1 will force A1PSEL1,0=(00), i.e. S10C,S9C, S8C will be opened.

OPA1 switch control:

The following table illustrates the relationship between OPA1 control register settings and the switches:

OPA1 Control Bits in OPA1C0, OPA1C1						Switch Description				Results
A1PS	A1NS	A1OFM	A1RS	A1PSEL1 A1PSEL0	A1O2N	S4C	S5C	S6C	S8C~ S10C	OPA1 connections
1	1	1	0	00	0	ON	ON	OFF	OFF	Calibration mode, A1N
1	1	1	1	00	0	ON	ON	OFF	OFF	Calibration mode, A1P
1	1	0	×	00	0	ON	ON	OFF	OFF	Input = A1N, A1P
0	1	0	×	01	0	OFF	ON	OFF	S10C ON	Input = A1N, VH1
0	1	0	×	10	0	OFF	ON	OFF	S9C ON	Input = A1N, VM
0	1	0	×	11	0	OFF	ON	OFF	S8C ON	Input = A1N, VL1
1	1	0	×	00	1	ON	ON	ON	OFF	Input = A1N, A1P, Connect A1N, A1E
1	0	0	×	00	1	ON	OFF	ON	OFF	Input = A1P, OPA1 as buffer
0	1	0	×	01	0	OFF	ON	OFF	S10C ON	Input = A1N, VH1
0	1	0	×	10	0	OFF	ON	OFF	S9C ON	Input = A1N, VM
0	1	0	×	11	0	OFF	ON	OFF	S8C ON	Input = A1N, VL1

Note: 1. S7C is closed when A1O2CIN=1, S7C is opened when A1O2CIN =0.
2. When A1OFM=1, S3C, S4C, S5C are closed, S6C, S8C~S10C are always opened.

OPA1 & I/O status description:

The following table illustrates the OPA1 & I/O settings.

A1EN	A1NS	A1PS	Description
0	×	×	PA2 and PA3 and PA4 are I/Os
1	0	0	PA2 and PA3 are I/Os, PA4 is OPA1 A1E output
1	0	1	PA3 is I/O. PA2 is OPA1 A1P input, PA4 is OPA1 A1E output
1	1	0	PA2 is I/O. PA3 is OPA1 A1N input, PA4 is OPA1 A1E output
1	1	1	PA2 is OPA1 A1P input and PA3 is OPA1 A1N input, PA4 is OPA1 A1E output

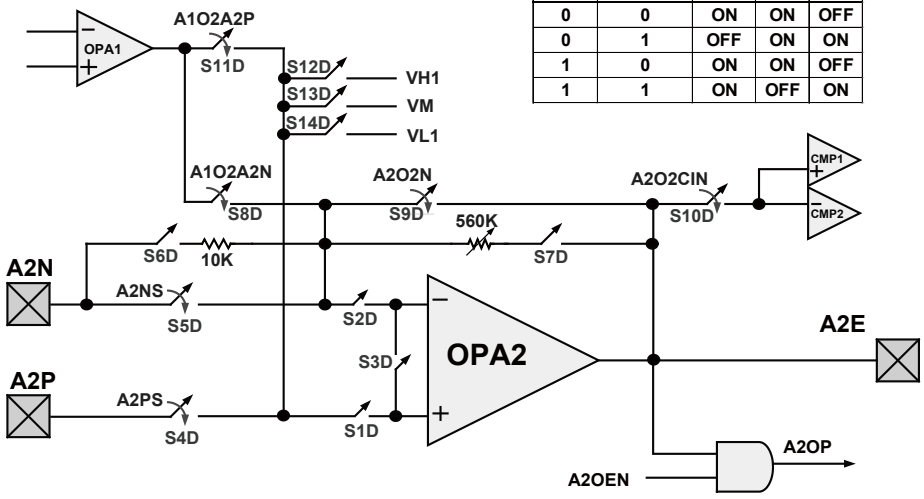
OPA2 Switch Control

The following diagram and table illustrate the OPA2 switch control setting and the corresponding connections. Note that some switch control selections will force some switches to be controlled by hardware automatically.

For example:

- The PGAEN=1 will force S6D, S7D to close and the PGAEN=0 will force S6D, S7D to open.
- When the A2EN=0, these switches, S6D, S7D and S9D, are opened by hardware, then the related I/O pins can be used as the other functions.

A2RS	A2OFM	S1D	S2D	S3D
0	0	ON	ON	OFF
0	1	OFF	ON	ON
1	0	ON	ON	OFF
1	1	ON	OFF	ON



Switch priority : S4D > S11D > (S12D, S13D, S14D); If A2PS=1, S11D~S14D will be opened by hardware.
Switch priority : S5D > S8D; If A2NS=1, S8D will be opened by hardware.

OPA2 switch control:

The following table illustrates the relationship between OPA2 control register settings and the switches:

