Brushless DC Motor Flash Type 8-Bit MCU

HT45FM2C

Revision: V1.20 Date: November 06, 2012

www.holtek.com

Table of Contents

Features

CPU Features

- Operating Voltage:
  - f_{SYS}=32kHz ~ 20MHz: 4.5V~5.5V
- Up to 0.2μs instruction cycle with 20MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Two oscillators:
  - Internal 20MHz RC - HIRC
  - Internal 32kHz RC - LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- Up to 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

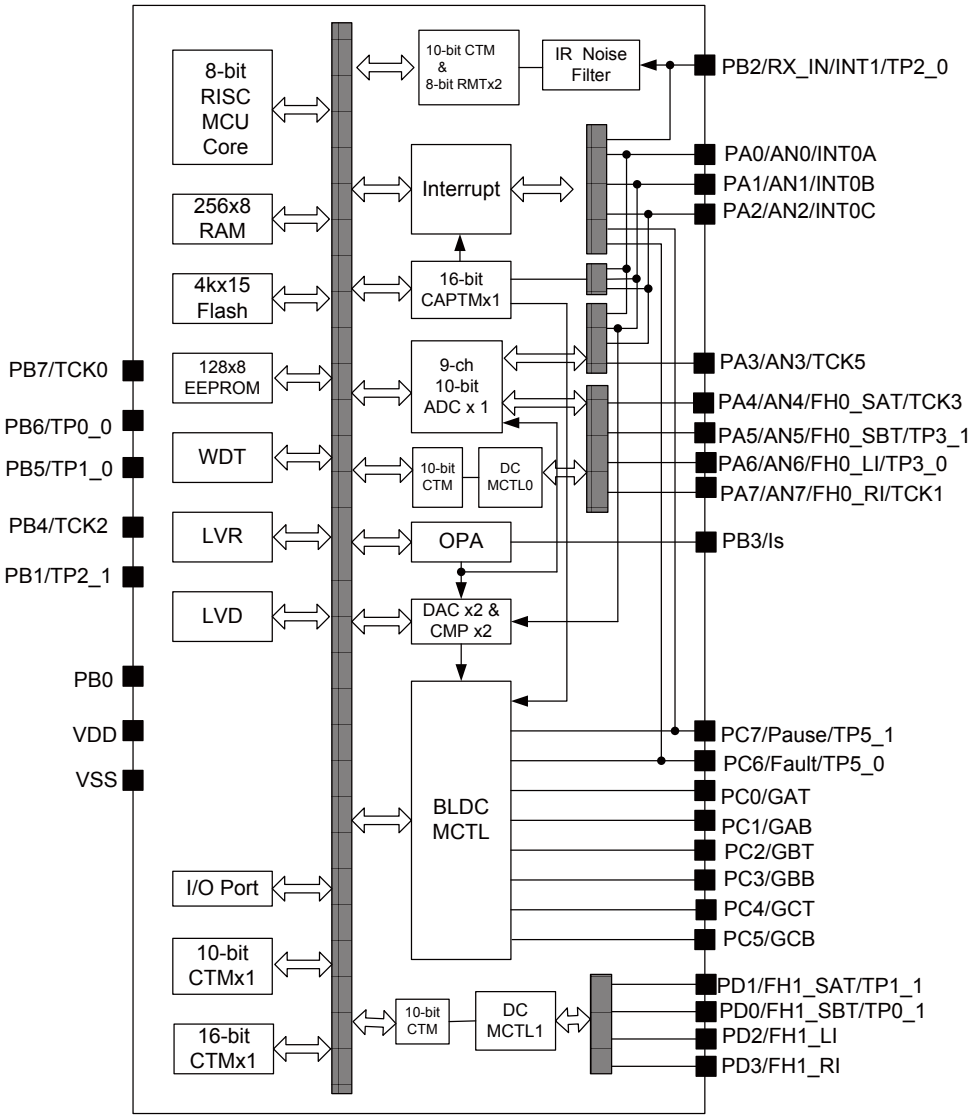
- Flash Program Memory: 4K×15
- RAM Data Memory: 256×8
- EEPROM Memory: 128×8
- Watchdog Timer function
- Up to 28 bidirectional I/O lines
- Six pin-shared external interrupts
- Support IR cord Noise Filter function
- Four 10-bit CTMs for Buzzer, RMT, up/down or left/right fan-head
- Single 16-bit CTM for BLDC sensorless application
- Single 16-bit CAPTM for motor protect
- Two 8-bit RMTs for IR decode
- A pair of 10-bit PWM with comlementary outputs for BLDC application
- 9-channel 10-bit resolution A/D converter
- Time-Base function for generation of fixed time interrupt signal
- Single operational Amplifier for current detect
- Two comparators with interrupt functions
- Dual 8-bit D/A Converter
- Low voltage reset function
- Low voltage detect function
- Package types: 16-pin NSOP, 28-pin SOP/SSOP/SKDIP, 44-pin LQFP

General Description

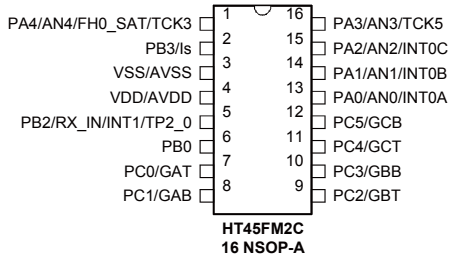
This device is Flash Memory with 8-bit high performance RISC architecture microcontroller device which includes a host of fully integrated special features specifically designed for the brushless DC motor applications.

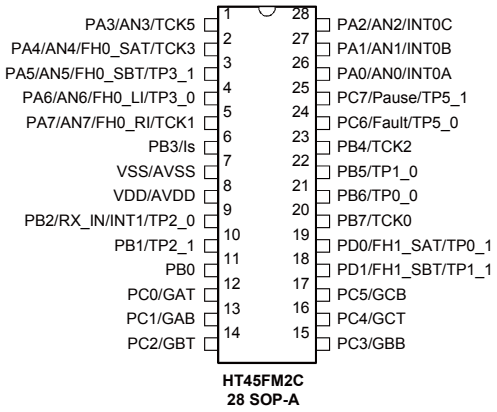
The advantages of low power consumption, I/O flexibility, Multiple and extremely flexible Timer Modules, oscillator options, multi-channel A/D and D/A Converter, Pulse Width Modulation function, 16-bit Capture Timer Module function, dual comparator functions, Motor Protect Module, Liner Hall Sensor detection, 8-bit RMT Module, Time Base function, LVD, EEPROM, power-down and wake-up functions, although especially designed for brushless DC motor applications, the enhanced versatility of this device also makes it applicable for using in a wide range of A/D application possibilities such as sensor signal processing, motor driving, industrial control, consumer products, subsystem controllers, etc.

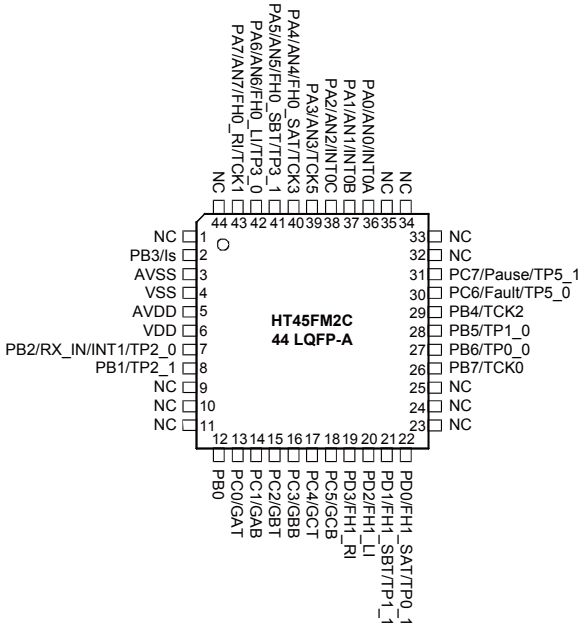
Block Diagram

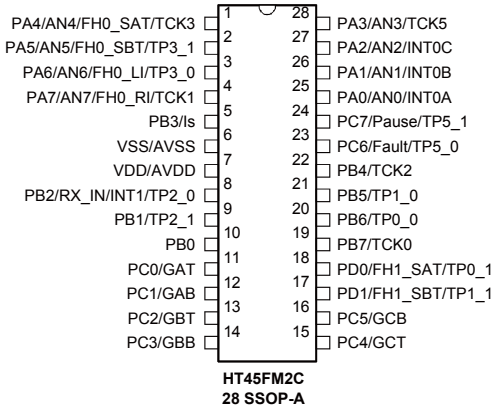


Pin Assignment









Note: 1. If the pin-shared pin functions have multiple outputs simultaneously, its pin names at the right side of the "/" sign can be used for higher priority

2. VDD&AVDD means the VDD and AVDD are the double bonding.

3. VSS&AVSS means the VSS and AVSS are the double bonding.

Pin Description

Pin Name	Function	OP	I/T	O/T	Description
PA0/AN0/ INT0A	PA0	PAWU PAPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up and wake-up.
	AN0	ANCSR0	AN	—	A/D channel 0
	INT0A	INTC0	ST	—	External interrupt input
PA1/AN1/ INT0B	PA1	PAPU PAWU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up and wake-up.
	AN1	ANCSR0	AN	—	A/D channel 1
	INT0B	INTC0	ST	—	External interrupt input
PA2/AN2/ INT0C	PA2	PAPU PAWU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up and wake-up.
	AN2	ANCSR0	AN	—	A/D channel 2
	INT0C	INTC0	ST	—	External interrupt input
PA3/AN3/ TCK5	PA3	PAPU PAWU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up and wake-up.
	AN3	ANCSR0	AN	—	A/D channel 3
	TCK5	—	ST	—	TM5 input
PA4/AN4/ FH0_SAT /TCK3	PA4	PAPU PAWU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up and wake-up.
	AN4	ANCSR0	AN	—	A/D channel 4
	FH0_SAT	DCMCR1	—	—	DC FAN Head port output
	TCK3	—	ST	—	TM3 input
PA5/AN5/ FH0_SBT/ TP3_1	PA5	PAPU PAWU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up and wake-up.
	AN5	ANCSR0	AN	—	A/D channel 5
	FH0_SBT	DCMCR1	—	—	DC FAN Head port output
	TP3_1	TMPC0	ST	CMOS	TM3 I/O
PA6/AN6/ FH0_LI/ TP3_0	PA6	PAPU PAWU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up and wake-up.
	AN6	ANCSR0	AN	—	A/D channel 6
	FH0_LI	DCMCR1	—	—	DC FAN Head port output
	TP3_0	TMPC0	ST	CMOS	TM3 I/O
PA7/AN7/ FH0_RI /TCK1	PA7	PAPU PAWU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up and wake-up.
	AN7	ANCSR0	AN	—	A/D channel 7
	FH0_RI	DCMCR1	—	—	DC FAN Head port output
	TCK1	—	ST	CMOS	TM1 input
PB0	PB0	PBPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
PB1/TP2_1	PB1	PBPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	TP2_1	TMPC0	—	CMOS	TM2 I/O

Pin Name	Function	OP	I/T	O/T	Description
PB2/RX_IN/ INT1/TP2_0	PB2	PBPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	RX_IN	INTC0	ST	—	IR Receive input pin
	INT1	INTC0	ST	—	External interrupt input
	TP2_0	TMPC0	ST	CMOS	TM2 I/O
PB3/Is	PB3	PBPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	Is	OPOMS	ST	—	Operational amplifier input pin
PB4/TCK2	PB4	PBPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	TCK2	—	ST	—	TM2 input
PB5/TP1_0	PB5	PBPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	TP1_0	TMPC0	ST	CMOS	TM1 I/O
PB6/TP0_0	PB6	PBPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	TP0_0	TMPC0	ST	CMOS	TM0 I/O
PB7/TCK0	PB7	PBPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	TCK0	—	ST	—	TM0 input
PC0/GAT	PC0	PCPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	GAT	PWMC	—	CMOS	Pulse Width Modulation complimentary output
PC1/GAB	PC1	PCPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	GAB	PWMC	—	CMOS	Pulse Width Modulation complimentary output
PC2/GBT	PC2	PCPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	GBT	PWMC	—	CMOS	Pulse Width Modulation complimentary output
PC3/GBB	PC3	PCPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	GBB	PWMC	—	CMOS	Pulse Width Modulation complimentary output
PC4/GCT	PC4	PCPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	GCT	PWMC	—	CMOS	Pulse Width Modulation complimentary output
PC5/GCB	PC5	PCPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	GCB	PWMC	—	CMOS	Pulse Width Modulation complimentary output
PC6/Fault/ TP5_0	PC6	PCPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	Fault	MPTC1	ST	—	PWM Disable input pin. Active Low
	TP5_0	TMPC1	ST	CMOS	CAPTM I/O

Pin Name	Function	OP	I/T	O/T	Description
PC7/Pause/ TP5_1	PC7	PCPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	Pause	MPTC1	ST	—	PWM Pause input pin
	TP5_1	TMPC1	ST	CMOS	CAPTM I/O
PD0/ FH1_SAT/ TP0_1	PD0	PDPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	FH1_SAT	DCMCR1	—	—	DC FAN Head port output
	TP0_1	TMPC0	ST	CMOS	TM0 I/O
PD1/ FH1_SBT/ TP1_1	PD1	PDPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	FH1_SBT	DCMCR1	—	—	DC FAN Head port output
	TP1_1	TMPC0	ST	CMOS	TM1 I/O
PD2/FH1_LI	PD2	PDPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	FH1_LI	DCMCR1	—	—	DC FAN Head port output
PD3/FH1_RI	PD3	PDPU	ST	CMOS	Bidirectional 8-bit I/O port. Register enabled pull-up.
	FH1_RI	DCMCR1	—	—	DC FAN Head port output
VSS	VSS	—	PWR	—	Negative power supply, ground
AVSS	AVSS	—	PWR	—	Ground connection for A/D converter. The VSS and AVSS are the same pin at 28 pin package
VDD	VDD	—	PWR	—	Positive power supply
AVDD	AVDD	—	PWR	—	Power supply connection for A/D converter. The VDD and AVDD are the same pin at 28 pin package

Note: I/T: Input type; O/T: Output type

OP: Optional by configuration option (CO) or register option

PWR: Power; CO: Configuration option; ST: Schmitt Trigger input

CMOS: CMOS output; NMOS: NMOS output

AN: Analog input pin

VDD is the device power supply while AVDD is the ADC power supply.

VSS is the device ground pin while AVSS is the ADC ground pin.

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

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to this 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 devices reliability.

D.C. Characteristics

Ta=25˚C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=32 ~ 20000kHz	4.5	—	5.5	V
I_{DD}	Operating Current (HIRC OSC)	5V	No load, f_H=20MHz, ADC off, WDT enable, Motor_CTL off, IR_RX off	—	8	10	mA
I_{STB}	Standby Current	—	LIRC and LVR on, LVD off, WDT enable	—	60	100	μA
V_{IL}	Input Low Voltage for I/O Ports, TCKn, INT0A, INT0B, INT0C, INT1	—	—	0	—	0.3V_{DD}	V
V_{IH}	Input High Voltage for I/O Ports, TCKn, INT0A, INT0B, INT0C, INT1	—	—	0.7V_{DD}	—	V_{DD}	V
V_{LVR}	LVR Voltage Level	—	LVR Enable, 3.15V option	-5%	3.15	+5%	V
V_{LVD}	LVD Voltage Level	—	LVDEN=1, V_{LVD}=3.6V	-5%	3.6	+5%	V
V_{OL}	Output Low Voltage for I/O Ports	5V	I_{OL}=20mA	—	—	0.5	V
V_{OH}	Output High Voltage for I/O Ports	5V	I_{OH}=-7.4mA	4.5	—	—	V
R_{PH}	Pull-high Resistance for I/O Ports	5V	—	10	30	50	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock	—	4.5V~5.5V	32	—	20000	kHz
f_{HIRC}	System Clock (HIRC)	4.5V~5.5V	Ta=-40°C~85°C	-12%	20	+4%	MHz
			Ta=-20°C~85°C	-9%	20	+4%	MHz
			Ta=25°C	-2%	20	+2%	MHz
f_{TIMER}	Timer Input Pin Frequency	—	—	—	—	1	f_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	t_{SYS}
t_{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Low Voltage Width to Interrupt	—	—	20	45	90	μs
t_{LVDS}	LVDO stable time	—	—	15	—	—	μs
t_{EERD}	EEPROM Read Time	—	—	—	45	90	μs
t_{EEWR}	EEPROM Write Time	—	—	—	2	4	ms
t_{SST}	System Start-up Timer Period (Wake-up from HALT)	—	f_{SYS}=HIRC	—	15~16	—	t_{SYS}
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

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

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

A/D Converter Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
AV_{DD}	A/D Converter Operating Voltage	—	—	V_{LVR}	5.0	5.5	V
I_{OP}	A/D Converter Operating Current	3V	—	—	0.8	—	mA
		5V	—	—	1	—	mA
I_{STBY}	ADC Standby Current	—	digital input no change	—	—	1	μA
V_{REF}	A/D Converter Reference Voltage	—	—	2	AV_{DD}	AV_{DD}+0.1	V
T_{conv}	A/D Conversion Time	—	—		14		T_{adck}
DNL	A/D Differential non-linearity	—	—	—	—	±2	LSB
INL	A/D Integral non-linearity	—	—	—	±2	—	LSB
G_{err}	Gain Error	—	—	—	—	±2	LSB
T_{adck}	ADCLK period	—	—	—	0.166	—	μs
T_{ckh}	ADCLK high width	—	—	—	83	—	ns
T_{ckl}	ADCLK low width	—	—	—	83	—	ns
T_{st1}	Setup time for ADON	—	—	2	—	—	ns
T_{st2}	Setup time for START latch	—	—	2	—	—	ns
T_{sth}	START high width	—	—	25	—	—	ns
T_{deoc}	EOCB output delay	—	AV_{DD}=5V	—	3	—	ns
T_{dout}	Output delay	—	AV_{DD}=5V	—	3	—	ns
T_{on}	ADC wake up time	—	—	2	—	—	μs
T_{off}	ADC sleep time	—	—	—	—	5	ns

D/A Converter Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	D/A Operating Current	—	—	V_{LVR}	—	5.5	V
V_{DA}	D/A Output Voltage	—	00h ~ FFh, no load	0.01	—	0.99	V_{DD}
t_{DAC}	D/A Conversion Time	—	V_{DD}=5V, C_L=10P	—	—	2	μs
R_O	D/A Output Resistance	—	—	—	10	—	kΩ

8-bit R-2R D/A Converter(Analog Conditon V_{DD}=5V, C_L=10P)

Model Corner	TT	SF	FS	SS	FF
Temperature	25	25	25	90	-40
Operating Average Current (V_{DD}=5V, C_L=10P)	352μA	330μA	374μA	297μA	413μA
Analog Output 00000000 (B) ~11111111 (B)	0~4.98V	0~4.981V	0~4.98V	0~4.98V	0~4.981V
Conversion Time	≤2μs	≤2μs	≤2μs	≤2μs	≤2μs

Operational Amplifier Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OPR1}	Operating Current	5V	—	—	250	—	μA
I_{OFF1}	Power Down Current	5V	—	—	—	0.1	μA
V_{OPOS1}	Input Offset Voltage	5V	Without calibration, OPOF[3:0]=1000B	-15	—	+15	mV
V_{OPOS2}	Input Offset Voltage	5V	By calibration	-4	—	+4	mV
V_{CM}	Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}-1.4V	V
PSRR	Power Supply Rejection Ratio	5V	—	60	80	—	dB
CMRR	Common Mode Rejuction Ratio	5V	V_{CM}=0 ~ V_{DD}-1.4V	60	80	—	dB
SR	Slew Rate+, Slew Rate-	5V	No load	1.8	2.5	—	V/μs
GBW	Gain Band Width	5V	R_L=1MΩ, C_L=100pF	—	2.5	—	MHz

Comparator Electrical Characteristics

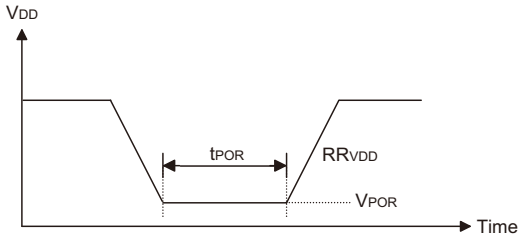
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
I_{OPR0}	Comparator operating voltage	5V	—	—	200	300	μA
I_{OFF0}	Comparator power down current	5V	—	—	—	0.1	μA
V_{OS}	Comparator input offset voltage	—	—	-10	—	+10	mV
V_{CM}	Comparator common mode input voltage range	—	—	V_{SS}	—	V_{DD}-1.4V	V
t_{PD}	Comparator response time (100mV overdrive)	—	—	—	4	8	μs

Power on Reset Electrical Characteristics

Ta=25°C

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



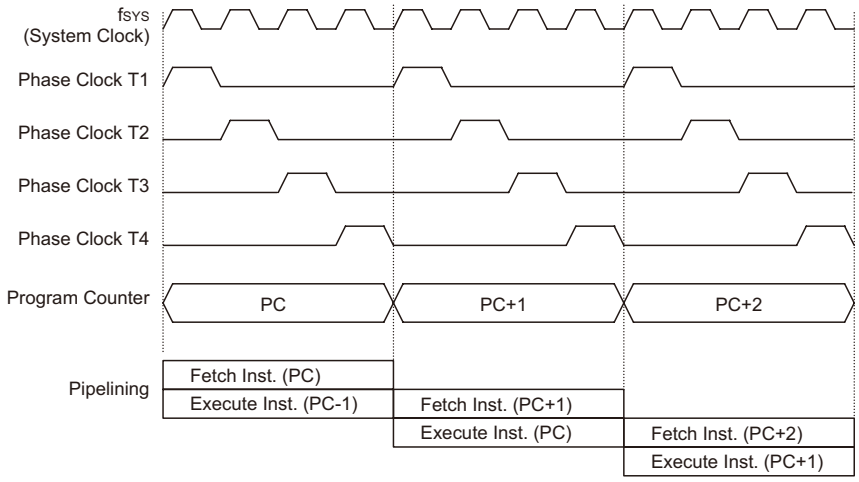
System Architecture

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

Clocking and Pipelining

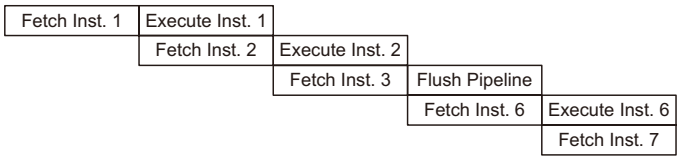
The main system clock, derived from either a HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the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

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



System Clocking and Pipelining

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



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demand a jump to a non-consecutive Program Memory address. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

Program Counter	
Program Counter High Byte	**PCL Register**
PC11~PC8	PCL7~PCL0

Program Counter

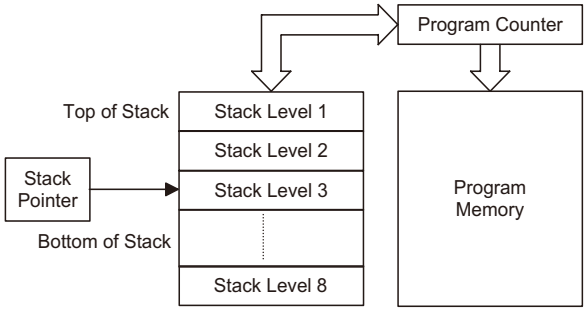
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack has eight levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

If the stack is full and an enabled interrupt takes place, the interrupt request flag will be recorded but the acknowledge signal will be inhibited. When the Stack Pointer is decremented, by RET or RETI, the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. However, when the stack is full, a CALL subroutine instruction can still be executed which will result in a stack overflow. Precautions should be taken to avoid such cases which might cause unpredictable program branching.

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



Arithmetic and Logic Unit – ALU

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

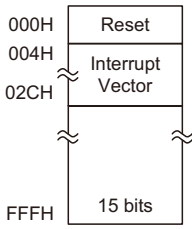
The Program Memory is the location where the user code or program is stored. For this device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate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

Structure

The Program Memory has a capacity of 4K×15 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.

Special Vectors

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



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.

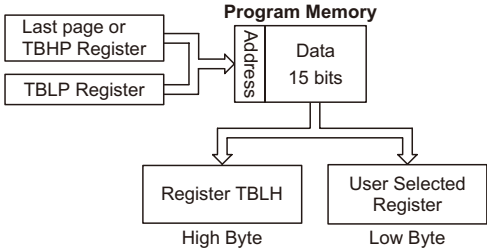


Table Program Example

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 "F00H" which refers to the start address of the last page within the 4K 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 "F06H" 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 "TABRDC [m]" instruction is executed.

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

Table Read Program Example

```
tempreg1 db    ?          ; temporary register #1
tempreg2 db    ?          ; temporary register #2
:
:
mov a,06h                 ; initialise low table pointer - note that this address
                          ; is referenced
mov tblp, a               ; to the last page or present page
mov a, 07h                ; initialise high table pointer
mov tbhp, a
:
:
tabrdl tempreg1           ; transfers value in table referenced by table pointer
                          ; data at program memory address "F06H" transferred to
                          ; tempreg1 and TBLH
dec tblp                  ; reduce value of table pointer by one
tabrdl tempreg2           ; transfers value in table referenced by table pointer
                          ; data at program memory address "F05H" transferred to
                          ; tempreg2 and TBLH in this example the data "1AH" is
                          ; transferred to tempreg1 and data "0FH" to register tempreg2
:
:
org F00h                  ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

In Circuit Programming

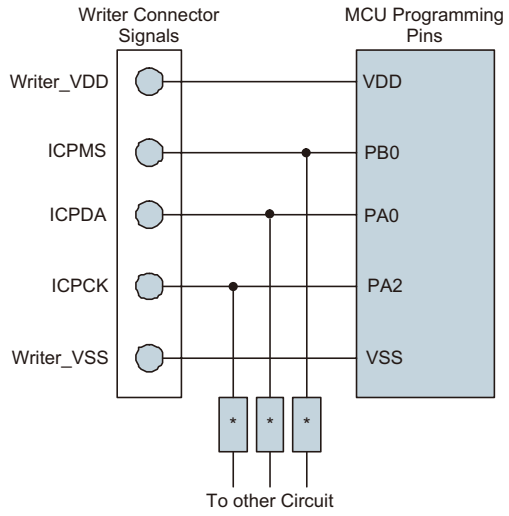
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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 sup plied with the latest program releases without removal and re-insertion of the device.

MCU Programming Pins	Function
PA0	Serial Data Input/Output
PA2	Serial Clock
PB0	In Circuit Programming Mode Set
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 in circuit programming Mode set. The technical details regarding the incircuit programming of the device is beyond the scope of this document and will be supplied in supple mentary literature.

During the programming process the PB0 pin will be used to set the in circuit programing Mode and taking control of the PA0 and PA2 I/O pins for data and clock programming purposes. The user must there take care to ensure that no other outputs are connected to these two pins.



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

Programmer Pin	MCU Pins
ICPMS	PB0
ICPDA	PA0
ICPCK	PA2

Programmer and MCU Pins

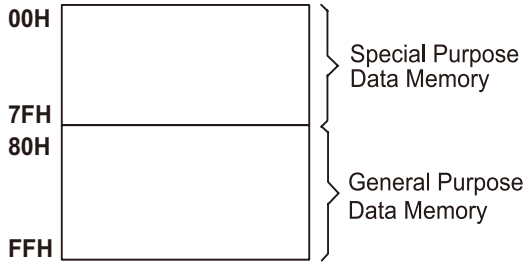
RAM Data Memory

The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored. The capacity of this device is 256×8.

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



Data Memory Structure

	Bank 0, 1
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	SMOD
0CH	LVDC
0DH	LVRC
0EH	WDTC
0FH	TBC
10H	INTC0
11H	INTC1
12H	INTC2
13H	INTC3
14H	MFI0
15H	MFI1
16H	MFI2
17H	MFI3
18H	PAWU
19H	PAPU
1AH	PA
1BH	PAC
1CH	PBPU
1DH	PB
1EH	PBC
1FH	PCPU
20H	PC
21H	PCC
22H	PDPU
23H	PD
24H	PDC
25H	MFI4
26H	MFI5
27H	MFI6
28H	MFI7
29H	MFI8
2AH	NF_VIH
2BH	NF_VIL
2CH	RMTC
2DH	RMT0
2EH	RMT1
2FH	HCHK_NUM
30H	HNF_MSEL
31H	CAPTC0
32H	CAPTC1
33H	CAPTMDL
34H	CAPTMDH
35H	CAPTMAL
36H	CAPTMAH
37H	CAPTMCL
38H	CAPTMCH
39H	OPOMS
3AH	OPCM
3BH	LHMC
3CH	HACM
3DH	TMPC0
3EH	TMPC1
3FH	CTRL

	Bank 0	Bank 1
40H	Unused	EEC
41H	EEA	
42H	EED	
43H	ADRL	
44H	ADRH	
45H	ADCR0	
46H	ADCR1	
47H	ANCSR0	
48H	ANCSR1	
49H	ADDL	
4AH	ADLVDL	
4BH	ADLVDH	
4CH	ADHVDL	
4DH	ADHVDH	
4EH	PWMC	
4FH	DUTRL	
50H	DUTRH	
51H	PRDRL	
52H	PRDRH	
53H	PWMRL	
54H	PWMRH	
55H	MCF	
56H	MCD	
57H	DTS	
58H	PLC	
59H	HDCR	
5AH	HDCD	
5BH	HDCT0	
5CH	HDCT1	
5DH	HDCT2	
5EH	HDCT3	
5FH	HDCT4	
60H	HDCT5	
61H	HDCT6	
62H	HDCT7	
63H	HDCT8	
64H	HDCT9	
65H	HDCT10	
66H	HDCT11	
67H	MPTC1	
68H	MPTC2	
69H	TM5C0	
6AH	TM5C1	
6BH	TM5DL	
6CH	TM5DH	
6DH	TM5AL	
6EH	TM5AH	
6FH	TM5RP	
70H	OPACAL	
71H	DCMCR0	
72H	DCMCR1	
73H	TM0C0	TM3C0
74H	TM0C1	TM3C1
75H	TM0DL	TM3DL
76H	TM0DH	TM3DH
77H	TM0AL	TM3AL
78H	TM0AH	TM3AH
79H	Unused	
7AH	TM1C0	TM2C0
7BH	TM1C1	TM2C1
7CH	TM1DL	TM2DL
7DH	TM1DH	TM2DH
7EH	TM1AL	TM2AL
7FH	TM1AH	TM2AH

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 Bank 0 while the IAR1 and MP1 register pair can access data from any bank. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of "00H" and writing to the registers indirectly will result in no operation.

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
org  00h
start:
     mov a,04h              ; setup size of block
     mov block,a
     mov a,offset adres1    ; Accumulator loaded with first RAM address
     mov mp0,a              ; setup memory pointer with first RAM address
loop:
     clr IAR0               ; clear the data at address defined by MP0
     inc mp0                ; increment memory pointer
     sdz block              ; check if last memory location has been cleared
     jmp loop
continue:
```

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

Bank Pointer – BP

The Data Memory is divided into two banks. Selecting the Data Memory area is achieved using the Bank Pointer. Bit 0 of the Bank Pointer is used to select Data Memory Banks 0 or 1.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the Power Down Mode, in which case, the Data Memory bank remains unaffected. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from banks other than Bank 0 must be implemented using Indirect addressing.

BP Register

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **DMBP0:** 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

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	—	—	—	—	×	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 occurred
1: Occurred
This bit can be cleared to “0”, but can not be set to “1”.

Bit 1 **LRF:** LVR Control register software reset flag
0: Not occurred
1: Occurred
This bit can be cleared to “0”, but can not be set to “1”.

Bit 0 **WRF:** WDT Control register software reset flag
0: Not occurred
1: Occurred
This bit can be cleared to “0”, but can not be set to “1”.

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 128×8 bits. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and data register in Bank 0 and a single control register in Bank 1.

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same was as any other Special Function Register. The EEC register however, being located in Bank1, cannot be addressed directly and can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the 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 EEC register are executed.

