Brushless DC Motor Flash MCU

HT66FM5240

Revision: V1.30 Date: July 18, 2023

www.holtek.com

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
- Oscillators
  - Internal 20MHz – HIRC
  - Internal 32kHz – LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

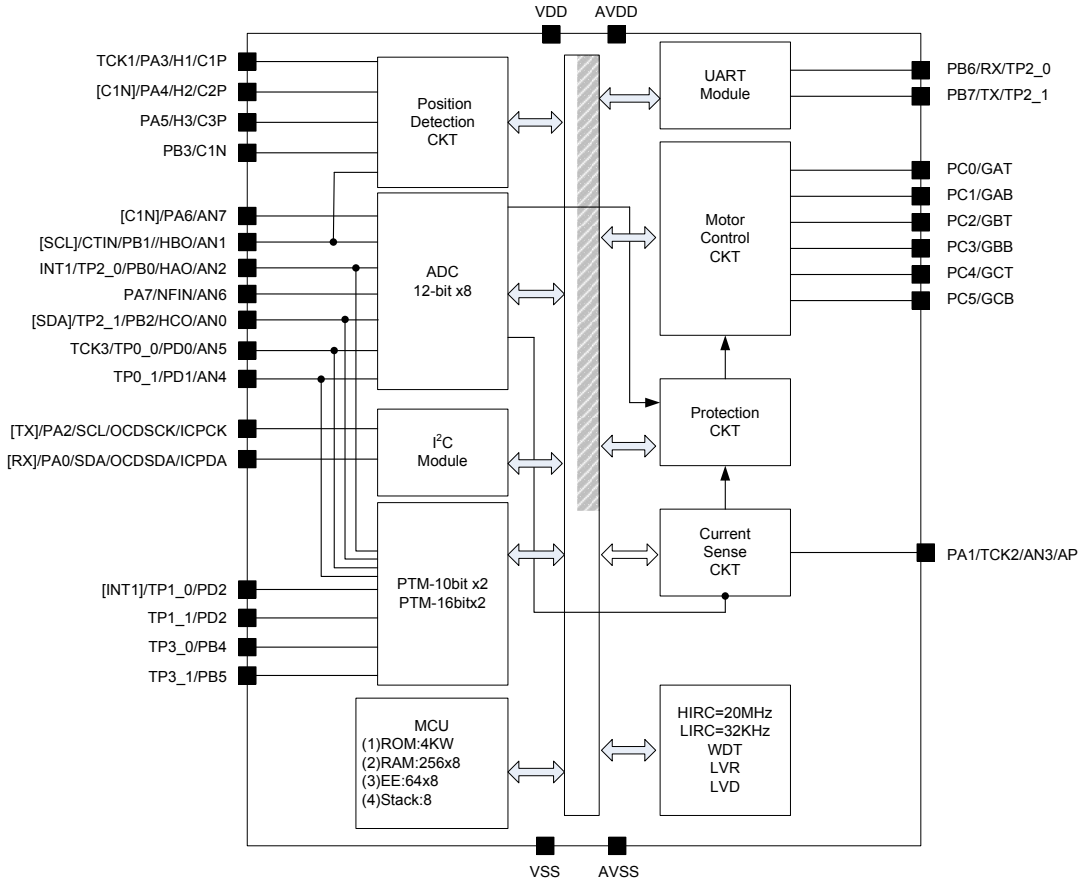
- Flash Program Memory: 4K×16
- RAM Data Memory: 256×8
- True EEPROM Memory: 64×8
- Watchdog Timer function
- Up to 26 bidirectional I/O lines
- Five pin-shared external interrupts
- Two 16-bit Periodic Timer Module – 16-bit PTM
- Two 10-bit Periodic Timer Module – 10-bit PTM
- Single 16-bit CAPTM for motor protect
- 3-channel 10-bit PWM with comlementary outputs for BLDC application
- 8-channel 12-bit resolution A/D converter
- UART interface
- Time-Base function for generation of fixed time interrupt signal
- Single operational Amplifier for current detection
- Four comparators function – Comparator 0 has interrupt function
- Single 8-bit D/A Converter
- I^2C interface
- Low voltage reset function
- Low voltage detect function
- Package types: 20/28-pin SSOP

General Description

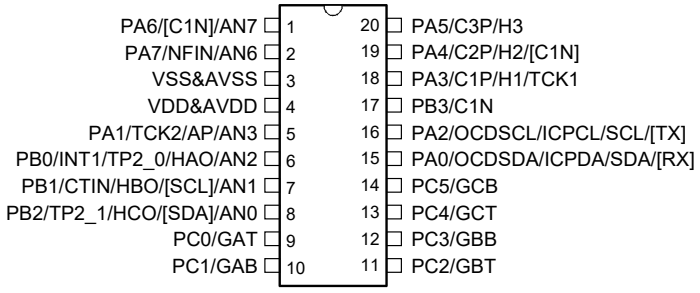
This device is Flash Memory with 8-bit high performance RISC architecture microcontroller device which includes a host of fully integrated special features specifically designed for the brushless DC motor applications.

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, comparator functions, Motor Protect Module, Hall sensor position detect function, Time Base function, LVD, true EEPROM, power-down and wake-up functions, Communication with the outside world is catered for by including fully integrated I^2C and UART interface 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.

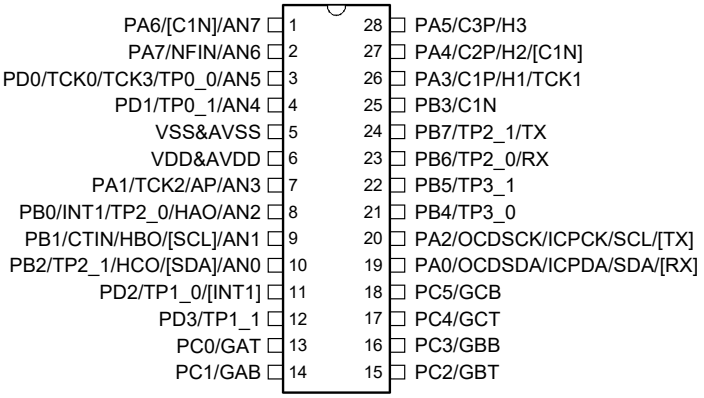
Block Diagram



Pin Assignment



HT66FM5240
20 SSOP-A



HT66FM5240
28 SSOP-A

Note: 1. The “VDD&AVDD” means that the VDD and AVDD are internally bonded while the “VSS&AVSS” means that the VSS and AVSS are internally bonded.

2. For the less pin count package type there will be unbounded pins which should be properly configured to avoid unwanted power consumption resulting from floating input conditions. Refer to the "Standby Current Considerations" and "Input/Output Ports" sections.

Pin Descriptions

With the exception of the power pins and some relevant transformer control pins, all pins on these devices can be referenced by their Port name, e.g. PA0, PA1 etc, which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Analog to Digital Converter, Timer Module pins etc. The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/OCDSDA/ICPDA/ SDA/[RX]	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	OCDSDA	—	ST	CMOS	OCDS Address/Data, for EV chip only.
	ICPDA	—	ST	CMOS	ICP Address/Data
	SDA	PAPS0	ST	NMOS	I^2C data/address line
	RX	PAPS0	ST	—	External UART RX serial data input pin
PA1/TCK2/AP/AN3	PA1	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	TCK2	PAPS0	ST	—	TM2 input
	AP	PAPS0	AN	—	Operational Amplifier input
	AN3	PAPS0	AN	—	A/D channel input
PA2/OCDSCK/ICPCK/ SCL/[TX]	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	OCDSCK	—	ST	—	OCDS Clock pin, for EV chip only.
	ICPCK	—	ST	—	ICP Clock pin
	SCL	PAPS0	ST	NMOS	I^2C clock line
	TX	PAPS0	—	CMOS	External UART TX serial data output pin
PA3/C1P/H1/TCK1	PA3	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	C1P	PAPS0	AN	—	Comparator 1 input
	H1	PAPS0	ST	—	Hall sensor input
	TCK1	PAPS0	ST	—	TM1 input
PA4/C2P/H2/[C1N]	PA4	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	C2P	PAPS1	AN	—	Comparator 2 input
	H2	PAPS1	ST	—	Hall sensor input
	C1N	PAPS1	AN	—	Comparator 1 input
PA5/C3P/H3	PA5	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	C3P	PAPS1	AN	—	Comparator 3 input
	H3	PAPS1	ST	—	Hall sensor input
PA6/[C1N]/AN7	PA6	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	C1N	PAPS1	AN	—	Comparator 1 input
	AN7	PAPS1	AN	—	A/D channel input

Pin Name	Function	OPT	I/T	O/T	Description
PA7/NFIN/AN6	PA6	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	NFIN	PAPS1	—	CMOS	External Noise Filtered input
	AN6	PAPS1	AN	—	A/D channel input
PB0/INT1/TP2_0/ HAO/AN2	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	INT1	INTEG	ST	—	External interrupt 1 input
	TP2_0	PBPS0	ST	CMOS	TM2 input/output
	HAO	PBPS0	—	CMOS	Test pin for SA
	AN2	PBPS0	AN	—	A/D channel input
PB1/CTIN/HBO/[SCL]/ AN1	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	CTIN	—	ST	—	CAPTM input
	HBO	PBPS0	ST	CMOS	Test pin for SB
	SCL	PBPS0	ST	NMOS	I^2C clock line
	AN1	PBPS0	AN	—	A/D channel input
PB2/TP2_1/HCO/[SDA]/ AN0	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	TP2_0	PBPS0	ST	CMOS	TM2 input/output
	HCO	PBPS0	—	CMOS	Test pin for SC
	SDA	PBPS0	ST	NMOS	I^2C data/address line
	AN0	PBPS0	AN	—	A/D channel input
PB3/C1N	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	C1N	PBPS0	AN	—	Comparator 1 input
PB4/TP3_0	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	TP3_0	PBPS1	ST	CMOS	TM3 input/output
PB5/TP3_1	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	TP3_1	PBPS1	ST	CMOS	TM3 input/output
PB6/TP2_0/RX	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	TP2_0	PBPS1	ST	CMOS	TM2 input/output
	RX	PBPS1	ST	—	External UART RX serial data input pin
PB7/TP2_1/TX	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	TP2_1	PBPS1	ST	CMOS	TM2 input/output
	TX	PBPS1	—	CMOS	External UART TX serial data input pin
PC0/GAT	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	GAT	PCPS0	—	CMOS	Pulse width modulation complementary output
PC1/GAB	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	GAB	PCPS0	—	CMOS	Pulse width modulation complementary output

Pin Name	Function	OPT	I/T	O/T	Description
PC2/GBT	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	GBT	PCPS0	—	CMOS	Pulse width modulation complementary output
PC3/GBB	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	GBB	PCPS0	—	CMOS	Pulse width modulation complementary output
PC4/GCT	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	GCT	PCPS1	—	CMOS	Pulse width modulation complementary output
PC5/GCB	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	GCB	PCPS1	—	CMOS	Pulse width modulation complementary output
PD0/TCK3/TCK0/ TP0_0/AN5	PD0	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	TCK3	PDPS0	ST	—	TM3 input
	TCK0	PDPS0	ST	—	TM2 input
	TP0_0	PDPS0	ST	CMOS	TM0 input/output
	AN5	PDPS0	AN	—	A/D channel input
PD1/TP0_1/AN4	PD1	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	TP0_1	PDPS0	ST	CMOS	TM0 input/output
	AN4	PDPS0	AN	—	A/D channel input
PD2/TP1_0/[INT1]	PD2	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	TP1_0	PDPS0	ST	CMOS	TM1 input/output
	INT1	INTEG	ST	—	External interrupt 1 input
PD3/TP1_1	PD3	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	TP1_1	PDPS0	ST	CMOS	TM1 input/output
VSS	VSS	—	PWR	—	Negative power supply, ground.
AVSS	AVSS	—	PWR	—	Analog negative power supply, ground.
VDD	VDD	—	PWR	—	Positive power supply
AVDD	AVDD	—	PWR	—	Analog positive power supply, ground.

Note: I/T: Input type O/T: Output type;
OPT: Optional by configuration option (CO) or register option
PWR: Power supply ST: Schmitt trigger input AN: Analog input
CMOS: CMOS output NMOS: NMOS output
The AVSS line is internally connected to the VSS line while the AVDD line is internally connected to the VDD line.

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..... -60˚C to 150˚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 these devices. Functional operation of these devices at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect devices reliability.

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	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	—	9	12	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, INTn, TCKn, H1, H2, H3 and NFIN	—	—	0	—	$0.3V_{DD}$	V
V_{IH}	Input High Voltage for I/O Ports, INTn, TCKn, H1, H2, H3 and NFIN	—	—	$0.7V_{DD}$	—	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

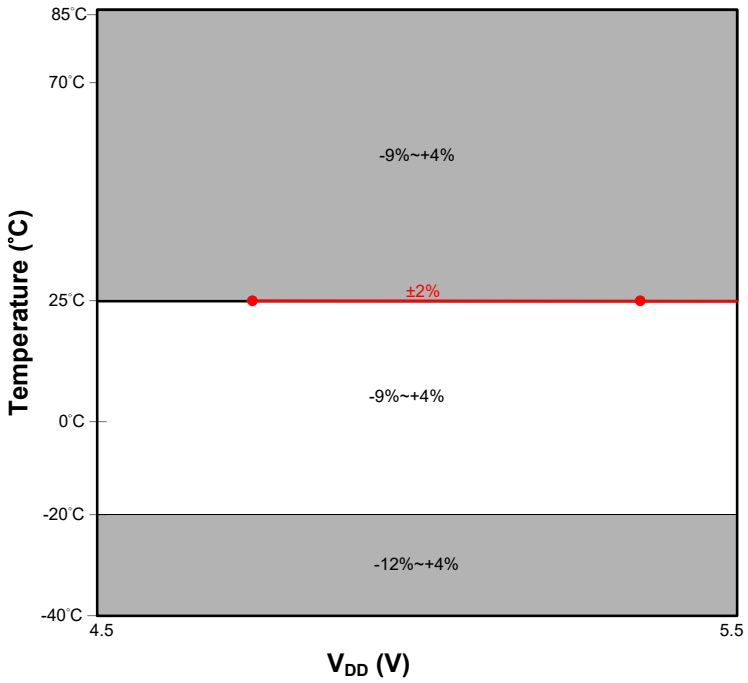
Ta=25°C

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	—	—	—	—	4	f_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	5	10	μs
t_{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Low Voltage Width to Interrupt	—	—	60	120	240	μs
t_{LVDS}	LVDO Stable Time	—	—	—	—	15	μs
t_{EERD}	EEPROM Read Time	—	—	—	2	4	t_{SYS}
t_{EEWR}	EEPROM Write Time	—	—	—	2	4	ms

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
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.

HIRC Frequency Accuracy over Device V_{DD} and Temperature



A/D Converter Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
AV_{DD}	A/D Converter Operating Voltage	—	—	V_{LVR}	5.0	5.5	V
I_{OP}	Additional Power Consumption if A/D Converter is Used	3V	—	—	0.5	—	mA
		5V		—	0.6	—	
I_{STB}	A/D Converter Standby Current	—	Digital input no change	—	—	4	μA
V_{REF}	A/D Converter Reference Voltage	—	—	2	AV_{DD}	AV_{DD}+0.1	V
t_{CONV}	A/D Conversion Time (Include Sample and Hold Time)	—	—	16			t_{ADCK}

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
DNL	Differential Non-linearity	4.5V 5.5V	V_{REF}=V_{AVDD}= V_{DD}, no load, t_{ADCK} =0.2μs, 10-bit	-1	—	+3	LSB
		4.5V 5.5V	V_{REF}=V_{AVDD}= V_{DD}, no load, t_{ADCK} =0.8μs, 12-bit				
		4.5V 5.5V	V_{REF}=V_{AVDD}= V_{DD}, no load, t_{ADCK} =0.8μs, 10-bit/12-bit				
INL	Integral Non-linearity	4.5V 5.5V	V_{REF}=V_{AVDD}= V_{DD}, no load, t_{ADCK} =0.2μs, 10-bit	-4	—	+4	LSB
		4.5V 5.5V	V_{REF}=V_{AVDD}= V_{DD}, no load, t_{ADCK} =0.8μs, 12-bit				
		4.5V 5.5V	V_{REF}=V_{AVDD}= V_{DD}, no load, t_{ADCK} =0.8μs, 10-bit/12-bit				
G_{ERR}	Gain Error	—	—	-10	—	+10	LSB
t_{ADCK}	A/D Converter Clock Period	—	12-bit	0.8	—	12.8	μs
			10-bit	0.2	—	12.8	
t_{CKH}	A/D Converter Clock High Pulse	—	—	225	—	—	ns
t_{CKL}	A/D Converter Clock Low Pulse	—	—	225	—	—	ns
t_{ST1}	A/D Converter Turn on Setup Time	—	—	2	—	—	ns
t_{ST2}	A/D Converter Start bit Setup Time	—	—	2	—	—	ns
t_{STH}	A/D Converter Start bit High Pulse	—	—	25	—	—	ns
t_{DEOC}	End of Conversion Output Delay	—	AV_{DD}=5V	—	3	—	ns
t_{DOUT}	Conversion Data Output Delay	—	AV_{DD}=5V	—	3	—	ns
t_{ON}	A/D Converter Wake-up Time	—	—	—	—	2	μs
t_{OFF}	A/D Converter 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=10pF	—	—	2	μs
R_O	D/A Output Resistance	—	—	—	10	—	kΩ

Operational Amplifier Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{OPR1}	Operating Voltage	3.3V	—	2.7	3.3	5.5	V
I_{OFF1}	Power Down Current	3.3V	—	—	—	0.1	μA
V_{OPOS1}	Input Offset Voltage	3.3V	Without calibration, AOF[4:0]=10000B	-15	—	+15	mV
V_{OPOS2}	Input Offset Voltage	3.3V	By calibration	-2	—	+2	mV
V_{CM}	Common Mode Voltage Range	3.3V	—	V_{SS}	—	V_{DD}-1.4V	V
PSRR	Power Supply Rejection Ratio	3.3V	—	90	—	96	dB

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
CMRR	Common Mode Rejuction Ratio	3.3V	V_{CM}=0~V_{DD}-1.4V	—	106	—	dB
SR	Slew Rate+, Slew Rate-	3.3V	R_L=600Ω, C_L=100pF	1.29	2.18	2.5	V/μs
GBW	Gain Band Width	3.3V	R_L=600Ω, C_L=100pF	2.05	3.70	7.16	MHz
A_{OL}	Open Loop Gain	3.3V	R_L=600Ω, C_L=100pF	—	96	—	dB
PM	Phase Margin	3.3V	R_L=600Ω, C_L=100pF	—	90	—	—

Comparator Electrical Characteristics

Ta=25°C

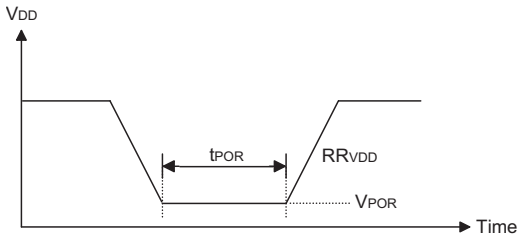
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
V_{CMP}	Comparator Operating Voltage	5V	—	2.2	5.0	5.5	V
I_{CMP}	Comparator Operatiing Current	5V	—	—	300	450	μA
I_{OFF}	Comparator Power Down Current	5V	Comparator disable	—	—	0.1	μA
V_{CMPOS}	Comparator Input Offset Voltage	5V	—	-10	—	+10	mV
V_{HYS0}	Hysteresis Width	5V	Comparator 0	TBC	100	TBC	mV
V_{HYS1}	Hysteresis Width	5V	Comparator 1,2,3	20	40	60	mV
V_{CM}	Input Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}-1.4	V
A_{OL}	Comparator Open Loop Gain	—	—	100	120	—	dB
t_{PD1}	Comparator Response Time	5V	V_M= 0~(V_{DD}-1.4)V With 10mV overdrive	—	—	1	μs
t_{PD2}	Comparator Response Time	5V	*With 100mV overdrive (note)	—	—	200	ns

*Note: Measured with comparator one input pin at V_M= (V_{DD}-1.4)/2 while the other pin input transition from V_{SS} to (V_M +100mV) or from V_{DD} to (V_M -100mV).