OPA2 Control Bits in OPA2C0, OPA2C1, OPA2C2									Switch Description							Results
A2PS	A2NS	A2OFM	A2RS	A2PSEL1,0	A1O2A2P	A1O2A2N	PGAEN	A2O2N	S4D	S5D	S6/7D	S8D	S9D	S11D	S12D~S14D	OPA2 connections
1	0	1	1	00	0	0	0	0	ON	ON	OFF	OFF	OFF	OFF	OFF	Calibration mode, Input = A2P
1	1	1	0	00	0	0	0	0	ON	ON	OFF	OFF	OFF	OFF	OFF	Calibration mode, Input = A2N
1	1	0	×	00	0	0	0	0	ON	ON	OFF	OFF	OFF	OFF	OFF	Normal mode, Input = A2N, A2P
0	1	0	×	01	0	0	0	0	OFF	ON	OFF	OFF	OFF	OFF	S12D ON	Input = A2N, VH1
0	1	0	×	10	0	0	0	0	OFF	ON	OFF	OFF	OFF	OFF	S13D ON	Input = A2N, VM
0	1	0	×	11	0	0	0	0	OFF	ON	OFF	OFF	OFF	OFF	S14D ON	Input = A2N, VL1
0	1	0	×	00	1	0	0	0	OFF	ON	OFF	OFF	OFF	ON	OFF	Input = A2N, A1E
1	0	0	×	00	0	1	0	0	ON	OFF	OFF	ON	OFF	OFF	OFF	Input = A1E, A2P
1	1	0	×	00	0	0	1	0	ON	ON	ON	OFF	OFF	OFF	OFF	Input = A2N, A2P
1	0	0	×	00	0	0	1	0	ON	OFF	ON	OFF	OFF	OFF	OFF	Input = A2N, A2P
1	0	0	×	00	0	0	0	1	ON	OFF	OFF	OFF	ON	OFF	OFF	Input = A2P, OPA2 as buffer
0	0	0	×	01	0	0	0	1	OFF	OFF	OFF	OFF	OFF	OFF	S12D ON	Input = VH1, OPA2 as buffer
0	0	0	×	10	0	0	0	1	OFF	OFF	OFF	OFF	OFF	OFF	S13D ON	Input = VM, OPA2 as buffer
0	0	0	×	11	0	0	0	1	OFF	OFF	OFF	OFF	OFF	OFF	S14D ON	Input = VL1, OPA2 as buffer

OPA2 & I/O status description:

The following table illustrates the OPA2 & I/O settings.

A2EN	PGAEN	A2NS	A2PS	Description
0	×	×	×	PA5 and PA6 and PA7 are I/Os
1	0	0	0	PA5 and PA6 are I/Os. PA7 is OPA2 A2E output
1	0	0	1	PA6 is I/O. PA5 is OPA2 A2P input, PA7 is OPA2 A2E output
1	0	1	0	PA5 is I/O. PA6 is OPA2 A2N input, PA7 is OPA2 A2E output
1	0	1	1	PA5 is OPA2 A2P input and PA6 is OPA2 A2N input, PA7 is OPA2 A2E output
1	1	0	0	PA5 is I/O. PA6 is OPA2 A2N input, PA7 is OPA2 A2E output
1	1	0	1	PA5 is OPA2 A2P input and PA6 is OPA2 A2N input, PA7 is OPA2 A2E output
1	1	1	0	PA5 is I/O. PA6 is OPA2 A2N input and bypass R1 (10kΩ), PA7 is OPA2 A2E output
1	1	1	1	PA5 is OPA2 A2P input and PA6 is OPA2 A2N input and bypass R1 (10kΩ), PA7 is OPA2 A2E output

Comparator

Two analog comparators are contained within this device. These functions offer flexibility via their register controlled features such as power-down, interrupt etc. In sharing their pins with normal I/O pins, the comparators do not waste precious I/O pins if there functions are otherwise unused. In addition, the device provides the calibration function to adjust the comparator offset.

Comparator Operation

The device contains two comparator functions which are used to compare two analog voltages and provide an output based on their difference. Full control over the two internal comparators is provided via control registers, CMP1C0, CMP1C1, CMP2C0 and CMP2C1. The comparator output is recorded via a bit in their respective control register, but can also be transferred out onto a shared I/O pin or to generate an interrupt trigger with edge control function. Additional comparator functions include the power down control.

Comparator Registers

The internal dual comparators are fully under the control of internal registers, CMP1C0, CMP1C1, CMP2C0 and CMP2C1. These registers control enable/disable function, input path selection, interrupt edge control and input offset voltage calibration function.

CMP1C0 register

Bit	7	6	5	4	3	2	1	0
Name	CMP1OP	C1OFM	C1RS	C1OF4	C1OF3	C1OF2	C1OF1	C1OF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **CMP1OP**: Comparator output; positive logic. This bit is read only.

Bit 6 **C1OFM**: Comparator mode or input offset voltage cancellation mode
0: Comparator mode
1: Input offset voltage cancellation mode
When the C1OFM=1, comparator inputs are always from I/O pins. i.e. the CNPSEL and C1NSEL will be forced to "1". That means disconnect the input from OPAs output.

Bit 5 **C1RS**: Comparator input offset voltage cancellation reference selection bit
0: Select C1N as the reference input
1: Select CNP as the reference input

Bit 4~0 **C1OF4~C1OF0**: Comparator input offset voltage cancellation control bits

CMP1C1 register

Bit	7	6	5	4	3	2	1	0
Name	CNPSEL	—	—	—	C1INTEN	C1OUTEN	C1NSEL	CMP1EN
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	1	—	—	—	0	0	1	0

Bit 7 **CNPSEL**: Comparator non-inverting input control bit
0: From OPA output
1: From CNP pin

Bit 6~4 Unimplemented, read as "0"

Bit 3 **C1INTEN**: Comparator 1 interrupt control bit
0: Disable
1: Enable

Bit 2 **C1OUTEN**: Comparator 1 output pin control bit
0: Disable
1: Enable

Bit 1 **C1NSEL**: Comparator 1 inverting input control bit
0: From VH0
1: From C1N pin

Bit 0 **CMP1EN**: Comparator 1 enable or disable control bit
0: Disable
1: Enable

CMP2C0 register

Bit	7	6	5	4	3	2	1	0
Name	CMP2OP	C2OFM	C2RS	C2OF4	C2OF3	C2OF2	C2OF1	C2OF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **CMP2OP**: Comparator output; positive logic. This bit is read only.

Bit 6 **C2OFM**: Comparator mode or input offset voltage cancellation mode
0: Comparator mode
1: Input offset voltage cancellation mode
When the C2OFM=1, comparator inputs are always from I/O pins. i.e. the CNPSEL and C1NSEL will be forced to "1". That means disconnect the input from OPAs output.