EEPROM Register List

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

EEA Register

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

"x" unknown

Bit 7 Unimplemented, read as "0"

Bit 6~0 Data EEPROM address
Data EEPROM address bit 6~bit 0

EEC Register

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

Bit 7~4 Undefined, read as "0"

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

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

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

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

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

Reading Data from the EEPROM

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

Writing Data to the EEPROM

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

Write Protection

Protection against inadvertent write operation is provided in several ways. After the device is powered-on the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on the Bank Pointer, 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 EPWE 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 EPWF 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 Consideration

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.

Programming Examples

Reading data from the EEPROM – polling method

```
MOV  A, EEPROM_ADRES            ; user defined address
MOV  EEA, A
MOV  A, 040H                    ; setup memory pointer MP1
MOV  MP1, A                     ; MP1 points to EEC register
MOV  A, 01H                     ; setup Bank Pointer
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 from the EEPROM – polling method

```
MOV  A, EEPROM_ADRES            ; user defined address
MOV  EEA, A
MOV  A, EEPROM_DATA             ; user defined data
MOV  EED, A
MOV  A, 040H                    ; setup memory pointer MP1
MOV  MP1, A                     ; MP1 points to EEC register
MOV  A, 01H                     ; setup Bank Pointer
MOV  BP, A
SET  IAR1.3                     ; set WREN bit, enable write operations
SET  IAR1.2                     ; start Write Cycle - set WR 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 a combination of configuration options and registers.

Oscillator Overview

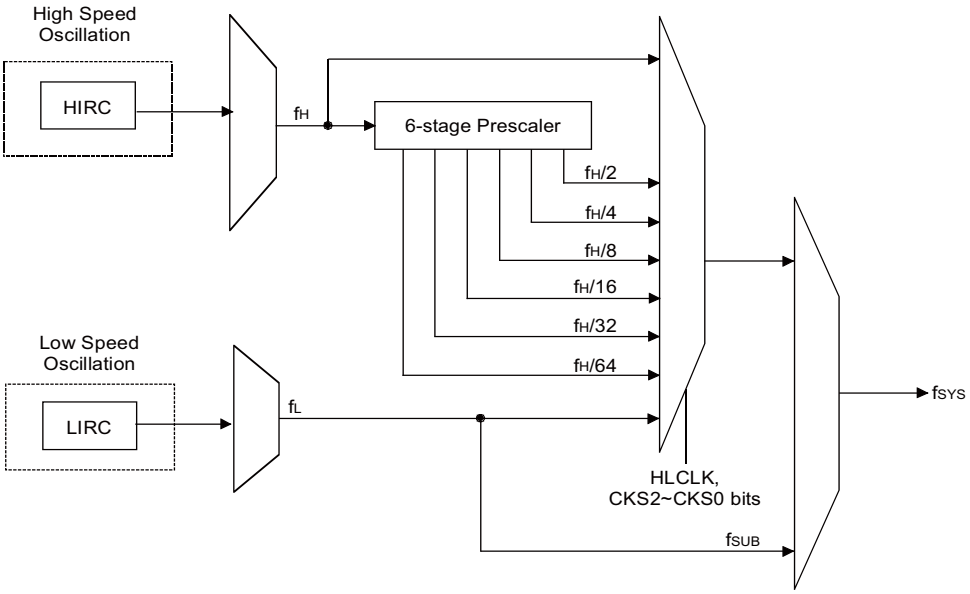
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupt. Fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The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

Type	Name	Freq.
Internal High Speed RC	HIRC	20MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two methods of generating the system clock, a high speed oscillator and a low speed oscillator. The high speed oscillator is the internal 20MHz RC oscillator. The low speed oscillator is the internal 32kHz RC oscillator. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and CKS2~CKS0 bits in the SMOD register and as the system clock can be dynamically selected.



System Clock Configurations

Internal 20MHz RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequencies of 20MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 5V and at a temperature of 25˚C degrees, the fixed oscillation frequency of 20MHz will have a tolerance within 2%.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is a low frequency oscillator choice. 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. As a result, at a power supply of 5V and at a temperature of 25˚C degrees, the fixed oscillation frequency of 32kHz will have a tolerance within 10%.

Supplementary Clocks

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

Operating Modes and System Clocks

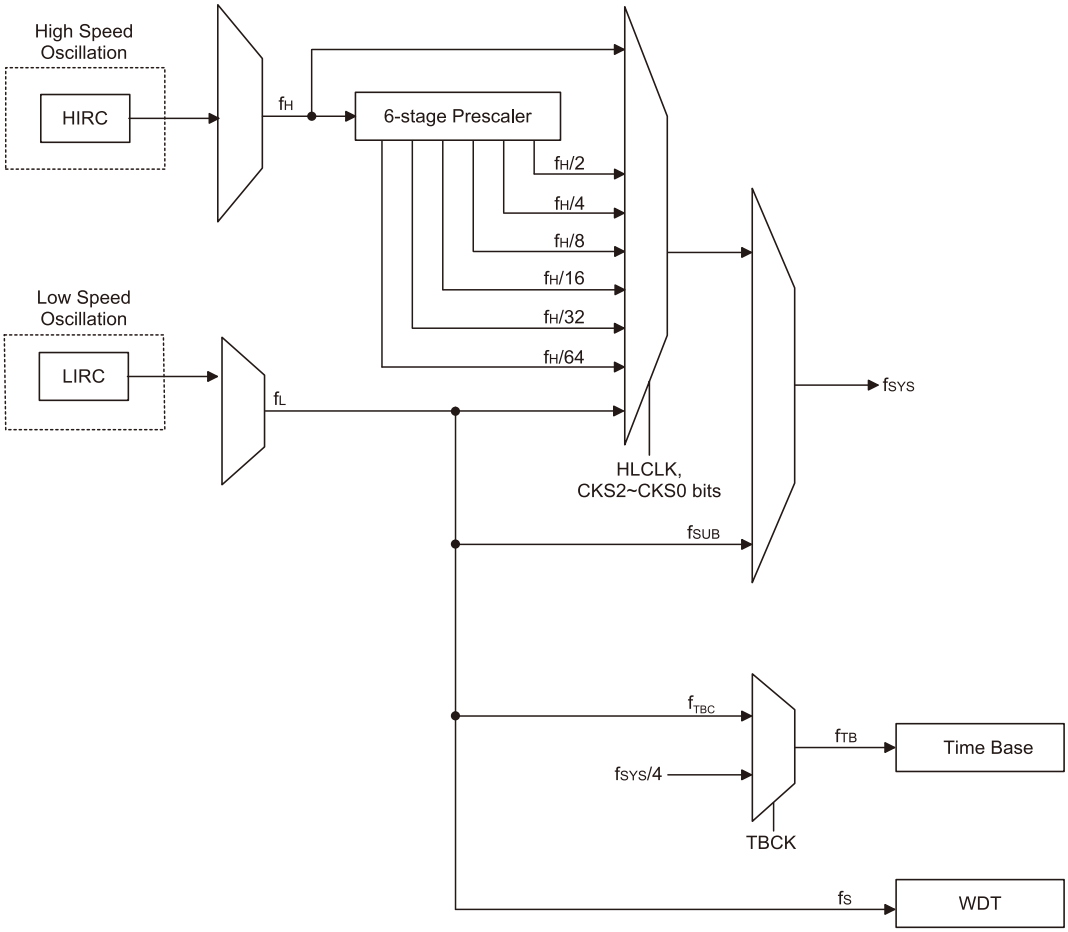
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice versa, lower speed clocks reduce current consumption. As Holtek has provided this 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 CKS2~CKS0 bits in the SMOD register. The high speed system clock can be sourced from HIRC oscillator. The low speed system clock source can be sourced from internal clock f_L. If f_L is selected then it can be sourced by the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.

There are two additional internal clocks for the peripheral circuits, the substitute clock, f_{SUB}, and the Time Base clock, f_{TBC}. Each of these internal clocks is sourced by the LIRC oscillator. The f_{SUB} clock is used to provide a substitute clock for the microcontroller just after a wake-up has occurred to enable faster wake-up times.



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.

The f_S is used as the clock source for the Watchdog timer. Together with $f_{SYS}/4$, the f_{TBC} clock is also used as a source for the Time Base interrupt function and for the TMs.

System Operation Modes

There are five 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 three modes, the SLEEP, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.

Operation Mode	Description				
	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
SLEEP Mode	Off	Off	On	On	On

- NORMAL Mode
  As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source will come from HIRC oscillator. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 and HLCLK bits in the SMOD register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.
- SLOW Mode
  This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from the low speed oscillator, 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.
- SLEEP Mode
  The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP Mode the CPU will be stopped.
- IDLE0 Mode
  The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD 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 TMs. In the IDLE0 Mode, the system oscillator will be stopped. In the IDLE0 Mode the Watchdog Timer clock, f_S, will be always on.
- IDLE1 Mode
  The IDLE1 Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD 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 TMs. 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, SMOD, is used for overall control of the internal clocks within this device.

SMOD Register

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	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 **CKS2~CKS0:** 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 the 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
0: Disable
1: Enable

This is the Fast Wake-up Control bit which determines if the f_{SUB} clock source is initially used after this device wake 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:** Low speed system oscillator ready flag
0: Not ready
1: Ready

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

Bit 2 **HTO:** High speed system oscillator ready flag
0: Not ready
1: Ready

This is the high speed system oscillator ready flag which indicates when the high speed system oscillator is stable. This flag is cleared to "0" by hardware when this device is powered on and then changes to a high level after the high speed system oscillator is stable. Therefore this flag will always be read as "1" by the application program after device power-on. The flag will be low when in the SLEEP or IDLE0 Mode but after a wake-up has occurred, the flag will change to a high level after 15~16 clock cycles if the 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 this 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 this device will enter the SLEEP Mode when a HALT instruction is executed.

Bit 0 **HLCLK:** System clock selection
0: $f_H/2$~$f_H/64$ or f_L
1: f_H
This bit is used to select if the f_H clock or the $f_H/2$~$f_H/64$ 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 $f_H/2$~$f_H/64$ 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.

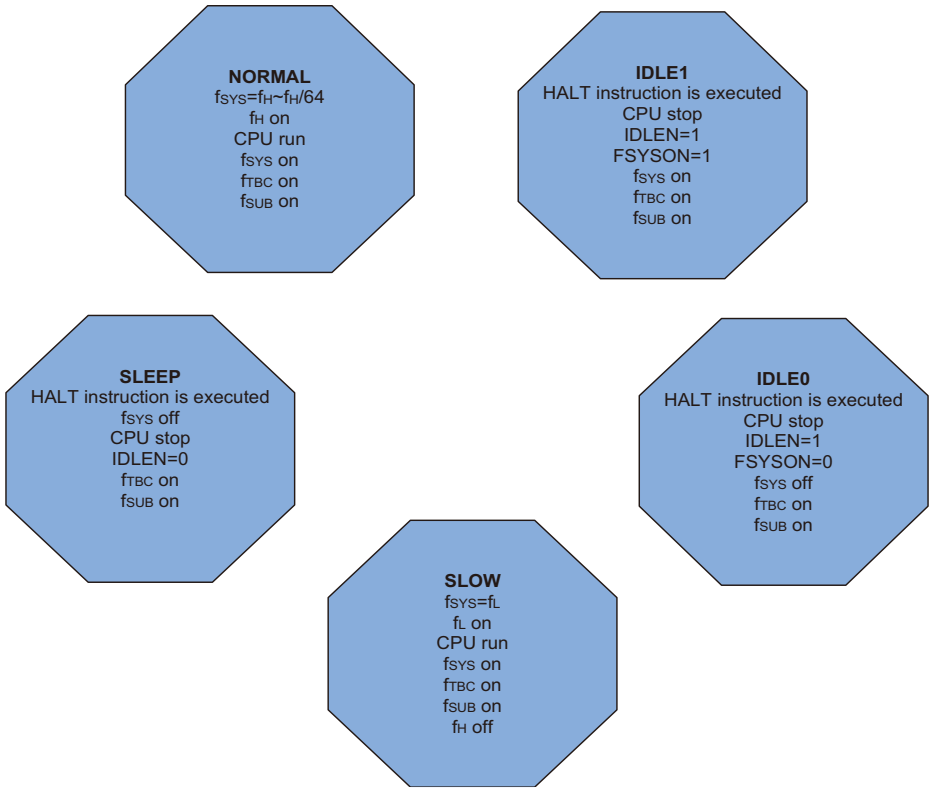
Fast Wake-up

To minimise power consumption this device can enter the SLEEP or IDLE0 Mode, where the system clock source to this device will be stopped. However when this 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 the 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 SLEEP and IDLE0 modes. The Fast Wake-up enable/disable function is controlled using the FSTEN bit in the SMOD register.

If the HIRC oscillator or LIRC oscillator is used as the system oscillator then it will take 15~16 clock cycles of the 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 (SLEEP Mode)	Wake-up Time (IDLE0 Mode)	Wake-up Time (IDLE1 Mode)
HIRC	x	15~16 HIRC cycles		1~2 HIRC cycles
LIRC	x	1~2 LIRC cycles		1~2 LIRC cycles

Wake-Up Times



Operating Mode Switching and Wake-up

This 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 CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether this device enter the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SMOD 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 such as the TMs. The accompanying flowchart shows what happens when this device move between the various operating modes.

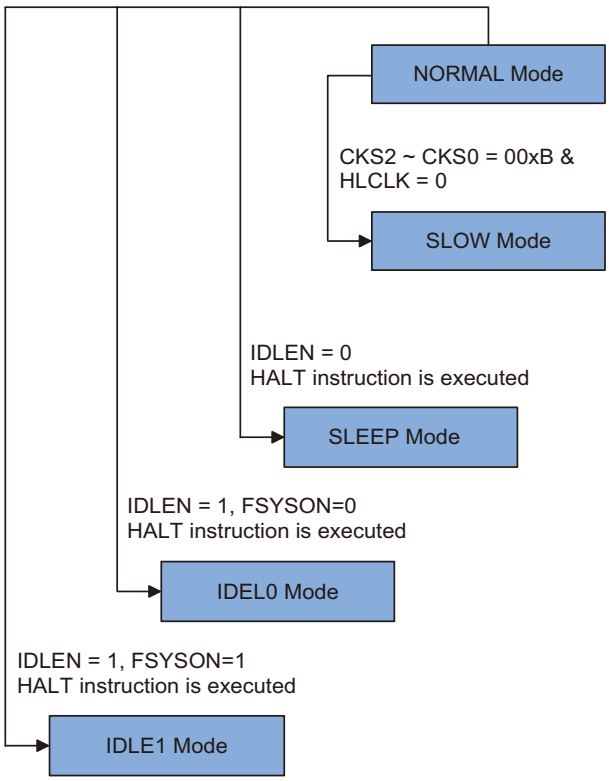
NORMAL Mode to SLOW Mode Switching

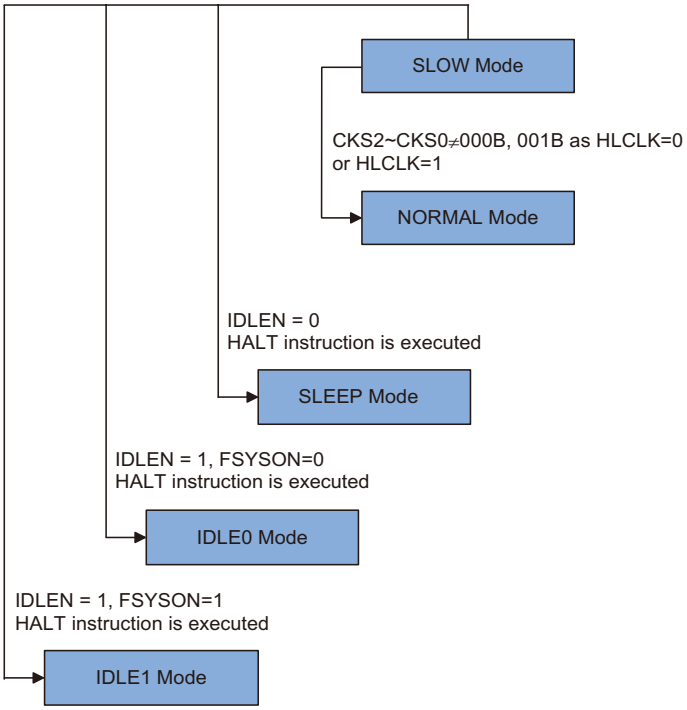
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to “0” and set the CKS2~CKS0 bits to “000” or “001” in the SMOD register. This will then use the low speed system oscillator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

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

SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system 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 CKS2~CKS0 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.


NORMAL Mode
CKS2 ~ CKS0 = 00xB &
HLCLK = 0
SLOW Mode
IDLEN = 0
HALT instruction is executed
SLEEP Mode
IDLEN = 1, FSYSON=0
HALT instruction is executed
IDEL0 Mode
IDLEN = 1, FSYSON=1
HALT instruction is executed
IDLE1 Mode


SLOW Mode
CKS2~CKS0≠000B, 001B as HLCLK=0
or HLCLK=1
NORMAL Mode
IDLEN = 0
HALT instruction is executed
SLEEP Mode
IDLEN = 1, FSYSON=0
HALT instruction is executed
IDLE0 Mode
IDLEN = 1, FSYSON=1
HALT instruction is executed
IDLE1 Mode

Entering the SLEEP Mode

There is only one way for this device to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0". 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.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting.
- 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 this device to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD 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.
- 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 this device to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD 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.
- 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 this 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 this device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to devices which have different package types, as there may be 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, this device will experience a full system reset, however, if this device are 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 this 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 can be sourced from the LIRC oscillator. 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. The LIRC internal oscillator has an approximate period of 32kHz 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.

Watchdog Timer Control Register

A single register, WDTC, controls the overall operation of the Watchdog Timer. Any reset of this device, the initial value of the WDTC is always "01010011", and it will not be changed in Power Down Mode.

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 operation
10101 or 01010: Enable
Other values: MCU reset (Reset will be active after 1~2 LIRC clock for debounce time.)
If the MCU reset caused by the WE[4:0] in WDTC software reset, the WRF flag of CTRL register will be set)

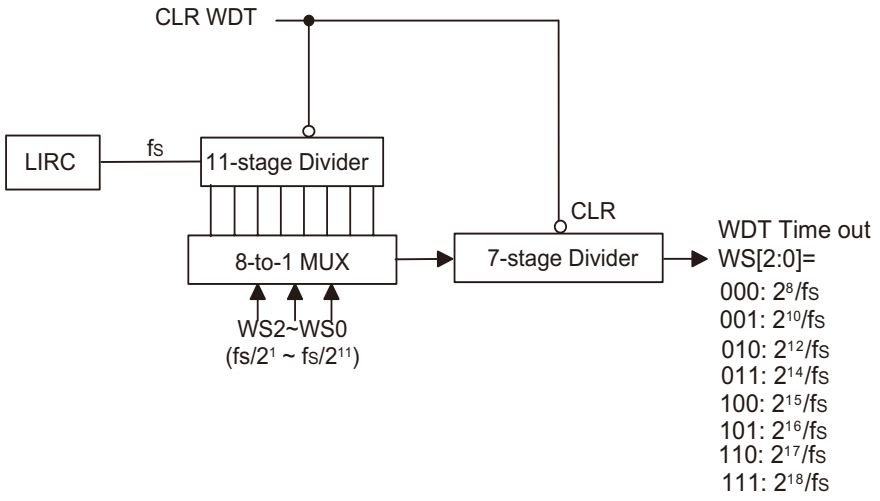
Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^8/f_S$
001: $2^{10}/f_S$
010: $2^{12}/f_S$
011: $2^{14}/f_S$
100: $2^{15}/f_S$
101: $2^{16}/f_S$
110: $2^{17}/f_S$
111: $2^{18}/f_S$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the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

Watchdog Timer Operation

Note that the Watchdog Timer function is always enabled. The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unkown 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. There are five bits, WE4~WE0, in the WDTC register to offer a control of the Watchdog Timer. If WE4~WE0 bits are set to a specific value of "10101" or "01010", the WDT is alway enable. Any other values for these bits will keep the MCU reset.

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is an external hardware reset, the second is using the Watchdog Timer software clear instruction and the third is via a HALT instruction.

To clear the Watchdog Timer is to use the single “CLR WDT” instruction. A simple execution of "CLR WDT" will clear the WDT.



Watchdog Timer

Reset and Initialisation

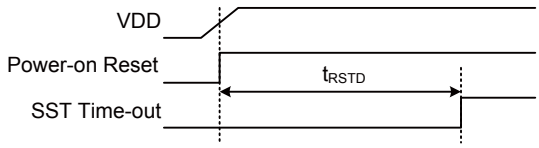
A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontroller. In this case, internal circuitry will ensure 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 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:

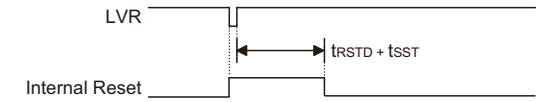
- Power-on Reset
  The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to 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
  This microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of this device, which is controlled by LVRC register. 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 set the LVRF in the CTRL register to high .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.



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

Low Voltage Reset Timing Chart

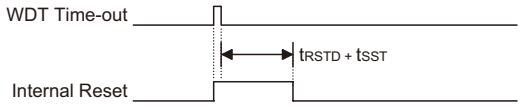
LVRC Register

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 7~3 **LVS7~LVS0**: LVR voltage select
01010101: 3.15V
00110011: 3.15V
10011001: 3.15V
10101010: 3.15V
Other values: MCU reset (Reset will be active after 2~3 LIRC clock for debounce time.)

If the MCU reset caused by the LVRC software reset, the LRF flag of CTRL register will be set.

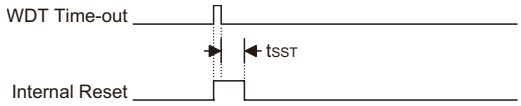
- Watchdog Time-out Reset during Normal Operation
  The Watchdog time-out flag TO will be set to "1".



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 15~16 clock cycles if the system clock source is provided by HIRC.
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

"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	TM modules will be turned off
Input/Output Ports	I/O ports will be setup as inputs, and AN0~AN7 is as A/D input pin.
Stack Pointer	Stack Pointer will point to the top of the stack

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

Register	Reset (Power On)	WDT Time-out (Normal Operation)	LVR Reset	WDT Time-out (IDLE)
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	---- ---0	---- ---0	---- ---0	---- ---u
ACC	xxxx xxxx	uuuu uuuu	xxxx xxxx	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	xxxx xxxx	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-xxx xxxx	-uuu uuuu
TBHP	---- xxxx	---- uuuu	---- xxxx	---- uuuu
STATUS	--00 xxxx	--1u uuuu	--uu xxxx	--11 uuuu
SMOD	0000 0011	0000 0011	0000 0011	uuuu uuuu
LVDC	--00 -000	--00 -000	--00 -000	--uu –uuu
LVRC	0101 0101	0101 0101	0101 0101	uuuu uuuu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	0011 ----	0011 ----	0011 ----	uuuu ----
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC3	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI0	-000 -000	-000 -000	-000 -000	-uuu -uuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	LVR Reset	WDT Time-out (IDLE)
MFI1	--00 --00	--00 --00	--00 --00	--uu —-uu
MFI2	--00 --00	--00 --00	--00 --00	--uu --uu
MFI3	--00 --00	--00 --00	--00 --00	--uu --uu
MFI4	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI5	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI6	-000 -000	-000 -000	-000 -000	-uuu -uuu
MFI7	--00 --00	--00 --00	--00 --00	--uu --uu
MFI8	--00 --00	--00 --00	--00 --00	--uu --uu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPU	---- 0000	---- 0000	---- 0000	---- uuuu
PD	---- 1111	---- 1111	---- 1111	---- uuuu
PDC	---- 1111	---- 1111	---- 1111	---- uuuu
NF_VIH	0--11001	0--11001	0--11001	u--u uuuu
NF_VIL	---0 1010	---0 1010	---0 1010	---u uuuu
RMTC	0000 0000	0000 0000	0000 0000	uuuu uuuu
RMT0	0000 0000	0000 0000	0000 0000	uuuu uuuu
RMT1	0000 0000	0000 0000	0000 0000	uuuu uuuu
HCHK_NUM	---0 0000	---0 0000	---0 0000	---u uuuu
HNF_MSEL	---- 0000	---- 0000	---- 0000	---- uuuu
CAPTC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
CAPTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
CAPTMDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
CAPTMDH	0000 0000	0000 0000	0000 0000	uuuu uuuu
CAPTMAL	0000 0000	0000 0000	0000 0000	uuuu uuuu
CAPTMAH	0000 0000	0000 0000	0000 0000	uuuu uuuu
CAPTMCL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
CAPTMCH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
OPOMS	00-- -010	00-- -010	00-- -010	uu-- -uuu
OPCM	0000 0000	0000 0000	0000 0000	uuuu uuuu
LHMC	--00 --00	--00 --00	--00 --00	--uu --uu
HACM	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMPC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMPC1	---- --00	---- --00	---- --00	---- --uu
CTRL	0--- -x00	0--- -x00	0--- -x00	u--- -uuu
EEC	---- 0000	---- 0000	---- 0000	---- uuuu
EEA	-xxx xxxx	-xxx xxxx	-xxx xxxx	-uuu uuuu
EED	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	LVR Reset	WDT Time-out (IDLE)
ADRH	---- --xx	---- --xx	---- --xx	---- --uu
ADCR0	011- 0000	011- 0000	011- 0000	uuu- uuuu
ADCR1	0000 0000	0000 0000	0000 0000	uuuu uuuu
ANCSR0	1111 1111	1111 1111	1111 1111	uuuu uuuu
ANCSR1	---- ---1	---- ---1	---- ---1	---- ---u
ADDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADLVDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADLVDH	xxxx xx00	xxxx xx00	xxxx xx00	uuuu uuuu
ADHVDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADHVDH	---- --00	---- --00	---- --00	---- --uu
PWMC	--00 0--0	--00 0--0	--00 0--0	--uu u--u
DUTRL	0000 0000	0000 0000	0000 0000	uuuu uuuu
DUTRH	---- --00	---- --00	---- --00	---- --uu
PRDRL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PRDRH	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWMRL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWMRH	---- --00	---- --00	---- --00	---- --uu
MCF	---- 0100	---- 0100	---- 0100	---- uuuu
MCD	--00 0000	--00 0000	--00 0000	--uu uuuu
DTS	0000 0000	0000 0000	0000 0000	uuuu uuuu
PLC	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCR	0001 0000	0001 0000	0001 0000	uuuu uuuu
HDCD	---- -000	---- -000	---- -000	---- -uuu
HDCT0	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCT1	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCT2	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCT3	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCT4	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCT5	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCT6	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCT7	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCT8	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCT9	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCT10	--00 0000	--00 0000	--00 0000	--uu uuuu
HDCT11	--00 0000	--00 0000	--00 0000	--uu uuuu
MPTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
MPTC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM5C0	0000 0---	0000 0---	0000 0---	uuuu u---
TM5C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM5DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM5DH	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM5AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM5AH	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM5RP	0000 0000	0000 0000	0000 0000	uuuu uuuu
OPACAL	-001 0000	-001 0000	-001 0000	-uuu uuuu
DCMCR0	0000 1010	0000 1010	0000 1010	uuuu uuuu
DCMCR1	---0 0000	---0 0000	---0 0000	---u uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	LVR Reset	WDT Time-out (IDLE)
TM0C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DH	---- --00	---- --00	---- --00	---- --uu
TM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0AH	---- --00	---- --00	---- --00	---- --uu
TM1C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DH	---- --00	---- --00	---- --00	---- --uu
TM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1AH	---- --00	---- --00	---- --00	---- --uu
TM2C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2DH	---- --00	---- --00	---- --00	---- --uu
TM2AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2AH	---- --00	---- --00	---- --00	---- --uu
TM3C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM3C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM3DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM3DH	---- --00	---- --00	---- --00	---- --uu
TM3AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM3AH	0000 0000	0000 0000	0000 0000	uuuu uuuu

Note: " - " stands for 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.

This device provide bidirectional input/output lines labeled with port names PA~PD 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.

I/O Register List

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	D7	D6	D5	D4	D3	D2	D1	D0
PB	D7	D6	D5	D4	D3	D2	D1	D0
PBC	D7	D6	D5	D4	D3	D2	D1	D0
PCPU	D7	D6	D5	D4	D3	D2	D1	D0
PC	D7	D6	D5	D4	D3	D2	D1	D0
PCC	D7	D6	D5	D4	D3	D2	D1	D0
PDPU	—	—	—	—	D3	D2	D1	D0
PD	—	—	—	—	D3	D2	D1	D0
PDC	—	—	—	—	D3	D2	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, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using registers, namely PAPU~PDPU, 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

PBPU 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

PCPU 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 I/O Port bit 7~bit 0 pull-high control

PDPU Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 Port D bit 3~bit 0 pull-high control

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~PDC, 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

PBC 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

PCC 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 I/O port bit 7~bit 0 Input/Output control

PDC Register

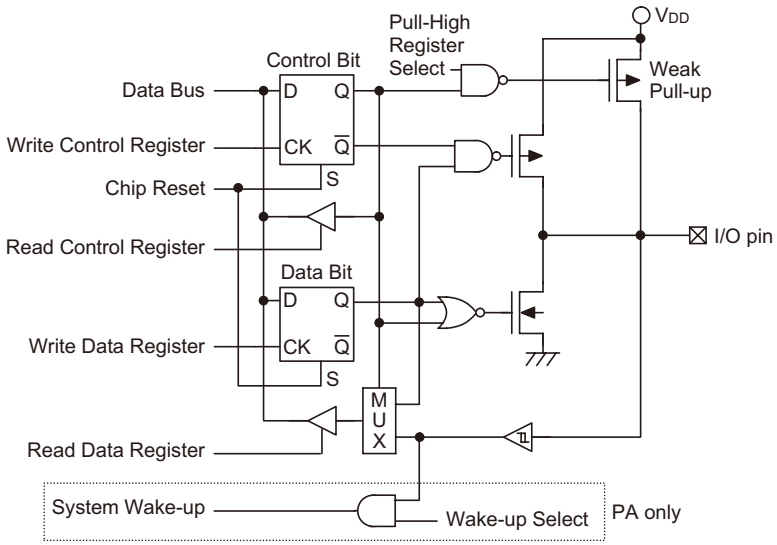
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D3	D2	D1	D0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

Bit 7~4 Unimplemented, read as "0"

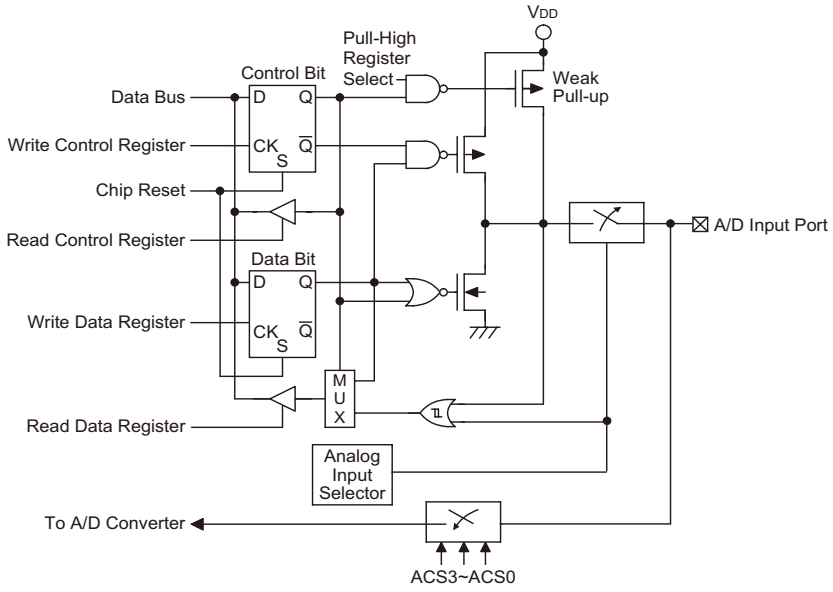
Bit 3~0 I/O Port bit 3~bit 0 Input/Output Control
0: Output
1: Input

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



A/D 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~PDC, 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~PD, are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

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

Timer Modules – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions each device includes several Timer Modules, abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has either multiple interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

The common features of the different TM types are described here with more detailed information provided in the individual Compact and Standard TM sections.