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

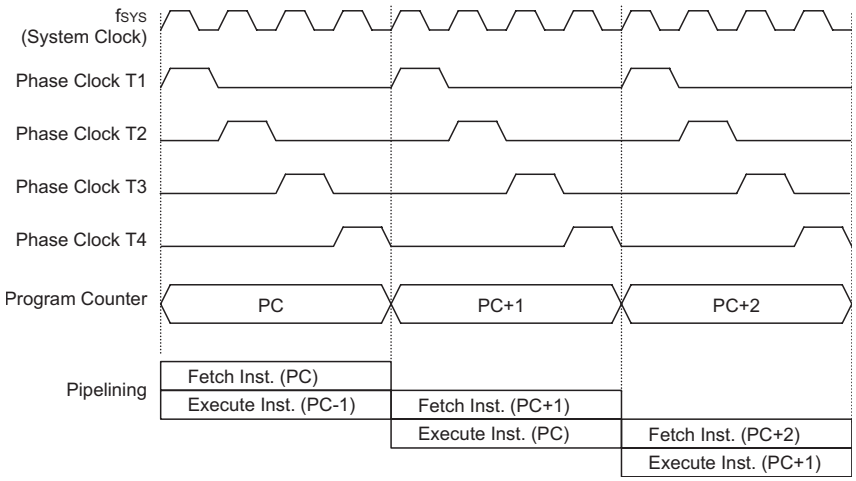


System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and Periodic 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 the device suitable for low-cost, high-volume production for controller applications.

Clocking and Pipelining

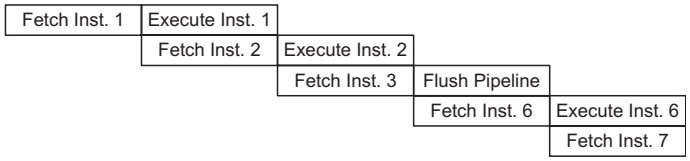
The main system clock, derived from either an HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the 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.



System Clock and Pipelining

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.

```
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

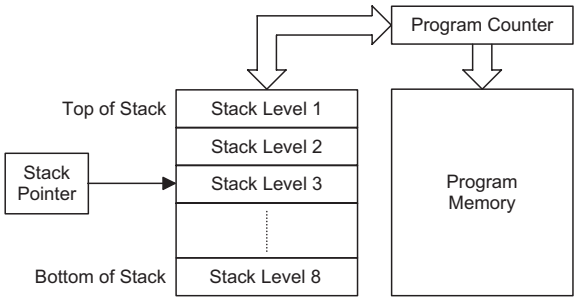
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available 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 is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



Arithmetic and Logic Unit – ALU

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

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

Flash Program Memory

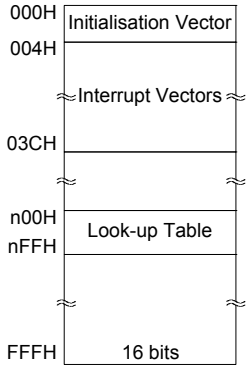
The Program Memory is the location where the user code or program is stored. For this device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate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

Structure

The Program Memory has a capacity of 4K×16 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 the 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 "TABRD [m]" or "TABRDL [m]" instructions, respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as "0".

The accompanying diagram illustrates the addressing data flow of the look-up table.

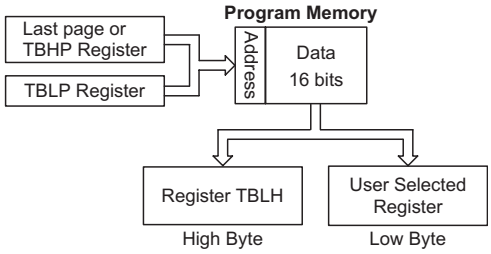


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "0F00H" which refers to the start address of the last page within the 4K words Program Memory of the device. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0F06H" 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 specific page determined by the TBHP register if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

Because the TBLH register is a read-only register and cannot be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use table read instructions. If using the table read instructions, the Interrupt Service Routines may change the

value of the TBLH and subsequently cause errors if used again by the main routine. As a rule it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

Table Read Program Example

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

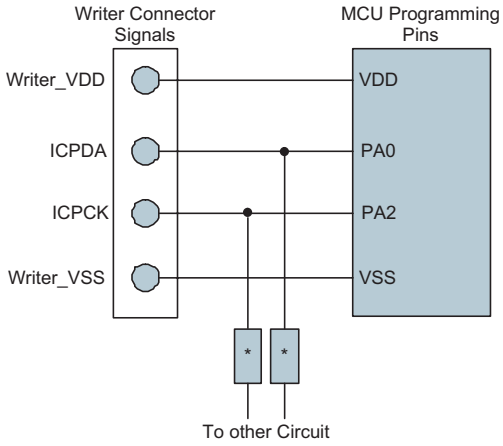
In Circuit Programming

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

The Holtek Flash MCU to Writer Programming Pin correspondence table is as follows:

Holtek Write Pins	MCU Programming Pins	Function
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Serial Clock
VDD	VDD	Power Supply
VSS	VSS	Ground

During the programming process, the user must there take care to ensure that no other outputs are connected to these two pins.The Program Memory and EEPROM data memory can both be programmed serially in-circuit using this 4-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply. The technical details regarding the in-circuit programming of the device are beyond the scope of this document and will be supplied in supplementary literature.



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

On-Chip Debug Support – OCDS

An EV chip exists for the purposes of device emulation. This EV chip device also provides an "On-Chip Debug" function to debug the device during the development process. The EV chip and the actual MCU devices are almost functionally compatible except for the "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the actual MCU device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For a more detailed OCDS description, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

RAM Data Memory

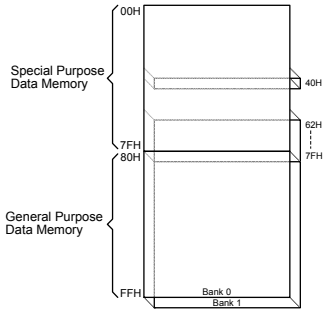
The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored. The RAM Data Memory capacity is up to 256 × 8 bits.

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 overall Data Memory is subdivided into two banks. The Special Purpose Data Memory registers are accessible in all banks, with the exception of the EEC register at address 40H, which is only accessible

in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory for the device is the address 00H.



Data Memory Structure

	Bank 0 ~ Bank1
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	INTEG1
11H	INTC0
12H	INTC1
13H	INTC2
14H	INTC3
15H	MFI0
16H	MFI1
17H	MFI2
18H	MFI3
19H	MFI4
1AH	MFI5
1BH	MFI6
1CH	PAWU
1DH	PAPU
1EH	PA
1FH	PAC
20H	PBPU
21H	PB
22H	PBC
23H	PCPU
24H	PC
25H	PCC
26H	PDPU
27H	PD
28H	PDC
29H	CAPTC0
2AH	CAPTC1
2BH	CAPTMDL
2CH	CAPTMDH
2DH	CAPTMAL
2EH	CAPTMAH
2FH	CAPTMCL
30H	CAPTMCH
31H	ADRL
32H	ADRH
33H	ADCR0
34H	ADCR1
35H	ADCR2
36H	ADDL
37H	ADLVDL
38H	ADLVDH
39H	ADHVDL
3AH	ADHVDH
3BH	PBPS0
3CH	PBPS1
3DH	PBPS2
3EH	INTEG0
3FH	CTRL

	Bank 0	Bank 1
40H	Unused	EEC
41H	EEA	
42H	EED	
43H	HDCR	
44H	HDCD	
45H	MPTC1	
46H	MPTC2	
47H	PTM1C0	
48H	PTM1C1	
49H	PTM1DL	
4AH	PTM1DH	
4BH	PTM1AL	
4CH	PTM1AH	
4DH	PTM1RPL	
4EH	PTM1RPH	
4FH	PTM0C0	
50H	PTM0C1	
51H	PTM0DL	
52H	PTM0DH	
53H	PTM0AL	
54H	PTM0AH	
55H	PTM0RPL	
56H	PTM0RPH	
57H	PTM2C0	
58H	PTM2C1	
59H	PTM2DL	
5AH	PTM2DH	
5BH	PTM2AL	
5CH	PTM2AH	
5DH	PTM2RPL	
5EH	PTM2RPH	
5FH	PWMC	
60H	DUTR0L	
61H	DUTR0H	
62H	DUTR1L	HDCT0
63H	DUTR1H	HDCT1
64H	DUTR2L	HDCT2
65H	DUTR2H	HDCT3
66H	PRDRL	HDCT4
67H	PRDRH	HDCT5
68H	PWMRL	HDCT6
69H	PWMRH	HDCT7
6AH	PWMME	HDCT8
6BH	PWMMD	HDCT9
6CH	MCF	HDCT10
6DH	MCD	HDCT11
6EH	DTS	OPOMS
6FH	PLC	OPCM
70H	IICC0	OPACAL
71H	IICC1	PTM3C0
72H	IICD	PTM3C1
73H	IICA	PTM3DL
74H	I2CTOC	PTM3DH
75H	USR	PTM3AL
76H	UCR1	PTM3AH
77H	UCR2	PTM3RPL
78H	BRG	PTM3RPH
79H	Unused	
7AH	TXR_RXR	PAPS0
7BH	CPC	PAPS1
7CH	HCHK_NUM	PCPS0
7DH	HNF_MSEL	PCPS1
7EH	NF_VIH	PDPS0
7FH	NF_VIL	PRM

: Unused, read as 00H

Special Purpose Data Memory Structure

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data 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´
org00h
start:
     mov a,04h            ; setup size of block
     mov block,a
     mov a,offset adres1  ; Accumulator loaded with first RAM address
     mov mp0,a            ; setup memory pointer with first RAM address
loop:
     clr IAR0             ; clear the data at address defined by mp0
     inc mp0              ; increment memory pointer
     sdz block            ; check if last memory location has been cleared
     jmp loop
continue:
```

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

Bank Pointer – BP

For this device, the Data Memory is divided into two banks, Bank0 and Bank1. Selecting the required Data Memory area is achieved using the Bank Pointer. Bit 0 of the Bank Pointer is used to select Data Memory Banks 0~1.

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. It should be noted that the Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within any bank. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from Bank1 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 pointers and indicate 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	×	×	×	×

"×" 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.

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is up to 64×8 bits. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped and is therefore not directly accessible in 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 way as any other Special Function Register. The EEC register however, being located in Bank1, cannot be directly addressed directly and can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the 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 Control Register List

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

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5~ 0 **D5~D0**: Data EEPROM address bit 5 ~ bit 0

• **EED Register**

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

Bit 7 ~ 0 Data EEPROM data
Data EEPROM data bit 7 ~ 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 Unimplemented, read as "0"

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

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

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

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

Note: 1. The EREN, ER, WREN, WR, RDEN and RD cannot be set high at the same time in one instruction.
2. Ensure that the fSUB clock is stable before executing the erase or write operation.
3. Ensure that the erase or write operation is totally complete before changing the contents of the EEPROM related registers.

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

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

Write Protection

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

EEPROM Interrupt

The EEPROM write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the EPWE bit in the relevant interrupt register. When an EEPROM write cycle ends, the EPWF request flag will be set. If the global and EEPROM interrupts are enabled and the stack is not full, a jump to the associated Interrupt vector will take place. When the interrupt is serviced, the EEPROM interrupt flag will be automatically reset. More details can be obtained in the Interrupt section.