Bit 5 **C2RS**: Comparator input offset voltage cancellation reference selection bit
0: Select C2P as the reference input
1: Select CNP as the reference input

Bit 4~0 **C2OF4~C2OF0**: Comparator input offset voltage cancellation control bits

CMP2C1 register

Bit	7	6	5	4	3	2	1	0
Name	CMPES1	CMPES0	—	—	C2INTEN	C2OUTEN	C2PSEL	CMP2EN
R/W	R/W	R/W	—	—	R/W	R/W	R/W	R/W
POR	0	0	—	—	0	0	1	0

Bit 7~6 **CMPES1, CMPES0**: Interrupt edge control bits
00: Disable
01: Rising edge trigger
10: Falling edge trigger
11: Dual edge trigger

Bit 5~4 Unimplemented, read as "0"

Bit 3 **C2INTEN**: Comparator 2 interrupt control bit
0: Disable
1: Enable

Bit 2 **C2OUTEN**: Comparator 2 output pin control bit
0: Disable
1: Enable

Bit 1 **C2PSEL**: Comparator 2 non-inverting input control bit
0: From VL0
1: From C2P pin

Bit 0 **CMP2EN**: Comparator 2 enable or disable control bit
0: Disable
1: Enable

Comparator Functions

Theses two comparators can operate together with the OPAs or standalone as shown in the main functional blocks of the OPAs and Comparators.

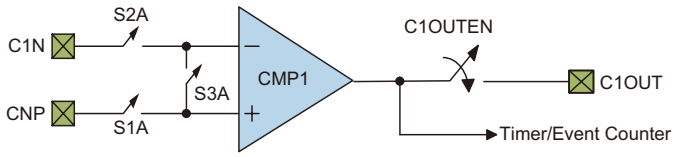
Each of the internal comparators allows for a common mode adjustment method of its input offset voltage.

The calibration steps are as following:

- Set C1OFM=1 to setup the offset cancellation mode, here S3A is closed.
- Set C1RS to select which input pin is to be used as the reference voltage - S1A or S2A is closed.
- Adjust C1OF0~C1OF4 until the output status changes.
- Set C1OFM = 0 to restore the normal comparator mode.
- Repeat the same procedure from steps 1 to 4 for comparator 2.

C1OFM	C1RS	S1A	S2A	S3A
0	X	ON	ON	OFF
1	0	OFF	ON	ON
1	1	ON	OFF	ON

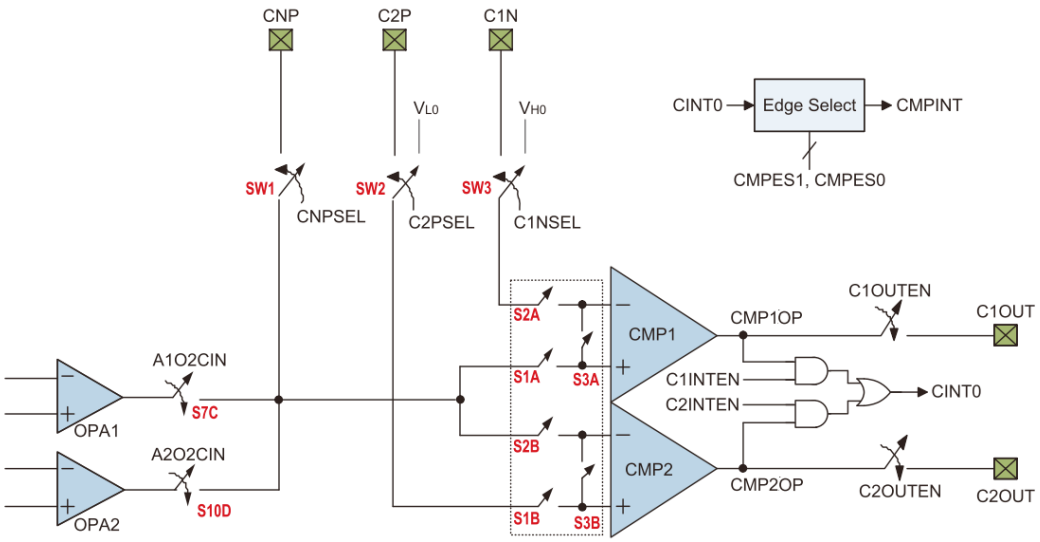
"X": don't care



The following diagram and table illustrate the comparators switch control setting and the corresponding connections. Note that some switch control selections will force some switches to be controlled by hardware automatically.

For example:

- When the CMP1 in calibration mode, i.e. C1OFM =1, then the SW1, SW3 will be forced to close. The CNPSEL and C1NSEL bits will be set “1” by hardware, and these two bits will be read out as “1”. After the offset voltage calibration, the CNPSEL and C1NSEL will be back to its original value.
- When the CMP2 in calibration mode, i.e. C2OFM =1, then the SW1, SW2 will be forced to close.
- The CNPSEL and C2PSEL bits will be set “1” by hardware, and these two bits will be read out as “1”. After the offset voltage calibration, the CNPSEL and C2PSEL will be back to its original value.
- If the CNPSEL=1, the A1O2CIN and A2O2CIN will be forced to “0”, i.e. If the SWI is closed, and that will force S7C and S10D to open.
- If the CNPSEL=0 and the A1O2CIN=1, the A2O2CIN will be forced to “0”, i.e. If the S7C is closed, and that will force S10D to open.



CMP1 & I/O status description:

The following table illustrates the CMP1 & I/O settings.

CMP1EN	C1OUTEN	CNPSEL	C1NSEL	Description
0	×	×	×	PC5 and PA0 and PA1 are I/Os
1	0	0	0	CNP is from OPA1 or OPA2 output, C1N is from VH0 input, PA1 is I/O
1	0	0	1	CNP is from OPA1 or OPA2 output, C1N is from PC5 input, PA1 is I/O
1	0	1	0	CNP is from PA0 input, C1N is from VH0 input, PA1 is I/O
1	0	1	1	CNP is from PA0 input, C1N is from PC5 input, PA1 is I/O
1	1	0	0	CNP is from OPA1 or OPA2 output, C1N is from VH0 input, PA1 is comparator output
1	1	0	1	CNP is from OPA1 or OPA2 output, C1N is from PC5 input, PA1 is comparator output
1	1	1	0	CNP is from PA0 input, C1N is from VH0 input, PA1 is comparator output
1	1	1	1	CNP is from PA0 input, C1N is from PC5 input, PA1 is comparator output

CMP2 & I/O status description:

The following table illustrates the CMP2 & I/O settings.