Introduction

The device contains five TMs having a reference name of TM0, TM1, TM2, TM3 and TM5. Each individual TM can be categorised as a certain type, namely Compact Type TM, four 10-bit CTM and one 16-bit CTM. The main features are summarised in the accompanying table.

Function	CTM
Timer/Counter	√
I/P Capture	—
Compare Match Output	√
PWM Channels	1
Single Pulse Output	—
PWM Alignment	Edge
PWM Adjustment Period & Duty	Duty or Period

TM Function Summary

TM Operation

TM offer a diverse range of functions, from simple timing operations to PWM signal generation. The key to understanding how the TM operates is to see it in terms of a free running counter whose value is then compared with the value of pre-programmed internal comparators. When the free running counter has the same value as the pre-programmed comparator, known as a compare match situation, a TM interrupt signal will be generated which can clear the counter and perhaps also change the condition of the TM output pin. The internal TM counter is driven by a user selectable clock source, which can be an internal clock or an external pin.

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the TnCK2~TnCK0 bits in the TM control registers. The clock source can be a ratio of either the system clock f_{SYS} or the internal high clock f_H, the f_{TBC} clock source or the external TCKn pin. Note that setting these bits to the value 101 will select an undefined clock input, in effect disconnecting the TM clock source. The TCKn pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

The Compact type TM has two internal interrupts, one for each of the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs.

TM External Pins

Each of the TMs, irrespective of what type, has one TM input pin, with the label TCKn. The TM input pin, is essentially a clock source for the TM and is selected using the TnCK2~TnCK0 bits in the TMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. This external TM input pin is shared with other functions but will be connected to the internal TM if selected using the TnCK2~TnCK0 bits. The TM input pin can be chosen to have either a rising or falling active edge.

The TMs each have two output pins with the label TPn. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external TPn output pin is also the pin where the TM generates the PWM output waveform. As the TM output pins are pin-shared with other function, the TM output function must first be setup using registers. A single bit in one of the registers determines if its associated pin is to be used as an external TM output pin or if it is to have another function.

All TM output pin names have a "_n" suffix. Pin names that include a "_0" or "_1" suffix indicate that they are from a TM with multiple output pins. This allows the TM to generate a complimentary output pair, selected using the I/O register data bits.

CTM0	CTM1	CTM2	CTM3	CTM5
TP0_0,TP0_1	TP1_0,TP1_1	TP2_0,TP2_1	TP3_0,TP3_1	TP5_0,TP5_1

TM Output Pins

TM Input/Output Pin Control Registers

Selecting to have a TM input/output or whether to retain its other shared function, is implemented using one or two registers, with a single bit in each register corresponding to a TM input/output pin. Setting the bit high will setup the corresponding pin as a TM input/output, if reset to zero the pin will retain its original other function.

Registers	Bit							
	7	6	5	4	3	2	1	0
TMPC0	T3CP1	T3CP0	T2CP1	T2CP0	T1CP1	T1CP0	T0CP1	T0CP0
TMPC1	—	—	—	—	—	—	T5CP1	T5CP0

TM Input/Output Pin Control Registers List

TMPC0 Register

Bit	7	6	5	4	3	2	1	0
Name	T3CP1	T3CP0	T2CP1	T2CP0	T1CP1	T1CP0	T0CP1	T0CP0
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 **T3CP1:** TP3_1 pin Control
0: Disable
1: Enable

Bit 6 **T3CP0:** TP3_0 pin Control
0: Disable
1: Enable

Bit 5 **T2CP1:** TP2_1 pin Control
0: Disable
1: Enable

Bit 4 **T2CP0:** TP2_0 pin Control
0: Disable
1: Enable

Bit 3 **T1CP1:** TP1_1 pin Control
0: Disable
1: Enable

Bit 2 **T1CP0:** TP1_0 pin Control
0: Disable
1: Enable

Bit 1 **T0CP1:** TP0_1 pin Control
0: Disable
1: Enable

Bit 0 **T0CP0:** TP0_0 pin Control
0: Disable
1: Enable

TMPC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	T5CP1	T5CP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

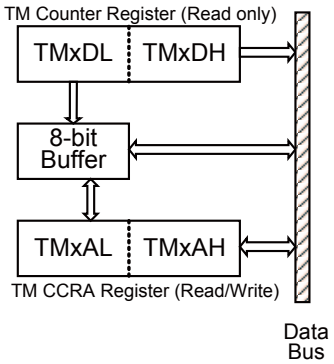
Bit 7~2 Unimplemented, read as "0"

Bit 1 **T5CP1:** TP5_1 pin Control
0: Disable
1: Enable

Bit 0 **T5CP0:** TP5_0 pin Control
0: Disable
1: Enable

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA is 10-bit or 16-bit register, have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed.

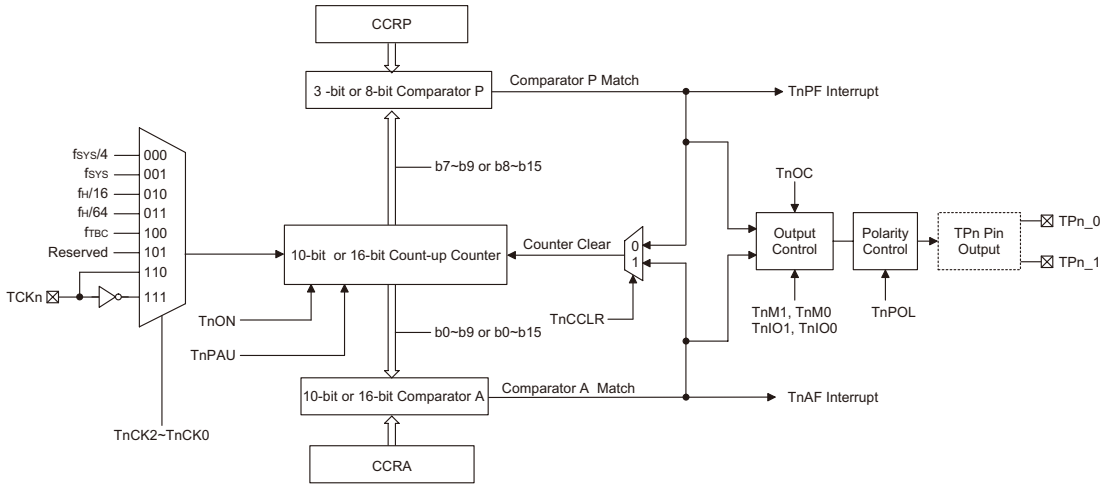


The following steps show the read and write procedures:

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

Compact Type TM – CTM

Although the simplest form of the TM types, the Compact TM type still contains three operating modes, which are Compare Match Output, Timer/Event Counter and PWM Output modes. The Compact TM can also be controlled with an external input pin and can drive two external output pins. These two external output pins can be the same signal or the inverse signal.



Compact Type TM Block Diagram (n=0, 1, 2, 3, 5)

Compact TM Operation

At its core is a 10-bit or 16-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is three bits wide whose value is compared with the highest three bits or eight bits in the counter while the CCRA is the ten bits or sixteen bits and therefore compares with all counter bits.

The only way of changing the value of the 10-bit or 16-bit counter using the application program, is to clear the counter by changing the TnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a TM interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control an output pin. All operating setup conditions are selected using relevant internal registers.

Compact Type TM Register Description

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

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMnC0	TnPAU	TnCK2	TnCK1	TnCK0	TnON	TnRP2	TnRP1	TnRP0
TMnC1	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnDPX	TnCCLR
TMnDL	D7	D6	D5	D4	D3	D2	D1	D0
TMnDH	—	—	—	—	—	—	D9	D8
TMnAL	D7	D6	D5	D4	D3	D2	D1	D0
TMnAH	—	—	—	—	—	—	D9	D8

10-bit Compact TM Register List(n=0, 1, 2, 3)

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TM5C0	T5PAU	T5CK2	T5CK1	T5CK0	T5ON	T5RP2	T5RP1	T5RP0
TM5C1	T5M1	T5M0	T5IO1	T5IO0	T5OC	T5POL	T5DPX	T5CCLR
TM5DL	D7	D6	D5	D4	D3	D2	D1	D0
TM5DH	D15	D14	D13	D12	D11	D10	D9	D8
TM5AL	D7	D6	D5	D4	D3	D2	D1	D0
TM5AH	D15	D14	D13	D12	D11	D10	D9	D8
TM5RP	D7	D6	D5	D4	D3	D2	D1	D0

16-bit Compact TM Register List

TMnDL Register(n=0, 1, 2, 3) — 10-bit CTM

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

Bit 7~0 **TMnDL:** TMn Counter Low Byte Register bit 7~bit 0
TMn 10-bit Counter bit 7~bit 0

TMnDH Register(n=0, 1, 2, 3) — 10-bit CTM

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **TMnDH:** TMn Counter High Byte Register bit 1~bit 0
TMn 10-bit Counter bit 9~bit 8

TMnAL Register(n=0, 1, 2, 3) — 10-bit CTM

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 **TMnAL:** TMn CCRA Low Byte Register bit 7~bit 0
TMn 10-bit CCRA bit 7~bit 0

TMnAH Register(n=0, 1, 2, 3) — 10-bit CTM

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **TMnAH:** TMn CCRA High Byte Register bit 1~bit 0
TMn 10-bit CCRA bit 9~bit 8

TM5DL Register — 16-bit CTM

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

Bit 7~0 **TM5DL:** TM5 Counter Low Byte Register bit 7~bit 0
TM5 16-bit Counter bit 7~bit 0

TM5DH Register — 16-bit CTM

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **TM5DH:** TM5 Counter High Byte Register bit 7~bit 0
TM5 16-bit Counter bit 15~bit 8

TM5AL Register — 16-bit CTM

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 **TM5AL:** TM5 CCRA Low Byte Register bit 7~bit 0
TM5 16-bit CCRA bit 7~bit 0

TM5AH Register — 16-bit CTM

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **TM5AH:** TM5 CCRA High Byte Register bit 8~bit 0
TM5 16-bit CCRA bit 15~bit 8

TMnC0 Register(n=0, 1, 2, 3) — 10-bit CTM

Bit	7	6	5	4	3	2	1	0
Name	TnPAU	TnCK2	TnCK1	TnCK0	TnON	TnRP2	TnRP1	TnRP0
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 **TnPAU:** TMn Counter Pause Control
0: Run
1: Pause
The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the TM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **TnCK2~TnCK0:** Select TMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: Reserved
110: TCKn rising edge clock
111: TCKn falling edge clock
These three bits are used to select the clock source for the TM0. Selecting the Reserved clock input will effectively disable the internal counter. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{TBC} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **TnON:** TMn Counter On/Off Control
0: Off
1: On
This bit controls the overall on/off function of the TMn. Setting the bit high enables the counter to run, clearing the bit disables the TMn. Clearing this bit to zero will stop the counter from counting and turn off the TMn which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal

counter will retain its residual value. If the TMn is in the Compare Match Output Mode then the TMn output pin will be reset to its initial condition, as specified by the TnOC bit, when the TnON bit changes from low to high.

Bit 2~0 **TnRP2~TnRP0:** TMn CCRP 3-bit register, compared with the TMn Counter bit 9~bit 7 Comparator P Match Period
000: 1024 TMn clocks
001: 128 TMn clocks
010: 256 TMn clocks
011: 384 TMn clocks
100: 512 TMn clocks
101: 640 TMn clocks
110: 768 TMn clocks
111: 896 TMn clocks

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

TMnC1 Register(n=0, 1, 2, 3) — 10-bit CTM

Bit	7	6	5	4	3	2	1	0
Name	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnDPX	TnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **TnM1~TnM0:** Select TMn Operating Mode
00: Compare Match Output Mode
01: Undefined
10: PWM Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the TM. To ensure reliable operation the TM should be switched off before any changes are made to the TnM1 and TnM0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5~4 **TnIO1~TnIO0:** Select TPn_0, TPn_1 output function
Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output
PWM Mode
00: PWM Output inactive state
01: PWM Output active state
10: PWM output
11: Undefined
Timer/counter Mode
unused

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

In the Compare Match Output Mode, the TnIO1 and TnIO0 bits determine how the TMn output pin changes state when a compare match occurs from the Comparator A. The TMn output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the TMn output pin should be setup using the TnOC bit in the TMnC1 register. Note that the output level requested by the TnIO1 and TnIO0 bits must be different from the initial value setup using the TnOC bit otherwise no change will occur on the TMn output pin when a compare match occurs. After the TMn output pin changes state it can be reset to its initial level by changing the level of the TnON bit from low to high.

In the PWM Mode, the TnIO1 and TnIO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the TnIO1 and TnIO0 bits only after the TMn has been switched off. Unpredictable PWM outputs will occur if the TnIO1 and TnIO0 bits are changed when the TM is running.

Bit 3 **TnOC:** TPn_0, TPn_1 Output control bit
Compare Match Output Mode
0: Initial low
1: Initial high
PWM Mode
0: Active low
1: Active high

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

Bit 2 **TnPOL:** TPn_0, TPn_1 Output polarity Control
0: Non-invert
1: Invert

This bit controls the polarity of the TPn_0 or TP0_1 output pin. When the bit is set high the TMn output pin will be inverted and not inverted when the bit is zero. It has no effect if the TMn is in the Timer/Counter Mode.

Bit 1 **TnDPX:** TMn PWM period/duty Control
0: CCRP - period; CCRA - duty
1: CCRP - duty; CCRA - period

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

Bit 0 **TnCCLR:** Select TMn Counter clear condition
0: TMn Comparator P match
1: TMn Comparator A match

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

TM5C0 Register—16-bit CTM

Bit	7	6	5	4	3	2	1	0
Name	T5PAU	T5CK2	T5CK1	T5CK0	T5ON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **T5PAU:** TM5 Counter Pause Control
0: Run
1: Pause
The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the TM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **T5CK2~T5CK0:** Select TM5 Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: Reserved
110: TCK5 rising edge clock
111: TCK5 falling edge clock
These three bits are used to select the clock source for the TM5. Selecting the Reserved clock input will effectively disable the internal counter. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{TBC} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **T5ON:** TM5 Counter On/Off Control
0: Off
1: On
This bit controls the overall on/off function of the TM5. Setting the bit high enables the counter to run, clearing the bit disables the TM5. Clearing this bit to zero will stop the counter from counting and turn off the TM5 which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value. If the TM5 is in the Compare Match Output Mode then the TM5 output pin will be reset to its initial condition, as specified by the T5OC bit, when the T5ON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

TM5C1 Register — 16-bit CTM

Bit	7	6	5	4	3	2	1	0
Name	T5M1	T5M0	T5IO1	T5IO0	T5OC	T5POL	T5DPX	T5CCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **T5M1~T5M0:** Select TM5 Operating Mode
00: Compare Match Output Mode
01: Undefined
10: PWM Mode
11: Timer/Counter Mode
These bits setup the required operating mode for the TM. To ensure reliable operation the TM should be switched off before any changes are made to the T5M1 and T5M0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5~4 **T5IO1~T5IO0:** Select TP5_0, TP5_1 output function
Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output
PWM Mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Undefined
Timer/counter Mode
unused

These two bits are used to determine how the TM5 output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the TM5 is running.

In the Compare Match Output Mode, the T5IO1 and T5IO0 bits determine how the TM5 output pin changes state when a compare match occurs from the Comparator A. The TM5 output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the TM5 output pin should be setup using the T5OC bit in the TM5C1 register. Note that the output level requested by the T5IO1 and T5IO0 bits must be different from the initial value setup using the T5OC bit otherwise no change will occur on the TM5 output pin when a compare match occurs. After the TM5 output pin changes state it can be reset to its initial level by changing the level of the T5ON bit from low to high.

In the PWM Mode, the T5IO1 and T5IO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the T5IO1 and T5IO0 bits only after the TM5 has been switched off. Unpredictable PWM outputs will occur if the T5IO1 and T5IO0 bits are changed when the TM is running.

Bit 3 **T5OC:** TP5_0, TP5_1 Output control bit
Compare Match output Mode
0: Initial low
1: Initial high
PWM Mode
0: Active low
1: Active high

This is the output control bit for the TM5 output pin. Its operation depends upon whether TM5 is being used in the Compare Match Output Mode or in the PWM Mode. It has no effect if the TM5 is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of he TM5 output pin before a compare match occurs. In the PWM Mode it determines if the PWM signal is active high or active low.

Bit 2 **T5POL:** TP5_0, TP5_1 output polarity Control
0: Non-invert
1: Invert

This bit controls the polarity of the TP5_0 or TP5_1 output pin. When the bit is set high the TM5 output pin will be inverted and not inverted when the bit is zero. It has no effect if the TMn is in the Timer/Counter Mode.

Bit 1 **T5DPX:** TM5 PWM period/duty Control
0: CCRP - period; CCRA - duty
1: CCRP - duty; CCRA - period

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

Bit 0 **T5CCLR:** Select TM5 Counter clear condition
0: TM5 Comparator P match
1: TM5 Comparator A match
This bit is used to select the method which clears the counter. Remember that the Compact TM5 contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the T5CCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The T5CCLR bit is not used in the PWM Mode.

TM5RP Register – 16-bit CTM

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 TM5 CCRP 8-bit register, compared with the TM5 Counter bit 15~bit 8 Comparator P Match Period
0: 1024 TMn clocks
1~255: 256×(1~255)TM5 clocks
These three bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight bits. The result of this comparison can be selected to clear the internal counter if the T5CCLR bit is set to zero. Setting the T5CCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all three bits to zero is in effect allowing the counter to overflow at its maximum value.

Compact Type TM Operating Modes

The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Mode or Timer/Counter Mode. The operating mode is selected using the TnM1 and TnM0 bits in the TMnC1 register.

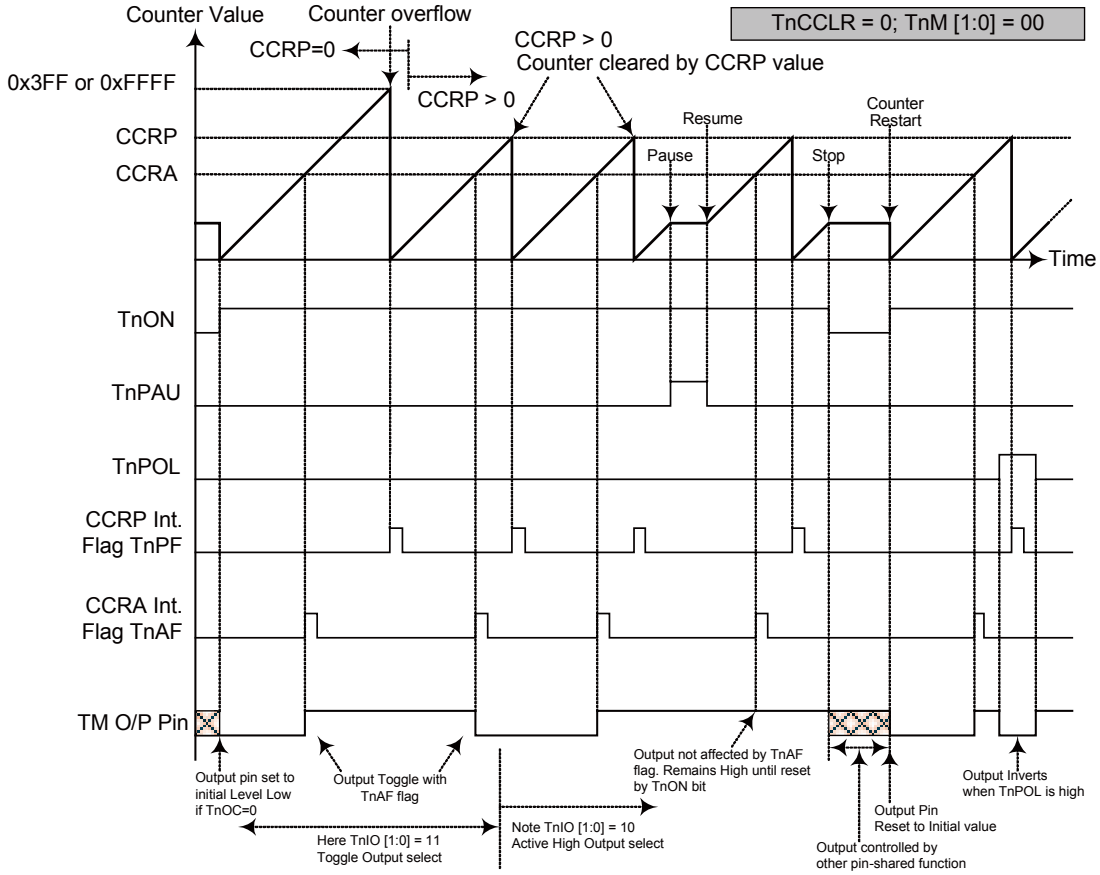
Compare Match Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register, should be set to "00" respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the TnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match occurs from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both TnAF and TnPF interrupt request flags for the Comparator A and Comparator P respectively, will both be generated.

If the TnCCLR bit in the TMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the TnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when TnCCLR is high no TnPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, or 16-bit, FFFF Hex ,value, however here the TnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the TM output pin will change state. The TM output pin condition however only changes state when a TnAF interrupt request flag

is generated after a compare match occurs from Comparator A. The TnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the TM output pin. The way in which the TM output pin changes state are determined by the condition of the TnIO1 and TnIO0 bits in the TMnC1 register. The TM output pin can be selected using the TnIO1 and TnIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the TM output pin, which is setup after the TnON bit changes from low to high, is setup using the TnOC bit. Note that if the TnIO1 and TnIO0 bits are zero then no pin change will take place.



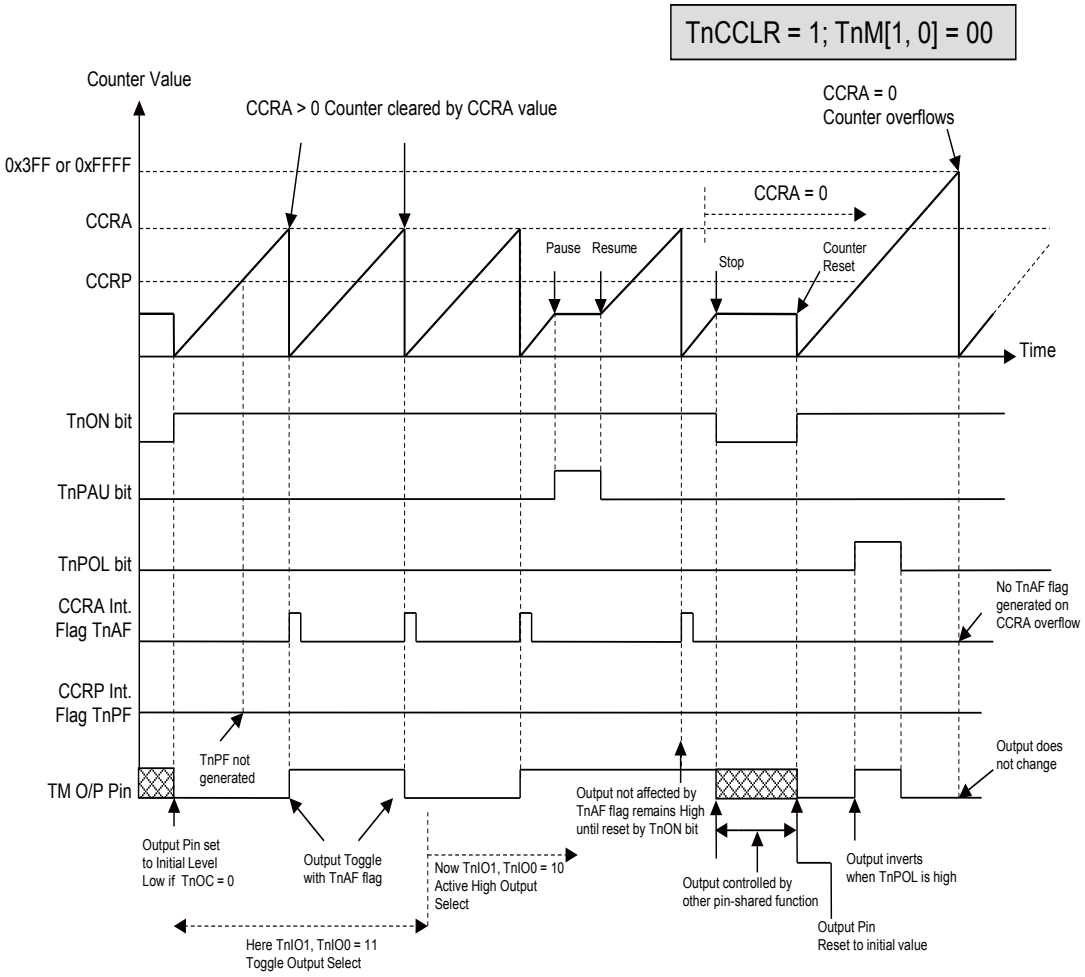
Compare Match Output Mode – TnCCLR=0

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

2. The TM output pin is controlled only by the TnAF flag

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

4. n=0, 1, 2, 3, 5



Compare Match Output Mode – TnCCLR=1

Note: 1. With TnCCLR=1, a Comparator A match will clear the counter
2. The TM output pin is controlled only by the TnAF flag
3. The output pin is reset to its initial state by a TnON bit rising edge
4. The TnPF flag is not generated when TnCCLR=1
5. n=0, 1, 2, 3, 5

Timer/Counter Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the TM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the TM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 10 respectively. The PWM function within the TM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the TM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM mode, the TnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the TnDPX bit in the TMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The TnOC bit in the TMnC1 register is used to select the required polarity of the PWM waveform while the two TnIO1 and TnIO0 bits are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The TnPOL bit is used to reverse the polarity of the PWM output waveform.

10-bit CTM, PWM Mode, Edge-aligned Mode, TnDPX=0

CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	128	256	384	512	640	768	896	1024
Duty	CCRA							

If f_{SYS}=16MHz, TM clock source is $f_{SYS}/4$, CCRP=100b and CCRA=128,

The CTM PWM output frequency=($f_{SYS}/4$)/512=$f_{SYS}/2048$=7.8125 kHz, duty=128/512=25%.

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

10-bit CTM, PWM Mode, Edge-aligned Mode, TnDPX=1

CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

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

16-bit CTM, PWM Mode, Edge-aligned Mode, TnDPX=0

CCRP	1~255	0
Period	CCRP×256	65536
Duty	CCRA	

If f_{SYS}=16MHz, TM clock source is f_{SYS}/4, CCRP=2 and CCRA=128,

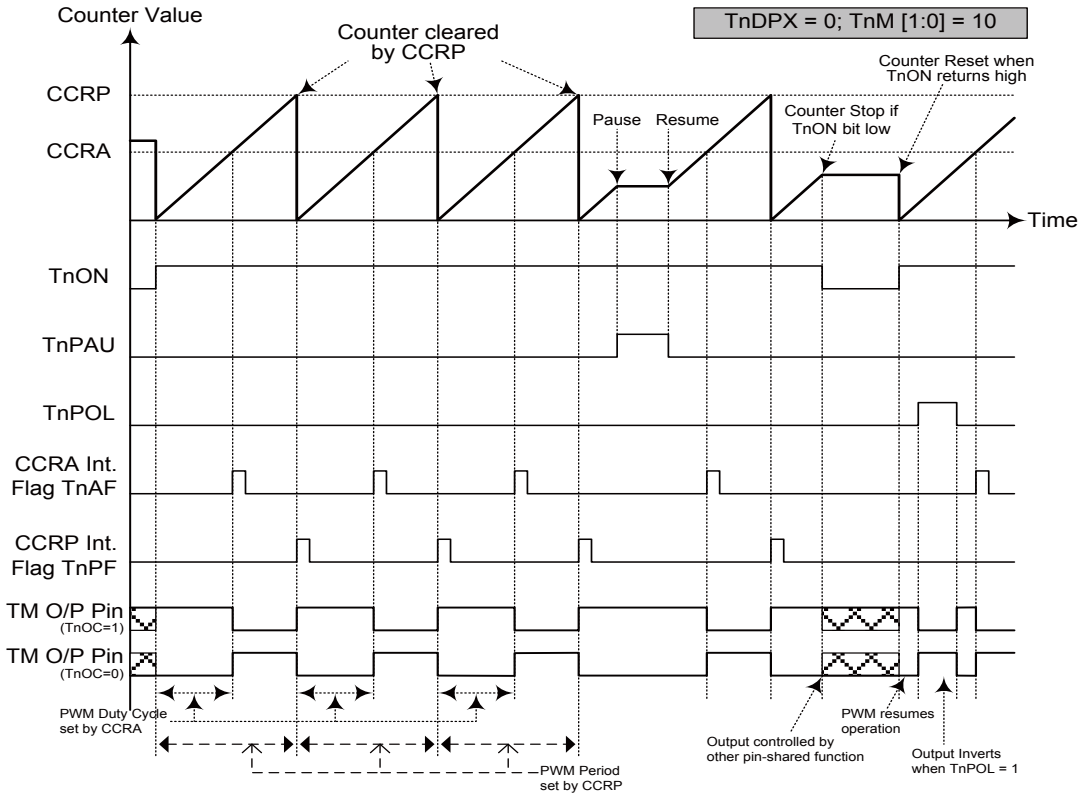
The CTM PWM output frequency=(f_{SYS}/4)/(2×256)=f_{SYS}/2048=7.8125 kHz, duty=128/(2×256)=25%.

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

16-bit CTM, PWM Mode, Edge-aligned Mode, TnDPX=1

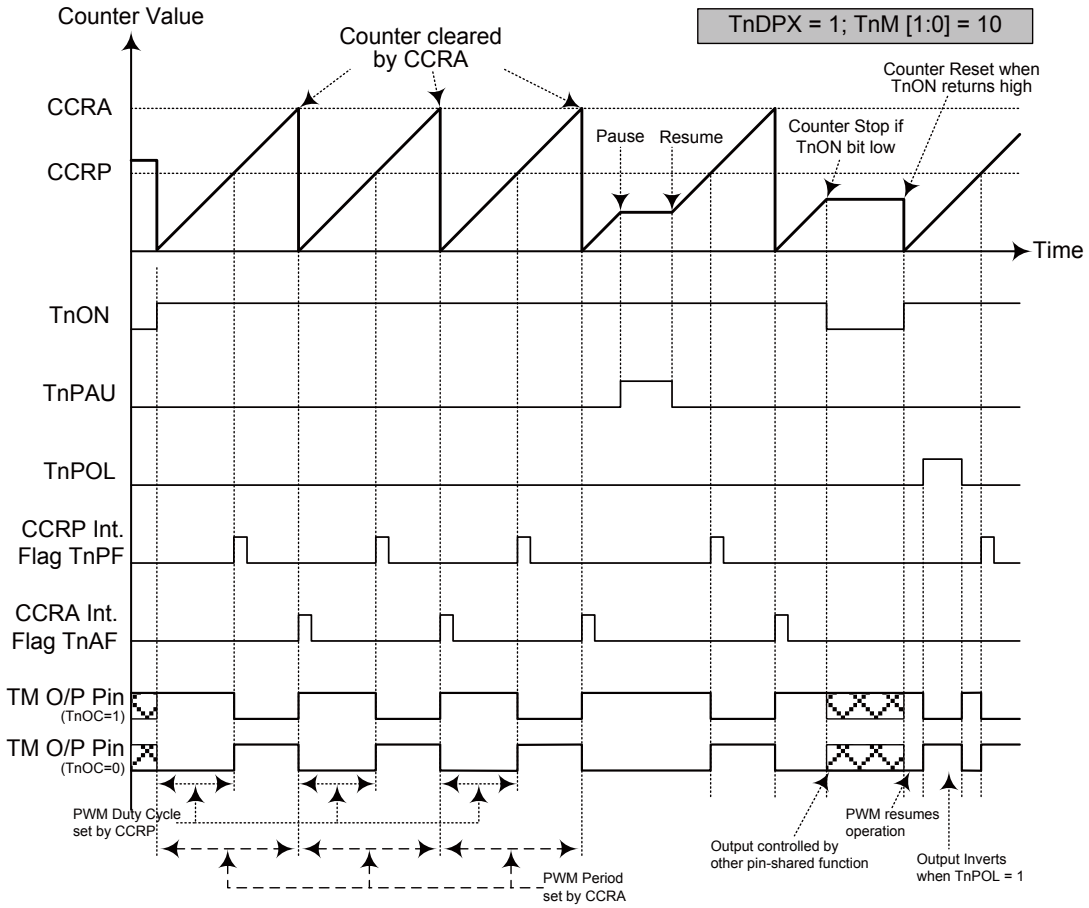
CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

The PWM output period is determined by the CCRA register value together with the TM clock while the PWM duty cycle is defined by the (CCRP×256) except when the CCRP value is equal to 0.