Programming Considerations

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

Programming 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 write
CLR BP
MOV A, EED                   ; move read data to register
MOV READ_DATA, A
```

Writing Data to the EEPROM - polling method

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

Oscillator

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

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. 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 oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

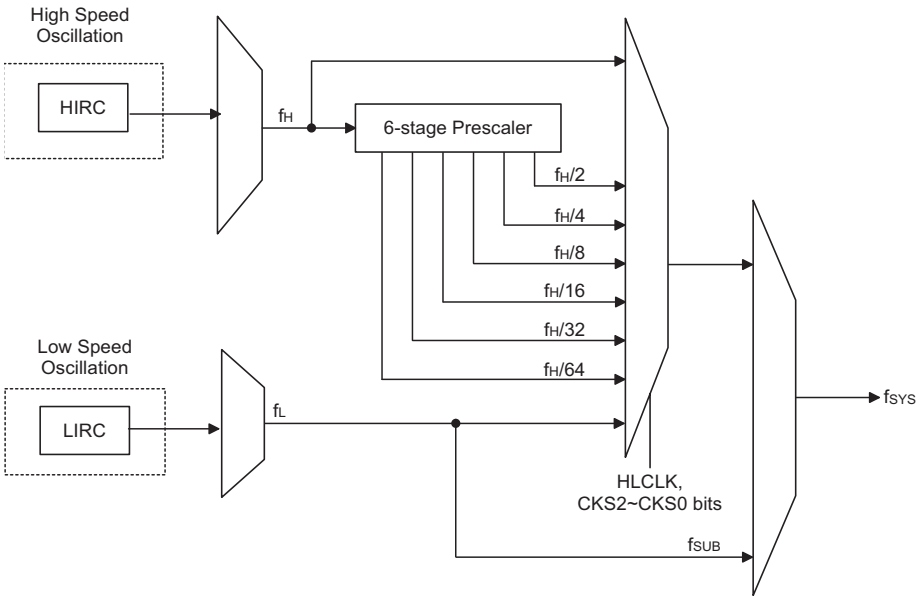
Type	Name	Freq.
Internal High Speed RC	HIRC	20MHz
Internal Low Speed RC	LIRC	32kHz

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 (LIRC) 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.

The actual source clock used for the high speed and the low speed oscillators is chosen via registers. The frequency of the slow speed or high speed system clock is also determined using the HLCLK bit and CKS2 ~ CKS0 bits in the SMOD register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.



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 frequency 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

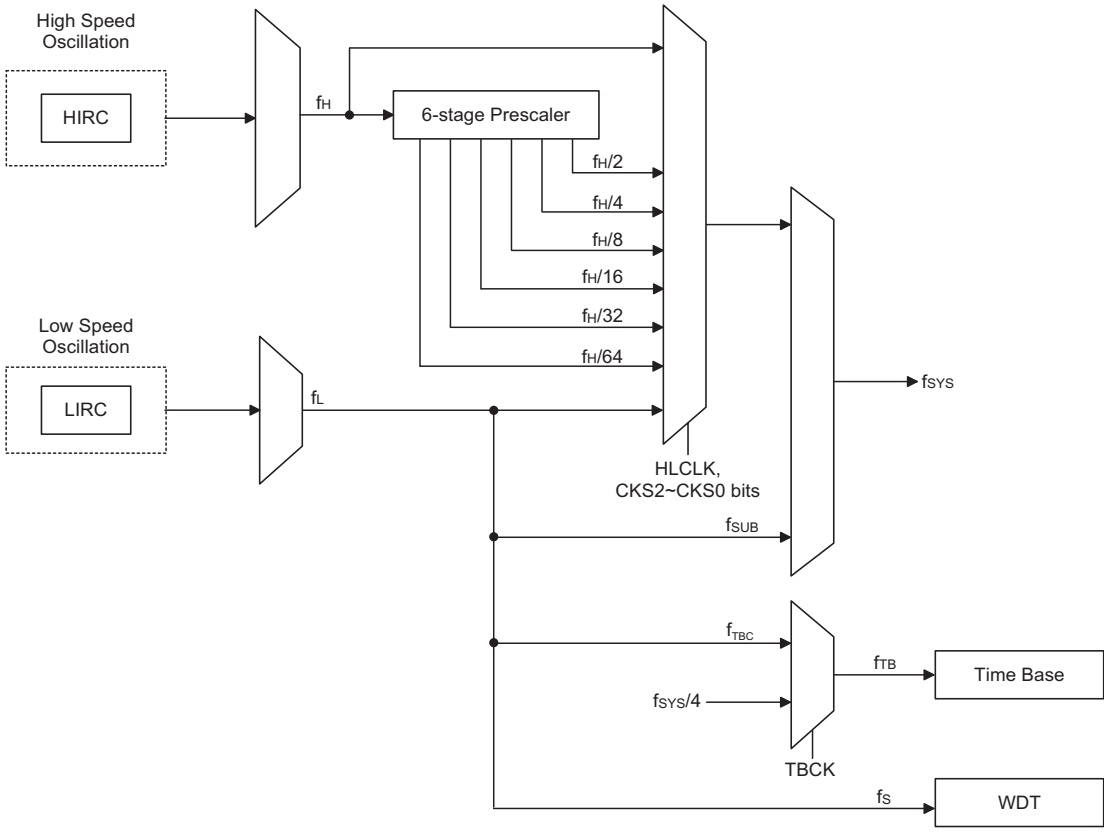
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their 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 the HIRC oscillator. The low speed system clock source can be sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.

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.



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.

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.

Operating 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	Off

NORMAL Mode

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

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_L. 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. However the f_L clocks will continue to operate.

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, TMs and IIC. In the IDLE0 Mode the system oscillator will be stopped, the Watchdog Timer clock, f_S, will be on.

IDLE1 Mode

The IDLE1 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 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 low frequency clock f_S will be on.

Control Register

The SMOD register is used to control the internal clocks within the device.

• **SMOD Register**

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

Bit 7 ~ 5 **CKS2 ~ CKS0**: The system clock selection when HLCLK is "0"
000: f_L (f_{LIRC})
001: f_L(f_{LIRC})
010: f_H/64
011: f_H/32
100: f_H/16
101: f_H/8
110: f_H/4
111: f_H/2

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

Bit 4 Unimplemented, read as "0".

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

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

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

This is the high speed system oscillator SST ready flag which indicates when the high speed system oscillator is stable after a wake-up has occurred. This flag is cleared to "0" by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable. Therefore this flag will always be read as "1" by the application program after device power-on. The flag will be low when in the SLEEP or IDLE0 Mode but after power on reset or a wake-up has occurred, the flag will change to a high level 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 the device will enter the IDLE Mode. In the IDLE1 Mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if FSYSON bit is high. If FSYSON bit is low, the CPU and the system clock will all stop in IDLE0 mode. If the bit is low the device will enter the SLEEP Mode when a HALT instruction is executed.

Bit 0 **HLCLK**: System Clock Selection
0: f_H/2 ~ 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.

• **CTRL Register**

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

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

Bit 6~3 Unimplemented, read as 0.

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

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

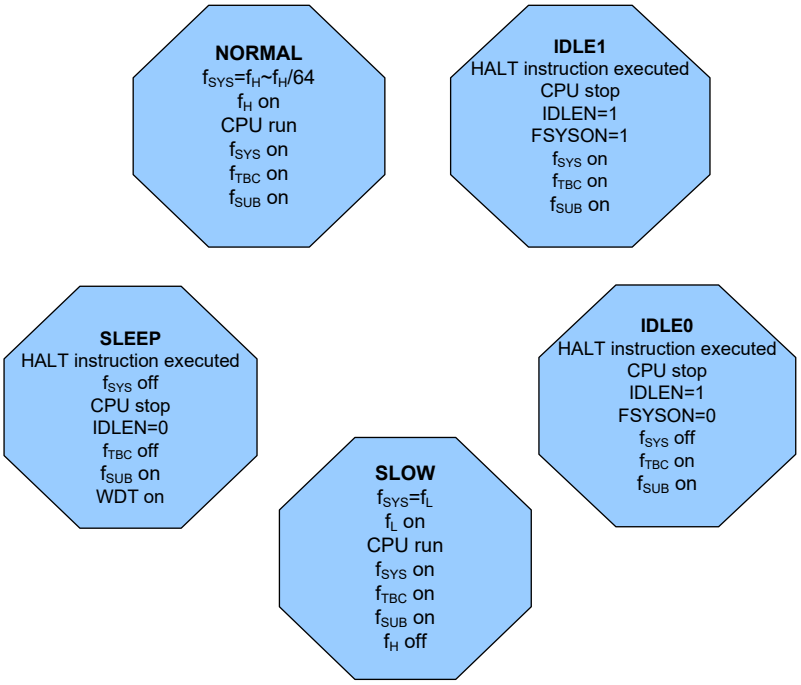
Bit 0 **WRF**: WDT Control register software reset flag
0: Not occur
1: Occurred
This bit is set to 1 by the WDT Control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

Operating Mode Switching

The device can switch between operating modes dynamically allowing the user to select the best performance/power ratio for the present task in hand. In this way microcontroller operations that do not require high performance can be executed using slower clocks thus requiring less operating current and prolonging battery life in portable applications.

In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SMOD register and FSYSON in the CTRL register.

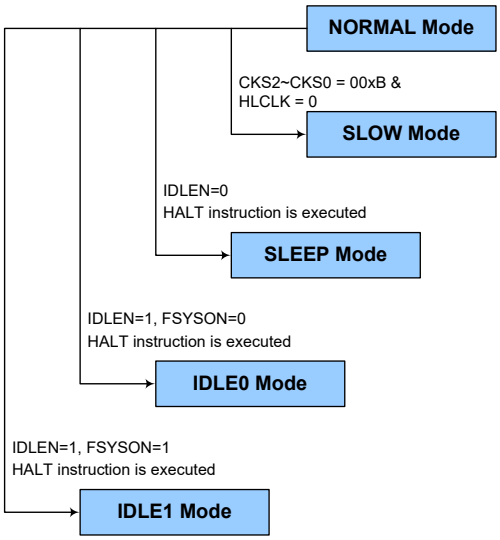
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~$f_H/64$ or f_L. If the clock is from the f_L, the high speed clock source will stop running to conserve power. When this happens it must be noted 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 the device moves between the various operating modes.



NORMAL Mode to SLOW Mode Switching

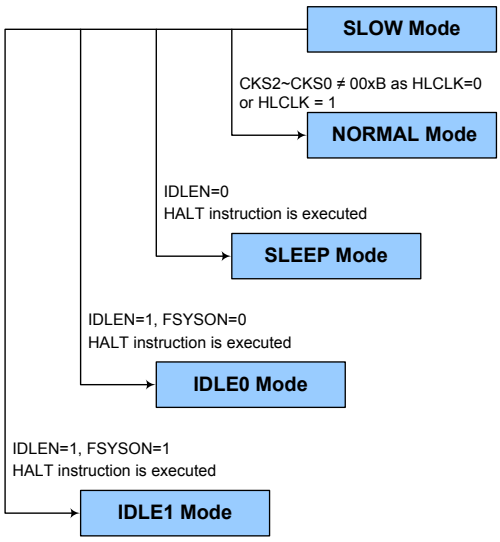
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by setting the HLCLK bit to "0" and setting the CKS2~CKS0 bits to "000" or "001" in the SMOD register.This will then use the low speed system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

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



SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system uses 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.



Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with 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. but the WDT or LVD will remain with the clock source coming from the f_L clock.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in 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 f_{TBC} and the low frequency f_L 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 the 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_{TBC} and the low frequency f_L 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 the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to devices which have different package types, as there may be unbonbed pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. In the IDLE1 Mode the 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 falling edge on Port A
- A system interrupt
- A WDT overflow

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

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a Port A pin wake-up occurs, the program will resume execution at

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal f_s clock which is in turn supplied by 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.

Note that the Watchdog Timer function is always enabled, it can be controlled by WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable operation. The WDTC register is initiated to 01010011B at any reset but keeps unchanged at the WDT time-out occurrence in a power down state.

- **WDTC Register**

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

Bit 7~ 3 **WE4 ~ WE0**: WDT function sorgware control
10101 or 01010: Enabled
Other values: Reset MCU (Reset will be active after 1~2 LIRC clock for debounce time.)
When these bits are changed by the environmental noise to reset the microcontroller, the WRF bit in the CTRL register will be set to 1.

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.

• **CTRL Register**

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

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

Bit 6~ 3 Unimplemented, read as "0"

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

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

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

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear WDT instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to enable the WDT function. When the WE4~WE0 bits value is equal to 01010B or 10101B, the WDT function is enabled. However, if the WE4~WE0 bits are changed to any other values except 01010B and 10101B, which is caused by the environmental noise, it will reset the microcontroller after 2~3 LIRC clock cycles.

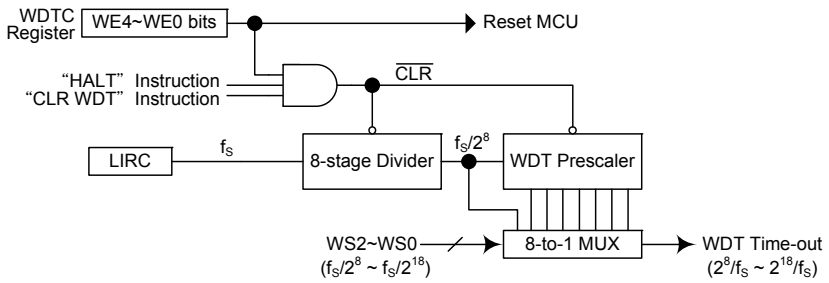
WE4 ~ WE0 Bits	WDT Function
01010B or 10101B	Enable
Any other values	Reset MCU

Watchdog Timer Enable/Disable Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT reset, which means a certain value is written into the WE4~WE0 bit filed except 01010B and 10101B, the second is using the Watchdog Timer software clear instruction and the third is via a HALT instruction.

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

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



Watchdog Timer

Reset and Initialisation

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

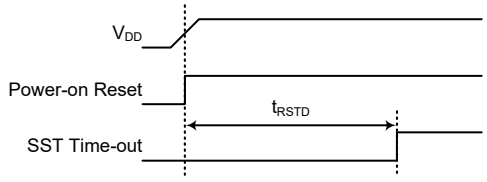
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply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

Reset Functions

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

Power-on Reset

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

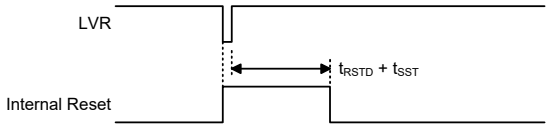


Note: t_{RSTD} is power-on delay, typical time=50ms

Power-On Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage, V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the CTRL register will also be set to1. 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 this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} is fixed at a voltage value of 3.15V by the LVS bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise, the LVR will reset the device after 2~3 LIRC clock cycles. When this happens, the LRF bit in the CTRL register will be set to 1. After power on the register will have the value of 01010101B.



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

Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7 ~ 0 **LVS7 ~ LVS0**: LVR Voltage Select control
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)
Note: S/W can write 00H~FFH to control LVR voltage, even to S/W reset MCU. If the MCU reset caused LVRC software reset, the LRF flag of CTRL register will be set.

- **CTRL Register**

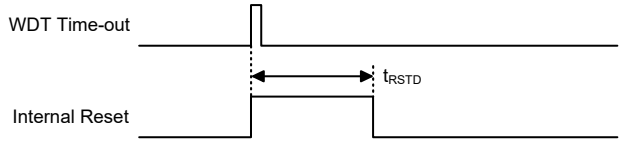
Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	—	LVRF	LRF	WRF
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	×	0	0

Bit 7 **FSYSON**: f_{SYS} Control IDLE Mode
Describe elsewhere
Bit 6~ 3 Unimplemented, read as "0"
Bit 2 **LVRF**: LVR function reset flag
0: Not occur
1: Occurred
This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.
Bit 1 **LRF**: LVR Control register software reset flag
0: Not occur
1: Occurred
This bit is set to 1 if the LVRC register contains any non defined LVR voltage register values. This in effect acts like a software reset function. This bit can only be cleared to 0 by the application program.

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

Watchdog Time-out Reset during Normal Operation

the Watchdog time-out flag TO will be set to "1" when Watchdog time-out Reset during normal operation.

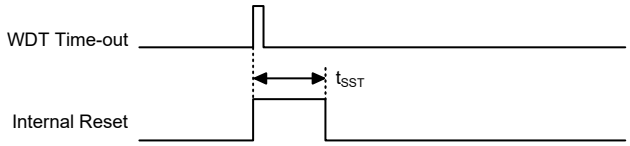


Note: t_{RSTD} is power-on delay, typical time=16.7ms