CMP2EN	C2OUTEN	CNPSEL	C2PSEL	Description
0	×	×	×	PC6 and PA0 and PA2 are I/Os
1	0	0	0	CNP is from OPA1 or OPA2 output, C2P is from VL0 input, PA2 is I/O
1	0	0	1	CNP is from OPA1 or OPA2 output, C2P is from PC6 input, PA2 is I/O
1	0	1	0	CNP is from PA0 input, C2P is from VL0 input, PA2 is I/O
1	0	1	1	CNP is from PA0 input, C2P is from PC6 input, PA2 is I/O
1	1	0	0	CNP is from OPA1 or OPA2 output, C2P is from VL0 input, PA2 is comparator output
1	1	0	1	CNP is from OPA1 or OPA2 output, C2P is from PC6 input, PA2 is comparator output
1	1	1	0	CNP is from PA0 input, C2P is from VL0 input, PA2 is comparator output
1	1	1	1	CNP is from PA0 input, C2P is from PC6 input, PA2 is comparator output

Comparators switch control:

The following table illustrates the relationship between comparators control register settings and the switches:

CMP1,CMP2 Control Bits					Switch Description								Results
CNPSEL	C2PSEL	C1NSEL	C1OFM	C1RS	SW1	SW2	SW3	S1A	S2A	S3A	S7C	S10D	Connections
1 (Forced to 1)	×	1 (Forced to 1)	1	1	ON CNP	×	ON C1N	ON	OFF	ON	OFF	OFF	Input common mode = CNP
1 (Forced to 1)	×	1 (Forced to 1)	1	0	ON CNP	×	ON C1N	OFF	ON	ON	OFF	OFF	Input common mode = C1N
1	0	1	0	×	ON	×	C1N	ON	ON	OFF	OFF	OFF	Input = CNP, C1N
1	0	0	0	×	ON	×	VH	ON	ON	OFF	OFF	OFF	Input = CNP, VH1
0	0	1	0	×	OFF	×	C1N	ON	ON	OFF	ON	OFF	Input = A1E, C1N
0	0	1	0	×	OFF	×	C1N	ON	ON	OFF	OFF	ON	Input = A2E, C1N

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions The external interrupts are generated by the action of the external $\overline{INT0}$~$\overline{INT1}$ pins, while the internal interrupts are generated by various internal functions such as the Comparator, Timer/Event Counter etc.

Interrupt Registers

Overall interrupt control, which basically means interrupt enabling and request flag setting, is controlled by the INTC0, INTC1, MFIC0 and MFIC1 registers, which are located in the Data Memory. By controlling the appropriate enable bits in these registers each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request flag will be set by the microcontroller. The global enable flag if cleared to zero will disable all interrupts.

Interrupt Operation

A Timer/Event Counter overflow, Time Base 0/1, the external interrupt line being triggered, a comparator output, an EEPROM Write or Read cycle ends will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP statement which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI statement, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

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

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded. If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full.

Interrupt Priority

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

Interrupt Source	Priority	Vector
External interrupt 0	1	04H
External interrupt 1	2	08H
Timer/Event Counter overflow	3	0CH
Multi-function Interrupt	4	18H

The Time Base interrupts, Comparator interrupt and EEPROM interrupt all share the same interrupt vector which is 18H. Each of these interrupts has their own individual interrupt flag but also share the same MFF interrupt flag. The MFF flag will be cleared by hardware once the Multi-function interrupt is serviced, however the individual interrupts that have triggered the Multi-function interrupt need to be cleared by the application program.

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	TF	EIF1	EIF0	ETI	EEI1	EEI0	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: Inactive
1: Active

Bit 5 **EIF1**: External interrupt 1 request flag
0: Inactive
1: Active

Bit 4 **EIF0**: External interrupt 0 request flag
0: Inactive
1: Active

Bit 3 **ETI**: Timer/Event counter interrupt enable
0: Disable
1: Enable

Bit 2 **EEI1**: External interrupt 1 enable
0: Disable
1: Enable

Bit 1 **EEI0**: External interrupt 0 enable
0: Disable
1: Enable

Bit 0 **EMI**: Master interrupt global enable
0: Global disable
1: Global enable

INTC1 Register

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

Bit 7 Unimplemented, read as "0"

Bit 6 **MFF**: Multi-function interrupt request flag
0: Inactive
1: Active

Bit 5~3 Unimplemented, read as "0"

Bit 2 **EMFI**: Multi-function interrupt enable
0: Disable
1: Enable

Bit 1~0 Unimplemented, read as "0"

MFIC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	TB1F	TB0F	—	—	TB1E	TB0E	—
R/W	—	R/W	R/W	—	—	R/W	R/W	—
POR	—	0	0	—	—	0	0	—

Bit 7 Unimplemented, read as "0"

Bit 6 **TB1F**: Time Base 1 interrupt request flag
0: Inactive
1: Active

Bit 5 **TB0F**: Time Base 0 interrupt request flag
0: Inactive
1: Active

Bit 4~3 Unimplemented, read as "0"

Bit 2 **TB1E**: Time Base 1 interrupt enable
0: Disable
1: Enable

Bit 1 **TB0E**: Time Base 0 interrupt enable
0: Disable
1: Enable

Bit 0 Unimplemented, read as "0"

MFIC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	E2F	CF	—	—	E2I	ECI
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **E2F**: EEPROM interrupt request flag
0: Inactive
1: Active

Bit 4 **CF**: Comparator interrupt request flag
0: Inactive
1: Active

Bit 3~2 Unimplemented, read as "0"