PWM Mode – TnDPX=0

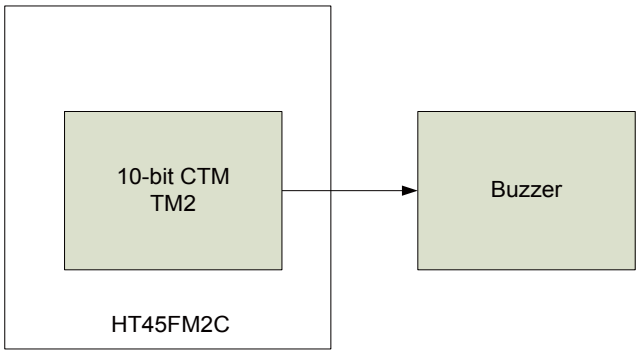
Note: 1. Here TnDPX=0 – Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues even when TnIO [1:0]=00 or 01
4. The TnCCLR bit has no influence on PWM operation
5. n=0, 1, 2, 3, 5



PWM Mode – TnDPX=1

Note: 1. Here TnDPX=1 – Counter cleared by CCRA
2. A counter clear sets the PWM Period
3. The internal PWM function continues even when TnIO [1:0]=00 or 01
4. The TnCCLR bit has no influence on PWM operation
5. n=0, 1, 2, 3, 5

Buzzer control



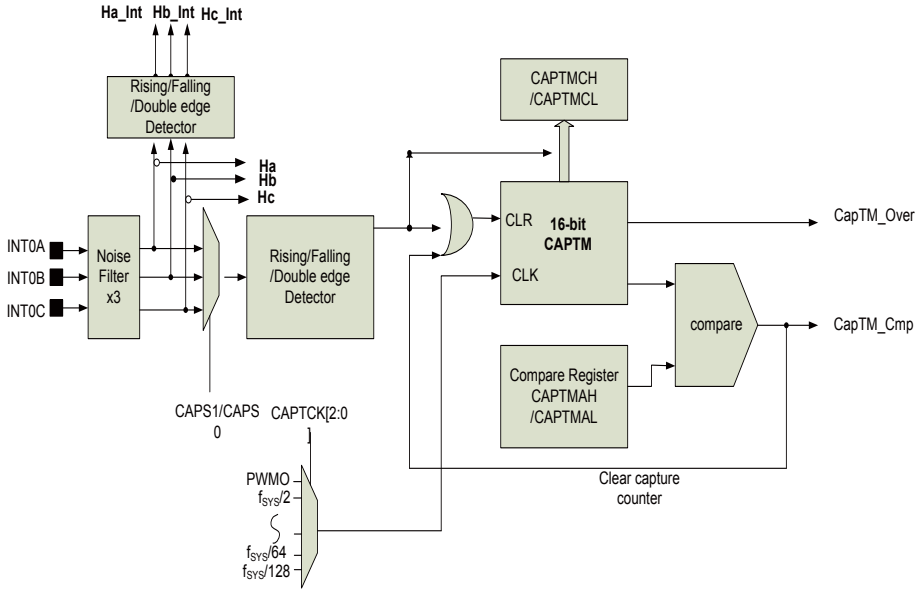
The 10-bit CTM can drive an external buzzer using its PWM mode to provide volume control.

Capture Timer Module – CAPTM

The Capture Timer Module is a timing unit specifically used for Motor Control purposes. The CAPTM is controlled by a program selectable clock source and by three interrupt sources from the motor positioning hall sensors.

Capture Timer Overview

At the core of the Capture Timer is a 16-bit count-up counter which is driven by a user selectable internal clock source which is some multiple of the system clock or by the PWM. There is also an internal comparator which compares the value of this 16-bit counter with a pre-programmed 16-bit value stored in two registers. There are two basic modes of operation, a Compare Mode and a Capture Mode, each of which can be used to reset the internal counter. When a compare match situation is reached a signal will be generated to reset the internal counter. The counter can also be cleared when a capture trigger is generated by the three external sources, INT0A, INT0B and INT0C.



Capture Timer Block Diagram

Capture Timer Register Description

Overall operation of the Capture Timer is controlled using eight registers. A read only register pair exists to store the internal counter 16-bit value, while a read/write register pair exists to store the internal 16-bit compare value. An additional read only register pair is used to store the capture value. The remaining two registers are control registers which setup the different operating and control modes.

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CAPTC0	CAPTPAU	CAPTCK2	CAPTCK1	CAPTCK0	CAPTON	—	CAPS1	CAPS0
CAPTC1	CAPEG1	CAPEG0	CAPEN	CAPNFT	CAPNFS	CAPFIL	CAPCLR	CAMCLR
CAPTMDL	D7	D6	D5	D4	D3	D2	D1	D0
CAPTMDH	D15	D14	D13	D12	D11	D10	D9	D8
CAPTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CAPTMAH	D15	D14	D13	D12	D11	D10	D9	D8
CAPTMCL	D7	D6	D5	D4	D3	D2	D1	D0
CAPTMCH	D15	D14	D13	D12	D11	D10	D9	D8

Capture Timer Register List

CAPTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	CAPTPAU	CAPTCK2	CAPTCK1	CAPTCK0	CAPTON	—	CAPS1	CAPS0
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 **CAPTPAU:** CAPTM Counter Pause Control
0: Run
1: Pause

The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the CAPTM will remain power up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again

Bit 6~4 **CAPTCK2~CAPTCK0:** Select CAPTM Counter clock
000: PWMO
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: $f_H/128$

These three bits are used to select the clock source for the CAPTM. The clock source f_H is the high speed system oscillator.

Bit 3 **CAPTON:** CAPTM Counter On/Off Control
0: Off
1: On

This bit controls the overall on/off function of the CAPTM. Setting the bit high enables the counter to run, clearing the bit disables the CAPTM. Clearing this bit to zero will stop the counter from counting and turn off the CAPTM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value.

Bit 2 Unimplemented, read as "0"

Bit 1~0 **CAPS1~CAPS0:** capture source select
00: INT0A
01: INT0B
10: INT0C
11: Unused

CAPTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	CAPEG1	CAPEG0	CAPEN	CAPNFT	CAPNFS	CAPFIL	CAPCLR	CAMCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **CAPEG1~CAPEG0:** Defines CAPTM capture active edge
00: Disabled CAPTM capture
01: Rising edge capture
10: Falling edge capture
11: Dual edge capture

Bit 5 **CAPEN:** CAPTM Capture input control
0: Disable
1: Enable
This bit enables/disables the CAPTM capture input source.

Bit 4 **CAPNFT:** Defines CAPTM Noise Filter sample times
0: Twice
1: 4 times
The CAPTM Noise Filter circuit requires sampling twice or 4 times continuously, when they are all the same, the signal will be acknowledged. The sample time is decided by CAPNFS.

Bit 3 **CAPNFS:** CAPTM Noise Filter clock source Select
0: t_{SYS}
1: $4t_{SYS}$
The clock source for Capture Timer Module Counter is provided by f_{SYS} or $f_{SYS}/4$.

Bit 2 **CAPFIL:** CAPTM capture input filter Control
0: Disable
1: Enable
This bit enables/disables the CAPTM capture input filter.

Bit 1 **CAPCLR:** CAPTM Counter capture auto-reset Control
0: Disable
1: Enable
This bit enables/disables the automatic reset of the counter when the value in CAPTMDL and CAPTMDH have been transferred into the capture registers CAPTMCL and CAPTMCH.

Bit 0 **CAMCLR:** CAPTM Counter compare match auto-reset Control
0: Disable
1: Enable
This bit enables/disables the automatic reset of the counter when the a compare match has occurred.

CAPTMDL Register

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

Bit 7~0 **CAPTMDL:** CAPTM Counter Low Byte Register bit 7~bit 0
CAPTM 16-bit Counter bit 7 ~ bit 0

CAPTMDH Register

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **CAPTMDH:** CAPTM Counter High Byte Register bit 7~bit 0
CAPTM 16-bit Counter bit 15 ~ bit 8.

CAPTMAL 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 **CAPTMAL:** CAPTM Compare Low Byte Register bit 7~bit 0
CAPTM 16-bit Compare Register bit 7~bit 0.

CAPTMAH Register

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **CAPTMAH:** CAPTM Compare High Byte Register bit 7~bit 0
CAPTM 16-bit Compare Register bit 15~bit 8.

CAPTMCL Register

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

"x"unknown

Bit 7~0 **CAPTMCL:** CAPTM Capture Low Byte Register bit 7~bit 0
CAPTM 16-bit Capture Register bit 7~bit 0

CAPTMCH Register

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x

"x"unknown

Bit 7~0 **CAPTMCH:** CAPTM Capture High Byte Register bit 7~bit 0
CAPTM 16-bit Capture Register bit 15~bit 8.

Capture Timer Operation

The Capture Timer is used to detect and measure input signal pulse widths and a periods. It can be used in both a Capture or Compare Mode. The timer inputs are the three capture inputs INT0A, INT0B and INT0C. Each of these capture inputs has its own edge detector selection, to choose between high, low or both edge trigger types.

The CAPTON bit is used to control the overall Capture Timer enable/disable function. Disabling the Capture Module when not used will reduce the device power consumption. Additionally the capture input control is enabled/disabled using the CAPEN control bit. The trigger edge option are setup using the CAPEG1 and CAPEG0 bits, to select either positive edge, negative edge or both edges.

Capture Mode Operation

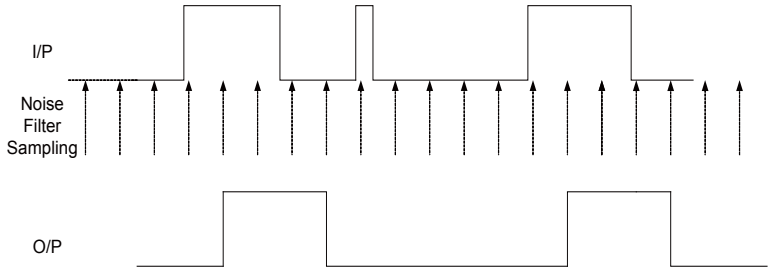
The capture timer module contains 2 capture registers, CAPTMCL and CAPTMCH, which are used to store the present value in the counter. When the Capture Module is enabled, then each time an external pin receives a valid trigger signal, the content of the free running 16-bit counter, which is contained in the CAPTMDL and CAPTMDH registers, will be captured into the capture registers, CAPTMCL and CAPTMCH. When this occurs, the CAPOF interrupt flag bit in the interrupt control register will be set. If this interrupt is enabled by setting the interrupt enable bit, CAPOE, high, an interrupt will be generated. If the CAPCLR bit is set high, then the 16-bit counter will be automatically reset after a capture event occurs.

Compare Mode Operation

When the timer is used in the compare mode, the CAPTMAL and CAPTMAH registers are used to store the 16-bit compare value. When the free running value of the count-up 16-bit counter reaches a value equal to the programmed values in these compare registers, the CAPCF interrupt flag will be set which will generate an interrupt if its related interrupt enable bit is set. If the CAMCLR bit is set high, then the counter will be reset to zero automatically when a compare match condition occurs. The rotor speed or a stalled motor condition can be detected by setting the compare registers to compare the captured signal edge transition time. If a rotor stall condition occurs, then a compare interrupt will be generated, after which the PWM motor drive circuit can be shut down to prevent a motor burn out situation.

Noise Filter

The timer also includes a noise Filter which is used to filter out unwanted glitches or pulses on the trigger input pins. This function is enabled using the CAPFIL bit. If the noise filter is enabled, the capture input signals must be sampled either 2 or 4 times, in order to recognize an edge as a valid capture event. The sampling 2 or 4 time units are based o either t_{SYS} or $4 \times t_{SYS}$ determined using the CAPNFS bit.



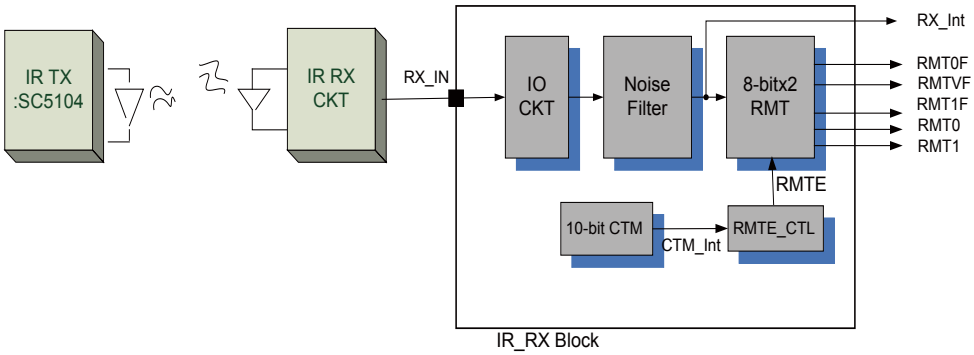
Noise Filter with CAPNFT and CATNFS = 0

Infrared Receiver

The device contains a function block to receive signals from infrared remote controls. These circuits assist with the implementation of integrated remote control functions for remote motor control.

Functional Description

The infrared receiver functional block contains a number of units to facilitate the implementation of infrared signal decoding such as IR code receiver circuit, noise filter, RMT capture circuit and RMTE control.



Infrared Receiver Block Diagram

The external RX_IN 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 RX_IN input signal. In order to ensure that the IR Code Rx circuit and the motor control circuit works normally.

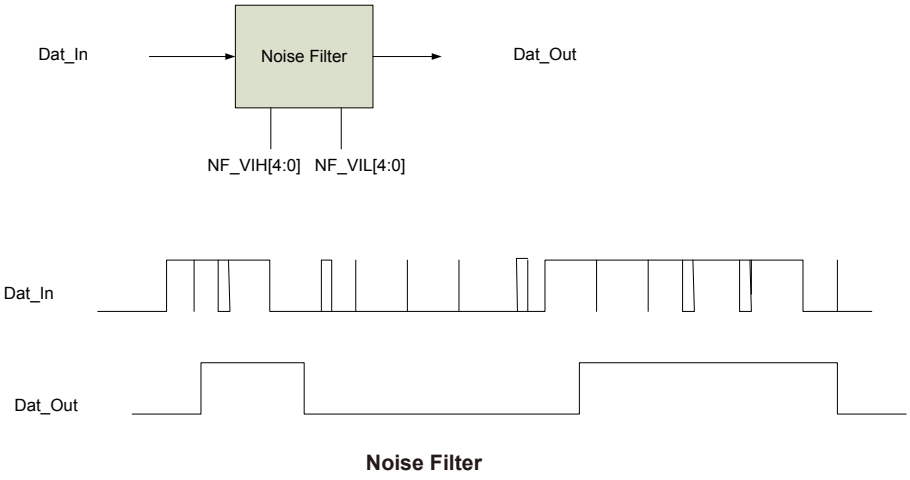
The RMT Capture circuit is implemented using two 8-bit RMT circuits, RMT0 and RMT1 registers to decode IR. As the IR code can be transmitted repeatedly, the RMTE control circuit can make the decoding time short and reduce the effects which generated by the remote controlling on the motor controlling. The noise filter circuit is a I/O filtering surge compare which can filter micro-second grade sharp-noise.

Antinoise pulse width maximum: (NF_VIH[5:0]-NF_VIL[5:0])×5μs

RMT Timing

Noise Filter

A noise filter circuit is included to reduce the possibility of noise spikes or erroneous signal inputs being decoded as genuine inputs signals.



Noise Filter

Noise Filter Registers Description

NF_VIH Register

Bit	7	6	5	4	3	2	1	0
Name	NF_BYPS	—	—	D4	D3	D2	D1	D0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	1	1	0	0	1

Bit 7 **NF_BYPS:** Bypass Noise Filter Enable
0: Disable
1: Enable, Dat_Out=Dat_In

Bit 6~5 Unimplemented, read as "0"

Bit 4~0 NF_VIH Bit 4~Bit 0

NF_VIL 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	1	0	1	0

Bit 7~5 Unimplement, read as "0"

Bit 4~0 NF_VIL Bit 4~Bit0

Remote Control Timer – RMT

The device contains two 8-bit RMT timer functions which are used for IR signal decoding. This function can be used to allow remote controllers to change the required motor operating mode.

The Remote Control Timer can detect an edge transition on the RX_IN pin, after which three interrupt.

signals can be generated. These are, rising edge interrupt signal, falling edge interrupt signal and a timer overflow interrupt signal. The control registers, RMT0 and RMT1, are used to store the captured data which measures the infrared input signal edge interval changes. The recorded data can then be used for IR decoding purposes.

The application program can be used to decode the IR code frame data in the following ways:

- **Disabling the RMTE**

  When an IR data frame has been decoded by the program, then the RMT can be disabled by the following: RME=0 → RMTE=0.

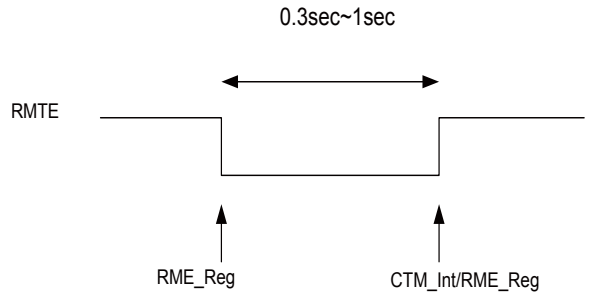
- **Enabling the RMTE**

  Use the 10-bit CTM as the IR Decode scan restart mechanism to improve the motor control efficiency.

  ♦ S/W Mode: RME is set to high by the S/W.

  ♦ H/W Mode: when a CTM_Int is detected by the H/W (about 0.3s~1s) , then set RMTE.

  Note: f_{TBC} is selected as the CTM clock source by S/W and adjusted using the TBC register.



Hardware Mode

RMT Register Description

Three registers are used for overall control of the Remote Control Timer. A control register, RMTC, is used to setup the timer, while registers, RMT0 and RMT1, are used to store the decoded signal data.

RMTC Register

Bit	7	6	5	4	3	2	1	0
Name	RMS1	RMS0	RMCS	RME	ERMTV	ERMT1	ERMT0	RMEMS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **RMS1, RMS0:** Selects the Remote Control Timer clock
00: $f_X/2^5$
01: $f_X/2^6$
10: $f_X/2^7$
11: $f_X/2^8$

Bit 5 **RMCS:** Selects the Remote Control Timer clock source f_X
0: $f_{sys}/4$
1: f_{sys}

Bit 4 **RME:** Controls the remote control timer
0: Disable and clear counter to 0
1: Enable and start counting

Bit 3 **ERMTV:** Controls the Remote Control Timer overflow interrupt
0: Disable
1: Enable

Bit 2 **ERMT1:** Controls the Remote Control Timer falling edge interrupt
0: Disable
1: Enable

Bit 1 **ERMT0:** Controls the Remote Control Timer rising edge interrupt
0: Disable
1: Enable

Bit 0 **RMEMS:** RMTE Circuit Mode Select
0: S/W Mode, RMTE start circuit is defined by RME bit via S/W
1: H/W Mode, RMTE start circuit is defined by CTM interrupt via H/W

RMT0 Register

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

Bit 7~0 **RMT0:** low level edge capture register Bit 7~Bit 0

RMT1 Register

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

Bit 7~0 **RMT1:** high level edge capture register Bit 7~Bit 0

Analog to Digital Converter

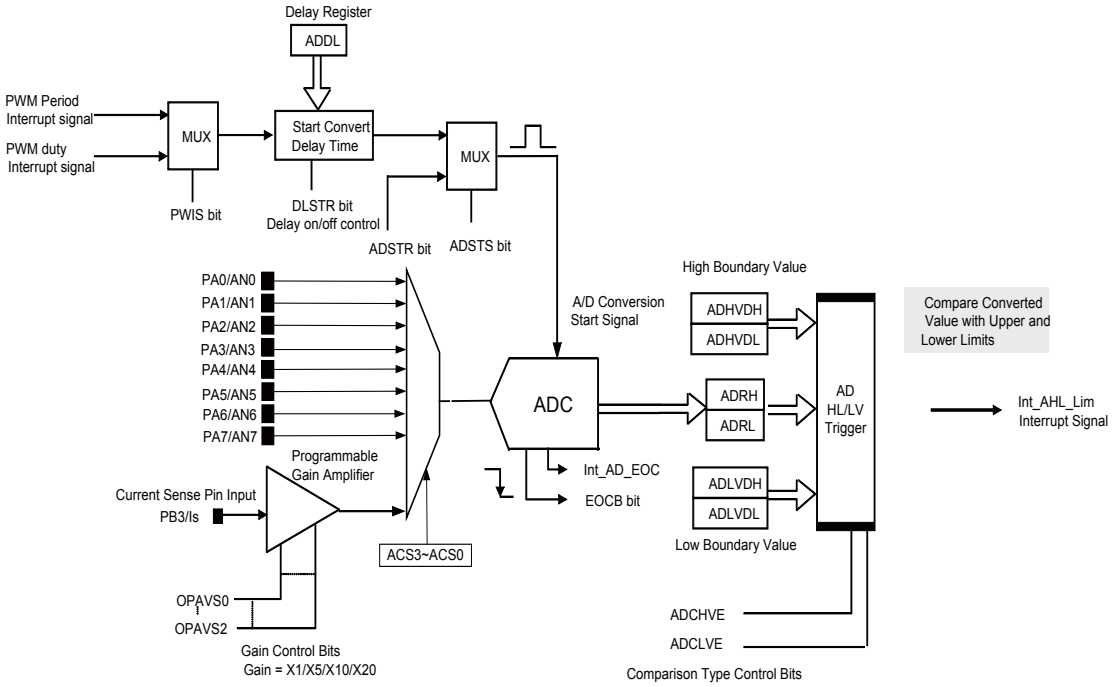
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

A/D Overview

This device contains a 9-channel analog to digital converter, 8-channel can be directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into either a 10-bit digital value. An additional channel is connected to the external current sense input pin, Is, via an internal operational amplifier for signal amplification, before being transferred to the A/D converter input. A set of what are known as high and low boundary registers, allow the A/D converter digital output value to be compared with upper and lower limit values and a corresponding interrupt to be generated. An additional delay function allows a delay to be inserted into the PWM triggered A/D conversion start process to reduce the possibility of erroneous analog value sampling when the output power transistors are switching large motor currents.

Input Channels	A/D Channel Select Bits	Input Pins
9	ACS3~ACS0	AN0~AN7, Is

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



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using several registers. A read only register pair ADRL/ADRH exists to store the ADC data 10-bit value. The ADLVDL/ADLVDH and ADHVDL/ADHVDH registers are used to store the boundary limit values of the ADC interrupt trigger while the ADDL register is used to setup the start conversion delay time. The remaining registers are control registers which setup the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
ADRL	D7	D6	D5	D4	D3	D2	D1	D0
ADRH	—	—	—	—	—	—	D9	D8
ADCR0	ADSTR	EOCB	ADOFF	—	ACS3	ACS2	ACS1	ACS0
ADCR1	ADSTS	DLSTR	PWIS	ADCHVE	ADCLVE	ADCK2	ADCK1	ADCK0
ANCSR0	PCR7	PCR6	PCR5	PCR4	PCR3	PCR2	PCR1	PCR0
ANCSR1	—	—	—	—	—	—	—	PCR8
ADDL	D7	D6	D5	D4	D3	D2	D1	D0
ADLVDL	D7	D6	D5	D4	D3	D2	D1	D0
ADLVDH	—	—	—	—	—	—	D9	D8
ADHVDL	D7	D6	D5	D4	D3	D2	D1	D0
ADHVDH	—	—	—	—	—	—	D9	D8

A/D Converter Register List

A/D Converter Data Registers – ADRL, ADRH

As this device contains an internal 10-bit A/D converter, it requires two data registers to store the converted value. These are a high byte register, known as ADRH, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value.

ADRL Register

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

"x"unknown

Bit 7~0 A/D Low Byte Register Bit 7~Bit 0

ADRH Register

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

"x"unknown

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 A/D High Byte Register Bit 1, Bit 0

A/D Converter Control Registers – ADCR0, ADCR1, ANCSR0, ANCSR1, ADDL

To control the function and operation of the A/D converter, four control registers known as ADCR0, ADCR1, ANCSR0 and ANCSR1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter end of conversion status. The ACS3~ACS0 bits in the ADCR0 register define the ADC input channel number. As the device contains only one actual analog to digital converter hardware circuit, each of the individual 9 analog inputs must be routed to the converter. It is the function of the ACS3~ACS0 bits to determine which analog channel input pins or Is pin is actually connected to the internal A/D converter.

The ANCSR0 and ANCSR1 control registers contain the PCR8~PCR0 bits which determine which pins on Port A or PB3 is used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. Setting the corresponding bit high will select the A/D input function, clearing the bit to zero will select either the I/O or other pin-shared function. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistors connected to these pins will be automatically removed if the pin is selected to be an A/D input.

The ADDL register exists to store the ADC delay start time.

ADCR0 Register

Bit	7	6	5	4	3	2	1	0
Name	ADSTR	EOCB	ADOFF	—	ACS3	ACS2	ACS1	ACS0
R/W	R/W	R	R/W	—	R/W	R/W	R/W	R/W
POR	0	1	1	—	0	0	0	0

Bit 7 **ADSTR:** Start the A/D conversion
0→1→0: Start
0→1: Reset the A/D converter and set EOCB to "1"
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process. When the bit is set high the A/D converter will be reset.

Bit 6 **EOCB:** End of A/D conversion flag
0: A/D conversion ended
1: A/D conversion in progress
This read only flag is used to indicate when an A/D conversion process has completed. When the conversion process is running, the bit will be high.

Bit 5 **ADOFF** : ADC module power on/off control bit
0: ADC module power on
1: ADC module power off
This bit controls the power to the A/D internal function. This bit should be cleared to zero to enable the A/D converter. If the bit is set high then the A/D converter will be switched off reducing the device power consumption. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.
Note: 1. it is recommended to set ADOFF=1 before entering IDLE/SLEEP Mode for saving power.
2. ADOFF=1 will power down the ADC module.

Bit 4 Unimplemented, read as "0"

Bit 3~0 **ACS3 ~ ACS0:** Select A/D channel
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000: Is currrent sense input - via amplifier
These are the A/D channel select control bits. As there is only one internal hardware A/D converter each of the eight A/D inputs must be routed to the internal converter using these bits.

ADCR1 Register

Bit	7	6	5	4	3	2	1	0
Name	ADSTS	DLSTR	PWIS	ADCHVE	ADCLVE	ADCK2	ADCK1	ADCK0
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 **ADSTS:** Select ADC trigger circuit
0: Select ADSTR trigger circuit
1: Select DELAY trigger circuit

Bit 6 **DLSTR:** Delay start function control
0: Disable but need to set ADDL to "0"
1: Enable but need to set ADDL to non zero value

Bit 5 **PWIS:** Select PWM Module interrupt source
0: Select PWM period interrupt
1: Select PWM duty interrupt

Bit 4~3 **ADCHVE~ADCLVE:** Select ADC interrupt trigger source
00: ADLVD[9:0] < ADR[9:0] < ADHVD[9:0]
01: ADR[9:0] <= ADLVD[9:0]
10:ADR[9:0] >= ADHVD[9:0]
11: ADR[9:0] <= ADLVD[9:0] or ADR[9:0] >= ADHVD[9:0]

Bit 2~0 **ADCK2~ADCK0:** Select ADC clock source
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: Undefined
These three bits are used to select the clock source for the A/D converter.

ANCSR0 Register

Bit	7	6	5	4	3	2	1	0
Name	PCR7	PCR6	PCR5	PCR4	PCR3	PCR2	PCR1	PCR0
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 **PCR7:** A/D input pin select
0: Not A/D input
1: A/D input, AN7

Bit 6 **PCR6:** A/D input pin select
0: Not A/D input
1: A/D input, AN6

Bit 5 **PCR5:** A/D input pin select
0: Not A/D input
1: A/D input, AN5

Bit 4 **PCR4:** A/D input pin select
0: Not A/D input
1: A/D input, AN4

Bit 3 **PCR3:** A/D input pin select
0: Not A/D input
1: A/D input, AN3

Bit 2 **PCR2:** A/D input pin select
0: Not A/D input
1: A/D input, AN2

Bit 1 **PCR1:** A/D input pin select
0: Not A/D input
1: A/D input, AN1

Bit 0 **PCR0:** A/D input pin select
0: Not A/D input
1: A/D input, AN0

ANCSR1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	PCR8
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	1

Bit 7~1 Unimplemented, read as "0"

Bit 0 **PCR8:** A/D input pin select
0: Not A/D input
1: A/D input, Is input, AN8

ADDL 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 ADC Delay-Time register Bit 7~Bit 0
Delay-Time Value (count by system clock)

A/D Converter Boundary Registers – ADLVDL, ADLVDH, ADHVDL, ADHVDH

The device contains what are known as boundary registers to store fixed values for comparison with the A/D converter converted value stored in ADRL and ADRH. There are two pairs of registers, a high boundary pair, known as ADHVDL and ADHVDH and a low boundary pair known as ADLVDL and ADLVDH.

ADLVDL 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 ADC Low Boundary Low Byte Register Bit 7~Bit 0

ADLVDH Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 ADC Low Boundary High Byte Register Bit 1, Bit 0

ADHVDL 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 ADC High Boundary Low Byte Register Bit 7~Bit 0

ADHVDH Register

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

"x"unknown

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 ADC High Boundary High Byte Register Bit 1~Bit 0

A/D Operation

There are two ways to initiate an A/D Converter conversion cycle, selected using the ADSTS bit. The first of these is to use the ADSTR bit in the ADCR0 register used to start and reset the A/D converter. When the microcontroller program sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the ADSTR bit is brought from low to high but not low again, the EOCB bit in the ADCR0 register will be set high and the analog to digital converter will be reset.

The second method of initiating a conversion is to uese the PWM interrupt signal. This can be sourced from either the PWM period or duty interrupt signal, selected using the PWIS bit. The DLSTR bit can activate a delay function which inserts a delay time between the incoming PWM interrupt signal and the actual start of the A/D conversion process, with the actual time being setup using the ADDL register. The actual delay time is calculated by the register content multiplied by the system clock period. The delay between the PWM interrupt and the start of the A/D conversion is to reduce the possibility of erroneous analog samples being taken during the time of large transient current switching by the motor drive transistors. Note that if the DLSTR bit selects no delay the ADDL register must be cleared to zero and vice-versa if the delay is selected, then a non-zero value must be programmed into the ADDL register.

The EOCB bit in the ADCR0 register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to zero by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the EOCB bit in the ADCR0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the ADCK2~ADCK0 bits in the ADCR1 register. Although the A/D clock source is determined by the system clocky, f_{SYS}, and by bits ADCK2~ADCK0, there are some limitations on the maximum A/D clock source speed that can be selected. As the minimum value of permissible A/D clock period, t_{ADCK}, is 0.5μs, care must be taken for system clock frequencies equal to or greater than 4MHz. For example, if the system clock operates at a frequency of 4MHz, the ADCK2~ADCK0 bits should not be set to "000". Doing so will give A/D clock periods that are less than the minimum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

f_{SYS}	A/D Clock Period (t_{ADCK})							
	ADCK2, ADCK1, ADCK0 =000 (f_{SYS})	ADCK2, ADCK1, ADCK0 =001 (f_{SYS}/2)	ADCK2, ADCK1, ADCK0 =010 (f_{SYS}/4)	ADCK2, ADCK1, ADCK0 =011 (f_{SYS}/8)	ADCK2, ADCK1, ADCK0 =100 (f_{SYS}/16)	ADCK2, ADCK1, ADCK0 =101 (f_{SYS}/32)	ADCK2, ADCK1, ADCK0 =110 (f_{SYS}/64)	ADCK2, ADCK1, ADCK0 =111
5MHz	200ns*	400ns*	800ns	1.6μs	3.2μs	6.4μs	12.8μs	Undefined
10MHz	100ns*	200ns*	400ns*	800ns	1.6μs	3.2μs	6.4μs	Undefined
20MHz	50ns*	100ns*	200ns*	400ns*	800ns	1.6μs	3.2μs	Undefined

A/D Clock Period Examples

Controlling the power on/off function of the A/D converter circuitry is implemented using the ADOFF bit in the ADCR0 register. This bit must be zero to power on the A/D converter. Even if no pins are selected for use as A/D inputs by clearing the PCR7~PCR0 bits in the ANCSR0 register and PCR8 in the ANCSR1 register, if the ADOFF bit is zero then some power will still be consumed. In power conscious applications it is therefore recommended that the ADOFF is set high to reduce power consumption when the A/D converter function is not being used.