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 the HIRC.
The t_{SST} is 1~2 clock for the LIRC.

WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	LVR reset during NORMAL or SLOW Mode operation
1	u	WDT time-out reset during NORMAL or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

Note: "u" stands for unchanged

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

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT	Clear after reset, WDT begins counting
Timer Modules	Timer Modules will be turned off
Input/Output Ports	I/O ports will be setup as inputs and AN0~AN7 as A/D input pins
Stack Pointer	Stack Pointer will point to the top of the stack

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

Register	Power On Reset	WDT Time-out (Normal Operation)	LVR Reset	WDT Time-out (IDLE/SLEEP mode)
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	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---- xxxx	---- xxxx	---- xxxx	---- uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--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 ----
INTEG1	---- --00	---- --00	---- --00	---- --uu
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 --00	-000 --00	-000 --00	-uuu --uu
MFI1	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI2	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI3	--00 --00	--00 --00	--00 --00	--uu --uu
MFI4	--00 --00	--00 --00	--00 --00	--uu --uu
MFI5	--00 --00	--00 --00	--00 --00	--uu --uu
MFI6	--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	--00 0000	--00 0000	--00 0000	--uu uuuu
PC	--11 1111	--11 1111	--11 1111	--uu uuuu
PCC	--11 1111	--11 1111	--11 1111	--uu uuuu
PDPU	---- 0000	---- 0000	---- 0000	---- uuuu
PD	---- 1111	---- 1111	---- 1111	---- uuuu
PDC	---- 1111	---- 1111	---- 1111	---- uuuu
CAPTC0	0000 0-00	0000 0-00	0000 0-00	uuuu u-uu
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	0000 uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	LVR Reset	WDT Time-out (IDLE/SLEEP mode)
CAPTMAL	0000 0000	0000 0000	0000 0000	uuuu uuuu
CAPTMAH	0000 0000	0000 0000	0000 0000	0000 uuuu
CAPTMCL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
CAPTMCH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL (ADRFS=0)	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRL (ADRFS=1)	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH (ADRFS=0)	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH (ADRFS=1)	---- xxxx	---- xxxx	---- xxxx	---- uuuu
ADCR0	0110 0000	0110 0000	0110 0000	uuuu uuuu
ADCR1	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADCR2	---- --00	---- --00	---- --00	---- --uu
ADDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADLVDL (ADRFS=0)	0000 ----	0000 ----	0000 ----	uuuu ----
ADLVDL (ADRFS=1)	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADLVDH (ADRFS=0)	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADLVDH (ADRFS=1)	---- 0000	---- 0000	---- 0000	---- uuuu
ADHVDL (ADRFS=0)	0000 ----	0000 ----	0000 ----	uuuu ----
ADHVDL (ADRFS=1)	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADHVDH (ADRFS=0)	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADHVDH (ADRFS=1)	---- 0000	---- 0000	---- 0000	---- uuuu
PBPS0	-000 0000	-000 0000	-000 0000	-uuu uuuu
PBPS1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBPS2	---- --00	---- --00	---- --00	---- --uu
INTEG0	-000 0000	-000 0000	-000 0000	-uuu uuuu
CTRL	0--- -x00	0--- -x00	0--- -x00	u--- -uuu
EEA	--xx xxxx	--xx xxxx	--xx xxxx	--uu uuuu
EED	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
EEC	---- 0000	---- 0000	---- 0000	---- uuuu
HDCR	0001 0000	0001 0000	0001 0000	uuuu uuuu
HDCD	---- -000	---- -000	---- -000	---- -uuu
MPTC1	0000 00--	0000 00--	0000 00--	uuuu uu--
MPTC2	---0 0000	---0 0000	---0 0000	---u uuuu
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

Register	Power On Reset	WDT Time-out (Normal Operation)	LVR Reset	WDT Time-out (IDLE/SLEEP mode)
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
PTM1C0	0000 0---	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DH	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1AH	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0C0	0000 0---	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DH	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0AH	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2C0	0000 0---	0000 0---	0000 0---	uuuu u---
PTM2C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DH	---- --00	---- --00	---- --00	---- --uu
PTM2AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2AH	---- --00	---- --00	---- --00	---- --uu
PTM2RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	---- --00	---- --00	---- --00	---- --uu
PTM3C0	0000 0---	0000 0---	0000 0---	uuuu u---
PTM3C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3DH	---- --00	---- --00	---- --00	---- --uu
PTM3AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3AH	---- --00	---- --00	---- --00	---- --uu
PTM3RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3RPH	---- --00	---- --00	---- --00	---- --uu
PWMC	0000 0000	0000 0000	0000 0000	uuuu uuuu
DUTR0L	0000 0000	0000 0000	0000 0000	uuuu uuuu
DUTR0H	---- --00	---- --00	---- --00	---- --uu
DUTR1L	0000 0000	0000 0000	0000 0000	uuuu uuuu
DUTR1H	---- --00	---- --00	---- --00	---- --uu
DUTR2L	0000 0000	0000 0000	0000 0000	uuuu uuuu
DUTR2H	---- --00	---- --00	---- --00	---- --uu
PRDRL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PRDRH	---- --00	---- --00	---- --00	---- --uu

Register	Power On Reset	WDT Time-out (Normal Operation)	LVR Reset	WDT Time-out (IDLE/SLEEP mode)
PWMRL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWMRH	---- --00	---- --00	---- --00	---- --uu
PWMME	--00 0000	--00 0000	--00 0000	--uu uuuu
PWMMD	--00 0000	--00 0000	--00 0000	--uu uuuu
MCF	---- 0100	---- 0100	---- 0100	---- uuuu
MCD	--00 0111	--00 0111	--00 0111	--uu uuuu
DTS	0000 0000	0000 0000	0000 0000	uuuu uuuu
PLC	--00 0000	--00 0000	--00 0000	--uu uuuu
IICC0	---- 000-	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
I2CTOC	0000 0000	0000 0000	0000 0000	uuuu uuuu
USR	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
CPC	0000 0000	0000 0000	0000 0000	uuuu uuuu
HCHK_NUM	---0 0000	---0 0000	---0 0000	---u uuuu
HNF_MSEL	---- 0000	---- 0000	---- 0000	---- uuuu
HNF_VIH	00-1 1001	00-1 1001	00-1 1001	uu-u uuuu
HNF_VIL	00-0 1010	00-0 1010	00-0 1010	uu-u uuuu
OPOMS	00-- -010	00-- -010	00-- -010	uu-- -uuu
OPCM	0000 0000	0000 0000	0000 0000	uuuu uuuu
OPACAL	-001 0000	-001 0000	-001 0000	-uuu uuuu
PAPS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPS1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCPS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCPS1	---- 0000	---- 0000	---- 0000	---- uuuu
PDPS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PRM	-000 0000	-000 0000	-000 0000	-uuu uuuu

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

Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA, PB PC and 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.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	D7	D6	D5	D4	D3	D2	D1	D0
PAC	D7	D6	D5	D4	D3	D2	D1	D0
PAPU	D7	D6	D5	D4	D3	D2	D1	D0
PAWU	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
PBPU	D7	D6	D5	D4	D3	D2	D1	D0
PC	—	—	D5	D4	D3	D2	D1	D0
PCC	—	—	D5	D4	D3	D2	D1	D0
PCPU	—	—	D5	D4	D3	D2	D1	D0
PD	—	—	—	—	D3	D2	D1	D0
PDC	—	—	—	—	D3	D2	D1	D0
PDPU	—	—	—	—	D3	D2	D1	D0

I/O Register List

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 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

Bit 7 ~ 0 I/O Port A bit 7~ bit 0 Pull-High Control
0: Disable
1: Enable

• **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

Bit 7 ~ 0 I/O Port B bit 7~ bit 0 Pull-High Control
0: Disable
1: Enable

• **PCPU Register**

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

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 ~ 0 I/O Port C bit 5~ bit 0 Pull-High Control
0: Disable
1: Enable

• **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 I/O Port D bit 3~ bit 0 Pull-High Control
0: Disable
1: Enable

Port A Wake-up

The HALT instruction forces the microcontroller into the SLEEP or IDLE Mode which preserves power, a feature that is important for battery and other low-power applications. Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the Port A pins from high to low. This function is especially suitable for applications that can be woken up via external switches. Each pin on Port A can be selected individually to have this wake-up feature using the PAWU register.

• **PAWU Register**

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

Bit 7 ~ 0 I/O 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

Bit 7 ~ 0 I/O Port A bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

• PBC Register

Bit	7	6	5	4	3	2	1	0
Name	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 B bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

• PCC Register

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

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 ~ 0 I/O Port C bit 5~bit 0 Input/Output Control
0: Output
1: Input

• 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 D bit 3~bit 0 Input/Output Control
0: Output
1: Input

Pin-remapping Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function.

Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. The way in which the pin function of each pin is selected is different for each function and a priority order is established where more than one pin function is selected simultaneously. Additionally there are a series of PRM register to establish certain pin functions.

Pin-remapping Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. Some devices include PRM register which can select the functions of certain pins.

- **PRM Register**

Bit	7	6	5	4	3	2	1	0
Name	—	INT1PS	RXPS	TXPS0	C1NPS1	C1NPS0	SDAPS	SCLPS
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 **INT1PS**: INT1 Pin Select
0: INT1 on PB0
1: INT1 on PD2

Bit 5 **RXPS**: RX Pin Select
0: RX on PB6
1: RX on PA0

Bit 4 **TXPS**: TX Pin Select
0: TX on PB7
1: TX on PA2

Bit 3~2 **C1NPS1~C1NPS0**: C1N Pin Select
00: C1N on PB3
01: C1N on PA6
10: C1N on PA4
11: Reserved

Bit 1 **SDAPS**: SDA Pin Select
0: SDA on PA0
1: SDA on PB2

Bit 0 **SCLPS**: SCL Pin Select
0: SCL on PA2
1: SCL on PB1

Pin-sharing Functions

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAPS0	PA3S1	PA3S0	PA2S1	PA2S0	PA1S1	PA1S0	PA0S1	PA0S0
PAPS1	PA7S1	PA7S0	PA6S1	PA6S0	PA5S1	PA5S0	PA4S1	PA4S0
PBPS0	—	PB2S2	PB2S1	PB2S0	PB1S1	PB1S0	PB0S1	PB0S0
PBPS1	PB6S1	PB6S0	PB5S1	PB5S0	PB4S1	PB4S0	PB3S1	PB3S0
PBPS2	—	—	—	—	—	—	PB7S1	PB7S0
PCPS0	PC3S1	PC3S0	PC2S1	PC2S0	PC1S1	PC1S0	PC0S1	PC0S0
PCPS1	—	—	—	—	PC5S1	PC5S0	PC4S1	PC4S0
PDPS0	PD3S1	PD3S0	PD2S1	PD2S0	PD1S1	PD1S0	PD0S1	PD0S0

Pin-sharing Function Register List

• PAPS0 Register

Bit	7	6	5	4	3	2	1	0
Name	PA3S1	PA3S0	PA2S1	PA2S0	PA1S1	PA1S0	PA0S1	PA0S0
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 **PA3S1~PA3S0**: PA3 pin-shared control
00: PA3/TCK1/H1
01: C1P
10: PA3/TCK1/H1
11: PA3/TCK1/H1

Bit 5~4 **PA2S1~PA2S0**: PA2 pin-shared control
00: PA2
01: SCL
10: [TX] – if the pin-remapping function is enabled.
11: PA2
If the relevant pin-remapping function is not enabled, the pin function is used as the default function which is specified by the "00" configuration.

Bit 3~2 **PA1S1~PA1S0**: PA1 pin-shared control
00: PA1/TCK2
01: AN3/AP
10: PA1/TCK2