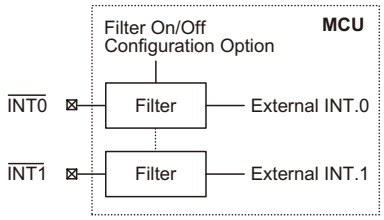
Bit 1 **E2I**: EEPROM interrupt enable
0: Disable
1: Enable

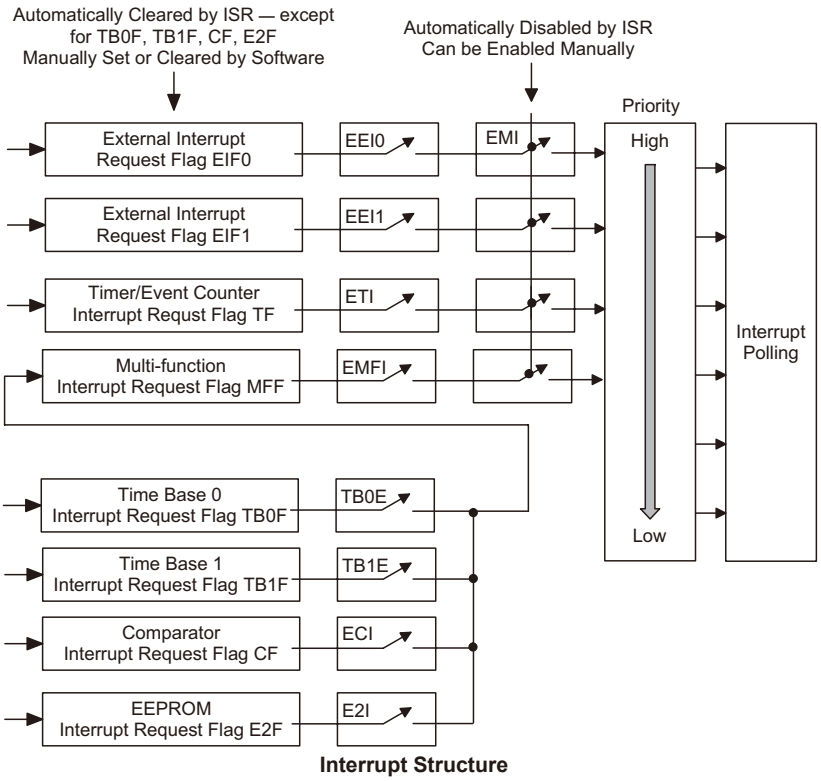
Bit 0 **ECI**: Comparator interrupt enable
0: Disable
1: Enable

External Interrupt

The external interrupts are controlled by signal transitions on the pins $\overline{\text{INT0}}$~$\overline{\text{INT1}}$. An external interrupt request will take place when the external interrupt request flags, EIF0~EIF1, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, EEI0~EEI1, must first be set. Additionally the correct interrupt edge type must be selected using the INTEDGE register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup 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 flags, EIF0~EIF1, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

The INTEDGE 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 INTEDGE register can also be used to disable the external interrupt function.





Interrupt Structure

The external interrupt pins are connected to an internal filter to reduce the possibility of unwanted external interrupts due to adverse noise or spikes on the external interrupt input signal. As this internal filter circuit will consume a limited amount of power, a configuration option is provided to switch off the filter function, an option which may be beneficial in power sensitive applications, but in which the integrity of the input signal is high. Care must be taken when using the filter on/off configuration option as it will be applied not only to both the external interrupt pins but also to the Timer/Event Counter external input pin. Individual external interrupt or Timer/Event Counter pin cannot be selected to have a filter on/off function.

INTEDGE Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INT1S1, INT1S0**: $\overline{\text{INT1}}$ Edge select
00: Disable
01: Rising edge trigger
10: Falling edge trigger
11: Dual edge trigger

Bit 1~0 **INT0S1, INT0S0**: $\overline{\text{INT0}}$ Edge select
00: Disable
01: Rising edge trigger
10: Falling edge trigger
11: Dual edge trigger

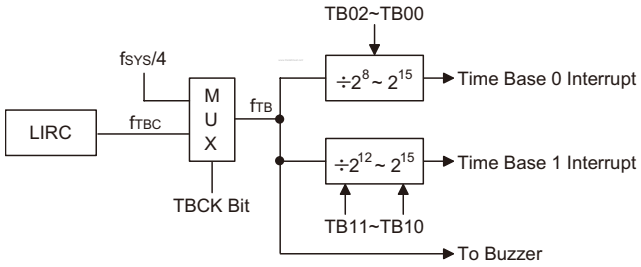
Timer/Event Counter Interrupt

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, ETI, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TF, is set, asituation that will occur when the Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter overflow occurs, a subroutine call to the timer interrupt vector at location 0CH, will take place. When the interrupt is serviced, the timer interrupt request flag, TF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Time Base Interrupt

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, 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. Their clock sources originate from the internal clock source f_{TB}. This f_{TB} input clock 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 that generates f_{TB}, which in turn controls the Time Base interrupt period, can originate from several different sources, as shown in the System Operating Mode section.



Time Base Interrupt

TBC Register

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

bit 7 **TBON**: TB0 and TB1 Control bit
0: Disable
1: Enable

bit 6 **TBCK**: Select f_{TB} Clock
0: f_{TBC}
1: $f_{SYS}/4$

bit 5~4 **TB11 ~ TB10**: Select Time Base 1 Time-out Period
00: $4096/f_{TB}$
01: $8192/f_{TB}$
10: $16384/f_{TB}$
11: $32768/f_{TB}$

bit 3 Unimplemented, read as "0"

bit 2~0 **TB02 ~ TB00**: Select Time Base 0 Time-out Period
000: $256/f_{TB}$
001: $512/f_{TB}$
010: $1024/f_{TB}$
011: $2048/f_{TB}$
100: $4096/f_{TB}$
101: $8192/f_{TB}$
110: $16384/f_{TB}$
111: $32768/f_{TB}$

Multi-function Interrupt

An additional interrupt known as the Multi-function interrupt is provided. Unlike the other interrupts, this interrupt has no independent source, but rather is formed from four other existing interrupt sources, namely the Time Base interrupts, Comparator interrupt and EEPROM interrupt.