The boundary register pairs, ADHVDL/ADHVDH and ADLVDL/ADLVDH contain preset values which can be compared with the A/D converted values in the ADRL/ADRH registers. Various types of comparisons can be made as defined by the ADCLVE and ADCHVE bits and an interrupt generated to inform the system that either the lower or higher boundary has been exceeded. This function can be used to ensure that the motor current operates within safe working limits.

A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins on Port A as well as other functions. The PCR7~PCR0 bits in the ANCSR0 register and PCR8 bit in the ANCSR1 register, determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the PCR8~PCR0 bits for its corresponding pin is set high then the pin will be setup to be an A/D converter input and the original pin functions disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the PAC or PBC port control registers to enable the A/D input as when the PCR8~PCR0 bits enable an A/D input, the status of the port control register will be overridden.

Summary of A/D Conversion Steps

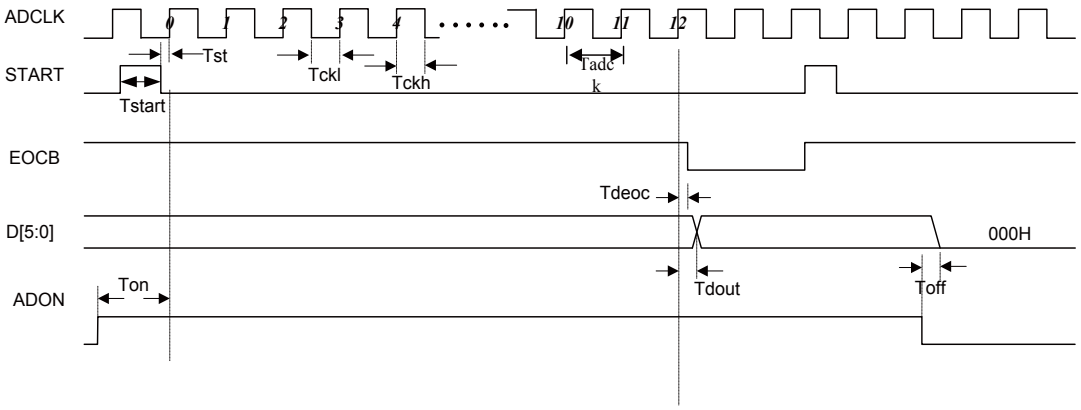
The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCK2~ADCK0 in the ADCR1 register.
- Step 2
  Enable the A/D by clearing the ADOFF bit in the ADCR0 register to zero.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS3~ACS0 bits which are also contained in the ADCR0 register.
- Step 4
  Select which pins are to be used as A/D inputs and configure them by correctly programming the PCR7~PCR0 bits in the ANCSR0 register and PCR8 in the ANCSR1.
- Step 5
  Select which trigger circuit is to be used by correctly programming the ADSTS bits in the ADCR1.
- Step 6
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bit, AEOCE, must both be set high to do this.
- Step 7
  If the step 5 selects ADSTR trigger circuit, the analog to digital conversion process can be initialised by setting the ADSTR bit in the ADCR0 register from low to high and then low again. Note that this bit should have been originally cleared to zero. If the step 5 selects PWM interrupt trigger Delay circuit, the Delay start function can be enabled by setting the DLSTR bit in the ADCR1 register.
- Step 8
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR0

register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data register ADRL and ADRH can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

Note: When checking for the end of the conversion process, if the method of polling the EOCB bit in the ADCR0 register is used, the interrupt enable step above can be omitted.

The accompanying diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is $16t_{ADCK}$ where t_{ADCK} is equal to the A/D clock period.



A/D Conversion Timing

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by setting bit ADOFF high in the ADCR0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Transfer Function

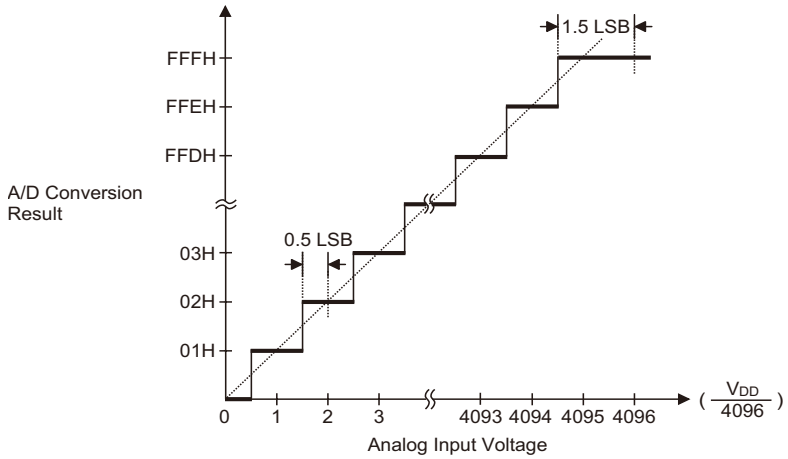
As the device contains a 10-bit A/D converter, its full-scale converted digitised value is equal to 3FFH. Since the full-scale analog input value is equal to the V_{DD} voltage, this gives a single bit analog input value of V_{DD} divided by 4096.

$$1\ LSB=V_{DD} \div 4096$$

The A/D Converter input voltage value can be calculated using the following equation:

$$A/D\ input\ voltage=A/D\ output\ digital\ value \times V_{DD} \div 4096$$

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the V_{DD} level.



Ideal A/D Transfer Function

A/D Programming Example

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the EOCB bit in the ADCR0 register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

Example: using an EOCB polling method to detect the end of conversion

```
clr  AEOCE                  ; disable ADC interrupt
mov  a,03H
mov  ADCR1,a                ; select fSYS/8 as A/D clock
clr  ADOFF
mov  a,0Fh                  ; setup ANCSR0 and ANCSR1 to configure pins AN0~AN3
mov  ANCSR0,a
mov  a,00h
mov  ANCSR1,a
mov  a,00h
mov  ADCR0                  ; enable and connect AN0 channel to A/D converter
:
start_conversion:
     clr ADSTR              ; high pulse on start bit to initiate conversion
     set ADSTR              ; reset A/D
     clr ADSTR              ; start A/D
polling_EOC:
     sz  EOCB               ; poll the ADCR0 register EOCB bit to detect end
                            ; of A/D conversion
     jmp polling_EOC        ; continue polling
     mov a,ADRL             ; read low byte conversion result value
     mov ADRL_buffer,a      ; save result to user defined register
     mov a,ADRH             ; read high byte conversion result value
     mov ADRH_buffer,a      ; save result to user defined register
:
:
jmp  start_conversion       ; start next a/d conversion
```

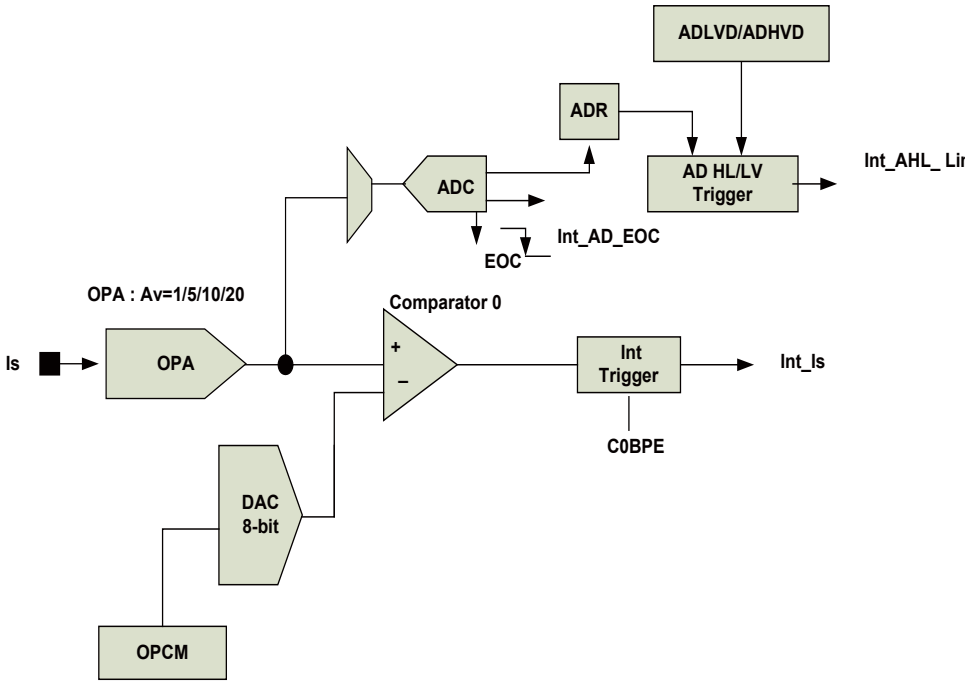
Example: using the interrupt method to detect the end of conversion

```
clr  MF1E                   ; disable ADC interrupt
CLR  AEOCE
mov  a,03H
```

```
mov  ADCR1,a                ; select fSYS/8 as A/D clock
Clr  ADOFF
mov  a,0Fh                  ; setup ANCSR0 and ANCSR1 to configure pins AN0~AN3
mov  ANCSR0,a
mov  a,00h
mov  ANCSR1,a
mov  a,00h
mov  ADCR0,a                ; enable and connect AN0 channel to A/D converter
Start_conversion:
     clr ADSTR              ; high pulse on START bit to initiate conversion
     set ADSTR              ; reset A/D
     clr ADSTR              ; start A/D
     clr AEOCF              ; clear ADC interrupt request flag
     set AEOCE              ; enable ADC interrupt
     set MF1E               ; enable Multi_interrupt 1
     set EMI                ; enable global interrupt
:
:
                            ; ADC interrupt service routine
ADC_ISR:
     mov acc_stack,a        ; save ACC to user defined memory
     mov a,STATUS
     mov status_stack,a     ; save STATUS to user defined memory
:
:
     mov a,ADRL             ; read low byte conversion result value
     mov adrl_buffer,a      ; save result to user defined register
     mov a,ADRH             ; read high byte conversion result value
     mov adrh_buffer,a      ; save result to user defined register
:
:
EXIT_INT_ISR:
     mov a,status_stack
     mov STATUS,a           ; restore STATUS from user defined memory
     mov a,acc_stack        ; restore ACC from user defined memory
     reti
```

Over-current Detection

The device contains an fully integrated over-current detect circuit which is used for motor protection.



Over-current Detector Block Diagram

Over-current Functional Description

The over-current functional block includes an amplifier, 10-bit A/D Converter, 8-bit D/A Converter and comparator. If an over-current situation is detected then the motor external drive circuit can be switched off immediately to prevent damage to the motor. Two kinds of interrupts are generated which can be used for over-current detection.

1. A/D Converter interrupt - Int_AHL_Lim
2. Comparator 0 interrupt - Int_Is

Over-current Register Description

There are three registers to control the function and operation of the over current detection circuits, known as OPOMS, OPCM and OPACAL. These 8-bit registers define functions such as the OPA operation mode selection, OPA calibration and comparison. OPCM is an 8-bit DAC register used for OPA comparison.

OPOMS Register

Bit	7	6	5	4	3	2	1	0
Name	CMP0_EG1	CMP0_EG0	—	—	—	OPAVS2	OPAVS1	OPAVS0
R/W	R/W	R/W	—	—	—	R/W	R/W	R/W
POR	0	0	—	—	—	0	1	0

Bit 7~6 **CMP0_EG1, CMP0_EG0:** Defines Comparator active edge
00: Disable Comparator 0 and DAC0
01: Rising edge
10: Falling edge
11: Dual edge

Bit 5~3 Unimplemented, read as "0"

Bit 2~0 **OPAVS2~OPAVS0:** OPA Av mode select
000: Disable OPA
001: Av=5
010: Av=10
011: Av=20
111: AV=1

OPCM 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 8-bit OPA comparison register bit 7 ~ bit 0

OPACAL Register

Bit	7	6	5	4	3	2	1	0
Name	—	ARS	AOFM	AOF4	AOF3	AOF2	AOF1	AOF0
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 Unimplemented, read as "0"

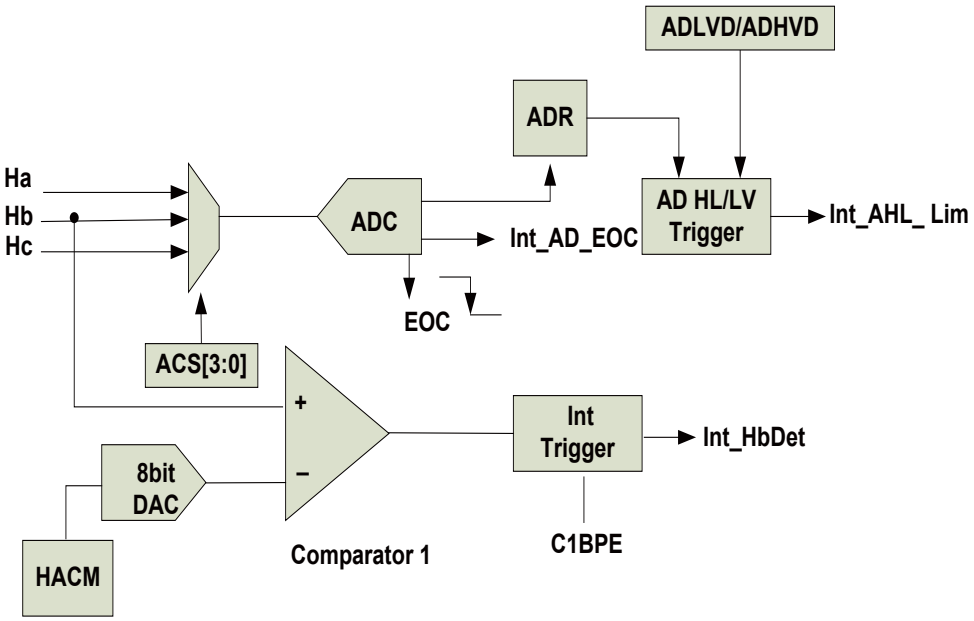
Bit 6 **ARS:** Comparator input offset calibration reference select
0: Comparator negative input
1: Comparator positive input

Bit 5 **AOFM:** Normal or Calibration Mode select
0: Opamp or Comparator Mode
1: Offset Calibration Mode

Bit 4~0 **AOF4~AOF0:** Comparator input offset voltage calibration control
00000: Minimum
10000: Center
11111: Maximum

Linear Hall Sensor Detection

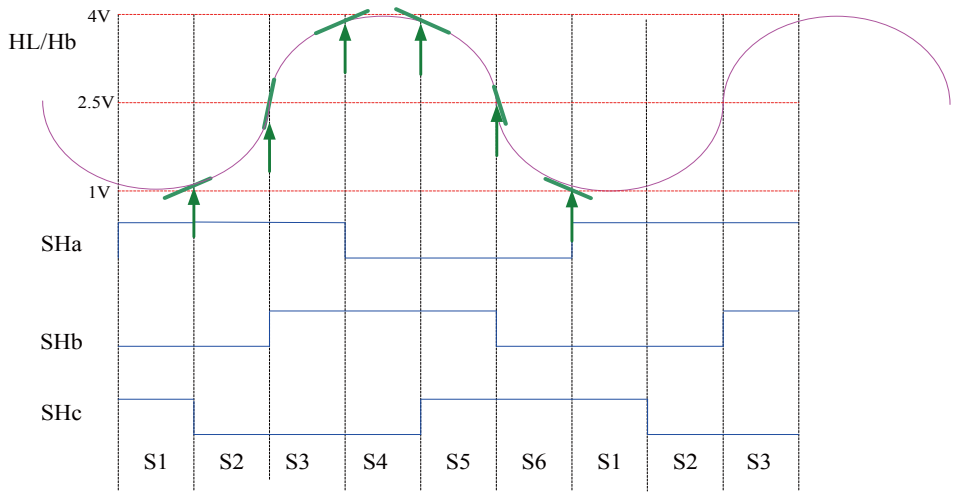
The motor position is detected using Hall Sensors for which the device includes circuitry to process signals from these sensors.



Hall Sensor Detector Block Diagram

Hall Sensor Detection Function Description

The signals from the external linear Hall Sensors are monitored using the internal 8-bit DAC, the internal 10-bit ADC and internal comparator 1. The motor position is monitored by two interrupts Int_HbDet or Int_AHL_Lim which are enabled/disabled by the LHMC and HACM registers. The six step rotational change of state of the motor position can be tracked by setting the DAC data and the LHMC register, to control the motor direction and speed, as shown in the figure.



linear Hall Sensor Output

Linear Hall Sensor Control Register Description

The LHMC is the linear Hall sensor Mode control register and the HACM is the 8-bit DAC register for Linear Hall Sensor comparison.

LHMC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	CMP1_EG1	CMP1_EG0	—	—	C1BPE	C0BPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **CMP1_EG1, CMP1_EG0:** Defines Comparator active edge
00: Disable Comparator 1 and DAC1
01: Rising edge
10: Falling edge
11: Dual edge

Bit 3~2 Unimplemented, read as "0"

Bit 1 **C1BPE:** Comparator 1 Interrupt Bypass(test option)
0: Disable Comparator 1 interrupt
1: Enable Comparator 1 interrupt

Bit 0 **C0BPE:** Comparator 0 Interrupt Bypass(test option)
0: Disable Comparator 0 interrupt
1: Enable Comparator 0 interrupt

HACM 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 8-bit Linear Hall Sensor comparison register bit 7 ~ bit 0

BLDC Motor Control Circuit

This sections describes how the device can be used to control Brushless DC Motors, otherwise known as BLDC Motors. Its high level of functional integration and flexibility offer a full range of driving features for motor driving.

Functional Description

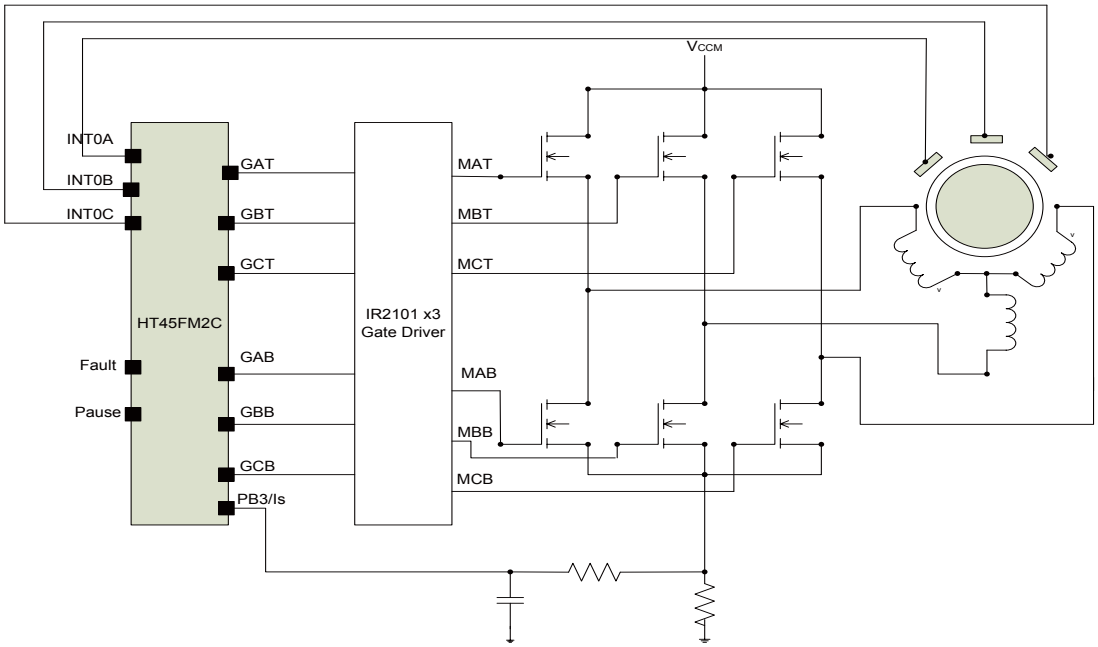
The PWM counter circuit output PWMO is has an adjustable PWM Duty to control the output motor power thus controlling the motor speed. Changing the PWM frequency can be used to enhance the motor drive efficiency or to reduce noise and resonance generated during physical motor operation.

The internal Mask circuit is used to determine which PWM modulation signals are enabled or disabled for the motor speed control. The PWM modulation signal can be output both the upper arms, GAT/GBT/GCT and the lower arms, GAB/GBB/GCB, of the external Gate Driver Transistor Pairs under software control. The Dead-Time insertion circuit is used to ensure the upper and lower Gate Driver Transistor Pairs are not enabled simultaneously to prevent the occurrence of a virtual power short circuit. The dead time is selected under software control.

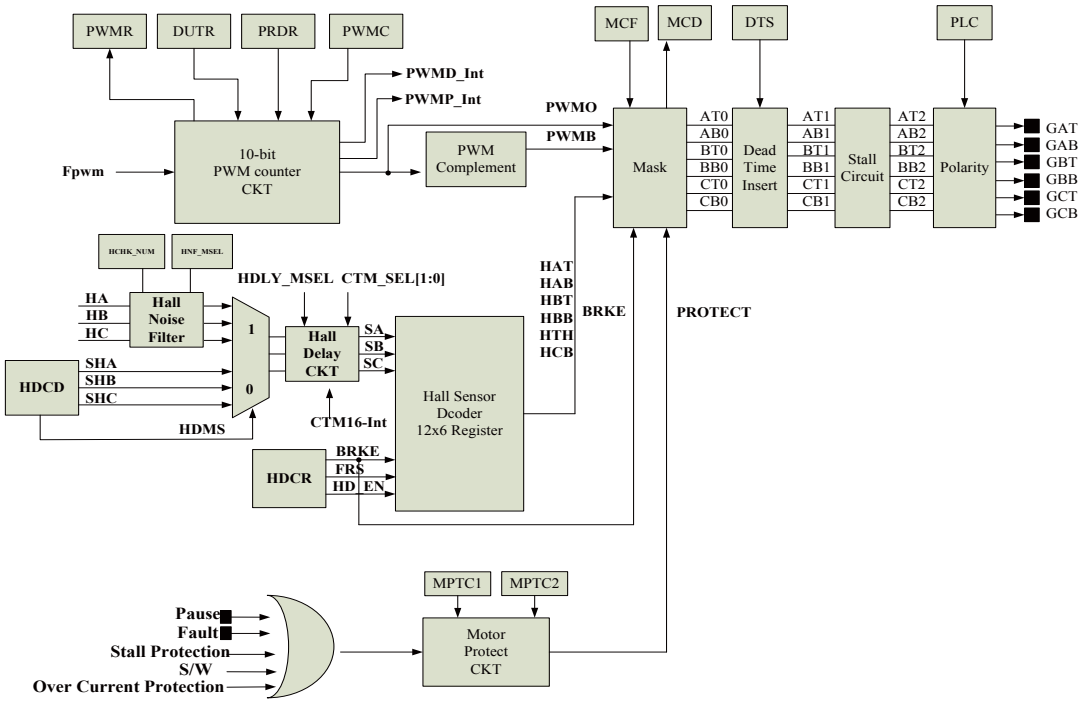
The Staggered circuit can force all the outputs to an off status if the software detects an error condition which could be due to external factors such as ESD problems or both upper and lower external Gate Driver Transistor pairs being simultaneously on. The Polarity circuit can select the output polarity of the BLDC motor output control port to support many different types of external MOS gate drive device circuit combinations.

The Motor Protect circuit includes many detection circuits for functions such as a motor stall condition, over current protection, external edge triggered Pause pin, external level trigger Fault pin etc. The Hall Sensor Decoder circuit is a six-step system which can be used control the motor direction.

Twelve registers, each using 6 bits, are used to control the direction of the motor. The motor forward, backward, brake and free functions are controlled by the HDCD/HDCR registers. The Ha/Hb/Hc or SHA/SHB/SHC can be selected as the Hall Sensor Decoder circuit inputs.



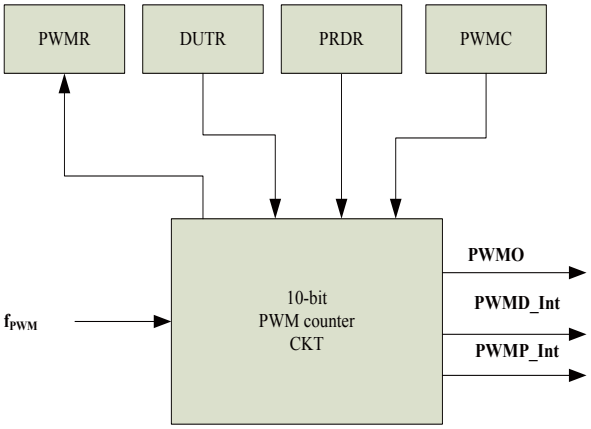
BLDC Application Circuit



BLDC Motor Control Block Diagram

PWM Counter Control Circuit

The device includes a 10-bit PWM generator. The PWM signal has both adjustable duty cycle and frequency that can be setup by programming 10-bit values into the corresponding PWM registers.



PWM Block Diagram

PWM Register Description

Overall PWM operation is controlled by a series of registers. The DUTRL/DUTRH register pair is used for PWM duty control for adjustment of the motor output power. The PRDRL/PRDRH register pair are used together to form a 10-bit value to setup the PWM period for PWM Frequency adjustment. Being able to change the PWM frequency is useful for motor characteristic matching for problems such as noise reduction and resonance. The PWMRL/PWMRH registers are used to monitor the PWM counter dynamically. The PWMON bit in the PWMC register is the 10-bit PWM counter on/off bit. The PWM clock source for the PWM counter can be selected by PCKS1~PCKS0 bits in the PWMC register. It should be noted that the order of writing data to PWM register is MSB.

PWMC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PCKS1	PCKS0	PWMON	—	—	GATSEL
R/W	—	—	R/W	R/W	R/W	—	—	R/W
POR	—	—	0	0	0	—	—	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PCKS1, PCKS0:** Clock source of the PWM counter select
000: f_{PWM}, PWM frequency Min.=20kHz, f_{PWM} base on 20MHz
001: $f_{PWM}/2$, PWM frequency Min.=10kHz
010: $f_{PWM}/4$, PWM frequency Min.=5kHz
011: $f_{PWM}/8$, PWM frequency Min.=2.5kHz

Bit 3 **PWMON:** PWM Circuit On/Off control
0: Off
1: On
This bit controls the overall on/off function of the PWM. Setting the bit high enables the counter to run, clearing the bit disables the PWM. Clearing this bit to zero will stop the counter from counting and turn off the PWM which will reduce its power consumption.

Bit 2~1 Unimplemented, read as "0"

Bit 0 **GATSEL:** GATE Driver output select
0: GAT/GAB/GBT/GBB/GCT/GCB are used for Gate driver output pins
1: GAT/GAB/GBT/GBB/GCT/GCB are used as PC[5:0]

DUTRL 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 10-bit PWM Duty register low byte register bit 7 ~ bit 0
10-bit DUTR register bit 7 ~ bit 0

DUTRH Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 10-bit PWM Duty register high byte register bit 1 ~ bit 0
10-bit DUTR register bit 9 ~ bit 8

PRDRL 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 10-bit PWM Period register low byte register bit 7~bit 0
10-bit PRDR register bit 7 ~ bit 0

PRDRH Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 PWM Period high byte register Bit 1~Bit 0
10-bit DUTR register Bit 9 ~ Bit 8

PWMRL Register

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

Bit 7~0 10-bit PWM Counter register low byte register Bit 7~Bit 0
10-bit PWM Counter Bit 7 ~ Bit 0

PWMRH Register

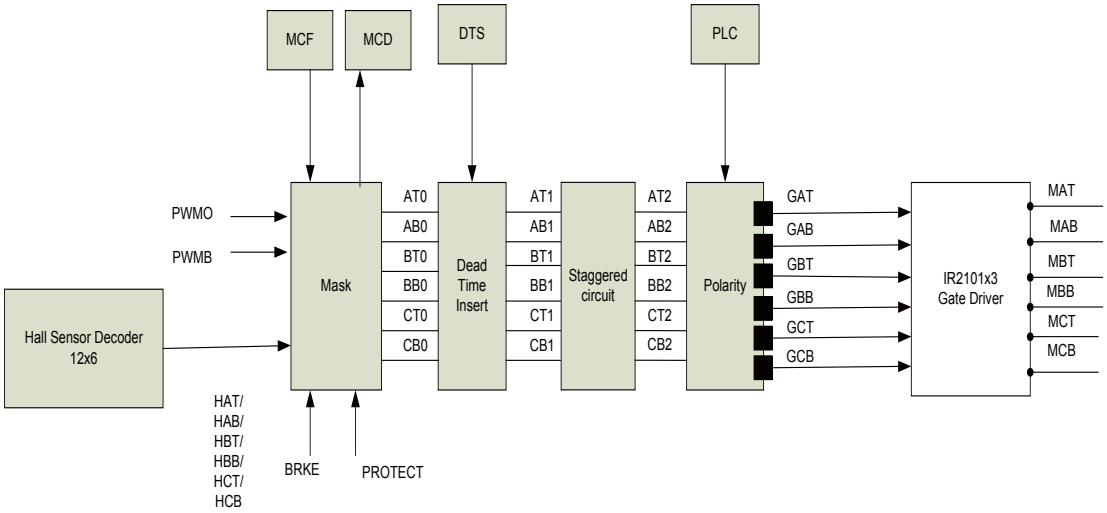
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

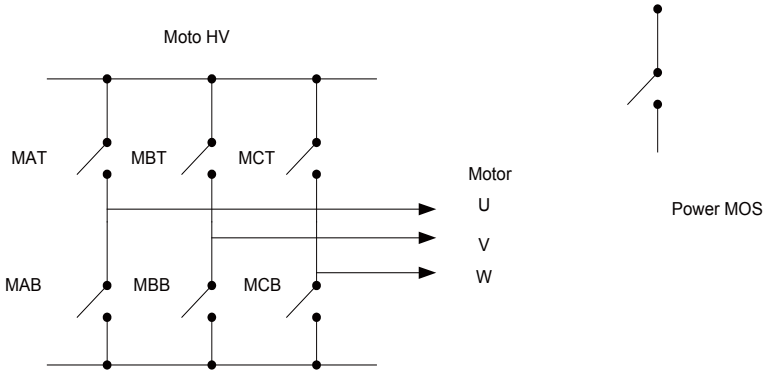
Bit 1~0 10-bit PWM Counter register high byte register Bit 1 ~ Bit 0
PWM 10-bit Counter Bit 9 ~ Bit 8

Mask Function

The device includes a Motor Control Mask Function for increased control flexibility.



Mask Function Block Diagram



Mask Switching

Functional Description

The internal MASK circuit has three operation modes, which are known as the Normal Mode, Brake Mode and Motor Protect Mode.

- Normal Mode

In the Normal Mode, the motor speed control method is determined by the PWMS/MPWE bits in the MCF register.

When PWMS =0, the bottom port PWM output selects transistor pair bottom arm GAB/ GBB/ GCB.

When PWMS =1, the top port PWM output selects transistor pair top arm, GAB/ GBB/ GCB.

When MPWE =0, the PWM output is disabled and AT0/BT0/CT0/AB0/BB0/CB0 are all on.

When MPWE =1, the PWM output is enabled and AT0/BT0/CT0/AB0/BB0/CB0 can output a variable PWM signal for speed control.