11: PA1/TCK2

Bit 1~0 **PA0S1~PA0S0**: PA0 pin-shared control
00: PA0
01: SDA
10: [RX] – if the pin-remapping function is enabled.
11: PA0
If the relevant pin-remapping function is not enabled, the pin function is used as the default function which is specified by the "00" configuration.

• **PAPS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PA7S1	PA7S0	PA6S1	PA6S0	PA5S1	PA5S0	PA4S1	PA4S0
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 **PA7S1~PA7S0**: PA7 pin-shared control
00: PA7/NFIN
01: AN6
10: PA7/NFIN
11: PA7/NFIN

Bit 5~4 **PA6S1~PA6S0**: PA6 pin-shared control
00: PA6
01: [C1N] – if the pin-remapping function is enabled.
10: AN7
11: PA6
If the relevant pin-remapping function is not enabled, the pin function is used as the default function which is specified by the "00" configuration.

Bit 3~2 **PA5S1~PA5S0**: PA5 pin-shared control
00: PA5/H3
01: PA5/H3
10: C3P
11: PA5/H3

Bit 1~0 **PA4S1~PA4S0**: PA4 pin-shared control
00: PA4/H2
01: PA4/H2
10: C2P
11: [C1N] – if the pin-remapping function is enabled.
If the relevant pin-remapping function is not enabled, the pin function is used as the default function which is specified by the "00" configuration.

• **PBPS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PB2S2	PB2S1	PB2S0	PB1S1	PB1S0	PB0S1	PB0S0
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 6~4 **PB2S2~PB2S0**: PB2 pin-shared control
000: PB2
001: HCO
010: AN0
011: TP2_1
100: [SDA] – if the pin-remapping function is enabled.
101~111: PB2
If the relevant pin-remapping function is not enabled, the pin function is used as the default function which is specified by the "000" configuration.

Bit 3~2 **PB1S1~PB1S0**: PB1 pin-shared control
00: PB1/CTIN
01: HBO
10: AN1
11: [SCL] – if the pin-remapping function is enabled.
If the relevant pin-remapping function is not enabled, the pin function is used as the default function which is specified by the "00" configuration.

Bit 1~0 **PB0S1~PB0S0**: PB0 pin-shared control
00: PB0/INT1
01: HAO
10: AN2
11: TP2_0

• **PBPS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PB6S1	PB6S0	PB5S1	PB5S0	PB4S1	PB4S0	PB3S1	PB3S0
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 **PB6S1~PB6S0**: PB6 pin-shared control
00: PB6
01: RX – if the pin-remapping function is disabled.
10: TP2_0
11: PB6
If the relevant pin-remapping function is enabled to remap to other pins, this pin function is used as the default function which is specified by the "00" configuration.

Bit 5~4 **PB5S1~PB5S0**: PB5 pin-shared control
00: PB5
01: TP3_1
10: PB5
11: PB5

Bit 3~2 **PB4S1~PB4S0**: PB4 pin-shared control
00: PB4
01: TP3_0
10: PB4
11: PB4

Bit 1~0 **PB3S1~PB3S0**: PB3 pin-shared control
00: PB3
01: C1N – if the pin-remapping function is disabled.
10: PB3
11: PB3
If the relevant pin-remapping function is enabled to remap to other pins, this pin function is used as the default function which is specified by the "00" configuration.

• **PBPS2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PB7S1	PB7S0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as 0

Bit 1~0 **PB7S1~PB7S0**: PB7 pin-shared control
00: PB7
01: TX – if the pin-remapping function is disabled.
10: TP2_1
11: PB7
If the relevant pin-remapping function is enabled to remap to other pins, this pin function is used as the default function which is specified by the "00" configuration.

• **PCPS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PC3S1	PC3S0	PC2S1	PC2S0	PC1S1	PC1S0	PC0S1	PC0S0
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 **PC3S1~PC3S0**: PC3 pin-shared control
00: PC3
01: PC3
10: GBB
11: PC3

Bit 6~5 **PC2S1~PC2S0**: PC2 pin-shared control
00: PC2
01: PC2
10: GBT
11: PC2

Bit 3~2 **PC1S1~PC1S0**: PC1 pin-shared control
00: PC1
01: PC1
10: GAB
11: PC1

Bit 1~0 **PC0S1~PC0S0**: PC0 pin-shared control
00: PC0
01: PC0
10: GAT
11: PC0

• **PCPS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PC5S1	PC5S0	PC4S1	PC4S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as 0

Bit 3~2 **PC5S1~PC5S0**: PC5 pin-shared control
00: PC5
01: PC5
10: GCB
11: PC5

Bit 1~0 **PC4S1~PC4S0**: PC4 pin-shared control
00: PC4
01: PC4
10: GCT
11: PC4

• **PDPS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PD3S1	PD3S0	PD2S1	PD2S0	PD1S1	PD1S0	PD0S1	PD0S0
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 **PD3S1~PD3S0**: PD3 pin-shared control
00: PD3
01: TP1_1
10: PD3
11: PD3

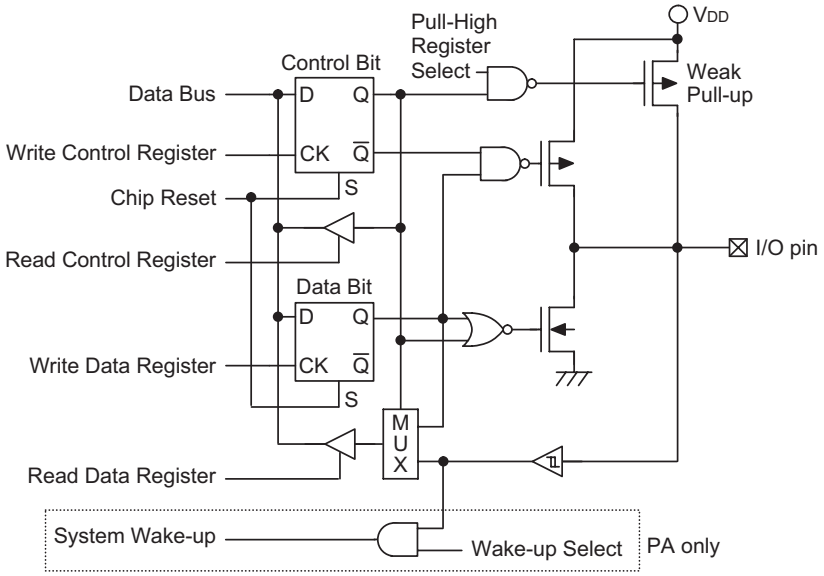
Bit 6~5 **PD2S1~PD2S0**: PD2 pin-shared control
00: PD2/[INT1] – if the pin-remapping function is enabled.
01: TP1_0
10: PD2/[INT1] – if the pin-remapping function is enabled.
11: PD2/[INT1] – if the pin-remapping function is enabled.
If the relevant pin-remapping function is not enabled, the pin function is used as the default function which is specified by the "000" configuration.

Bit 3~2 **PD1S1~PD1S0**: PD1 pin-shared control
00: PD1
01: AN4
10: TP0_1
11: PD1

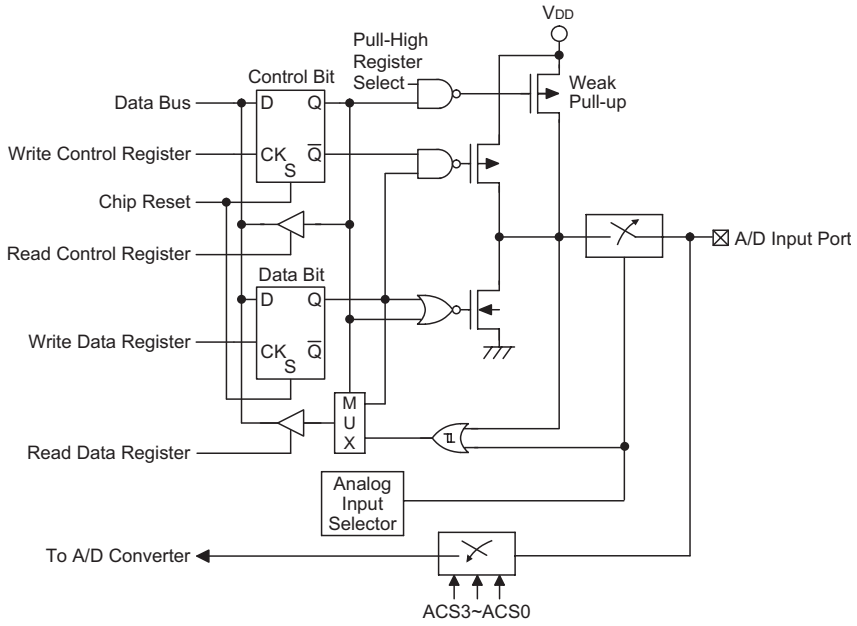
Bit 1~0 **PD0S1~PD0S0**: PD0 pin-shared control
00: PD0
01: AN5
10: TP0_0
11: PD0

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 the 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 two individual 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 Periodic TM section.

Introduction

The device contains four TMs with each TM having a reference name of TMn. The TM0 and TM1 are 16-bit Periodic Type TMs (PTM) while the TM2 and TM3 are 10-bit Periodic Type TMs (PTM). The common features to the 16-bit and 10-bit Periodic TMs will be described in this section and the detailed operation will be described in corresponding sections. The main features are summarised in the accompanying table.

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

TM Function Summary

TM0	TM1	TM2	TM3
16-bit PTM	16-bit PTM	10-bit PTM	10-bit PTM

TM Name/Type Reference

TM Operation

The TMs 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 PTnCK2~PTnCK0 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. 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. The TCKn pin can also be used as an input capture trigger pin.

TM Interrupts

The TMs have two internal interrupts, the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs. When a TM interrupt is generated, it can be used to clear the counter and also to change the state of the TM output pin.

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 PTnCK2~PTnCK0 bits in the PTMnC0 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 PTnCK2~PTnCK0 bits. The TM input pin can be chosen to have either a rising or falling active edge.

The TMs each have two output pins. 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. The number of output pins for each TM type is different, the details are provided in the accompanying table.

All TM output pin names have an "_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.

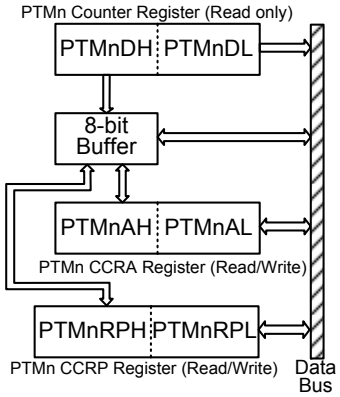
TM0	TM1	TM2	TM3
TCK0; TP0_0, TP0_1	TCK1; TP1_0, TP1_1	TCK2; TP2_0, TP2_1	TCK3; TP3_0, TP3_1

TM External Pins

Programming Considerations

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

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



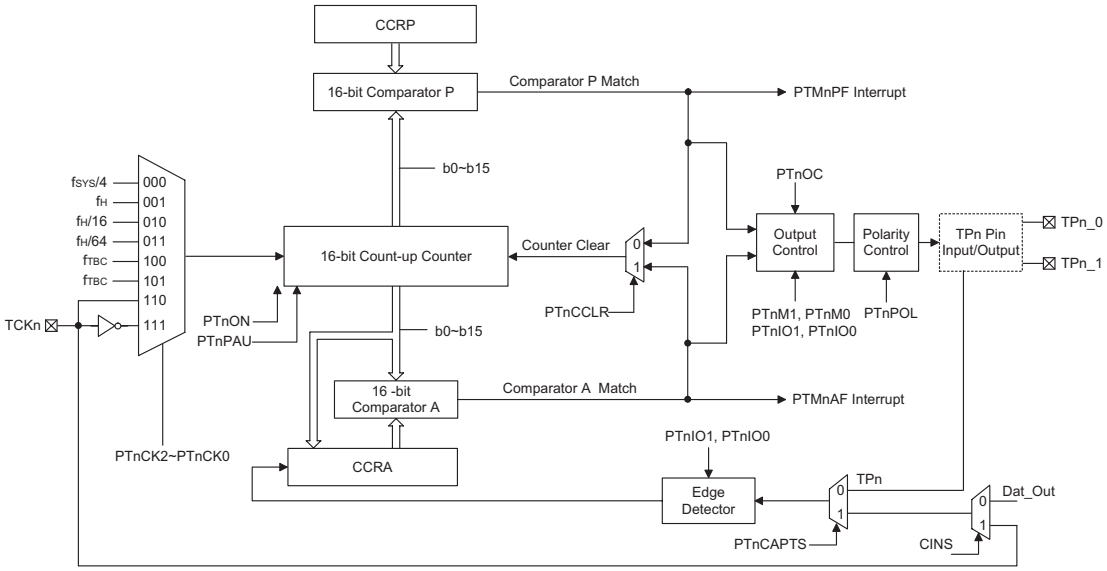
The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - Step 1. Write data to Low Byte PTMnAL or PTMnRPL
    – Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte PTMnAH or PTMnRPH
    – 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 or CCRP
  - Step 1. Read data from the High Byte PTMnDH, PTMnAH or PTMnRPH
    – 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 PTMnDL, PTMnAL or PTMnRPL
    – This step reads data from the 8-bit buffer.

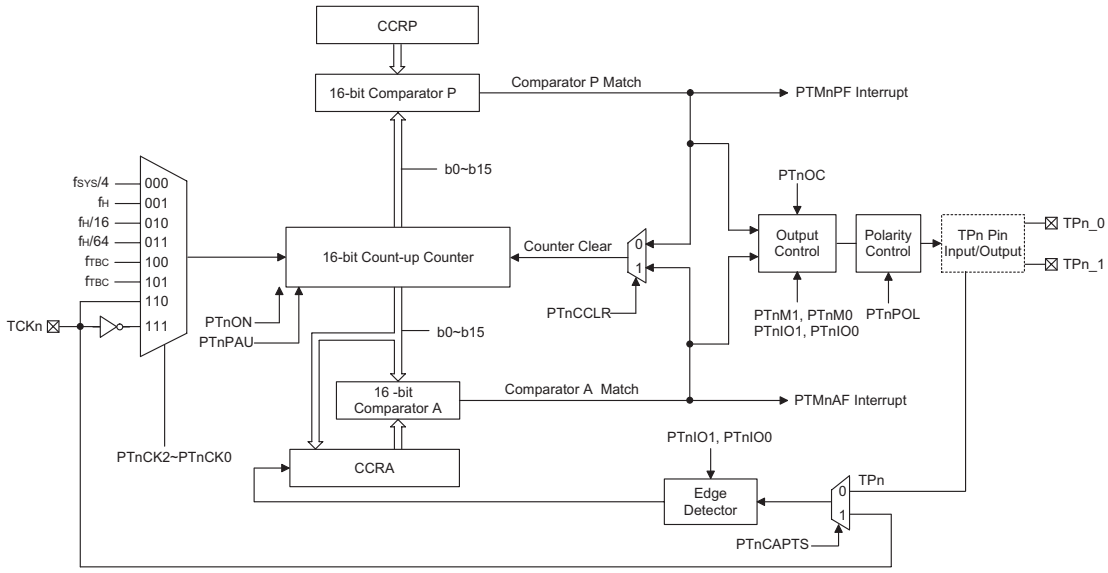
Periodic Type TM – PTM

The Periodic Type TM contains five operating modes, which are Compare Match Output, Timer/ Event Counter, Capture Input, Single Pulse Output and PWM Output modes. The Periodic TM can also be controlled with one or two external input pins and can drive one or two external output pin.

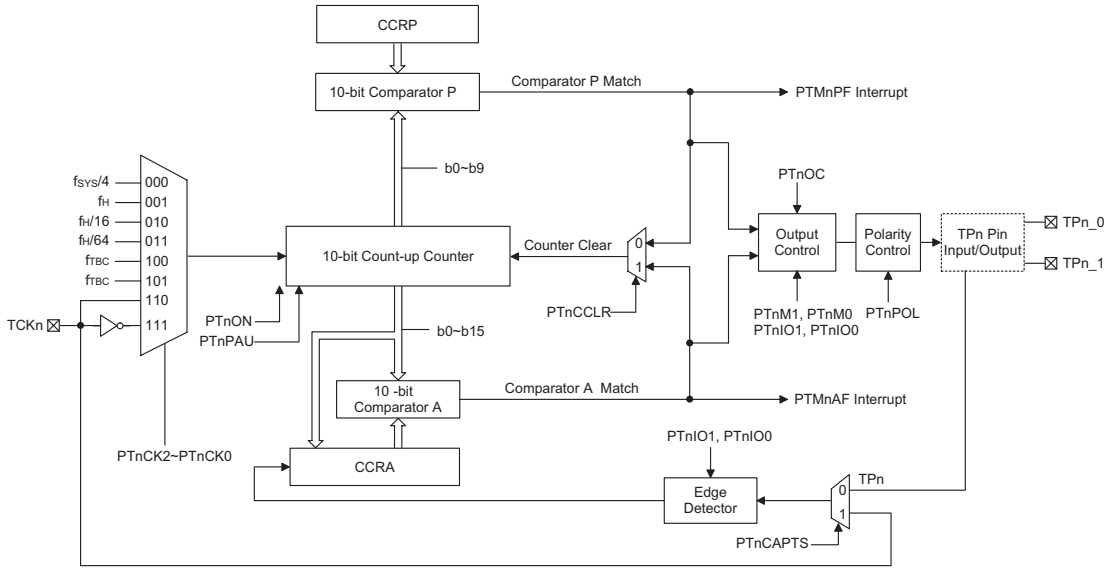
Name	TM No.	TM Input Pin	TM Output Pin
16-bit PTM	0, 1	TCK0; TCK1	TP0_0, TP0_1; TP1_0, TP1_1
10-bit PTM	2, 3	TCK2; TCK3	TP2_0, TP2_1; TP3_0, TP3_1



Periodic Type TM Block Diagram (n=0)



Periodic Type TM Block Diagram (n=1)



Periodic Type TM Block Diagram (n=2 or 3)

Periodic TM Operation

There are two sizes of Periodic TMs, one is 10-bit wide and the other is 16-bit wide. At its core is a 10 or 16-bit count-up counter which is driven by a user selectable internal or external clock source. There are two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with the CCRA and CCRP registers.

The only way of changing the value of the 10 or 16-bit counter using the application program, is to clear the counter by changing the PTnON 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 Periodic 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 the output pin. All operating setup conditions are selected using relevant internal registers.

Periodic Type TM Register Description

Overall operation of the Periodic TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10 or 16-bit value, while two read/write register pairs exist to store the internal 10 or 16-bit CCRA and CCRP 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
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	D15	D14	D13	D12	D11	D10	D9	D8

Periodic TM Register List (n=0 or 1)

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

Periodic TM Register List (n=2 or 3)

- **PTMnDL 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 **TMn Counter Low Byte Register bit 7 ~ bit 0**
TMn 10 or 16-bit Counter bit 7 ~ bit 0

- **PTMnDH Register (n=0 or 1)**

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 TMn Counter High Byte Register bit 7 ~ bit 0
TMn 16-bit Counter bit 15 ~ bit 8

- **PTMnDH Register (n=2 or 3)**

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

- **PTMnAL 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 TMn CCRA Low Byte Register bit 7 ~ bit 0
TMn 10 or 16-bit CCRA bit 7 ~ bit 0

• **PTMnAH Register (n=0 or 1)**

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 TMn CCRA High Byte Register bit 7 ~ bit 0
TMn 16-bit CCRA bit 15 ~ bit 8

• **PTMnAH Register (n=2 or 3)**

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 7~0 TMn CCRA High Byte Register bit 1 ~ bit 0
TMn 10-bit CCRA bit 9 ~ bit 8

• **PTMnRPL 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 TMn CCRP Low Byte Register bit 7 ~ bit 0
TMn 10 or 16-bit CCRP bit 7 ~ bit 0

• **PTMnRPH Register (n=0 or 1)**

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 TMn CCRP High Byte Register bit 7 ~ bit 0
TMn 16-bit CCRP bit 15 ~ bit 8

• **PTMnRPH Register (n=2 or 3)**

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 7~0 TMn CCRP High Byte Register bit 1 ~ bit 0
TMn 10-bit CCRP bit 9 ~ bit 8

• **PTMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTnPAU**: 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 TMn 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 **PTnCK2~TnCK0**: Select TMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: f_{TBC}
110: TCKn rising edge clock
111: TCKn falling edge clock