For a Multi-function interrupt to occur, the global interrupt enable bit, EMI, and the Multi-function interrupt enable bit, EMFI, must first be set. An actual Multi-function interrupt will take place when the Multi-function interrupt request flag, MFF, is set. This will occur when either a Time Base overflow, a Comparator output interrupt or an EEPROM Write or Read cycle ends interrupt is generated. When the interrupt is enabled and the stack is not full, and either one of the interrupts contained within the Multi-function interrupt occurs, a subroutine call to the Multi-function interrupt vector at location 018H will take place. When the interrupt is serviced, the Multi-Function request flag, MFF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. However, it must be noted that the request flags from the original source of the Multi-function interrupt, namely the Time-Base interrupt or a Comparator output interrupt, will not be automatically reset and must be manually reset by the application program.

Comparator Interrupt

The Comparator Interrupt is contained within the Multi-function Interrupt. The comparator interrupt is controlled by the two internal comparators. A comparator interrupt request will take place when the comparator interrupt request flag, CF, is set, a situation that will occur when the comparator output changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, the Multi-function interrupt enable bit EMFI and comparator

interrupt enable bit, ECI, must first be set. When the interrupt is enabled, the stack is not full and the comparator input generates a comparator output transition, a subroutine call to the comparator interrupt vector, will take place. When the Comparator Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the CF flag will not be automatically cleared, it has to be cleared by the application program.

EEPROM Interrupt

The EEPROM interrupt, is contained within the Multi-function Interrupt. An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, E2F, is set, which occurs when an EEPROM Write or Read cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, EEPROM Interrupt enable bit, E2I, and the Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the respective Multi-function Interrupt vector, will take place. When the EEPROM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the E2F flag will not be automatically cleared, it has to be cleared by the application program.

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 pins, a low power supply voltage or comparator input change may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

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

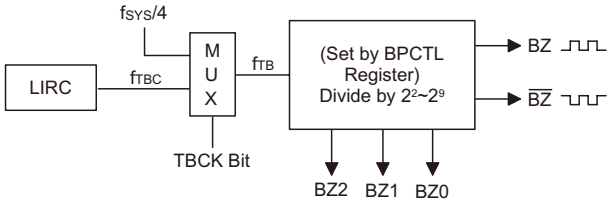
It is recommended that programs do not use the “CALL” instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged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

All of these interrupts have the capability of waking up the processor when in the Power Down Mode. Only the Program Counter is pushed onto the stack. If the contents of the status or other registers are altered by the interrupt service program, which may corrupt the desired control sequence, then the contents should be saved in advance.

Buzzer

Operating in a similar way to the Programmable Frequency Divider, the Buzzer function provides a means of producing a variable frequency output, suitable for applications such as Piezo-buzzer driving or other external circuits that require a precise frequency generator. The BZ and $\overline{BZ}$ pins form a complimentary pair, and are pin-shared with I/O pins, PA6 and PA7. A BPCTL register is used to select from one of three buzzer options. The first option is for both pins PA6 and PA7 to be used as normal I/Os, the second option is for both pins to be configured as BZ and $\overline{BZ}$ buzzer pins, the third option selects only the PA6 pin to be used as a BZ buzzer pin with the PA7 pin retaining its normal I/O pin function. Note that the $\overline{BZ}$ pin is the inverse of the BZ pin which together generates a differential output which can supply more power to connected interfaces such as buzzers.

The buzzer is driven by the internal clock source, f_{TB}, which then passes through a divider, the division ratio of which is selected by BPCTL register to provide a range of buzzer frequencies from $f_{TB}/2^2$ to $f_{TB}/2^9$. The clock source that generates f_{TB}, which in turn controls the buzzer frequency, can originate from two different sources, the LIRC oscillator or the System oscillator/4, the choice of which is determined by the f_{TB} clock source option. Note that the buzzer frequency is controlled by BPCTL register, which select the source clock for the internal clock f_{TB}.



Buzzer Function

If the BPCTL options have selected both pins PA6 and PA7 to function as a BZ and $\overline{BZ}$ complementary pair of buzzer outputs, then for correct buzzer operation it is essential that both pins must be setup as outputs by setting bits PAC6 and PAC7 of the PAC port control register to zero. The PA6 data bit in the PA data register must also be set high to enable the buzzer outputs, if set low, both pins PA6 and PA7 will remain low. In this way the single bit PA6 of the PA register can be used as an on/off control for both the BZ and $\overline{BZ}$ buzzer pin outputs. Note that the PA7 data bit in the PA register has no control over the $\overline{BZ}$ buzzer pin PA7.

BPCTL Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	BC1	BC0	BZ2	BZ1	BZ0
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 4~3 **BC1, BC0**: Buzzer or I/O
- 00: PA7 is I/O, PA6 is I/O
- 01: PA7 is I/O, PA6 is BZ
- 10: Reserved
- 11: PA7 is $\overline{BZ}$, PA6 is BZ