When MPWMS=0, the PWM has a Complementary output

When MPWMS=1, the PWM has a Non-complementary output

Complementary control, MPWMS=0

	HAT	HAB	AT0	AB0		HAT	HAB	AT0	AB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	PWMB	PWMO	PWMS=1	0	1	0	1
	1	0	1	0		1	0	PWMO	PWMB
	1	1	0	0		1	1	0	0

	HBT	HBB	BT0	BB0		HBT	HBB	BT0	BB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	PWMB	PWMO	PWMS=1	0	1	0	1
	1	0	1	0		1	0	PWMO	PWMB
	1	1	0	0		1	1	0	0

	HCT	HCB	CT0	CB0		HCT	HCB	CT0	CB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	PWMB	PWMO	PWMS=1	0	1	0	1
	1	0	1	0		1	0	PWMO	PWMB
	1	1	0	0		1	1	0	0

Non-complementary control, MPWMS=1

	HAT	HAB	AT0	AB0		HAT	HAB	AT0	AB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	0	PWMO	PWMS=1	0	1	0	1
	1	0	1	0		1	0	PWMO	0
	1	1	0	0		1	1	0	0

	HBT	HBB	BT0	BB0		HBT	HBB	BT0	BB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	0	PWMO	PWMS=1	0	1	0	1
	1	0	1	0		1	0	PWMO	0
	1	1	0	0		1	1	0	0

	HCT	HCB	CT0	CB0		HCT	HCB	CT0	CB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	0	PWMO	PWMS=1	0	1	0	1
	1	0	1	0		1	0	PWMO	0
	1	1	0	0		1	1	0	0

- Brake Mode

The Brake Mode has the highest priority. When activated, the external Gate Driver Transistor Pair Top arm will be off and the Bottom arm will be on. The Brake Truth decode table is shown below.

	AT0	BT0	CT0	AB0	BB0	CB0	1	0
BRKE=1	0	0	0	1	1	1	D9	D8

- Motor Protect Mode

When the Motor Protect Mode is activated, the external Gate Driver Transistor Pair can select the brake, where the top arm is off and the bottom arm is on, or select free running where the top and bottom arm are both off. The protection decode table is shown below.

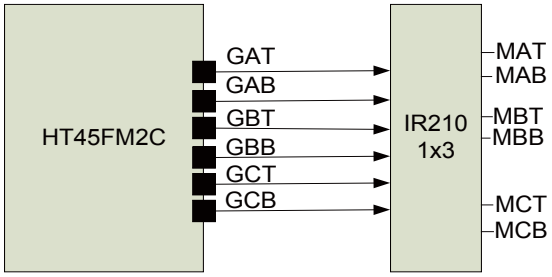
PROTECT =1	GAT	GBT	GCT	GAB	GBB	GCB
FMOS=0	0	0	0	0	0	0
FMOS=1	0	0	0	1	1	1

For 6-Step communication, if the U winding and W winding are on then turn off the V winding.

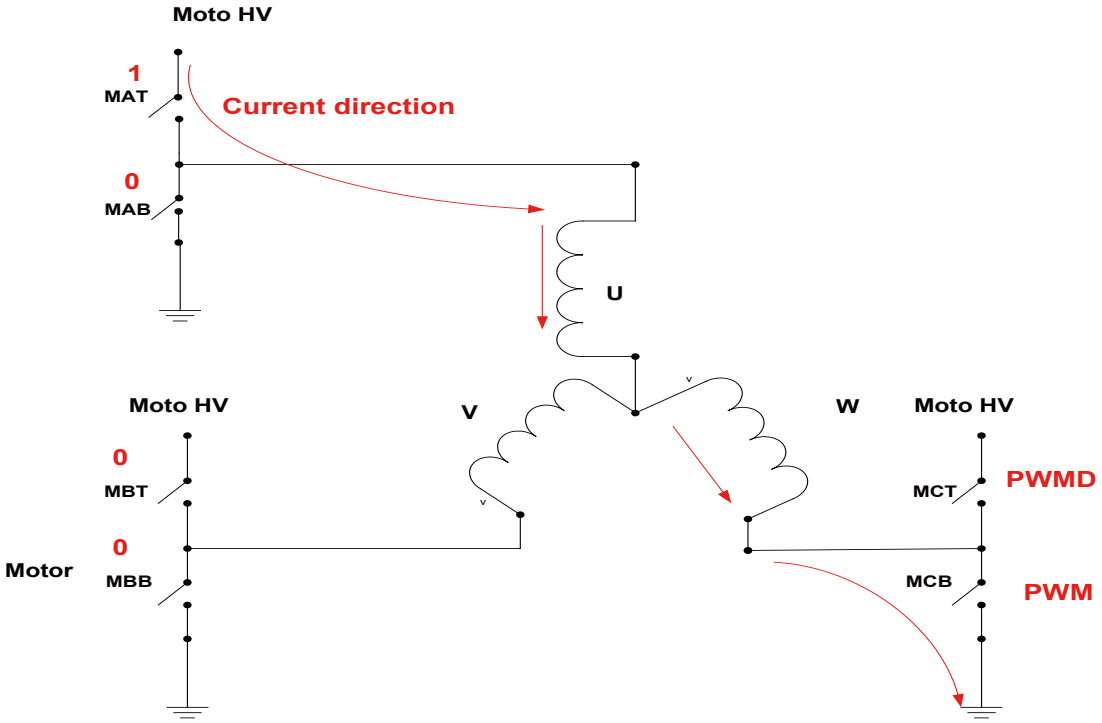
If GAT =1 and GAB =0, turn on the U winding

If GBT =0 and GBB =0, turn off the V Winding.

If GCT=PWMD and GCB=PWM, turn on the W winding and adjust the output power of the motor using the DUTR register to control the speed.



Drive Signal Block Diagram



Motor Winding Connection

Register Description

The device has two registers connected with the Mask Function control. These are the MCF register which is used for control and the MCD register which is used to read the status of the gate driver outputs.

MCF Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	MPWMS	MPWE	FMOS	PWMS
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	1	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **MPWMS:** Mask PWM Mode select
0: Complementary
1: Non-complementary

Bit 2 **MPWE:** PWM output control
0: PWM output disable (AT0/BT0/CT0/AB0/BB0/CB0 can not output PWM)
1: PWM output enable (AT0/BT0/CT0/AB0/BB0/CB0 can output PWM to control speed)

Bit 1 **FMOS:** Fault Mask output select
0: AT0/BT0/CT0=0, AB0/BB0/CB0=0
1: AT0/BT0/CT0=0, AB0/BB0/CB0=1

Bit 0 **PWMS:** Top port/Bottom port PWM select
0: Select Bottom port PWM output
1: Select Top port PWM output

MCD Register

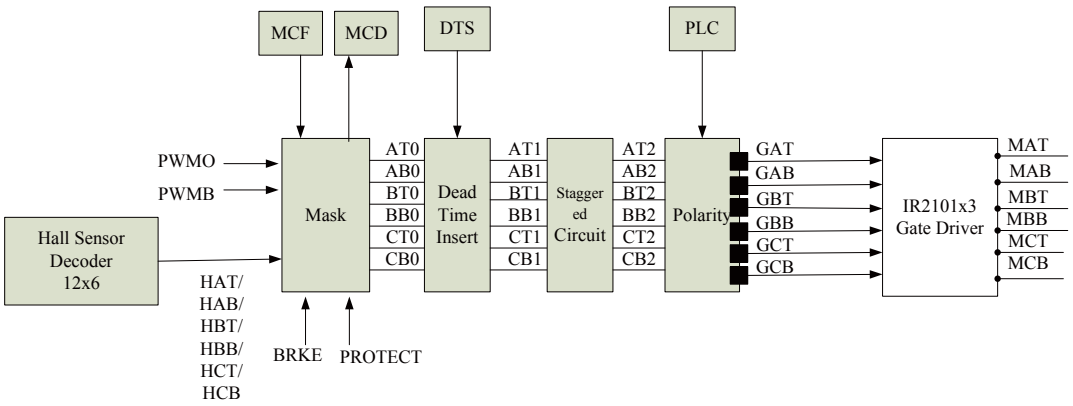
Bit	7	6	5	4	3	2	1	0
Name	—	—	GAT	GAB	GBT	GBB	GCT	GCB
R/W	—	—	R	R	R	R	R	R
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **GAT/GAB/GBT/GBB/GCT/GCB:** Gate diver output monitor

Other Functions

Several other functions exist for additional motor control drive signal flexibility. These are the Dead Time Function, Staggered Function and Polarity Function.



Dead Time, Staggered and Polarity Function Block Diagram

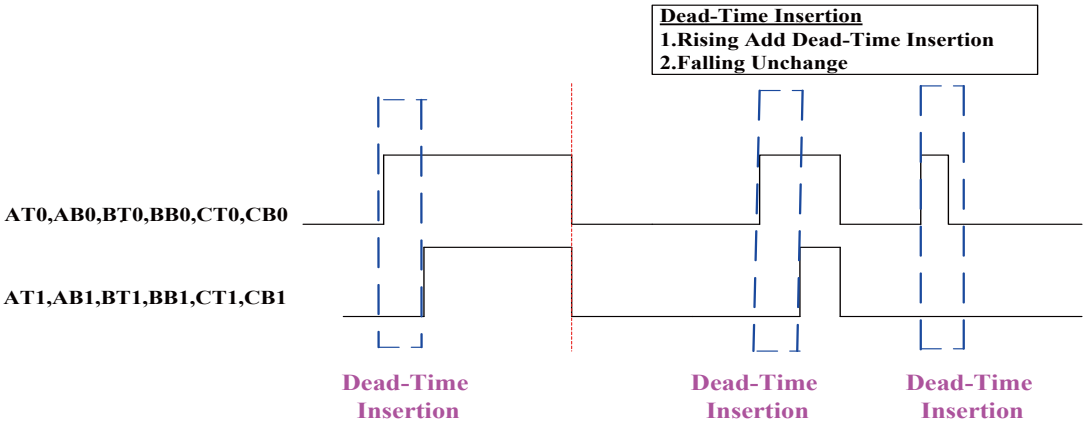
Dead Time Function

During transistor pair switching, the Dead Time function is used to prevent both upper and lower transistor pairs from conducting at the same time thus preventing a virtual short circuit condition from occurring. The actual dead time value can be setup to be within a value from 0.3μs to 5μs which is selected by the application program.

The Dead Time Insertion circuit requires six independent output circuits:

- When the AT0/AB0/BT0/BB0/CT0/CB0 outputs experience a rising edge, then a Dead Time is inserted.
- When the AT0/AB0/BT0/BB0/CT0/CB0 outputs experience a falling edge, then the outputs remain unchanged.

The Dead-Time Insertion Circuit is only during motor control. The Dead Time function is enabled/ disabled by the DTE bit in the DTS register.



Dead Time Insertion Timing

A single register, DTS, is dedicated for use by the Dead Time function.

DTS Register

Bit	7	6	5	4	3	2	1	0
Name	DTCKS1	DTCKS0	DTE	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 7~6 **DTCKS1, DTCKS0**: Dead-Time clock source selection
00: f_{DT} is f_{SYS}, f_{SYS} based on 20MHz
01: f_{DT} is $f_{SYS}/2$
10: f_{DT} is $f_{SYS}/4$
11: f_{DT} is $f_{SYS}/8$

Bit 5 **DTE:** Dead-Time Enable
0: Dead-Time=0
1: Dead-Time = (DTS[4:0]+1)/f_{DT}

Bit 4~0 **D4~D0:** Dead-Time Register bit 4 ~ bit 0
Dead-Time counter. 5-bit Dead-Time value bits for Dead-Time Unit.
Dead-Time = (DTS[4:0]+1)/f_{DT}

Staggered Function

The Staggered Function is used to force all output drive transistors to an off condition when a software error occurs or due to external factors such as ESD.

AT1	AB1	AT2	AB2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

The default condition for the BLDC motor control circuit is designed for default N-type transistor pairs. This means a “1” value will switch the transistor on and a “0”value will switch it off.

Polarity Function

This function allows setup of the external gate drive transistor On/Off polarity status. A single register, PLC, is used for overall control.

PLC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PCBC	PCTC	PBBC	PBTC	PABC	PATC
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 **PCBC:** C pair Bottom port Gate output inverse control

Bit 4 **PCTC:** C pair Top port Gate output inverse control

Bit 3 **PBBC:** B pair Bottom port Gate output inverse control

Bit 2 **PBTC:** B pair Top port Gate output inverse control

Bit 1 **PABC:** A pair Bottom port Gate output inverse control

Bit 0 **PATC:** A pair Top port Gate output inverse control

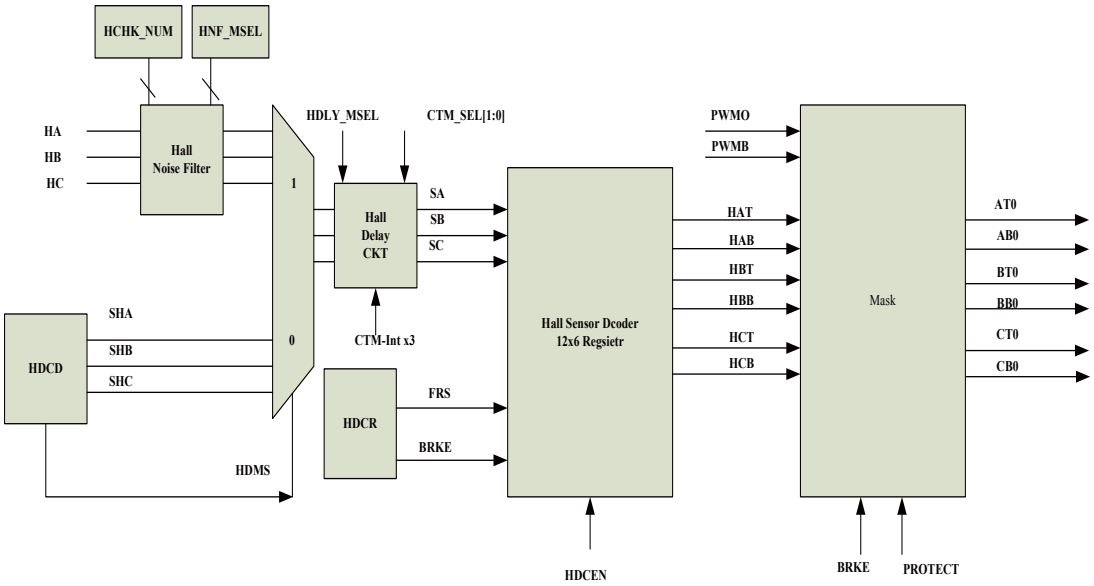
Bit Value	Status
0	Output not inverted
1	Output inverted

PLC Register Values

Note that the default output pin GAT/GAB/GBT/GBB/GCT/GCB status is high impedance.

Hall Sensor Decoder

This device contains a fully integrated Hall Sensor decoder function which interfaces to the Hall Sensors in the BLDC motor for directional and speed control.



Hall Sensor Decoder Block Diagram

The Hall Sensor input signals are selected by setting the HDMS bit high. If the HDMS bit is zero then SHA/SHB/SHC will be used instead of the actual Hall Sensor signals.

Hall Sensor Noise Filter

This device includes a Hall Noise Filter function to filter out the effects of noise generated by the large switching currents of the motor driver. This generated noise may affect the Hall Sensor inputs (HA/HB/HC), which in turn may result in incorrect Hall Sensor output decoding.

Several registers are used to control the noise filter. The HNF_EN bit in the HNF_MSEL register is used as the overall enable/disable bit for the noise filter.

HNF_EN bit	Status
0	Noise Filter Off – HA/HB/HC not used
1	Noise Filter On

Hall Sensor Noise Filter Enable

The sampling frequency of the Hall noise filter is setup using the HFR_SEL [3:0] bits.

The HCK_NUM [4:0] bits are used to setup the Hall Sensor input compare numbers.

HCK_NUM [4:0] × Sampling space = Anti-noise ability = Hall Delay-Time.

It should be noted that longer Hall delay times will result in higher rotor speed feedback signal distortion.

Hall Sensor Delay Function

The Hall sensor function in the device includes a Hall delay function which can implement a signal phase forward or phase backward operation. The following steps, which should be executed before the Hall Decoder is enabled, show how this function is activated.

- Step 1

Set the Hall Decode table to select either the phase forward or phase backward function.

- Step 2

Select which CTM is used to generate the Delay Time and set the selected CTM to run in the Compare Match Mode by programming the CTM_SEL1~CTM_SEL0 bits.

- Step 3

Use the HDLY_MSEL bit to select the Hall Delay circuit operating mode. The default value of HDLY_MSEL is zero which will disable the Hall Delay circuit. If the HDLY_MSEL bit is set high, then the Hall Delay circuit will be enabled.

- Step 4

Enable the Hall Decoder using the HDCEN bit.

The following points should be noted regarding the HDLY_MSEL bit.

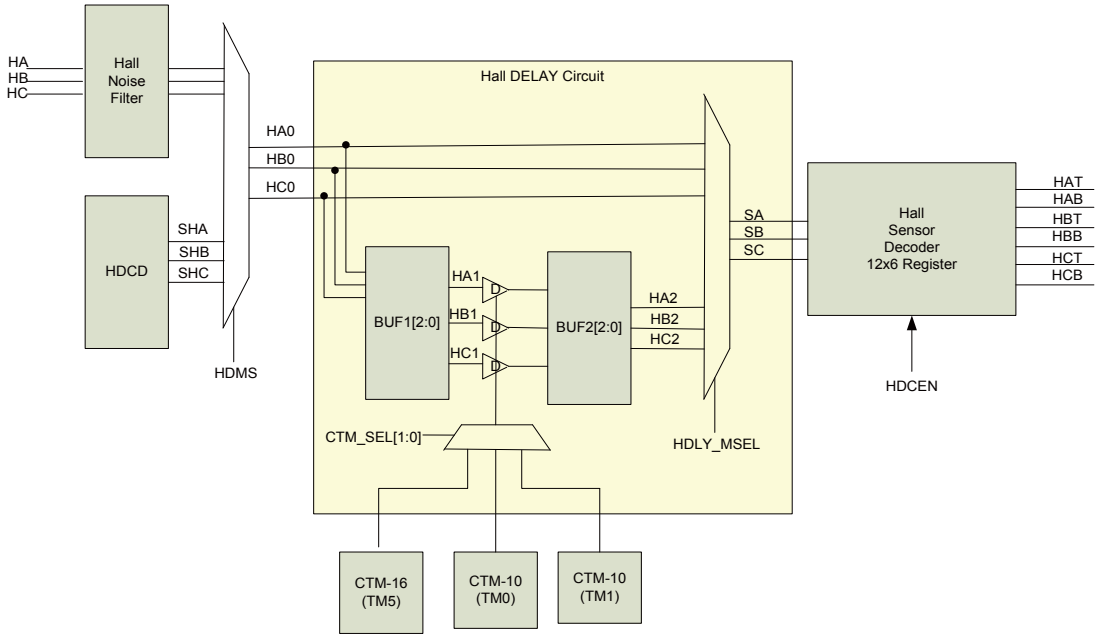
- When this bit is low, BUF1[2:0] and BUF2[2:0] will be cleared to zero.
- When this bit is low, TM0/TM1/TM5 retain their original TM functions.
- When the bit is high, the CTM which is selected by the Delay function will be dedicated for use by the Hall Delay circuit.

The original TM functions will still remain active except for the TnON bit which will be controlled automatically by the hardware.

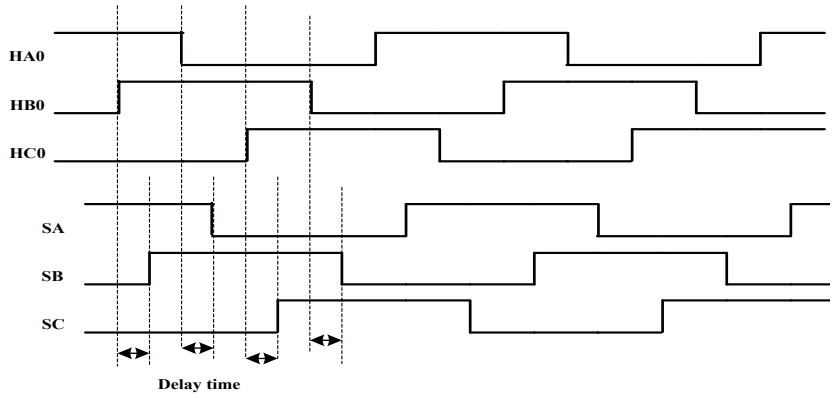
With regard to the TM functions the following steps should be taken before the Delay function is enabled.

1. Keep TnON and TnPAU = 0
2. The TM should be setup in the Compare Match Mode
3. TnCCLR=1, therefore the TM is cleared with a comparator A match condition.
4. Setup the Delay time using TMnA and TnCKx.

After the Delay function is enabled, HDLY_MSEL will change from low to high. The Delay time must not be more than one step time of the Hall input, which has six steps, Otherwise the output can not be anticipated, will drop out of step.



Delay Function Block Diagram



Delay Function Timing

Motor Control Drive Signals

The direction of the BLDC motor is controlled using the HDCR, HDCD registers and a set of six HDCT registers, HDCT0~HDCT11. When using the Hall Sensor Decoder function, the direction can be determined using the FRS bit and the brake can be controlled using the BRKE bit. Both bits are in the HDCR register. Six bits in the HDCT0~HDCT5 registers are used for the Motor Forward table, and six bits in the HDCT6~HDCT11 registers are used for the Motor Backward table. The accompanying tables show the truth tables for each of the registers.

	60 degree			120 degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
Forward (HDCEN=1, FRS=0, BRKE=0)	1	0	0	1	0	0	HDCT0[5:0]					
	1	1	0	1	1	0	HDCT1[5:0]					
	1	1	1	0	1	0	HDCT2[5:0]					
	0	1	1	0	1	1	HDCT3[5:0]					
	0	0	1	0	0	1	HDCT4[5:0]					
	0	0	0	1	0	1	HDCT5[5:0]					

Hall Sensor Decoder Forward Truth Table

	60 degree			120 degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
Backward (HDCEN=1, FRS=1, BRKE=0)	1	0	0	1	0	0	HDCT6[5:0]					
	1	1	0	1	1	0	HDCT7[5:0]					
	1	1	1	0	1	0	HDCT8[5:0]					
	0	1	1	0	1	1	HDCT9[5:0]					
	0	0	1	0	0	1	HDCT10[5:0]					
	0	0	0	1	0	1	HDCT11[5:0]					

Hall Sensor Decoder Backward Truth Table

The truth tables for the brake function, hall decoder disable function and hall decoder error function are also shown below.

	60 degree			120 degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
Brake (BRKE=1, HDCEN=X, FRS=X)	V	V	V	V	V	V	0	1	0	1	0	1

Brake Truth Table

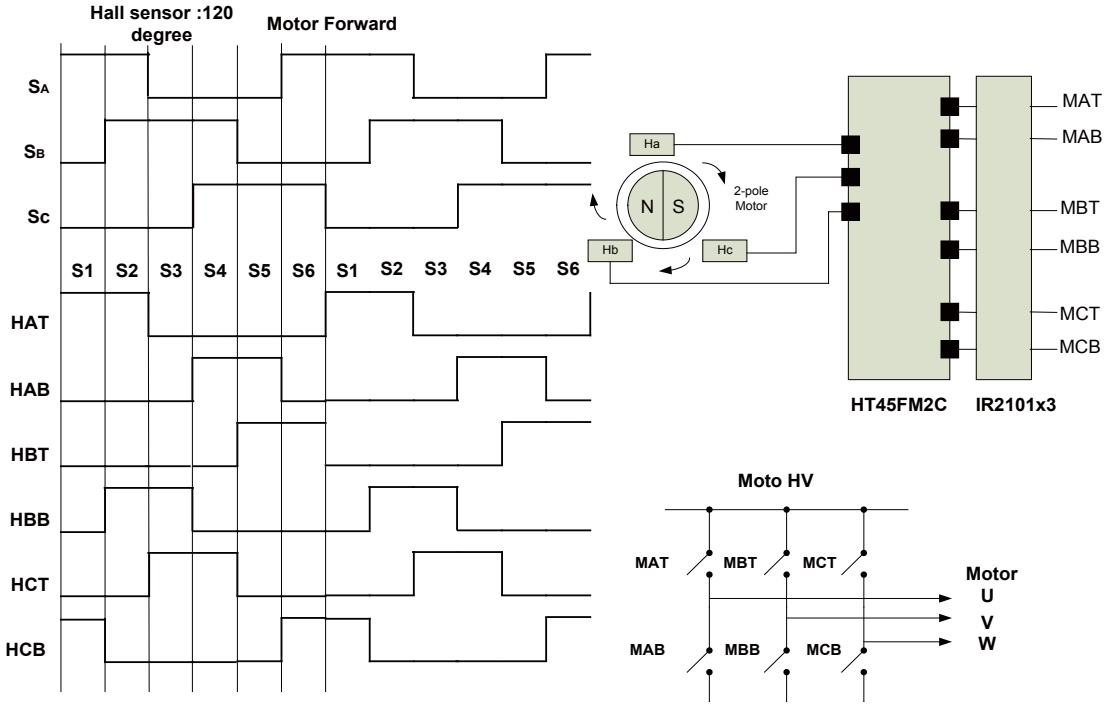
	60 degree			120 degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
Hall Decoder disable (HDCEN=0)	V	V	V	V	V	V	0	0	0	0	0	0

Hall Decoder Disable Truth Table

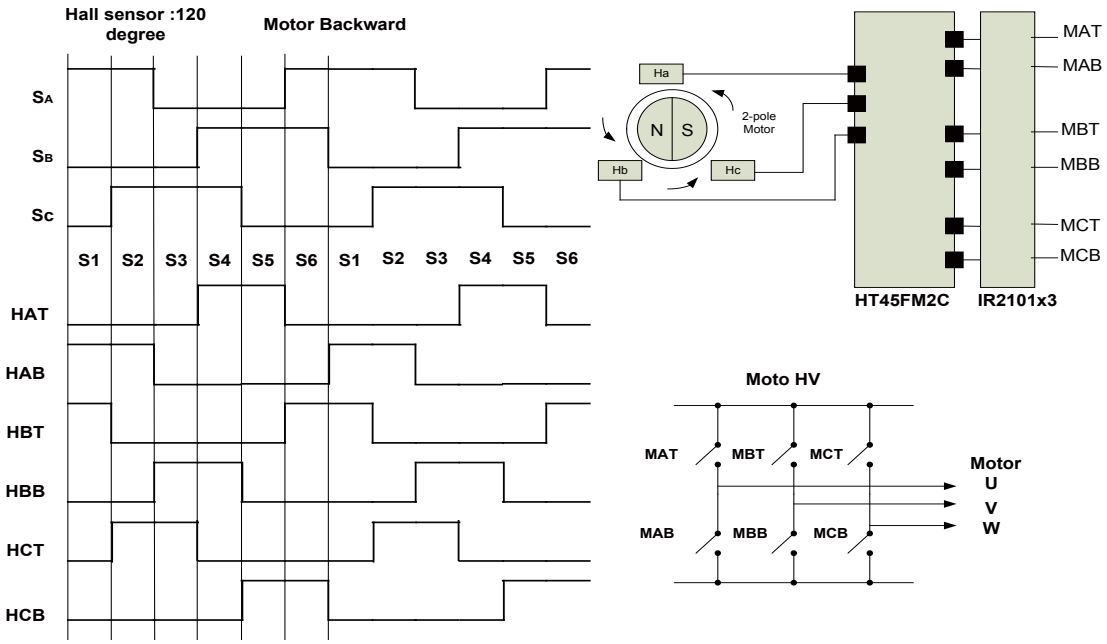
	60 degree			120 degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
Hall Decoder error (HDCEN=X)	1	0	1	1	1	1	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0

Hall Decoder Error Truth Table

The relationship between the data in the truth tables and how they relate to actual motor drive signals is shown in the accompanying timing diagram. The full 6 step cycle for both forward and backward motor rotation is provided.



Motor Drive Signal Timing Diagram - Forward Direction



Motor Drive Signal Timing Diagram - Backward Direction

Hall Sensor Decoder Register Description

The HDCR register is the Hall Sensor Decoder control register, HDCD is the Hall Sensor Decoder input data register, and HDCT0~HDCT11 are the Hall Sensor Decoder tables. The HCHK_NUM register is the Hall Noise Filter check number register and HNF_MSEL is the Hall Noise Filter Mode select register

HDCR Register

Bit	7	6	5	4	3	2	1	0
Name	CTM_SEL1	CTM_SEL0	HDLY_MSEL	HALS	HDMS	BRKE	FRS	HDCEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7~6 **CTM_SEL1~CTM_SEL0:** CTM Timer select of the Hall Delay Circuit
00:TM5(16-bit CTM)
01:TM0(10-bit CTM)
10:TM1(10-bit CTM)
11:Unused

Bit 5 **HDLY_MSEL:** Hall Delay Circuit select
0: Select original path
1: Select Hall Delay Circuit

Bit 4 **HALS:** Hall Sensor Decoder Mode select
0: Hall Sensor 60 degree
1: Hall Sensor 120 degree

Bit 3 **HDMS:** Hall Sensor Decoder Mode select
0: S/W Mode
1: Hall Sensor Mode

Bit 2 **BRKE:** motor brake control
0: GAT/GBT/GCT/GAB/GBB/GCB=V
1: GAT/GBT/GCT=0, GAB/GBB/GCB=1

Bit 1 **FRS:** Motor Forward/Backward select
0: Forward
1: Backward

Bit 0 **HDCEN:** Hall Sensor Decoder enable
0: Disable
1: Enable

HDCD Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	SHC	SHB	SHA
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **SHC:** S/W Hall C

Bit 1 **SHB:** S/W Hall B

Bit 0 **SHA:** S/W Hall A

HDCT11~0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	HATD	HABD	HBTD	HBBD	HCTD	HCBD
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 **HATD:** GAT output state control
Bit 4 **HABD:** GAB output state control
Bit 3 **HBTD:** GBT output state control
Bit 2 **HBBD:** GBB output state control
Bit 1 **HCTD:** GCT output state control
Bit 0 **HCBD:** GCB output state control

Bit Value	Status
0	Output is low
1	Output is high

Output Status

HCHK_NUM Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	HCK_N4	HCK_N3	HCK_N2	HCK_N1	HCK_N0
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 **HCK_N4~HCK_N0:** Hall Noise Filter check number

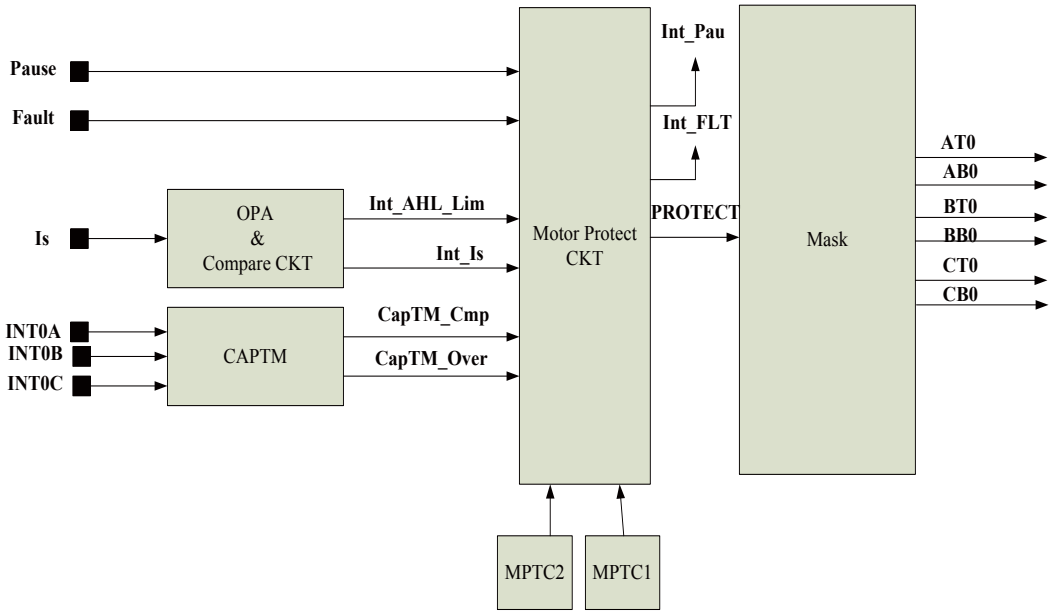
HNF_MSEL Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HNF_EN	HFR_SEL2	HFR_SEL1	HFR_SEL0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

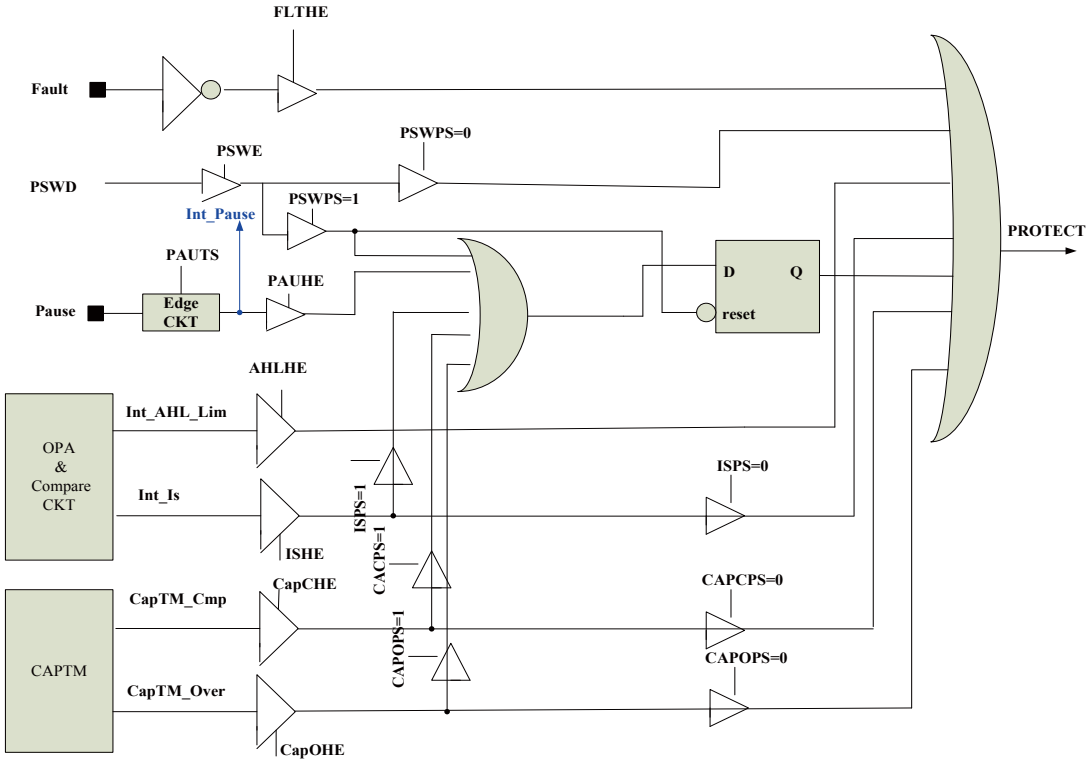
Bit 7~4 Unimplemented, read as "0"
Bit 3 **HNF_EN:** Hall noise filter enable
0: Disable(bypass)
1: Enable
Bit 2~0 **HFR_SEL2~HFR_SEL0:** Hall noise filter clock source select
000:f_{SYS}/2
001:f_{SYS}/4
010:f_{SYS}/8
011:f_{SYS}/16
100:f_{SYS}/32
101:f_{SYS}/64
110:f_{SYS}/128
111:Unused

Motor Protection Function

Motors normally require large currents for their operation and as such need to be protected from the problems of excessive drive currents, motor stalling etc to reduce motor damage or for safety reasons. This device includes a range of protection and safety features.