These three bits are used to select the clock source for the TMn. 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 **PTnON**: 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 until the bit returns high again. 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 PTnOC bit, when the PTnON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

• **PTMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
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 **PTnM1~TnM0**: Select TMn Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode
These bits setup the required operating mode for the TMn. To ensure reliable operation the TMn should be switched off before any changes are made to the PTnM1 and PTnM0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5~4 **PTnIO1~TnIO0**: Select TPn_0, TPn_1 Pin Function

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

PWM Mode/ Single Pulse Output Mode
- 00: PWM Output inactive state
- 01: PWM Output active state
- 10: PWM output
- 11: Single Pulse output

Capture Input Mode
- 00: Input capture at rising edge of TPn_0, TPn_1, TCKn
- 01: Input capture at falling edge of TPn_0, TPn_1, TCKn
- 10: Input capture at falling/rising edge of TPn_0, TPn_1, TCKn
- 11: Input capture disabled

Timer/Counter Mode
- Unused

These two bits are used to determine how the TMn pin functions 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 PTnIO1 and PTnIO0 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 PTnOC bit in the PTMnC1 register. Note that the output level requested by the PTnIO1 and PTnIO0 bits must be different from the initial value setup using the PTnOC 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 PTnON bit from low to high.

In the PWM Mode, the PTnIO1 and PTnIO0 bits determine how the TMn 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 PTnIO1 and PTnIO0 bits only after the TMn has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the TMn is running.

In the Capture Input Mode, the capture input trigger source is selected by the PTnCAPTS bit in the PTMnC1 register for TM1, TM2 and TM3. However, the capture input trigger source is also determined by the CINS bit in the NF_VIH register together with the PT0CAPTS bit in the PTM0C1 register for TM0.

Bit 3 **PTnOC**: TPn_0, TPn_1 Output Control

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

PWM Mode/ Single Pulse Output 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/ Single Pulse Output 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 the 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 **PTnPOL**: TPn_0, TPn_1 Output Polarity Control
0: Non-inverted
1: Inverted
This bit controls the polarity of the TPn_0 or TPn_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 **PTnCAPTS**: TMn Capture Input trigger source select
0: From TPn_0, TPn_1
1: From TCKn
This bit is used to select the TMn capture input trigger source. However, for TM0, the capture input trigger source is also determined by the CINS bit in the NF_VIH register. When the PT0CAPTS bit is set to 1, the capture input trigger source can be TCK0 or noise filtered Dat_Out signal determined using the CINS bit.

Bit 0 **PTnCCLR**: TMn Counter Clear condition select
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 Periodic TMn contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the PTnCCLR 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 PTnCCLR bit is not used in the PWM Mode, Single Pulse or Input Capture Mode.

Periodic Type TM Operating Modes

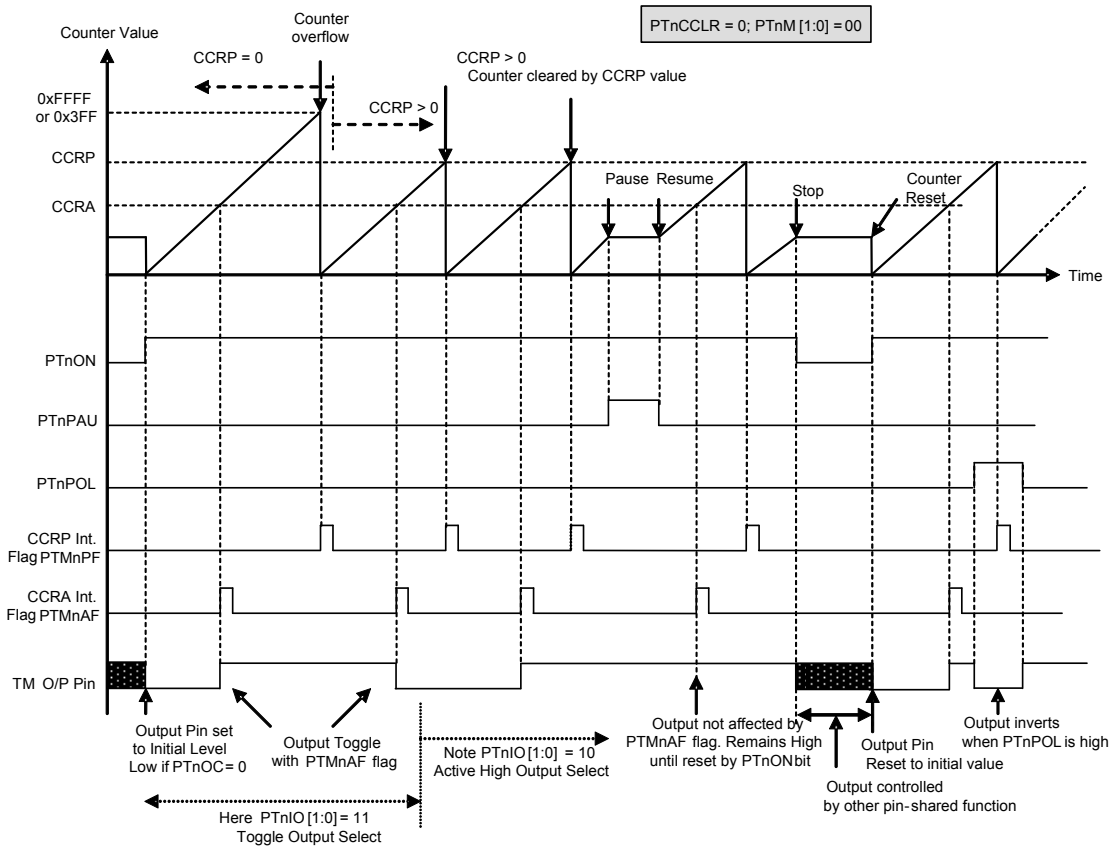
The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTnM1 and PTnM0 bits in the PTMnC1 register.

Compare Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 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 PTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both PTMnAF and PTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

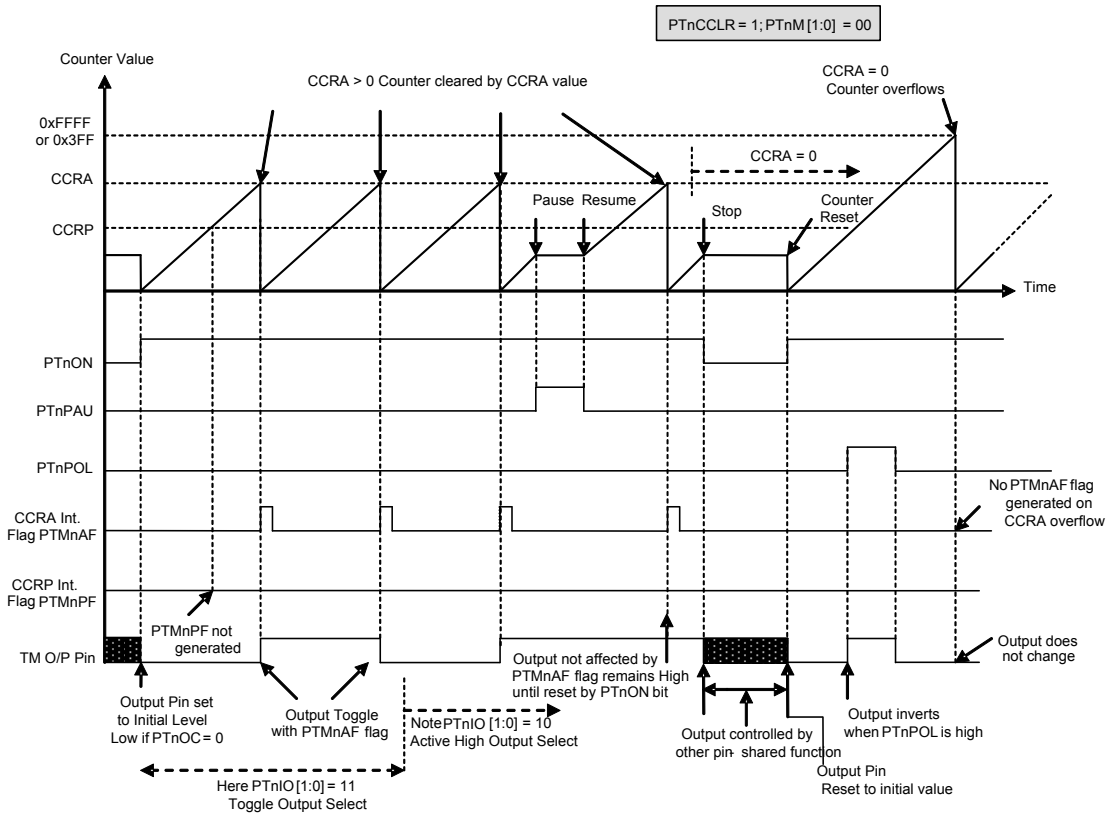
If the PTnCCLR bit in the PTMnC1 register is high, then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is high no PTMnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”.

As the name of the mode suggests, after a comparison is made, the TMn output pin will change state. The TMn output pin condition, however, only changes state when a PTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the TMn output pin. The way in which the TMn output pin changes state are determined by the condition of the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The TMn output pin can be selected using the PTnIO1 and PTnIO0 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 TMn output pin, which is setup after the PTnON bit changes from low to high, is setup using the PTnOC bit. Note that if the PTnIO1 and PTnIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – PTnCCLR=0

Note: 1. With PTnCCLR = 0 a Comparator P match will clear the counter
2. The TM output pin controlled only by the PTMnAF flag
3. The output pin reset to initial state by a PTnON bit rising edge



Compare Match Output Mode – PTnCCLR=1

Note: 1. With PTnCCLR = 1 a Comparator A match will clear the counter
2. The TM output pin controlled only by the PTMnAF flag
3. The output pin reset to initial state by a PTnON rising edge
4. The PTMnPF flags is not generated when PTnCCLR = 1

Timer/Counter Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 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 TMn 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 TMn 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 PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 10 respectively. The PWM function within the TMn 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 TMn 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 PTnCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA 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. 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 PTnOC bit in the PTMnC1 register is used to select the required polarity of the PWM waveform while the two PTnIO1 and PTnIO0 bits are used to enable the PWM output or to force the TMn output pin to a fixed high or low level. The PTnPOL bit is used to reverse the polarity of the PWM output waveform.

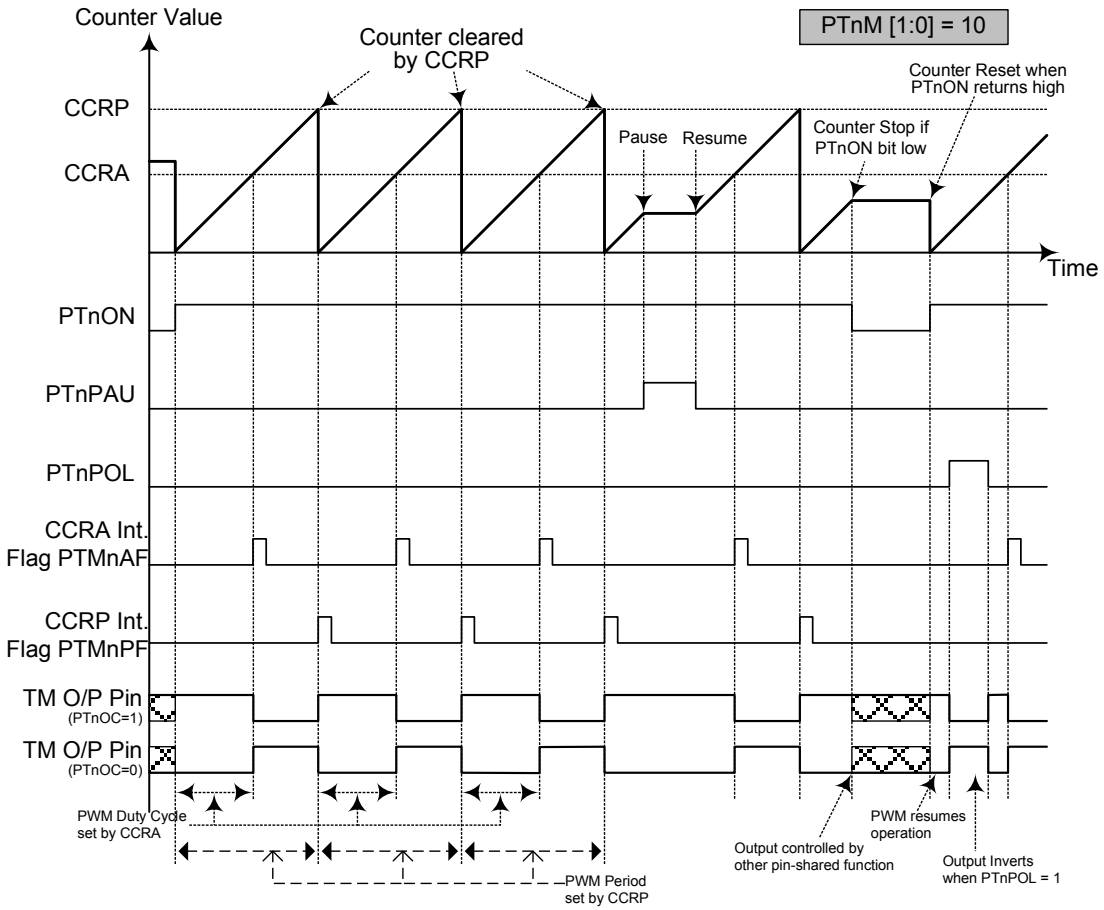
• 16-bit PTM, PWM Mode, Edge-aligned Mode

Period		Duty
CCRP=0 → Period = 65536	CCRP=1~65535 → Period = 1~65535	CCRA

• 10-bit PTM, PWM Mode, Edge-aligned Mode

Period		Duty
CCRP=0 → Period = 1024	CCRP=1~1023 → Period = 1~1023	CCRA

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%.



PWM Mode

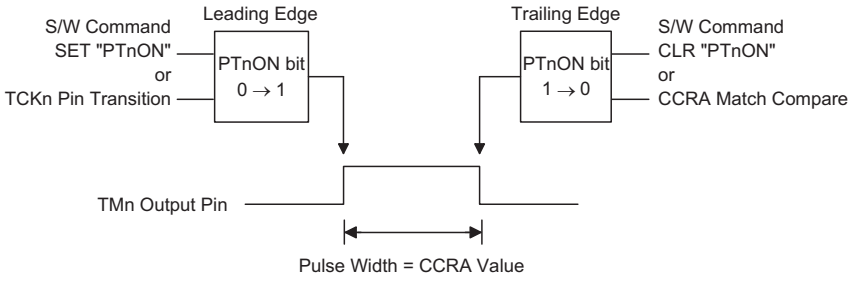
Note: 1. Here Counter cleared by CCRP
2. A counter clear sets PWM Period
3. The internal PWM function continues even when PTnIO[1:0] = 00 or 01
4. The PTnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

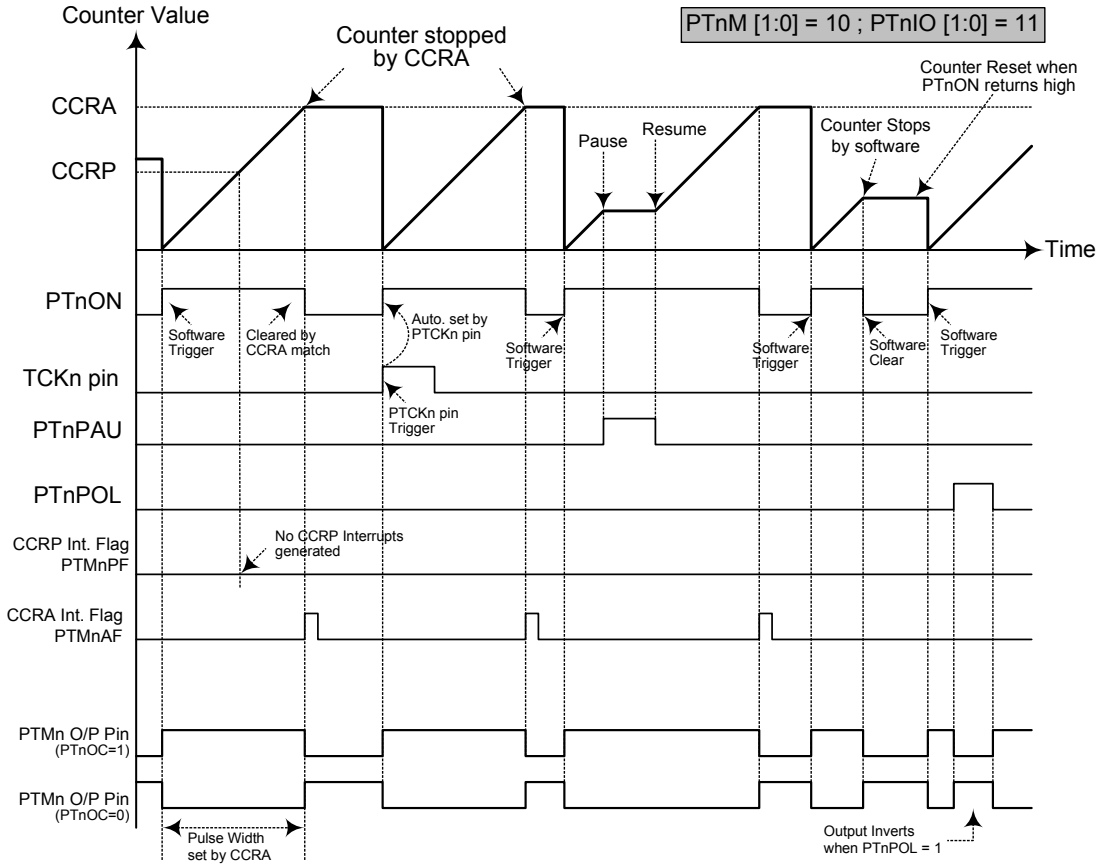
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the TMn output pin.

The trigger for the pulse output leading edge is a low to high transition of the PTnON bit, which can be implemented using the application program. However in the Single Pulse Mode, the PTnON bit can also be made to automatically change from low to high using the external TCKn pin, which will in turn initiate the Single Pulse output. When the PTnON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The PTnON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the PTnON bit is cleared to zero, which can be implemented using the application program or when a compare match occurs from Comparator A.

However a compare match from Comparator A will also automatically clear the PTnON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate a TM interrupt. The counter can only be reset back to zero when the PTnON bit changes from low to high when the counter restarts. In the Single Pulse Mode CCRP is not used. The PTnCCLR is not used in this Mode.



Single Pulse Generation

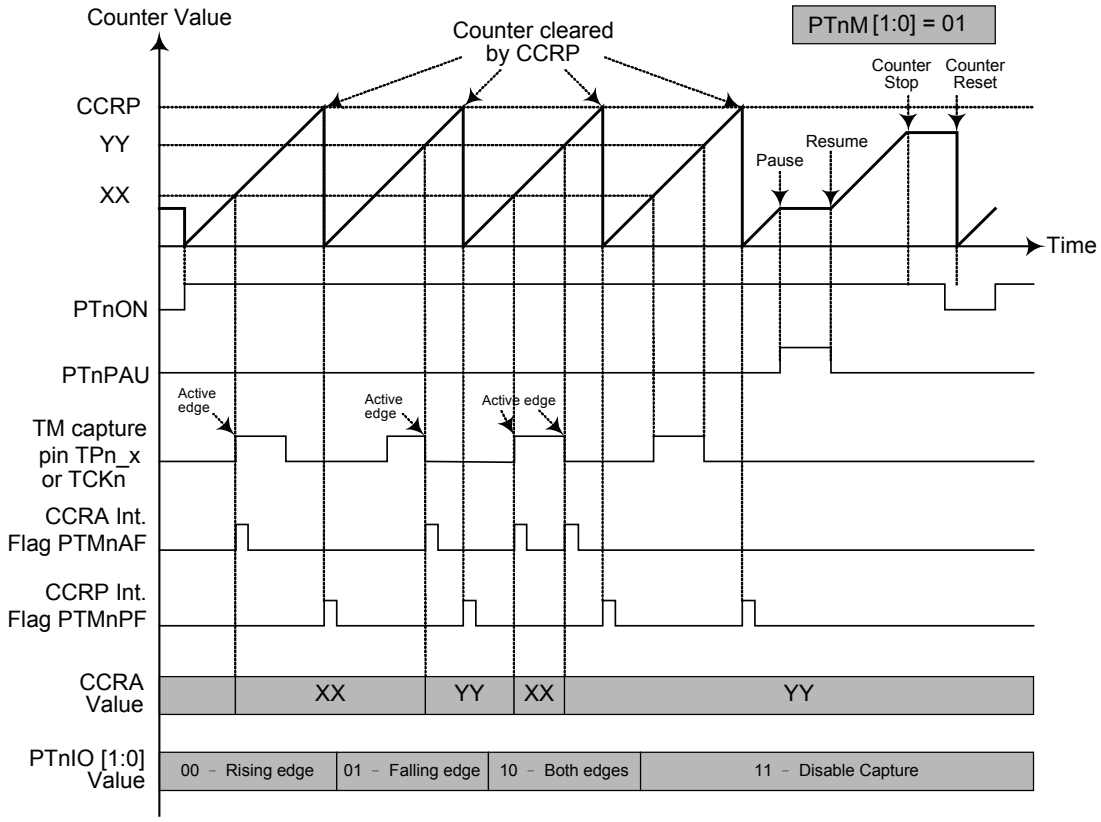


Single Pulse Mode

Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse is triggered by the TCKn pin or setting the PTnON bit high
4. A TCKn pin active edge will automatically set the PTnON bit high
5. In the Single Pulse Mode, PTnIO [1:0] must be set to "11" and can not be changed.

Capture Input Mode

To select this mode bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the TPn_0, TPn_1 or TCKn pin, selected by the PTnCAPTS bit in the PTM1C0 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The counter is started when the PTnON bit changes from low to high which is initiated using the application program. When the required edge transition appears on the TPn_0, TPn_1 or TCKn pin the present value in the counter will be latched into the CCRA registers and a TM interrupt generated. Irrespective of what events occur on the TPn_0, TPn_1 or TCKn pin the counter will continue to free run until the PTnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a TM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTnIO1 and PTnIO0 bits can select the active trigger edge on the TPn_0, TPn_1 or TCKn pin to be a rising edge, falling edge or both edge types. If the PTnIO1 and PTnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the TPn_0, TPn_1 or TCKn pin, however it must be noted that the counter will continue to run. As the TPn_0, TPn_1 or TCKn pin is pin shared with other functions, care must be taken if the TM1 is in the Input Capture Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The PTnCCLR, PTnOC and PTnPOL bits are not used in this Mode.



Capture Input Mode

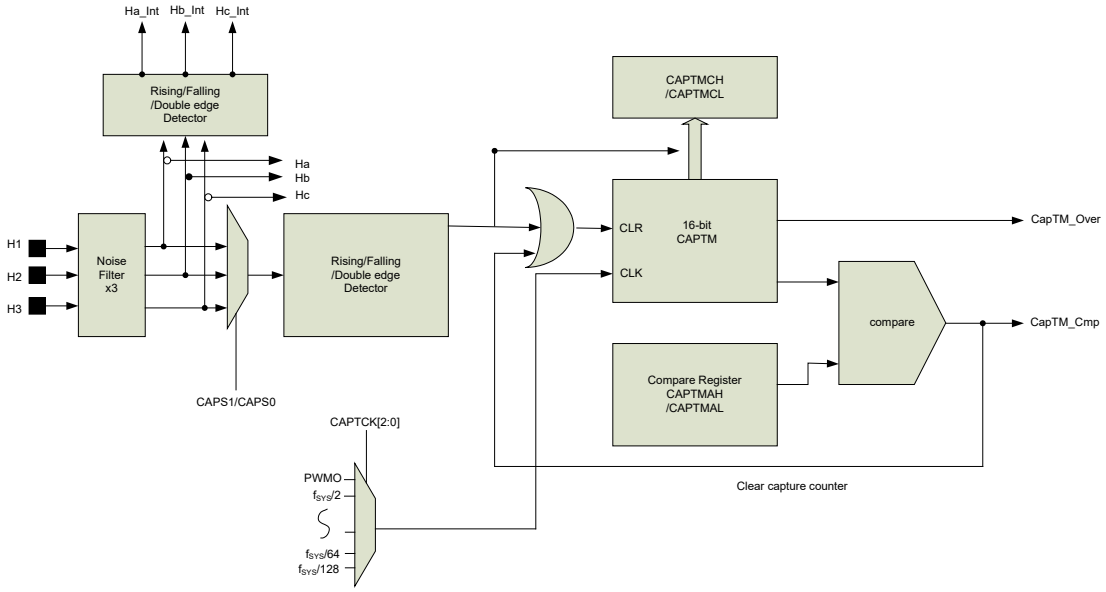
Note: 1. PTnM [1:0] = 01 and active edge set by the PTnIO [1:0] bits
2. A TM Capture input pin active edge transfers the counter value to CCRA
3. PTnCCLR bit is not used
4. No output function – PTnOC and PTnPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero.

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, H1, H2 and H3.



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	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
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
POR	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: H1
01: H2
10: H3
11: CTIN

• **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 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

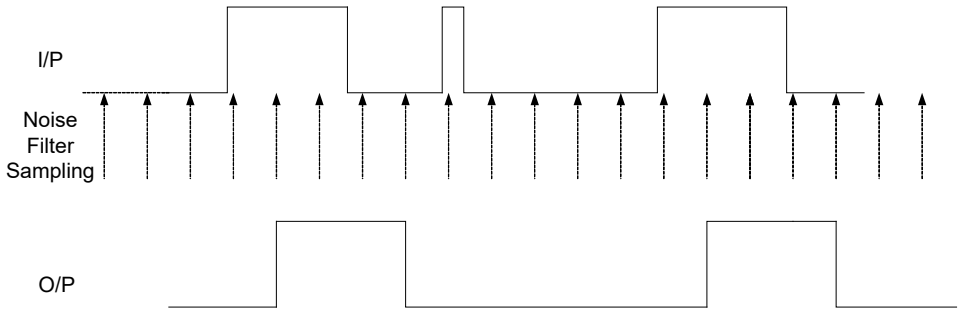
Bit 7~0 **CAPTMCH**: CAPTM Capture High Byte Register bit 7 ~ bit 0
CAPTM 16-bit Capture Register bit 15 ~ bit 8.

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 four capture inputs H1, H2 and H3. 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 options are setup using the CAPEG1 and CAPEG0 bits, to select either positive edge, negative edge or both edges.

The timer also includes a noise Filter which is used to filter out unwanted glitches or pulses on the H1, H2, H3 and NFIN 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

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

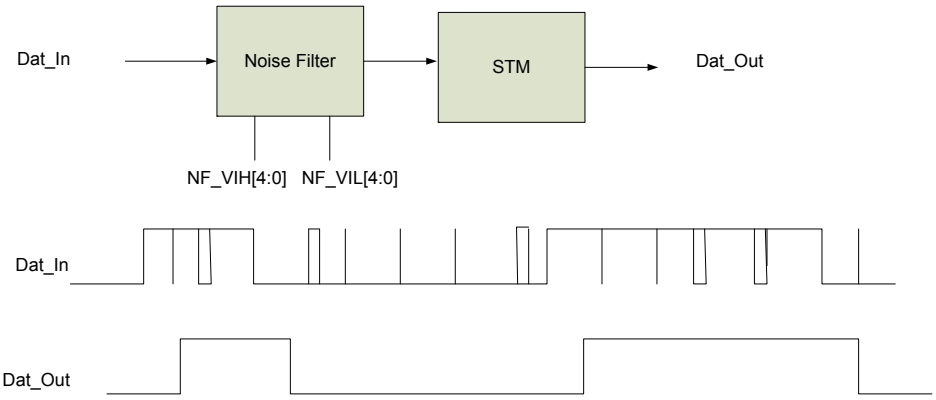
When the timer is used in the compare mode, the CAPTMAL and CAPTMAH registers are used to store the 16-bit compare value. When the free running value of the count-up 16-bit counter reaches a value equal to the programmed values in these compare registers, the CAPCF interrupt flag will be set which will 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 external NFIN 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 NFIN input signal and then output to 16-bit PTM, TM0, capture circuit. In order to ensure that the motor control circuit works normally, the noise filter circuit is an I/O filtering surge compare which can filter micro-second grade sharp-noise.