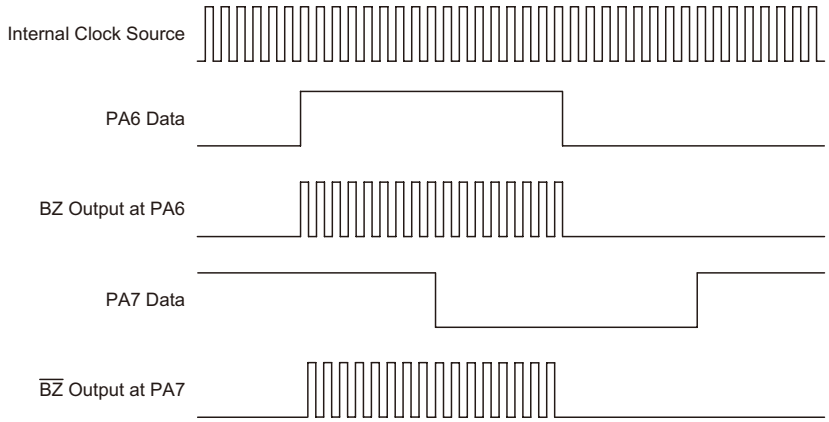
Bit 1~0 **BZ2~BZ0**: Buzzer output frequency selection
- 000: $f_{TB}/2^2$
- 001: $f_{TB}/2^3$
- 010: $f_{TB}/2^4$
- 011: $f_{TB}/2^5$
- 100: $f_{TB}/2^6$
- 101: $f_{TB}/2^7$
- 110: $f_{TB}/2^8$
- 111: $f_{TB}/2^9$

PAC Register PAC6	PAC Register PAC7	PA Data Register PA6	PA Data Register PA7	Output Function
0	0	1	×	PA6=BZ PA7=$\overline{BZ}$
0	0	0	×	PA6= "0" PA7= "0"
0	1	1	×	PA6=BZ PA7=input line
0	1	0	×	PA6= "0" PA7=input line
1	0	×	D	PA6=input line PA7=D
1	1	×	×	PA6=input line PA7=input line

"×" stands for don't care
"D" stands for Data "0" or "1"

If the options have selected that only the PA6 pin is to function as a BZ buzzer pin, then the PA7 pin can be used as a normal I/O pin. For the PA6 pin to function as a BZ buzzer pin, PA6 must be setup as an output by setting bit PAC6 of the PAC port control register to zero. The PA6 data bit in the PA data register must also be set high to enable the buzzer output, if set low pin PA6 will remain low. In this way the PA6 bit can be used as an on/off control for the BZ buzzer pin PA6. If the PAC6 bit of the PAC port control register is set high, then pin PA6 can still be used as an input even though the option has configured it as a BZ buzzer output.

Note that no matter what BPCTL option is chosen for the buzzer, if the port control register has setup the pin to function as an input, then this will override the BPCTL option selection and force the pin to always behave as an input pin. This arrangement enables the pin to be used as both a buzzer pin and as an input pin, so regardless of the BPCTL option chosen; the actual function of the pin can be changed dynamically by the application program by programming the appropriate port control register bit.



Buzzer Output Pin Control

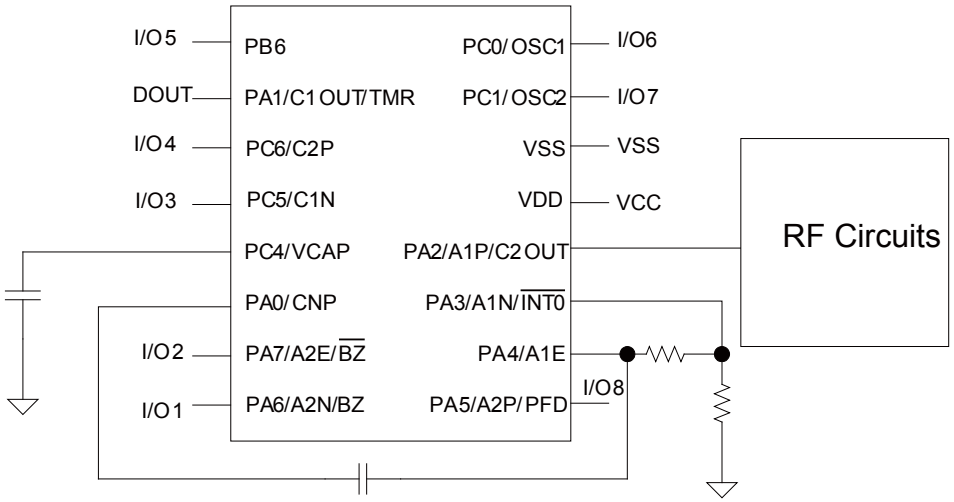
Note: The above drawing shows the situation where both pins PA6 and PA7 are selected by BPCTL option to be BZ and $\overline{BZ}$ buzzer pin outputs. The Port Control Register of both pins must have already been setup as output. The data setup on pin PA7 has no effect on the buzzer outputs.

Configuration Options

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

No.	Options
Oscillator Options	
1	OSC type selection: ERC or crystal or HIRC or EC (external clock) 00: HXT (Filter On) 01: ERC (Filter On) 10: HIRC (Filter Off) 11: EC (Filter Off)
2	HIRC frequency selection: 4MHz, 910kHz, 2MHz, 8MHz
3	HXT mode selection: 455kHz or 1M~12MHz
Watchdog Options	
4	Watchdog Timer Function: Always Enable By S/W Control
RC Filter	
5	RC filter for TMR & $\overline{\text{INT0}}$/$\overline{\text{INT1}}$, enable or disable
Lock Options	
6	Lock All
7	Partial Lock

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0 ~ 7 number of bits

addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None

Mnemonic	Description	Cycles	Flag Affected
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRD [m]	Read table 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
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

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

CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	[m] ← $\overline{[m]}$
Affected flag(s)	Z

CPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − C

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

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

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

Operation [m] ← ACC − [m] − C

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m] = 0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

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

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

Operation Skip if [m].i = 0

Affected flag(s) None

TABRD [m] Read table to TBLH and Data Memory

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

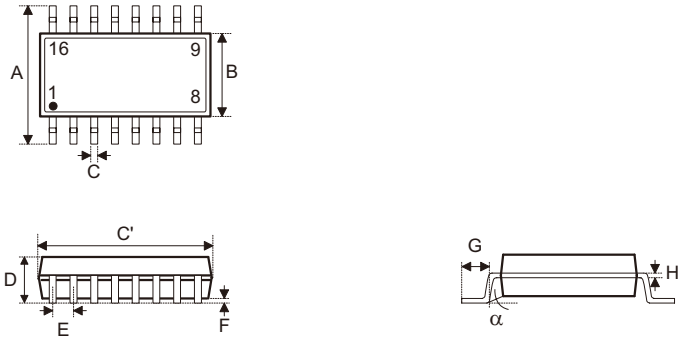
Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website (http://www.holtek.com.tw/english/literature/package.pdf) for the latest version of the package information.