Protection Function Block Diagram



Protection Function Control

Motor Protection Function Description

This device provides five kinds of protection features, allowing action to be taken to protect the motor from damage or to provide additional safety.

The protection features are:

1. An external edge trigger on the Pause pin - edge trigger
2. An external level trigger on Fault pin - level trigger
3. Stall detection function
4. Over current protection
5. Turn off the motor using software

When the motor protection circuit is on, the external Gate Drive transistor pair can be put into two different protection modes. The first is the Brake Mode which is where the top arm is off and the bottom arm is on, and the second is the Free Running Mode where both top and bottom arms are off. The FMOS bit in the MCF register determines which type is used.

The motor protection circuit operates in two modes, which is selected by the MPTC2 register. One mode is the Fault Mode and the other is Pause Mode. In the Fault Mode, activating the protect function is determined by the trigger source starting status. Ending the protect function is determined by the trigger source disarming status. In the Pause Mode, turning on the protect function is determined by the trigger source. Ending the protection function is determined by software.

Fault Pin Function

The Fault Pin is used to detect whether an external circuit has detected a motor stall or over current condition. The pin is a level trigger type and is active low and PROTECT is "1". The Fault pin and PROTECT are controlled by FLTHE bit in the MPTC1 register.

Pause Pin Function

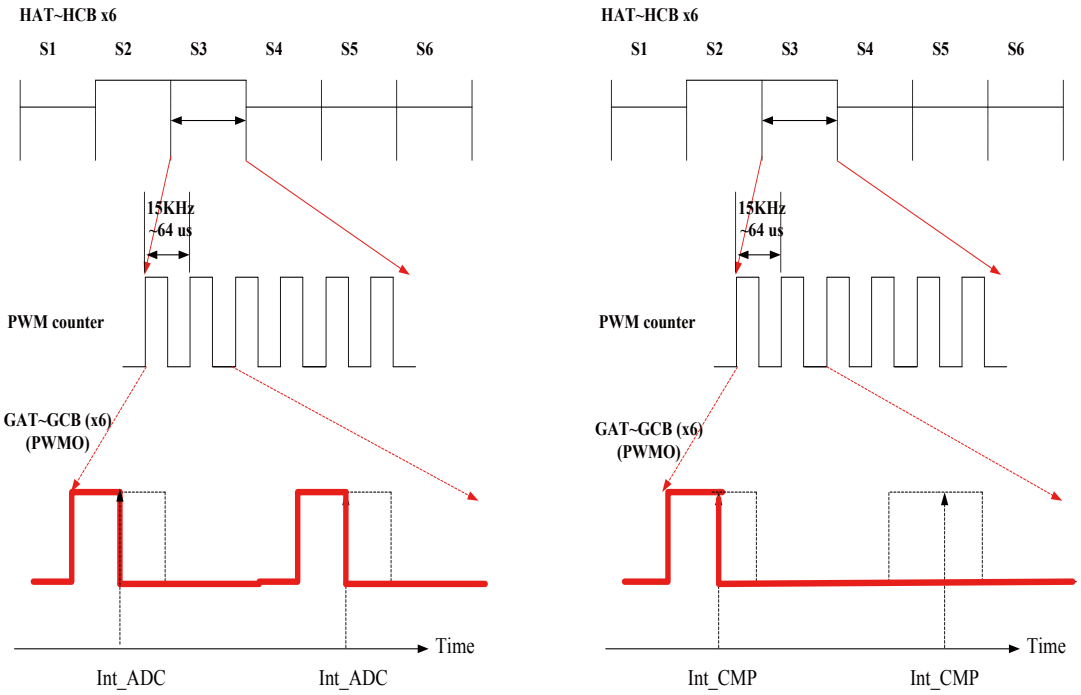
The Pause pin is used to detect whether an external circuit has detected a motor stall or over current condition. The Pause pin is edge triggered and PROTECT is "1". It will cause the external Gate Driver to be shut down. The Pause pin Mode condition is determined by the PSWD/PSWE/PSWPS bits and PROTECT is "0", then the external Gate Driver circuit is controlled by the Hall Sensor Decoder circuit. The Pause pin function is controlled by the PAUHE and PAUTS bits in the MPTC1 and MPTC2 registers.

Current Protection Function

As the device contains a 10-bit A/D Converter, an 8-bit D/A Converter and an amplifier, they can be used together to measure the motor current and to detect for excessive current values. If an over current situation should occur, then the external drive circuit can be shut down immediately to prevent motor damage.

The Int_AHL_Li MOS has a current limit protection mechanism. Disable the H/W Mode when AHLHE is "0" and enable the H/W Mode when AHLHE is "1". The limited current circuit is a hardware circuit, for which the A/D converter channel must select the operation amplifier to be active. If there is an over current during system startup, then this current limit circuit should be disabled.

The Int_Is MOS has an over current protection mechanism. Disable the H/W Model when ISHE is "0" and enable the H/W Mode when ISHE is "1". Select the Fault Mode when ISPS is "0" and select the Pause Mode when ISPS is "1", .



MOS limited current protect: (AHLHE=1;AHLPS=1)
Start the next cycle of the PWM output automaticly by hardware

MOS over current protection: (ISHE=1;ISPS=0)
Restar the PWM output must by software

Over Current

Motor Stall Detection Function

For 3-phase BLDC applications with Hall Sensors, the 16-bit CAPTM can be used to monitor the INT0A, INT0B and INT0C inputs for rotor speed detection. The over current signal is selected by the CAPTMAH and CAPTMAL registers which can monitor the Hall sensor input pins INT0A, INT0B and INT0C to detect the rotor speed. When an over current condition occurs, a CapTM_Cmp or CapTM_Over interrupt will be generated. Refer to the CAPTM chapter for details. In the CapTM_Cmp stall detect mechanism, disable the H/W Mode when CapCHE is "0", and enable the H/W Mode when CapCHE is "1". Select the Fault Mode when CAPCPS is "0" and select the Pause Mode when CAPCPS is "1".

In the CapTM_Over stall detect mechanism, disable the H/W Mode when CapOHE is "0" and enable the H/W Mode when CapOHE is "1". Select the Fault Mode when CAPOPS is "0" and select the Pause Mode when CAPOPS is "1".

Motor Protection Circuit Register Description

There are two registers, MPTC1 and MPTC2, which are used for the motor protection control function.

MPTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	PSWD	PSWE	CapOHE	CapCHE	ISHE	AHLHE	PAUHE	FLTHE
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 **PSWD:** Protect S/W Mode data
0: PSWD=0
1: PSWD=1

Bit 6 **PSWE:** Protect S/W Mode enable
0: Disable
1: Enable

Bit 5 **CapOHE:** CapTM_Over H/W Mode enable
0: Disable
1: Enable

Bit 4 **CapCHE:** CapTM_Cmp H/W Mode enable
0: Disable
1: Enable

Bit 3 **ISHE:** Int_Is H/W Mode enable
0: Disable
1: Enable

Bit 2 **AHLHE:** Int_AHL_Lim H/W Mode enable
0: Disable
1: Enable

Bit 1 **PAUHE:** Pause Pin H/W Mode enable
0: Disable
1: Enable

Bit 0 **FLTHE:** Fault Pin H/W Mode enable
0: Disable
1: Enable

MPTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	PAUTS1	PAUTS0	PSWPS	AHLPS	ISPS	CAPCPS	CAPOPS
R/W	R	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 Unimplemented, read as "0"

Bit 6~5 **PAUTS1~PAUTS0:** Pause trigger select
00: Disable Pause Int.
01: Rising edge
10: Falling edge
11: Dual edge

Bit 4 **PSWPS:** Pause/Fault Mode select
0: Select Fault Mode
1: Select Pause Mode

Bit 3 **AHLPS:** Int_AHL_Lim Pause/Fault Mode select
0: Select Fault Mode
1: Select Pause Mode

Bit 2 **ISPS:** Int_Is Pause/Fault Mode select
0: Select Fault Mode
1: Select Pause Mode

Bit 1 **CAPCPS:** CapTM_Cmp Pause/Fault Mode select
0: Select Fault Mode
1: Select Pause Mode

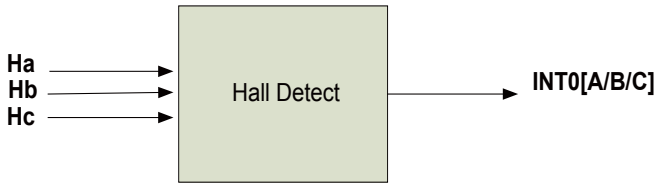
Bit 0 **CAPOPS:** CapTM_Over Pause/Fault Mode select
0: Select Fault Mode
1: Select Pause Mode

Motor Position Detection Methods

There are three methods of BLDC motor positioning control available. These are Digital Hall Sensor Method, Linear Hall Sensor Method and Sensorless Method.

Digital Hall Sensor Method

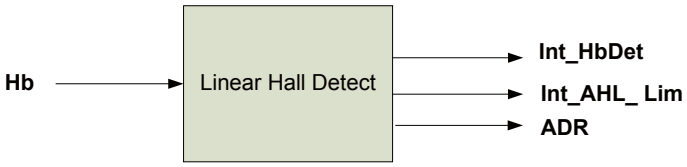
In this method there are three external digital outputs from the hall sensors to detect the rotor position. INT0A, INT0B and INT0C can detect rising, falling and dual edge trigger interrupts. The numerical changes from the Hall sensors is detected is controlled by the application program and can be used to monitor the rotor position . Here HA/HB/HC and the 12 Hall decoder registers are used to control the direction of motor. The PWM functional block is used to control the motor speed.



Digital Hall Sensor Method

Linear Hall Sensor Method:

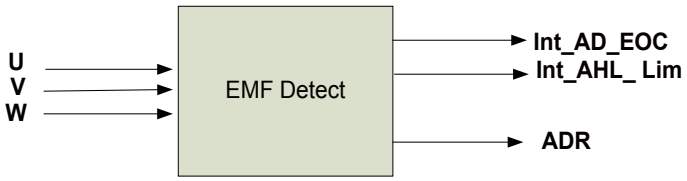
In this method a linear Hall sensor to detect the rotor position. The numerical changes of the external linear Hall sensor is monitored by the 8-bit DAC, 10-bit ADC and comparator 1. The LHMC register, HACM, and the Int_HbDet or Int_AHL_Lim, are used to monitor the motor position. Here SHA/SHB/SHC and the 12 Hall decoding registers are used to control the motor direction. The PWM functional block is used to control the motor speed.



Linear Hall Sensor Method

Sensorless Method

In this method the 3 channels, AN0/AN1/AN2, of the 10-bit A/D converter is used to detect the changes in the back EMF of the three-phase motor. The changes can be detected by the Int_AD_EOC, Int_AHL_Lim and ADRL/ADRH registers. There is a set of 16-bit CTMs to monitor the position and speed of the motor. Use SHA/SHB/SHC and the 12 Hall decoder registers to control the motor direction. The PWM functional block is used to control the motor speed.



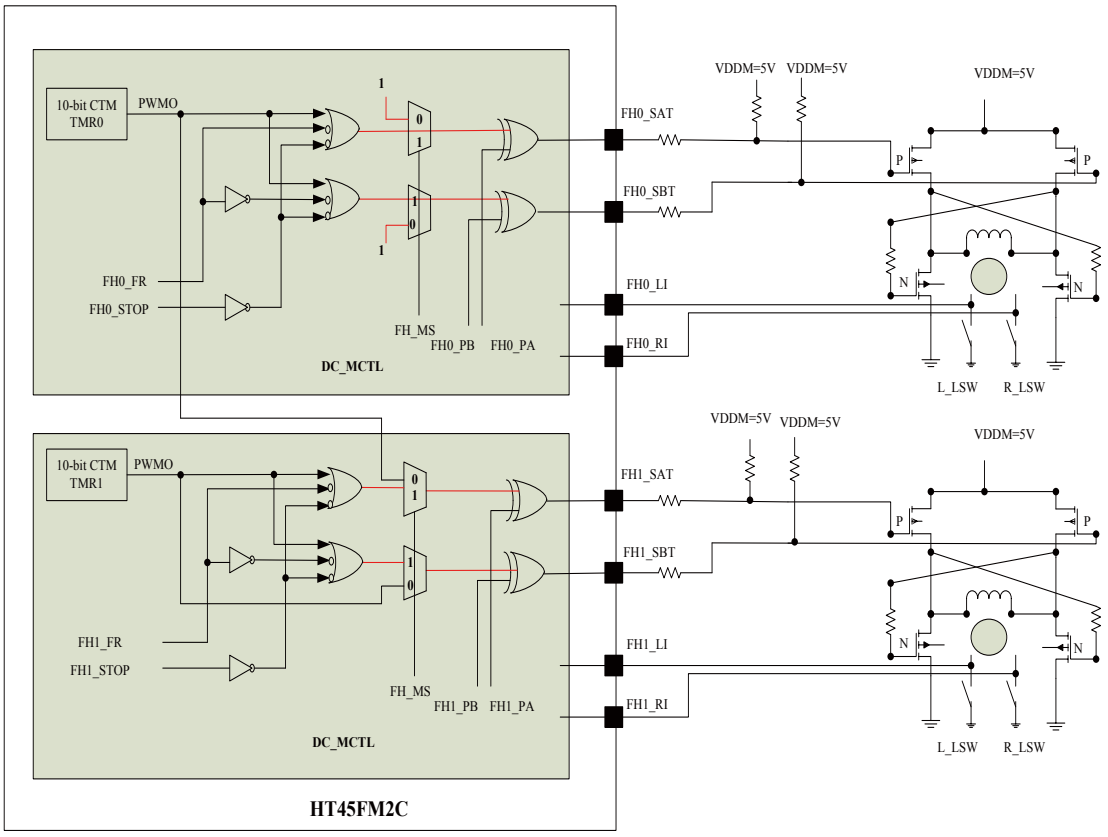
Sensorless Method

DC Motor Control

The device can control motors using 1 or 2 pins. Taking the example of DC Fan Head motor control. a 2-pin DC Motor Interface can control the fan head motor speed and direction, while the 1-pin DC Motor Interface can only control the motor speed. The FH_MS bit in the DCMCR1 register is used to select either a 1-pin or 2-pin DC Motor Interface.

2-pin DC Motor Control

In this case the 2-pin DC Motor interface can control both the motor speed and direction. An external circuit using a limit switch or a VR circuit can be used to control the motor direction. The 10-bit CTM output PWMO is used to adjust the PWM duty cycle to control DC motor speed. The DC_MCTL circuit controls the DC motor direction.



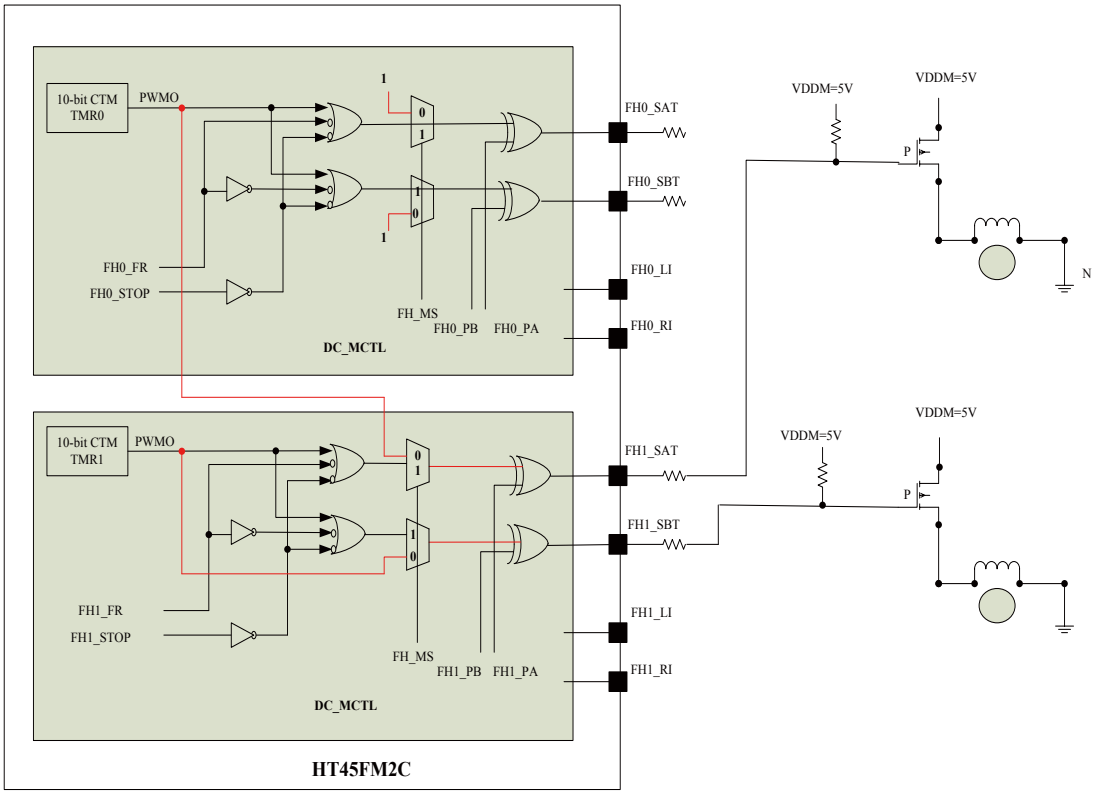
2-Pin DC Motor Interface Application Circuit

The DC control order for the Motor direction is that the motor should be free (FH [0/1] _STOP) before forward (FH [0/1] _FR) and then change direction. It is used to adjust the output polarity according to the external gate drive configuration (P type or N type). The DC Fan Head interface reset default value is FH0_SAT = hi-z/FH0_SBT = hi-z and FH1_SAT = hi-z/FH1_SBT = hi-z.

FH[0/1]_STOP	FH[0/1]_FR	DC_Motor
0	1	Forward
0	0	Backward
1	V	Free

1-pin DC Motor Control

The 1-pin DC Motor interface is used only to control the motor speed. The 10-bit CTM output PWMO is used to adjust the PWM duty cycle to control the DC motor speed. It is used to adjust the output polarity according to the external gate drive configuration (P type or N type). The DC Fan Head interface reset default value is FH1_SAT= hi-z /FH1_SBT=hi-z.



1-Pin DC Motor Interface Application Circuit

Register Description

Two registers, DCMCR0 and DCMCR1 are used for overall control.

DCMCR0 Register

Bit	7	6	5	4	3	2	1	0
Name	FH1_PB	FH1_PA	FH0_PB	FH0_PA	FH1_STOP	FH1_FR	FH0_STOP	FH0_FR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	1	0

Bit 7 **FH1_PB:** polarity output control
0: Non-inverse
1: Inverse

Bit 6 **FH1_PA:** polarity output control
0: Non-inverse
1: Inverse

Bit 5 **FH0_PB:** polarity output control
0: Non-inverse
1: Inverse

Bit 4 **FH0_PA:** polarity output control
0: Non-inverse
1: Inverse

Bit 3 **FH1_STOP:** FAN Head 1 stop enable
0: Motor normal operation
1: Motor free run

Bit 2 **FH1_FR:** FAN Head 1 direction select
0: Backward
1: Forward

Bit 1 **FH0_STOP:** FAN Head 0 stop enable
0: Motor normal operation
1: Motor free run

Bit 0 **FH0_FR:** FAN Head 0 direction select
0: Backward
1: Forward

DCMCR1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	FH1_BE	FH1_AE	FH0_BE	FH0_AE	FH_MS
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 4~1 **FH[1/0]_[A/B]E:** Fan Head Interface output enable
0: Disable Fan Head Interface output
1: Enable Fan Head Interface output

Bit 0 **FH_MS:** FAN Head Mode select
0: 1-pin Mode for 1 FAN head
1: 2-pin Mode for 1 FAN head

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. This device contains six external interrupt and 21 internal interrupt functions. The external interrupts are generated by the action of the external INT0A, INT0B, INT0C, INT1, Fault and Pause pins, while the internal interrupts are generated by various internal functions such as the 10-bit or 16-bit CTMs, Comparators, Motor Protect, Linear Hall Sensor detect, PWM Module, 16-bit CAPTM Module, 8-bit RMT Module, Time Base, LVD, EEPROM and the A/D converter.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers fall into two categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI8 registers which setup the Multi-function interrupts.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
External interrupt 0 (Hall Sensor)	HALLE	HALLF	—
	HALAE	HALAF	—
	HALBE	HALBF	—
	HALCE	HALCF	—
External interrupt 1	INT1E	INT1F	—
Comparator	CnE	CnF	n=0,1
Multifunction interrupt	MFnE	MFnF	n=1~8
A/D Converter	AEOCE	AEOCF	—
	ALIME	ALIMF	—
External Fault interrupt	FLTE	FLTF	—
External Pause interrupt	PAUE	PAUF	—
PWM	PWMDE	PWMDF	—
	PWMPE	PWMPF	—
Time Base	TBE	TBF	—
CAPTM	CAPOE	CAPOF	—
	CAPCE	CAPCF	—
TM	TMnAE	TMnAF	n=0,1,2,3,5
	TMnPE	TMnPF	n=0,1,2,3,5
RMT	RMT0E	RMT0F	—
	RMT1E	RMT1F	—
	RMTVE	RMTVF	—
LVD	LVDE	LVDF	—
EEPROM	EPWE	EPWF	—

Interrupt Register Bit Naming Conventions

Interrupt Register Contents

Name	Bit							
	7	6	5	4	3	2	1	0
INTC0	—	C0F	INT1F	HALLF	C0E	INT1E	HALLE	EMI
INTC1	PAUF	FLTF	MF1F	C1F	PAUE	FLTE	MF1E	C1E
INTC2	MF4F	MF3F	TBF	MF2F	MF4E	MF3E	TBE	MF2E
INTC3	MF8F	MF7F	MF6F	MF5F	MF8E	MF7E	MF6E	MF5E
MFI0	—	HALCF	HALBF	HALAF	—	HALCE	HALBE	HALAE
MFI1	—	—	ALIMF	AEOCF	—	—	ALIME	AEOCE
MFI2	—	—	PWMPF	PWMDF	—	—	PWMPE	PWMDE
MFI3	—	—	CAPCF	CAPOF	—	—	CAPCE	CAPOE
MFI4	TM1AF	TM1PF	TM0AF	TM0PF	TM1AE	TM1PE	TM0AE	TM0PE
MFI5	TM3AF	TM3PF	TM2AF	TM2PF	TM3AE	TM3PE	TM2AE	TM2PE
MFI6	—	RMTVF	RMT1F	RMT0F	—	RMTVE	RMT1E	RMT0E
MFI7	—	—	TM5AF	TM5PF	—	—	TM5AE	TM5PE
MFI8	—	—	EPWF	LVDF	—	—	EPWE	LVDE

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	C0F	INT1F	HALLF	C0E	INT1E	HALLE	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 **CP0F:** Comparator 0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **INT1F:** External 1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **HALLF:** Hall sensor global interrupt request flag
0: No request
1: Interrupt request

Bit 3 **C0E:** Comparator 0 interrupt control
0: Disable
1: Enable

Bit 2 **INT1E:** External 1 interrupt control
0: Disable
1: Enable

Bit 1 **HALLE:** Hall sensor global interrupt control
0: Disable
1: Enable

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

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	PAUF	FLTF	MF1F	C1F	PAUE	FLTE	MF1E	C1E
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 **PAUF:** Pause Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **FLTF:** Fault Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **MF1F:** Multi-function Interrupt 1 Request Flag
0: No request
1: Interrupt request

Bit 4 **C1F:** Comparator 1 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **PAUE:** Pause Interrupt Interrupt Control
0: Disable
1: Enable

Bit 2 **FLTE:** Fault Interrupt Control
0: Disable
1: Enable

Bit 1 **MF1E:** Multi-function Interrupt 1 Control
0: Disable
1: Enable

Bit 0 **C1E:** Comparator 1 Interrupt Control
0: Disable
1: Enable

INTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	MF4F	MF3F	TBF	MF2F	MF4E	MF3E	TBE	MF2E
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 **MF4F:** Multi-function interrupt 4 request flag
0: No request
1: Interrupt request

Bit 6 **MF3F:** Multi-function interrupt 3 request flag
0: No request
1: Interrupt request

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

Bit 4 **MF2F:** Multi-function interrupt 2 Request flag
0: No request
1: Interrupt request

Bit 3 **MF4E:** Multi-function interrupt 4 control
0: Disable
1: Enable

Bit 2 **MF3E:** Multi-function interrupt 3 control
0: Disable
1: Enable

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

Bit 0 **MF2E:** Multi-function interrupt 2 control
0: Disable
1: Enable

INTC3 Register

Bit	7	6	5	4	3	2	1	0
Name	MF8F	MF7F	MF6F	MF5F	MF8E	MF7E	MF6E	MF5E
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 **MF8F:** Multi-function interrupt 8 request flag
0: No request
1: Interrupt request

Bit 6 **MF7F:** Multi-function interrupt 7 request flag
0: No request
1: Interrupt request

Bit 5 **MF6F:** Multi-function interrupt 6 request flag
0: No request
1: Interrupt request

Bit 4 **MF5F:** Multi-function interrupt 5 request flag
0: No request
1: Interrupt request

Bit 3 **MF8E:** Multi-function interrupt 8 control
0: Disable
1: Enable

Bit 2 **MF7E:** Multi-function interrupt 7 control
0: Disable
1: Enable

Bit 1 **MF6E:** Multi-function interrupt 6 control
0: Disable
1: Enable

Bit 0 **MF5E:** Multi-function interrupt 5 control
0: Disable
1: Enable

MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	HALCF	HALBF	HALAF	—	HALCE	HALBE	HALAE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **HALCF:** Hall Sensor C interrupt request flag
0: No request
1: Interrupt request

Bit 5 **HALBF:** Hall Sensor B interrupt request flag
0: No request
1: Interrupt request

Bit 4 **HALAF:** Hall Sensor A interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **HALCE:** Hall Sensor C interrupt control
0: Disable
1: Enable

Bit 1 **HALBE:** Hall Sensor B interrupt control
0: Disable
1: Enable

Bit 0 **HALAE:** Hall Sensor A interrupt control
0: Disable
1: Enable

MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	ALIMF	AEOCF	—	—	ALIME	AEOCE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **ALIMF:** A/D Converter EOC compare interrupt request flag
0: No request
1: Interrupt request

Bit 4 **AEOCF:** A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **ALIME:** A/D Converter EOC compare interrupt control
0: Disable
1: Enable

Bit 0 **AEOCE:** A/D Converter interrupt control
0: Disable
1: Enable

MFI2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PWMPF	PWMDF	—	—	PWMPE	PWMDE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PWMPF:** PWM Period match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PWMDF:** PWM Duty match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PWMPE:** PWM Period match interrupt control
0: Disable
1: Enable

Bit 0 **PWMDE:** PWM Duty match interrupt control
0: Disable
1: Enable

MFI3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	CAPCF	CAPOF	—	—	CAPCE	CAPOE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **CAPCF:** CAPTM compare match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CAPOF:** CAPTM capture overflow interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **CAPCE:** CAPTM compare match interrupt control
0: Disable
1: Enable

Bit 0 **CAPOE:** CAPTM capture overflow interrupt control
0: Disable
1: Enable

MFI4 Register

Bit	7	6	5	4	3	2	1	0
Name	TM1AF	TM1PF	TM0AF	TM0PF	TM1AE	TM1PE	TM0AE	TM0PE
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 **TM1AF:** TM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TM1PF:** TM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TM0AF:** TM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TM0PF:** TM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TM1AE:** TM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **TM1PE:** TM1 Comparator P match interrupt control
0: Disable
1: Enable

Bit 1 **TM0AE:** TM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **TM0PE:** TM0 Comparator P match interrupt control
0: Disable
1: Enable

MFI5 Register

Bit	7	6	5	4	3	2	1	0
Name	TM3AF	TM3PF	TM2AF	TM2PF	TM3AE	TM3PE	TM2AE	TM2PE
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 **TM3AF:** TM3 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TM3PF:** TM3 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TM2AF:** TM2 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TM2PF:** TM2 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TM3AE:** TM3 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **TM3PE:** TM3 Comparator P match interrupt control
0: Disable
1: Enable

Bit 1 **TM2AE:** TM2 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **TM2PE:** TM2 Comparator P match interrupt control
0: Disable
1: Enable

MFI6 Register

Bit	7	6	5	4	3	2	1	0
Name	—	RMTVF	RMT1F	RMT0F	—	RMTVE	RMT1E	RMT0E
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **RMTVF:** RMT overflow interrupt request flag
0: No request
1: Interrupt request

Bit 5 **RMT1F:** RMT falling edge interrupt request flag
0: No request
1: Interrupt request

Bit 4 **RMT0F:** RMT rasing edge interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **RMTVE:** RMT overflow interrupt control
0: Disable
1: Enable

Bit 1 **RMT1E:** RMT falling edge interrupt control
0: Disable
1: Enable

Bit 0 **RMT0E:** RMT rasing edge interrupt control
0: Disable
1: Enable

MFI7 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TM5AF	TM5PF	—	—	TM5AE	TM5PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TM5AF:** TM5 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TM5PF:** TM5 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **TM5AE:** TM5 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **TM5PE:** TM5 Comparator P match interrupt control
0: Disable
1: Enable

MFI8 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	EPWF	LVDF	—	—	EPWE	LVDE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **EPWF:** Data EEPROM interrupt request flag
0: No request
1: Interrupt request

Bit 4 **LVDF:** LVD interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **EPWE:** Data EEPROM interrupt control
0: Disable
1: Enable

Bit 0 **LVDE:** LVD interrupt control
0: Disable
1: Enable

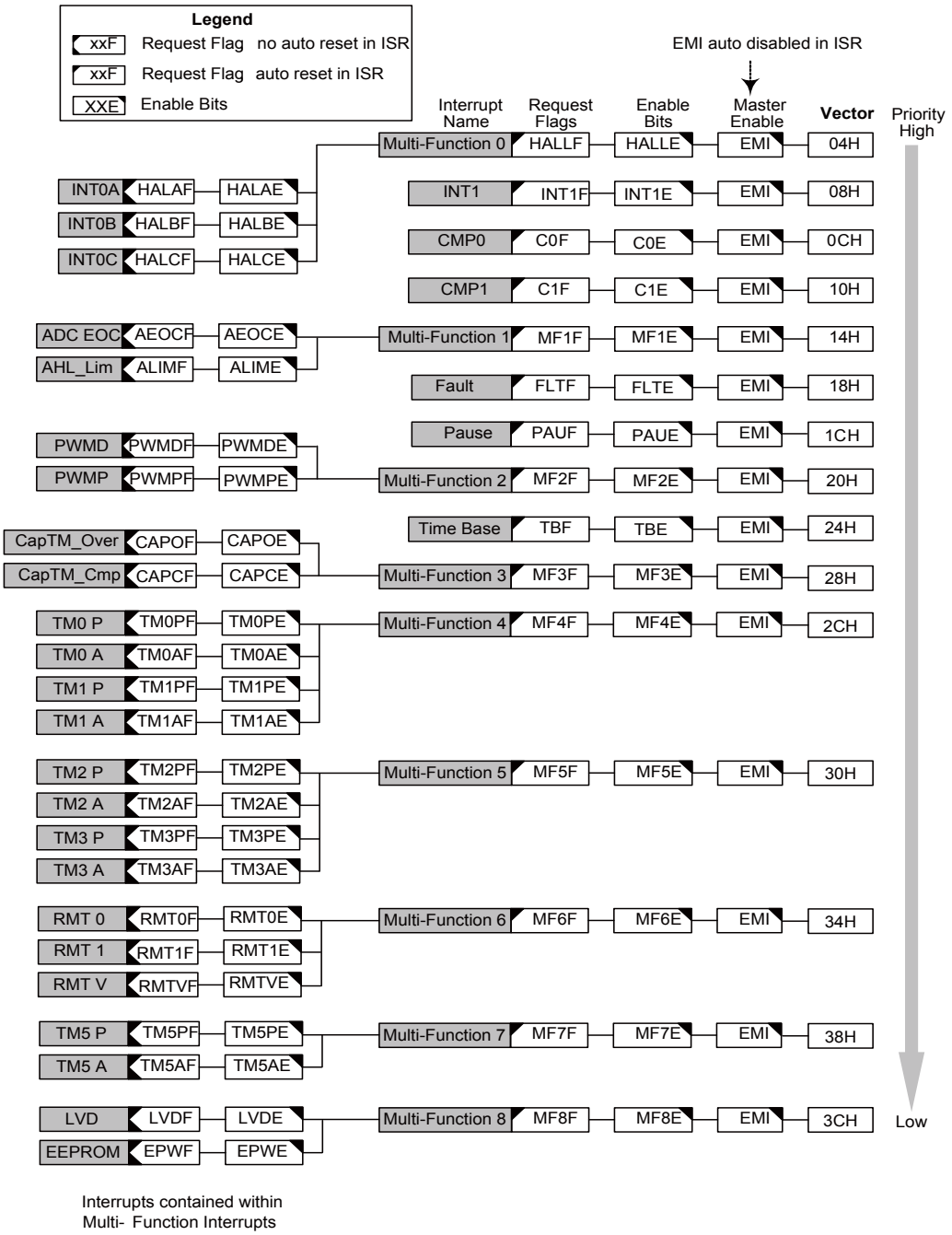
Interrupt Operation

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

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a “JMP” which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a “RETI”, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.