Antinoise pulse width maximum:

(NF_VIH[4:0]-NF_VIL[4:0])×5μs, (NF_VIH[4:0]-NF_VIL[4:0])>1



Noise Filter Registers Description

- **NF_VIH Register**

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

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

Bit 6 **CINS**: PTM 0 capture source selection
0: No select Noise Filter Dat_Out (remains the original PTM path)
1: Select Noise Filter Dat_Out

Bit 5 Unimplemented, read as "0"

Bit 4~0 NF_VIH register Bit 4~Bit 0

- **NF_VIL Register**

Bit	7	6	5	4	3	2	1	0
Name	NFIS1	NFIS0	—	D4	D3	D2	D1	D0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	1	0	1	0

Bit 7~6 **NFIS1~NFIS0**: NFIN Interrupt edge control
00: Disable
01: Rising edge trigger
10: Falling edge trigger
11: Dual edge trigger

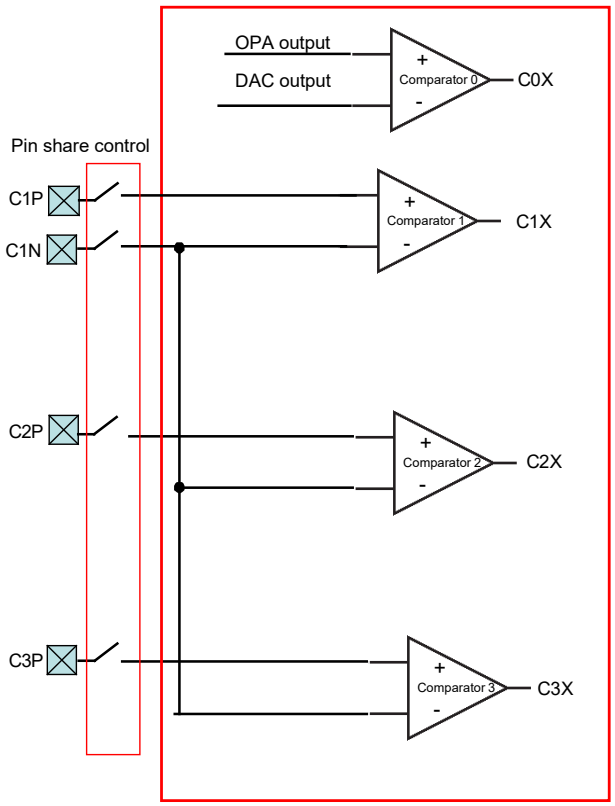
Bit 5 Unimplement, read as "0"

Bit 4~0 NF_VIL register Bit 4~Bit0

Comparators

Four independent analog comparators are contained within these devices. These functions offer flexibility via their register controlled features such as power-down, polarity select, hysteresis etc. In sharing their pins with normal I/O pins the comparators do not waste precious I/O pins if there functions are otherwise unused.

Comparators Block Diagram



Comparator Operation

The device contains four comparator functions which are used to compare two analog voltages and provide an output based on their difference. Additional comparator functions include, output polarity, hysteresis functions and power down control. Any pull-high resistors connected to the shared comparator input pins will be automatically disconnected when the comparator pin-share is enabled. As the comparator inputs approach their switching level, some spurious output signals may be generated on the comparator output due to the slow rising or falling nature of the input signals.

This can be minimised by selecting the hysteresis function will apply a small amount of positive feedback to the comparator. Ideally the comparator should switch at the point where the positive and negative inputs signals are at the same voltage level, however, unavoidable input offsets introduce some uncertainties here. The hysteresis function, if enabled, also increases the switching offset value.

• **CPC Register**

Bit	7	6	5	4	3	2	1	0
Name	C3HYEN	C2HYEN	C1HYEN	C0HYEN	C3EN	C2EN	C1EN	C0EN
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 **C3HYEN**: Comparator 3 Hysteresis Control
0: Off
1: On
This is the hysteresis control bit and if set high will apply a limited amount of hysteresis to the comparator, as specified in the Comparator Electrical Characteristics table. The positive feedback induced by hysteresis reduces the effect of spurious switching near the comparator threshold.

bit 6 **C2HYEN**: Comparator 2 Hysteresis Control
0: Off
1: On
This is the hysteresis control bit and if set high will apply a limited amount of hysteresis to the comparator, as specified in the Comparator Electrical Characteristics table. The positive feedback induced by hysteresis reduces the effect of spurious switching near the comparator threshold.

bit 5 **C1HYEN**: Comparator 1 Hysteresis Control
0: Off
1: On
This is the hysteresis control bit and if set high will apply a limited amount of hysteresis to the comparator, as specified in the Comparator Electrical Characteristics table. The positive feedback induced by hysteresis reduces the effect of spurious switching near the comparator threshold.

bit 4 **C0HYEN**: Comparator 0 Hysteresis Control
0: Off
1: On
This is the hysteresis control bit and if set high will apply a limited amount of hysteresis to the comparator, as specified in the Comparator Electrical Characteristics table. The positive feedback induced by hysteresis reduces the effect of spurious switching near the comparator threshold.

bit 3 **C3EN**: Comparator 3 On/Off control
0: Off
1: On
This is the Comparator on/off control bit. If the bit is zero the comparator will be switched off and no power consumed even if analog voltages are applied to its inputs. For power sensitive applications this bit should be cleared to zero if the comparator is not used or before the device enters the Power-down mode.

bit 2 **C2EN**: Comparator 2 On/Off control
0: Off
1: On
This is the Comparator on/off control bit. If the bit is zero the comparator will be switched off and no power consumed even if analog voltages are applied to its inputs. For power sensitive applications this bit should be cleared to zero if the comparator is not used or before the device enters the Power-down mode.

bit 1 **C1EN**: Comparator 1 On/Off control
0: Off
1: On
This is the Comparator on/off control bit. If the bit is zero the comparator will be switched off and no power consumed even if analog voltages are applied to its inputs. For power sensitive applications this bit should be cleared to zero if the comparator is not used or before the device enters the Power-down mode.

bit 0 **C0EN**: Comparator 0 On/Off control
0: Off
1: On
This is the Comparator on/off control bit. If the bit is zero the comparator will be switched off and no power consumed even if analog voltages are applied to its inputs. For power sensitive applications this bit should be cleared to zero if the comparator is not used or before the device enters the Power-down mode.

Analog to Digital Converter

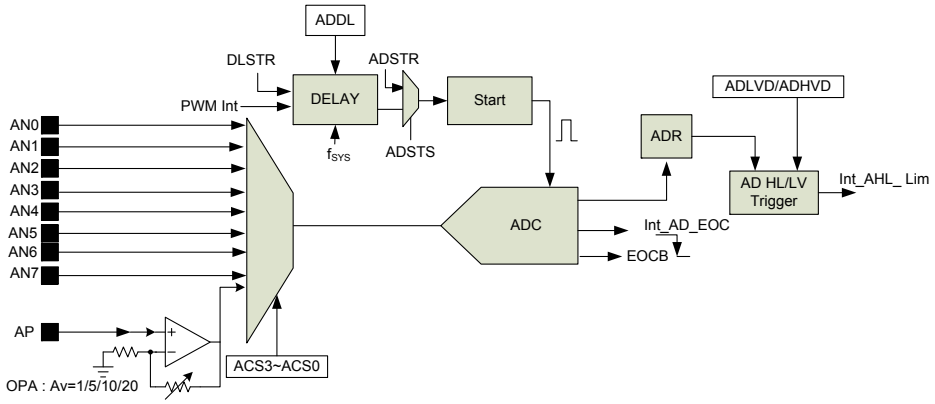
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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 8-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 12-bit digital value. An additional channel is connected to the external current sense input pin, AP, via an internal operational amplifier for signal amplification, before being transferred to the A/D converter input. A set of register pairs which 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
8+1	ACS3~ACS0	AN0~AN7; AP

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 (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
ADRL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADRH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADRH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
ADCR0	ADSTR	EOCB	ADOFF	ADRFS	ACS3	ACS2	ACS1	ACS0
ADCR1	ADSTS	DLSTR	PWIS	ADCHVE	ADCLVE	ADCK2	ADCK1	ADCK0
ADCR2	—	—	—	—	—	—	PWMDIS1	PWMDIS0
ADDL	D7	D6	D5	D4	D3	D2	D1	D0
ADLVDL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
ADLVDL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADLVDH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADLVDH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
ADHVDL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
ADHVDL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADHVDH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADHVDH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8

A/D Converter Register List

A/D Converter Data Registers – ADRL, ADRH

As this device contains an internal 12-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.

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

A/D Converter Data Registers

A/D Converter Control Registers – ADCR0, ADCR1, ADCR2, ADDL

To control the function and operation of the A/D converter, four control registers known as ADCR0, ADCR1 and ADCR2 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 8 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 AP pin is actually connected to the internal A/D converter.

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	ADRFS	ACS3	ACS2	ACS1	ACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	1	0	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 **ADRFS** : A/D output data format select bit
0: ADC output data format → ADRH=D [11:4]; ADRL=D[3:0]
1: ADC output data format → ADRH=D [11:8]; ADRL=D[7:0]

Bit 3~ 0 **ACS3 ~ ACS0**: A/D Converter channel selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: OPA output
0111: AN6
1000: AN7
1001~1xxx: Undefined
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.

• **ADCR2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PWDIS1	PWDIS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PWDIS1~PWDIS0**: PWIS=1, select PWMn duty cycle interrupt trigger source
00: PWM0
01: PWM1
10: PWM2
11: Reseved(select PWM2 duty cycle interrupt trigger source)

• **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.

ADRFS	ADLVDH								ADLVDL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D Low Boundary Data Registers

ADRFS	ADHVDH								ADHVDL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D High Boundary Data Registers

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 use the PWM interrupt signal. This can be sourced from either the PWM period or duty interrupt signal, selected using the PWIS bit. If selects PWM duty interrupt signal, interrupt trigger source can be selected by PWDIS1 and PWDIS2 in the ADCR2 register. 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 12-bit A/D converter clock period, t_{ADCK}, is 0.8μs, care must be taken for system clock frequencies equal to or greater than 1.25MHz. For example, if the system clock operates at a frequency of 5MHz, the ADCK2~ADCK0 bits should not be set to "000"and "001". 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})							
	ADCK[2:0] =000(f_{SYS})	ADCK[2:0] =001(f_{SYS}/2)	ADCK[2:0] =010(f_{SYS}/4)	ADCK[2:0] =011(f_{SYS}/8)	ADCK[2:0] =100(f_{SYS}/16)	ADCK[2:0] =101(f_{SYS}/32)	ADCK[2:0] =110(f_{SYS}/64)	ADCK[2:0] =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. 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.

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.

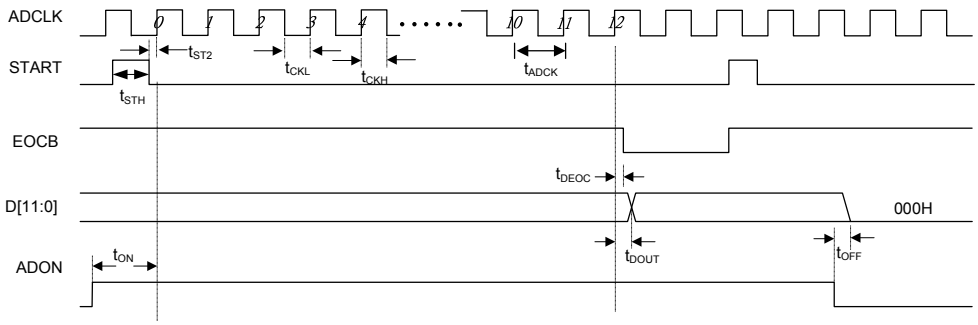
- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCK2~ADCK0 in the ADCR1 register.
- Step 2
  Enable the A/D by clearing the ADOFF bit in the ADCR0 register to zero.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming 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 correct bits in the pin share registers.
- 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 Diagram

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

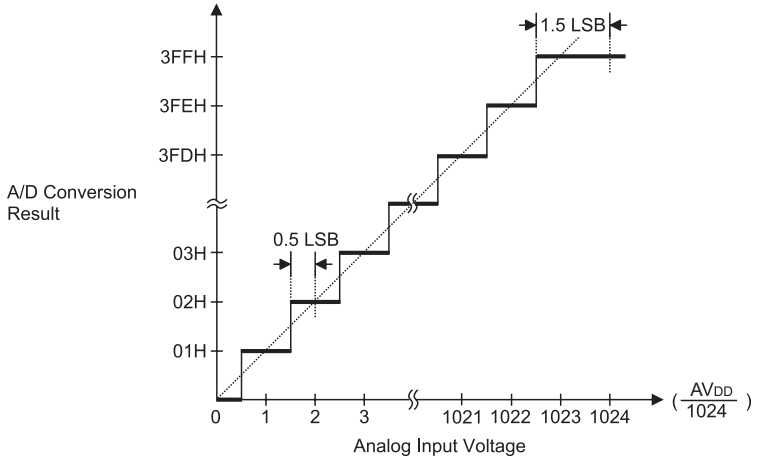
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the V_{DD} voltage, this gives a single bit analog input value of V_{DD} divided by 4096.

$$1\ LSB = AV_{DD} / 4096$$

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

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

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 AV_{DD} or V_{REF} level.



Ideal A/D Transfer Function

A/D Programming Examples

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

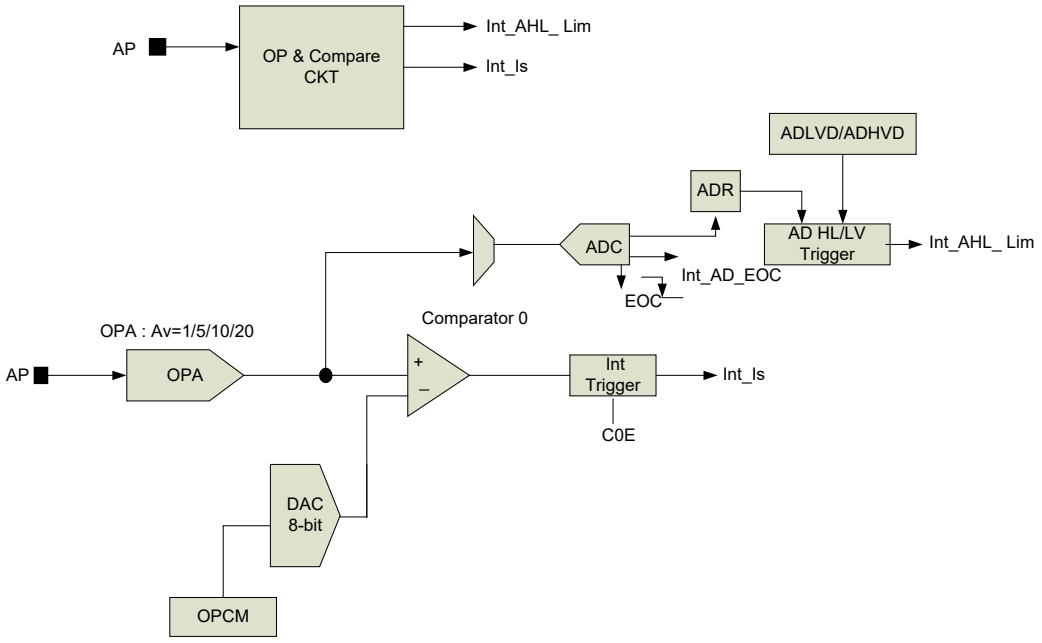
```
clr  AEOCE                  ; disable ADC interrupt
mov  a,03H
mov  ADCR1,a                ; select fSYS/8 as A/D clock
clr  ADOFF
mov  a,2Ah                  ; setup PBPS0 to configure pins AN0~AN2
mov  PBPS0,a
mov  a,00h
mov  ADCR0                  ; enable and connect AN0 channel to A/D converter
:
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,2Ah                   ; setup PBPS0 to configure pins AN0~AN2
mov  PBPS0,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, 12-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.