16-pin NSOP (150mil) Outline Dimensions

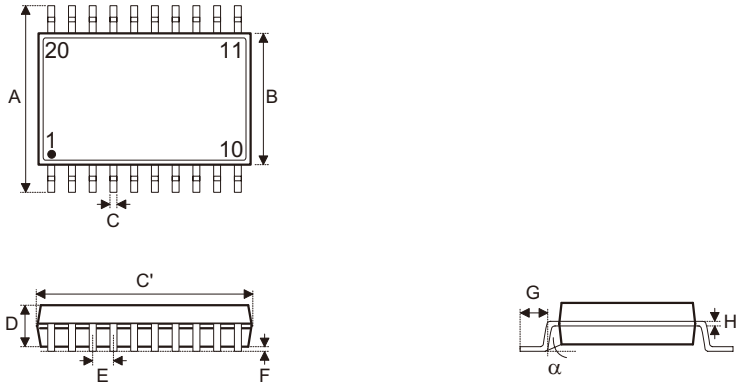


MS-012

Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.157
C	0.012	—	0.020
C'	0.386	—	0.402
D	—	—	0.069
E	—	0.050	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.007	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	3.99
C	0.30	—	0.51
C'	9.80	—	10.21
D	—	—	1.75
E	—	1.27	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.18	—	0.25
α	0°	—	8°

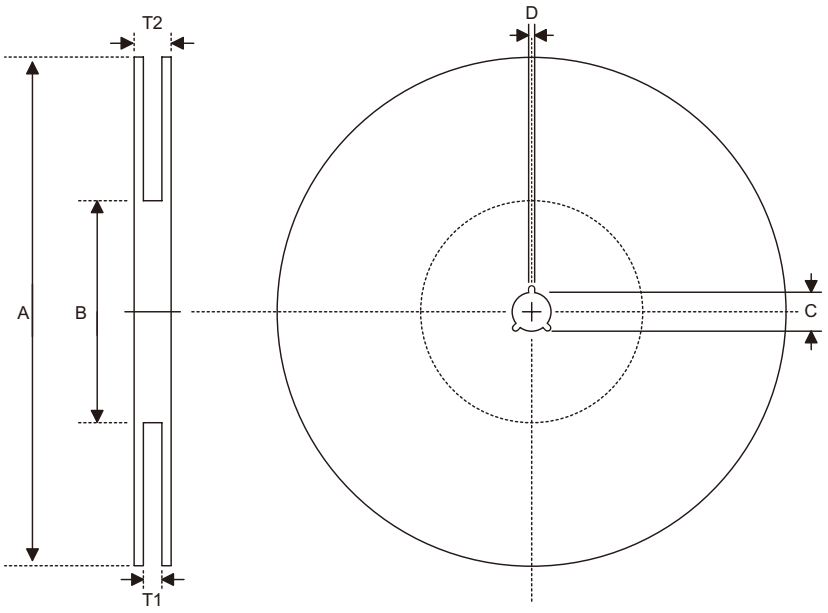
20-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.158
C	0.008	—	0.012
C'	0.335	—	0.347
D	0.049	—	0.065
E	—	0.025	—
F	0.004	—	0.010
G	0.015	—	0.050
H	0.007	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	4.01
C	0.20	—	0.30
C'	8.51	—	8.81
D	1.24	—	1.65
E	—	0.64	—
F	0.10	—	0.25
G	0.38	—	1.27
H	0.18	—	0.25
α	0°	—	8°

Reel Dimensions



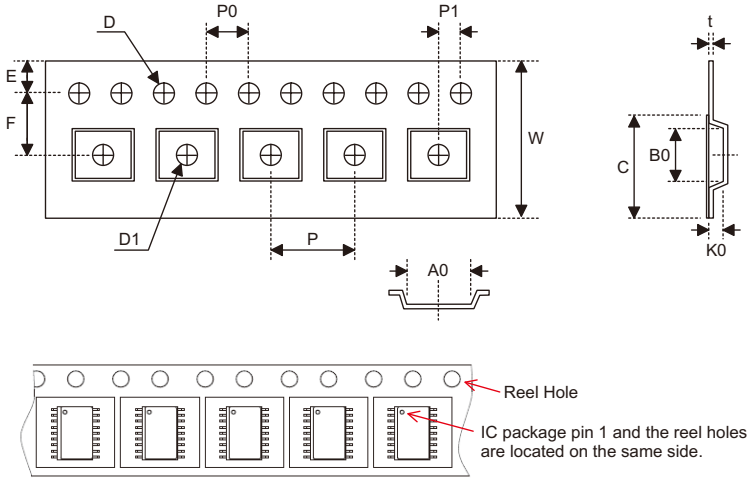
16-pin NSOP (150mil)

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

20-pin SSOP (150mil)

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

Carrier Tape Dimensions



16-pin NSOP (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0±0.3
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation(Width Direction)	7.5±0.1
D	Perforation Diameter	1.55 +0.10/-0.00
D1	Cavity Hole Diameter	1.50 +0.25/-0.00
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation(Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	10.3±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

20-pin SSOP (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0 +0.3/-0.1
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation(Width Direction)	7.50±0.10
D	Perforation Diameter	1.50 +0.10/-0.00
D1	Cavity Hole Diameter	1.50 +0.25/-0.00
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation(Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	9.0±0.1
K0	Cavity Depth	2.3±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

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

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

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5F, Unit A, Productivity Building, No.5 Gaoxin M 2nd Road, Nanshan District, Shenzhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

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