Interrupt Structure

External Interrupt 0

The external interrupt 0, also known as the Hall Sensor interrupt, is contained within the Multi-function Interrupt. It is controlled by signal transitions on the pins, Hall Sensor input pins, INT0A, INT0B and INT0C. An external interrupt request will take place when the external interrupt request flag, HALAF HALBF and HALCF is set, which will occur when a transition, 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 the Multi-function interrupt controlled bit, HALLE must first be set.

When the Multi-function interrupt controlled bit HALLE is enabled and the stack is not full, and either one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the related Multi-Function request flag HALLF, will be automatically reset, but the Multi-function interrupt request flags, HALAF,HALBF,HALCF, must be manually cleared by the application program..

External Interrupt 1

The external interrupt 1 is controlled by signal transitions on the pin INT1. An external interrupt request will take place when the external interrupt request flag, INT1F, is set, which will occurs when a transition appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT1E, must first be set. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INT1F, 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.

Comparator Interrupt

The comparator interrupts are controlled by the two internal comparators. A comparator interrupt request will take place when the comparator interrupt request flag, C0F or C1F, 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, and comparator interrupt enable bit, C0E or C1E, must first be set. When the interrupt is enabled, the stack is not full and the comparator inputs generate a comparator output transition, a subroutine call to the comparator interrupt vector, will take place. When the interrupt is serviced, the comparator interrupt request flags, C0F or C1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Multi-function Interrupt

Within this device are nine Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the Hall Sensor interrupts, A/D interrupts, PWM Module interrupts, CAPTM Interrupts, TM Interrupts, RMT Interrupts, EEPROM and LVD Interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags, HALLF and MF1F~MF8F are set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related Multi-Function request flag, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function Interrupt flags will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupts, namely the Hall Sensor interrupts, A/D interrupts, PWM Module interrupts, CAPTM Interrupts, TM Interrupts, RMT Interrupts, EEPROM and LVD Interrupt will not be automatically reset and must be manually reset by the application program.

A/D Converter Interrupt

The A/D Converter has two interrupts. All of them are contained in Multi-function interrupt. The one is controlled by the termination of an A/D conversion process. An A/D Converter interrupt request will take place when the A/D Converter Interrupt request flag, ALIMF, is set, which occurs when the A/D conversion process finishes. The other is controlled by the ADCHVE/ADCLVE bit in the ADCR1 register and the value in the ADLVDH/ADLVDL and ADHVDH/ADHVDL boundary control registers. An A/D Converter Interrupt request will take place after EOC comparing. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, AEOCE or ALIME, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended or after EOC comparing a subroutine call to the A/D Converter Interrupt vector, will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, AEOCF or ALIMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Fault Interrupt

Fault pin is Motor Control disable pin, it supports low active level trigger interrupt. When this happens, its interrupt request flag, FLTF will be set. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI and enable bit, FLTE, must first be set. When the interrupt is enabled, the stack is not full and a low active level appears on the pin, a subroutine call to this vector location will take place. When the interrupt is serviced, the interrupt request flag, FLTF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

Pause Interrupt

Pause pin is Motor Control enable pin, it support rising/falling/both trigger interrupt. When this happens, its interrupt request flag, PAUF will be set. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI and the corresponding enable bit, PAUE, must first be set. When the interrupt is enabled, the stack is not full and a rising/falling/both edge trigger appears on the pin, a subroutine call to this vector location will take place. When the interrupt is serviced, the interrupt request flag, PAUF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

PWM Module Interrupts

The PWM Module has two interrups. The two of them are contained in Multi-function interrupt, which is known as PWMD and PWMP. They are the Duty or the Period maching of the PWM Module. An PWM interrupt request will take place when the PWM interrupt request flag, PWMDF or PWMPF, is set, which occurs when the PWM Duty or PWM Period matches. When the interrupt is enabled, the stack is not full and PWM Duty or PWM Period maches, a subroutine call to this vector location will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the related Multi-Function request flag will be automatically reset, but the interrupt request flag, PWMDF or PWMPF, must be manually cleared by the application program.

Time Base Interrupt

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

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source originates 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.

TBC Register

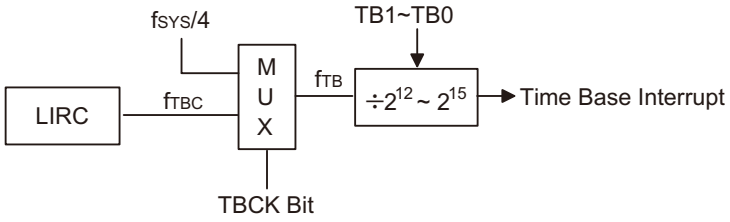
Bit	7	6	5	4	3	2	1	0
Name	TBON	TBCK	TB1	TB0	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	1	1	—	—	—	—

Bit 7 **TBON:** TB Control
0: Disable
1: Enable

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

Bit 5~4 **TB1~TB0:** Select Time Base Time-out Period
00: $4096/f_{TB}$
01: $8192/f_{TB}$
10: $16384/f_{TB}$
11: $32768/f_{TB}$

Bit 3~0 Unimplemented, read as "0"



Time Base Interrupt

CAPTM Module Interrupt

The CAPTM Module has two interrupts. All of them are contained within the Multi-function Interrupt, which are known as CapTM_Over and CapTM_Cmp. A CAPTM Interrupt request will take place when the CAPTM Interrupts request flag, CAPOF or CAPCF, is set, which occurs when CAPTM capture overflows or compare maches. To allow the program to branch to their respective interrupt vector address, the global interrupt enable bit, EMI, and the CAPTM Interrupt enable bit, and Muti-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and CAPTM capture overflows or compare maches, a subroutine call to the respective Multi-function Interrupt vector, will take place. When the CAPTM 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 CAPOF and CAPCF flag will not be automatically cleared, it has to be cleared by the application program.

TM Interrupt

The Compact TM has two interrupts. All of the TM interrupts are contained within the Multi-function Interrupts. For the Compact Type TM, there are two interrupt request flags TnPF and TnAF and two enable bits TnPE and TnAE. A TM interrupt request will take place when any of the TM request flags is set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

RMT Module Interrupt

The RMT Module has three interrupts. All of them are contained within the Multi-function Interrupt, which are known as RMT0, RMT1 and RMTV. The RMT Interrupts request will take place when the RMT Interrupts request flag, RMT0F, RMT1F or RMTVF, is set, which will occurs when raising edge transition or falling edge transition appears on the Rx_IN pin or Timer overflows. To allow the program to branch to their respective interrupt vector address, the global interrupt enable bit, EMI, RMT interrupt enable bit, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and raising edge transition or falling edge transition appears on the Rx_IN pin or Timer overflow, a subroutine call to the respective Multi-function Interrupt, will take place. When the RMT 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 RMT0, RMT1 and RMTV flags will not be automatically cleared, thy have 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, EPWF, 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, EPWE, and associated 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 EPWF flag will not be automatically cleared, it has to be cleared by the application program.

LVD Interrupt

The Low Voltage Detector Interrupt is contained within the Multi-function Interrupt. An LVD Interrupt request will take place when the LVD Interrupt request flag, LVDF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVDE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the Multi-function Interrupt vector, will take place. When the Low Voltage 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 LVDF 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 this device are in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins, a low power supply voltage or comparator input change may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

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

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, HALLF and MF1F~MF8F, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

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

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

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

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

Low Voltage Detector – LVD

Each device has a Low Voltage Detector function, also known as LVD. This enabled the device to monitor the power supply voltage, V_{DD}, and provide a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of eight fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

LVDC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	—	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **LVDO:** LVD Output Flag
0: No Low Voltage Detect
1: Low Voltage Detect

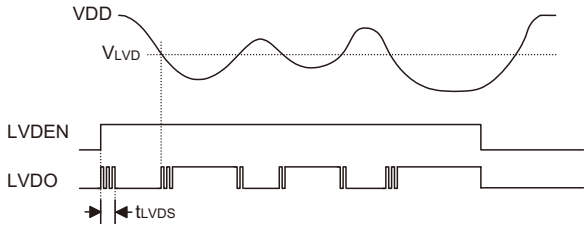
Bit 4 **LVDEN:** Low Voltage Detector Control
0: Disable
1: Enable

Bit 3 Unimplemented, read as "0"

Bit 2~0 **VLVD2~VLVD0:** Select LVD Voltage
000: 3.6V
001: 3.6V
010: 3.6V
011: 3.6V
100: 3.0V
101: 3.6V
110: 3.6V
111: 3.6V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a specified voltage 3.6V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is powered down the low voltage detector will remain active if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.

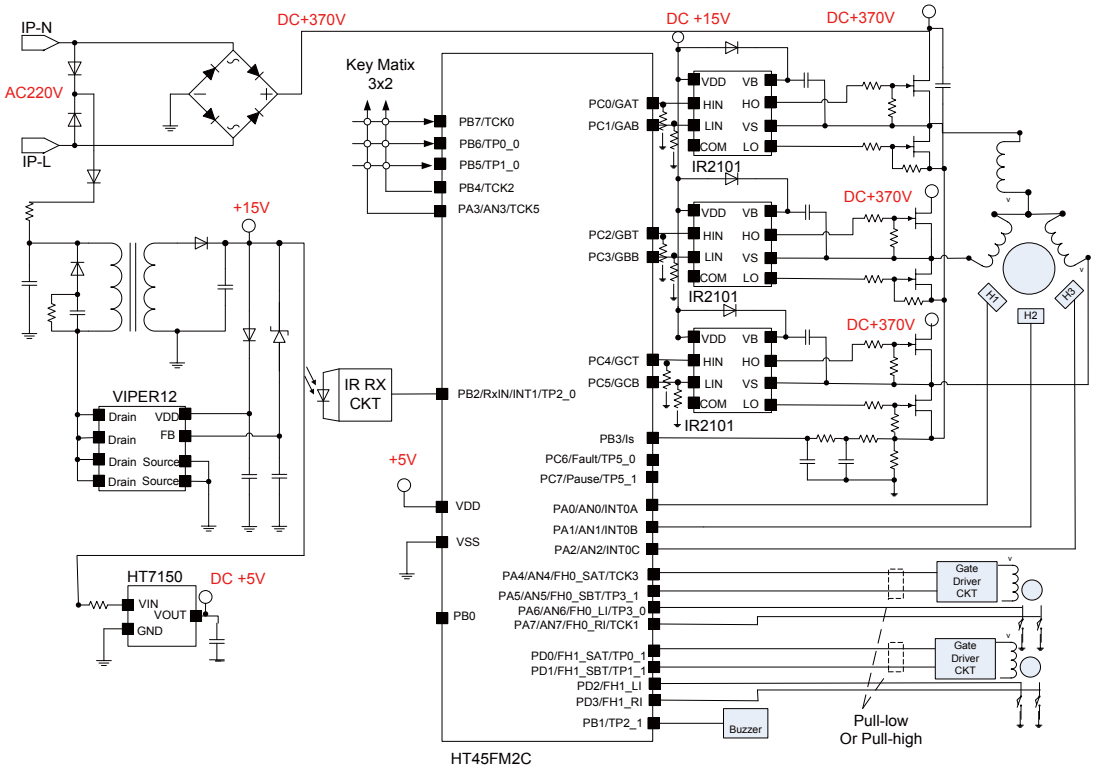


LVD Operation

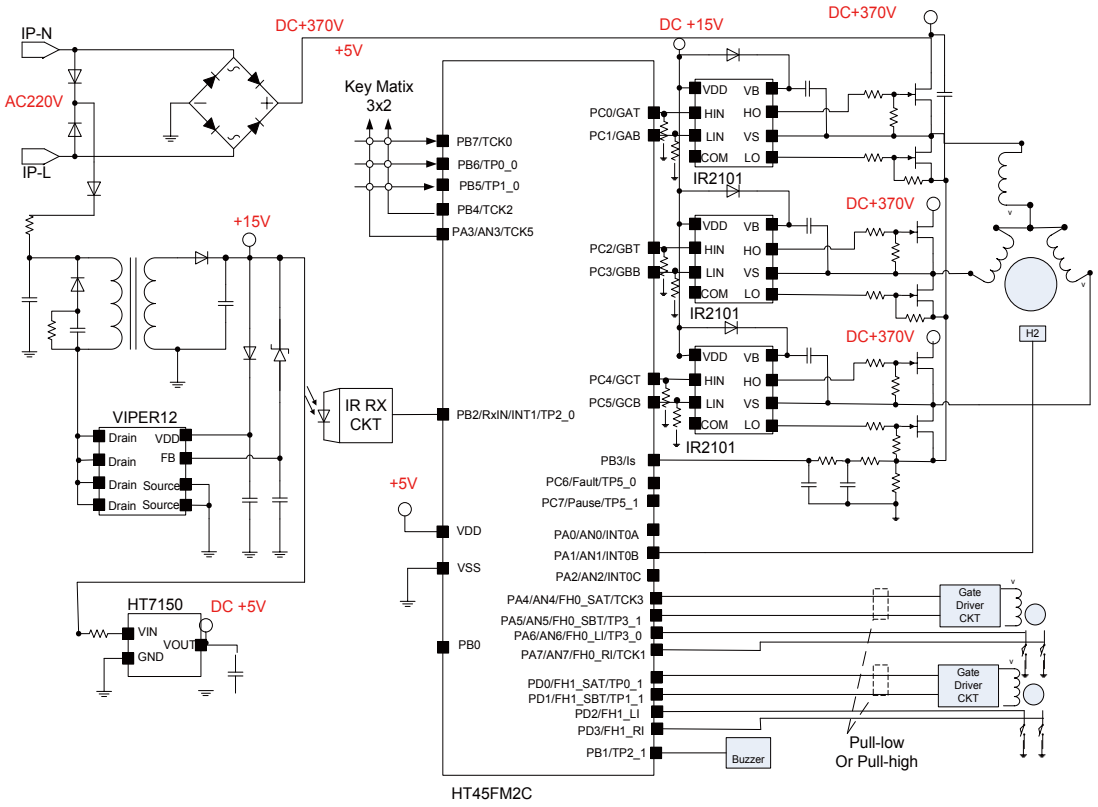
The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. When the device is powered down the Low Voltage Detector will remain active if the LVDEN bit is high. In this case, the LVDF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} falls below the preset LVD voltage. This will cause the device to wake-up from the SLEEP or IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVDF flag should be first set high before the device enters the SLEEP or IDLE Mode.

Application Circuits

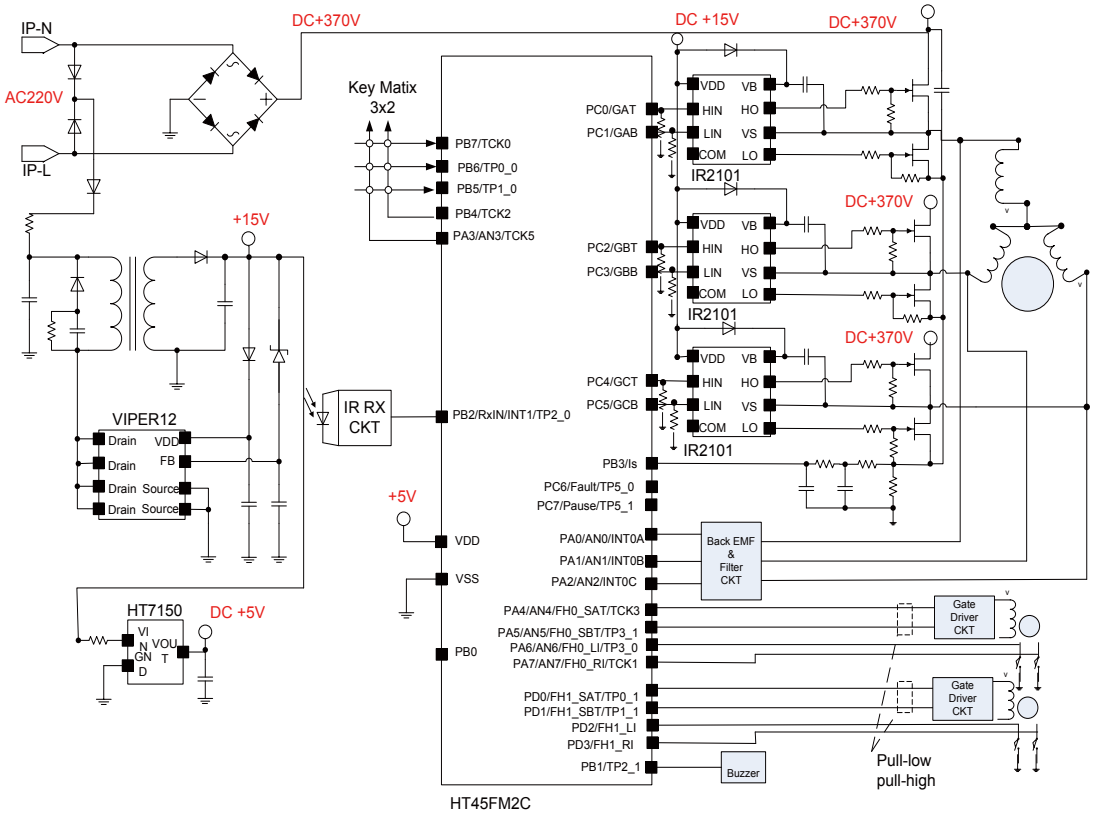
Hall Sensor × 3



Hall Sensor × 1



Non-Hall Sensor



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be 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

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

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

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

3. For the “CLR WDT1” and “CLR WDT2” instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both “CLR WDT1” and “CLR WDT2” instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry

Description: The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + [m] + C

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

ADCM A,[m] Add ACC to Data Memory with Carry

Description: The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation: [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

Description: The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

Description: The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

Description: The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation: [m] ← ACC + [m]

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

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

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

Operation: ACC ← ACC "AND" [m]

Affected flag(s): Z

AND A,x Logical AND immediate data to ACC

Description: Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.

Operation: ACC ← ACC "AND" x

Affected flag(s): Z

ANDM A,[m] Logical AND ACC to Data Memory

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

Operation: [m] ← ACC "AND" [m]

Affected flag(s): Z

CALL addr Subroutine call

Description Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

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

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

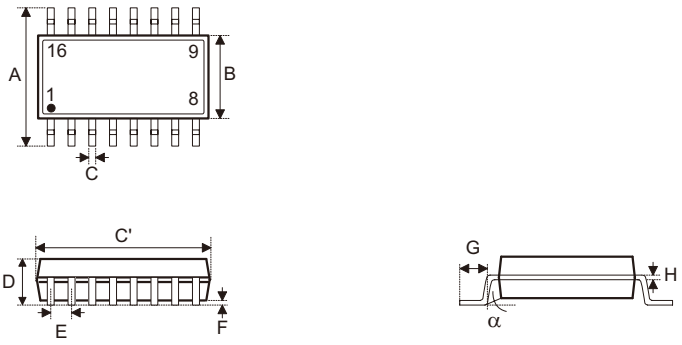
Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Package Information

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

16-pin NSOP (150mil) Outline Dimensions

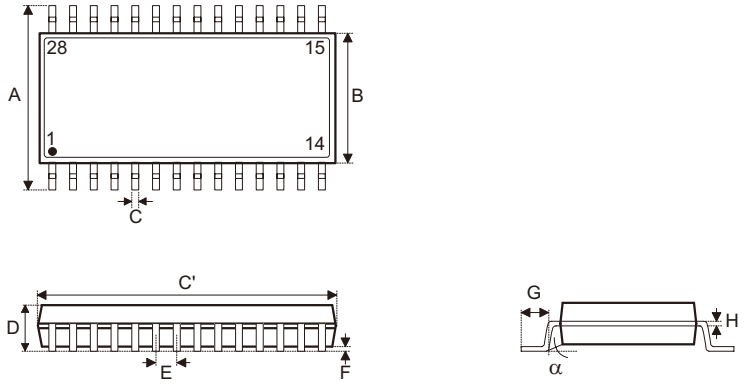


MS-012

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°

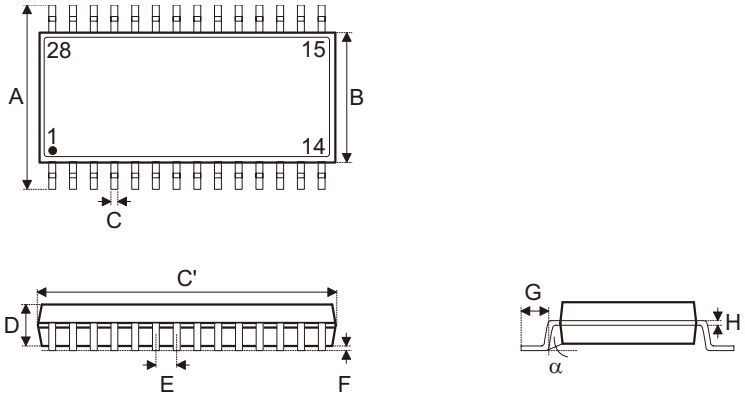
28-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.393	—	0.419
B	0.256	—	0.300
C	0.012	—	0.020
C′	0.697	—	0.713
D	—	—	0.104
E	—	0.050	—
F	4	—	0.012
G	16	—	0.050
H	8	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	9.98	—	10.64
B	6.50	—	7.62
C	0.30	—	0.51
C′	17.70	—	18.11
D	—	—	2.64
E	—	1.27	—
F	0.10	—	0.30
G	0.41	—	1.27
H	0.20	—	0.33
α	0°	—	8°

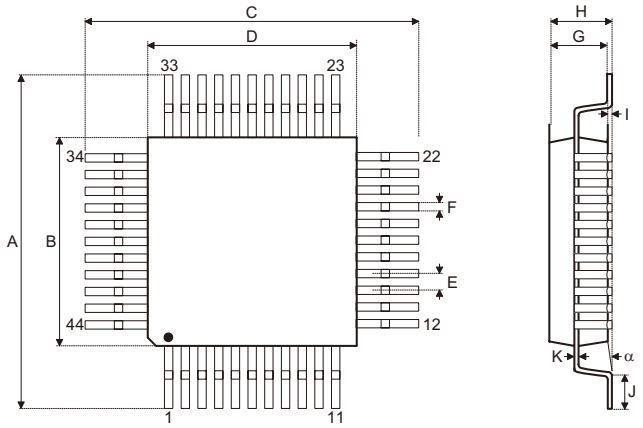
28-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.157
C	0.008	—	0.012
C′	0.386	—	0.394
D	0.054	—	0.060
E	—	0.025	—
F	0.004	—	0.010
G	0.022	—	0.028
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	0.20		0.30
C′	9.80	—	10.01
D	1.37	—	1.52
E	—	0.64	—
F	0.10	—	0.25
G	0.56	—	0.71
H	0.18	—	0.25
α	0°	—	8°

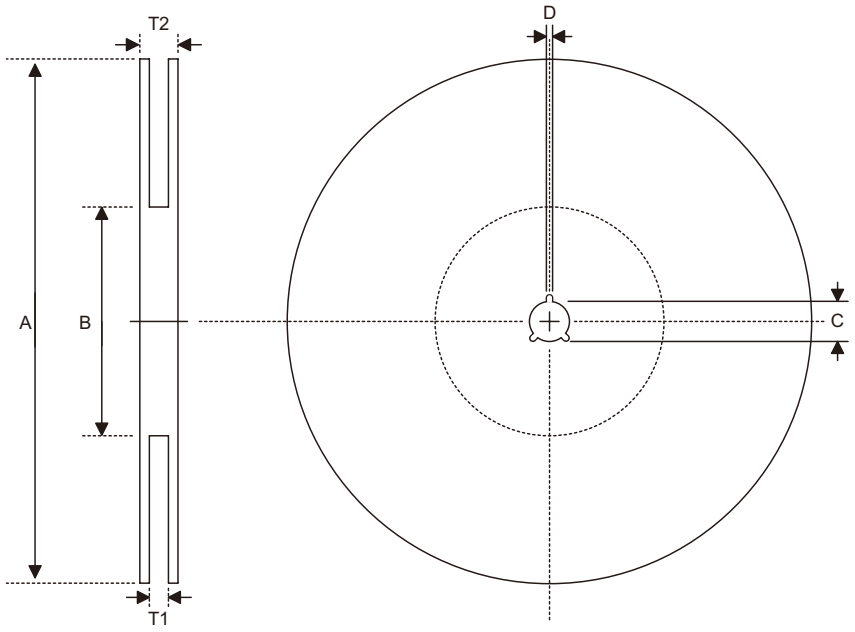
44-pin LQFP (10mm×10mm) (FP2.0mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.469	—	0.476
B	0.390	—	0.398
C	0.469	—	0.476
D	0.390	—	0.398
E	—	0.031	—
F	—	0.012	—
G	0.053	—	0.057
H	—	—	0.063
I	—	0.004	—
J	0.018	—	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	11.90	—	12.10
B	9.90	—	10.10
C	11.90	—	12.10
D	9.90	—	10.10
E	—	0.80	—
F	—	0.30	—
G	1.35	—	1.45
H	—	—	1.60
I	—	0.10	—
J	0.45	—	0.75
K	0.10	—	0.20
α	0°	—	7°

Reel Dimensions



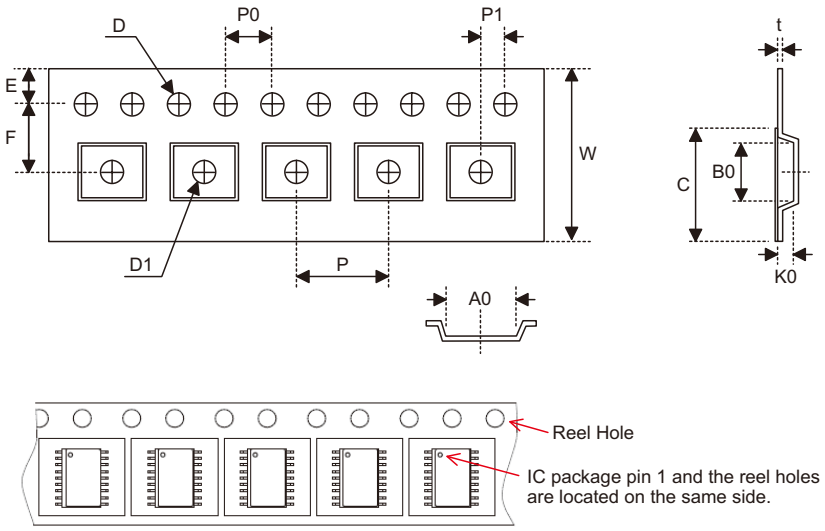
16-pin NSOP(150mil), SSOP 28S (150mil)

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

SOP 28W(300mil)

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	$24.8^{+0.3/-0.2}$
T2	Reel Thickness	30.2±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±0.1
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

SOP 28W (300mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0+0.3
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation(Width Direction)	11.5±0.1
D	Perforation Diameter	$1.5^{+0.10/-0.0}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation(Length Direction)	2.0±0.1
A0	Cavity Length	10.85±0.1
B0	Cavity Width	2.97±0.1
K0	Cavity Depth	2.97±0.10
t	Carrier Tape Thickness	0.35±0.01
C	Cover Tape Width	21.3±0.1

SSOP 28S (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0±0.3
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation(Width Direction)	7.5±0.1
D	Perforation Diameter	$1.55^{+0.1/-0.0}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
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

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 (China) Inc.
Building No.10, Xinzhu Court, (No.1 Headquarters), 4 Cuizhu Road, Songshan Lake, Dongguan, China 523808
Tel: 86-769-2626-1300
Fax: 86-769-2626-1311

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