- A/D Converter interrupt - Int_AHL_Lim
- 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.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OPOMS	CMP0_EG1	CMP0_EG0	—	—	—	OPAVS2	OPAVS1	OPAVS0
OPCM	D7	D6	D5	D4	D3	D2	D1	D0
OPACAL	—	ARS	AOFM	AOF4	AOF3	AOF2	AOF1	AOF0

Over-Current Function Register List

• **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 DAC
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

Note: The AN3/AP pin has to be enabled by setting the corresponding pin-shared register when using OPA function.

• **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/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	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

BLDC Motor Control Circuit

This section 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.

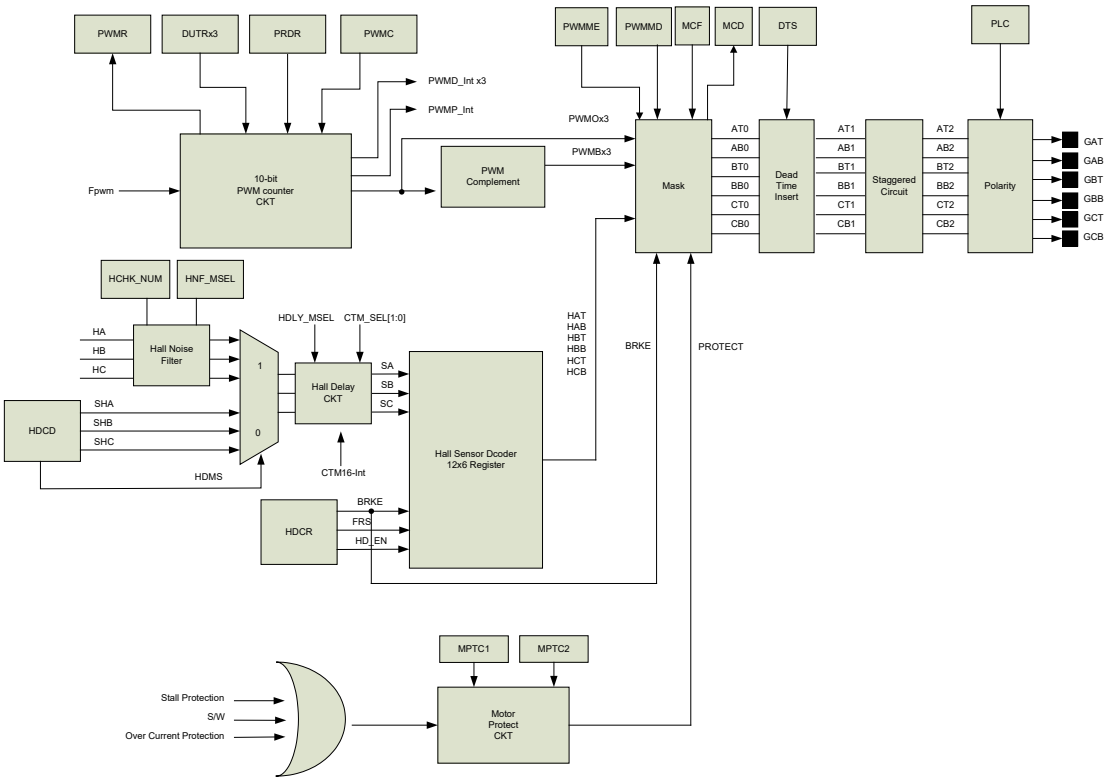
The internal Mask circuit is used to determine which PWM modulation signals are enabled or disabled for the motor speed control. The PWM modulation signal can be output both the upper arms, GAT/GBT/GCT and the lower arms, GAB/GBB/GCB, of the external Gate Driver Transistor Pairs under software control.

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.

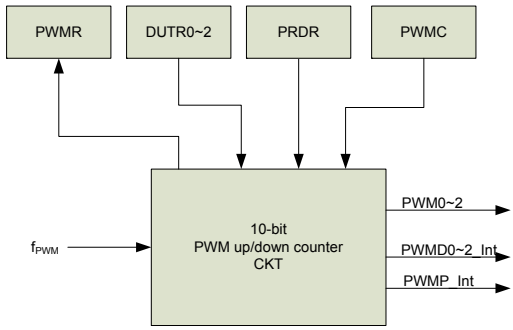


BLDC Motor Control Block Diagram

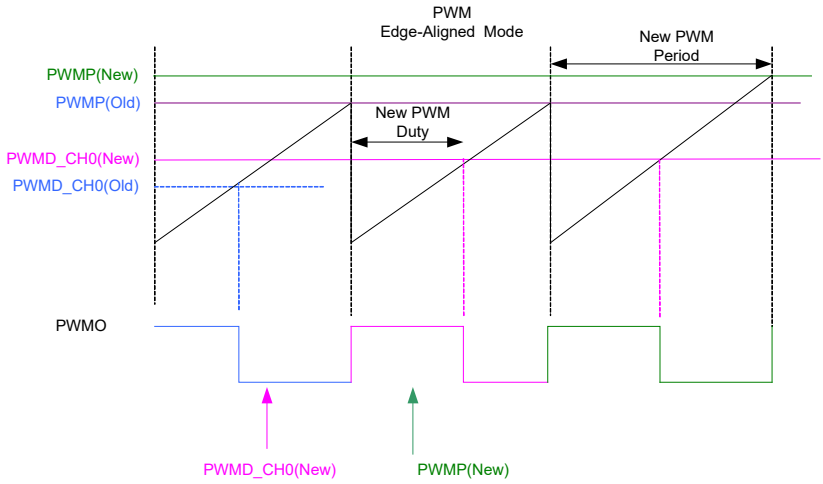
Note: GAT, GAB, GBT, GBB, GCT, GCB == PWM0H, PWM0L, PWM1H, PWM1L, PWM2H, PWM2L.

PWM Counter Control Circuit

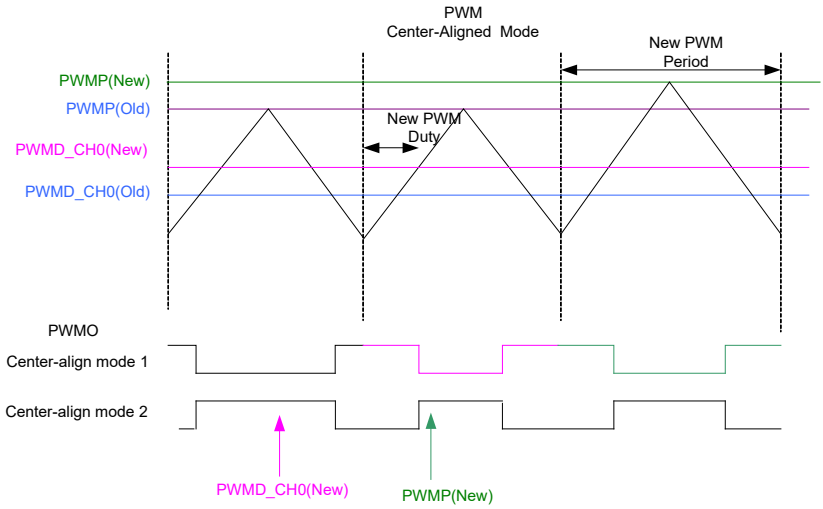
The device includes a 10-bit PWM generator. The PWM signal has both adjustable duty cycle and frequency that can be setup by programming 10-bit values into the corresponding PWM registers.



PWM Block Diagram



PWM Edge-Aligned mode Timing Diagram



PWM Center-Aligned mode Timing Diagram

PWM Register Description

Overall PWM operation is controlled by a series of registers. The DUTRnL/DUTRnH 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. The PWMMS bits in the PWMC register determine the PWM alignment type, which can be either edge or centre type. It should be noted that the order of writing data to PWM register is MSB.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWMC	PWMMS1	PWMMS0	PCKS1	PCKS0	PWMON	ITCMS1	ITCMS0	PWMLD
DUTR0L	D7	D6	D5	D4	D3	D2	D1	D0
DUTR0H	—	—	—	—	—	—	D9	D8
DUTR1L	D7	D6	D5	D4	D3	D2	D1	D0
DUTR1H	—	—	—	—	—	—	D9	D8
DUTR2L	D7	D6	D5	D4	D3	D2	D1	D0
DUTR2H	—	—	—	—	—	—	D9	D8
PRDRL	D7	D6	D5	D4	D3	D2	D1	D0
PRDRH	—	—	—	—	—	—	D9	D8
PWMRL	D7	D6	D5	D4	D3	D2	D1	D0
PWMRH	—	—	—	—	—	—	D9	D8

PWM Register List

- **PWMC Register**

Bit	7	6	5	4	3	2	1	0
Name	PWMMS1	PWMMS0	PCKS1	PCKS0	PWMON	ITCMS1	ITCMS0	PWMLD
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 **PWMMS**: PWM mode select bit
0x: Edge-aligned mode,
10: Centre-aligned mode 1
11: Centre-aligned mode 2

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 **ITCMS1~ITCMS0**:
00: disable center-aligned mode duty interrupt
01: center-aligned mode duty interrupt only in counter up condition
10: center-aligned mode duty interrupt only in counter down condition
11: center-aligned mode duty interrupt both in counter up or down condition

Bit 0 **PWMLD**: PWM PRDR&DUTRx,x=0~2 register update bit
0: The registers value of PRDR and DUTRx, x=0~2 are never loaded to counter and Comparator registers.
1: The PRDR register will be load value to counter register after counter underflow, and hardware will clear by next clock cycle.

• **DUTR0L 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 PWM0 Duty register low byte register
10-bit DUTR0 register bit 7 ~ bit 0

• **DUTR0H 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 PWM0 Duty register high byte register
10-bit DUTR0 register bit 9 ~ bit 8

• **DUTR1L 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 PWM1 Duty register low byte register
10-bit DUTR1 register bit 7 ~ bit 0

• **DUTR1H 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 PWM1 Duty register high byte register
10-bit DUTR1 register bit 9 ~ bit 8

• **DUTR2L 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 PWM2 Duty register low byte register
10-bit DUTR2 register bit 7 ~ bit 0

• **DUTR2H 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 PWM2 Duty register high byte register
10-bit DUTR2 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
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 10-bit PWM Period register high byte register
10-bit PRDR 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
10-bit PWM counter bit 7 ~ bit 0

• **PWMRH Register**

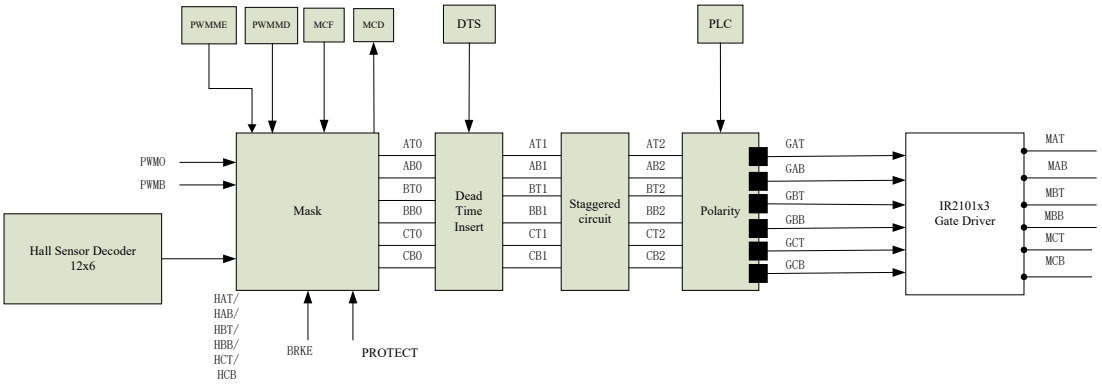
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

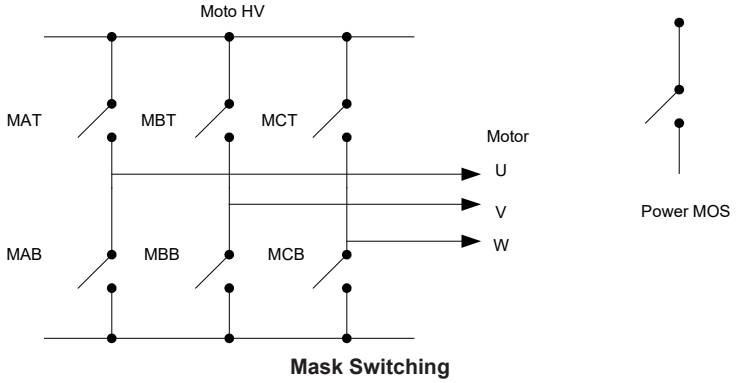
Bit 1~0 10-bit PWM Counter high byte register
10-bit PWM Counter bit 9 ~ bit 8

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, GAT/ GBT/ GCT.

  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.

  MSKMS=0: the Mask Mode selects H/W.

  MSKMS=1: the Mask Mode selects S/W.

• **H/W Mask Mode**

Complementary control, MPWMS=0

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

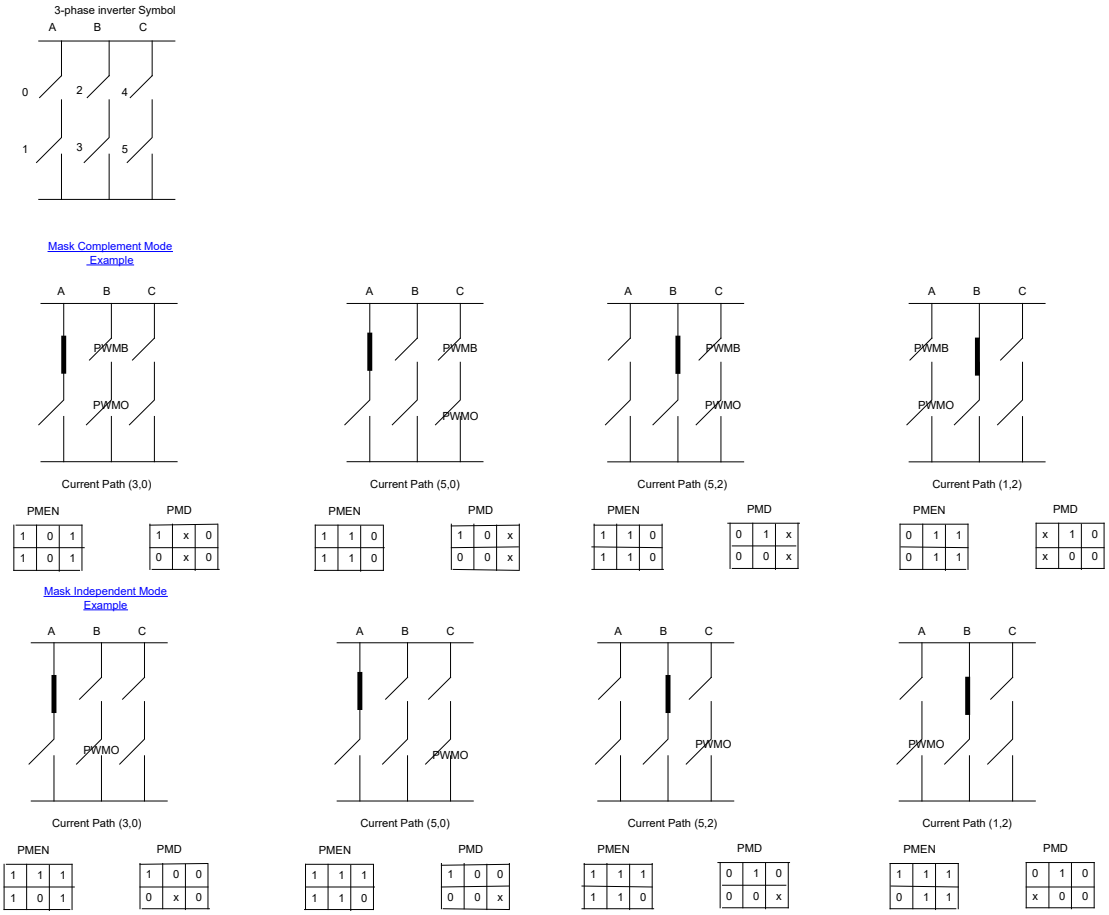
	HAT	HAB	AT0	AB0		HAT	HAB	AT0	AB0
PWMS=0	0	0	0	0	PWMS=1	0	0	0	0
	0	1	0	PWMO		0	1	0	1
	1	0	1	0		1	0	PWMO	0
	1	1	0	0		1	1	0	0

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

- **S/W Mask Mode**

  To control the Mask circuit, two registers known as PWMME, PWMMD, MCF and MCD are provided. PWMME register is used for control PWM signal and PWMMD is used to determine the MOS Gate Driver Circuit is on or off. Note that Setting PWMS and MPWMS or anything related to PWM function is effective to H/W or S/W mode.



Mask S/W Mode circuit

Note: 1. During masking enabled, when PWMxH and PWMxL are masked simultaneously, the two pins of each pair can not be set to "1" simultaneously, PMD.0 and PMD.1, PMD.2 and PMD.3, PMD.4 and PMD.5. If they are all high in the same time, switch 2n and switch 2n+1 will output "0".

2. If PWM and complementary PWM are enabled simultaneously, one of the two registers PWMxH and PWMxL output PWM and the other one can not be masked to "1" but output "0" automatically by hardware.

3. If the PWMxH and PWMxL are configured as I/O function, then PWM MASK function will be invalid.

S/W Mask Register Description

- **PWMME Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PME5	PME4	PME3	PME2	PME1	PME0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

bit 7~6 unimplemented, read as"0"

bit 5~0 **PME5~PME0**: PWM Mask enable register
0: PWM generator signal is output to next stage.
1: PWM generator signal is masked and PMDn is output to next stage.
The PWM generator signal will be masked when this bit is enabled. The corresponding PWMn channel will be output with PMD.n data.

- **PWMMD Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PMD5	PMD4	PMD3	PMD2	PMD1	PMD0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

bit 7~6 unimplemented, read as"0"

bit 5-0 **PMD5~PMD0**: PWM Mask Data bit
0: Output logic low to PWMn.
1: Output logic high to PWMn.
This data bit control the state of PWMn output pin, if corresponding PMEn = 1.

- **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.

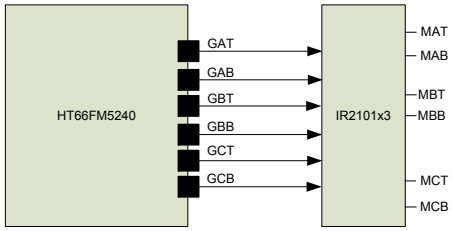
	AT0	BT0	CT0	AB0	BB0	CB0
BRKE=1	0	0	0	1	1	1

- **Motor Protect Mode**

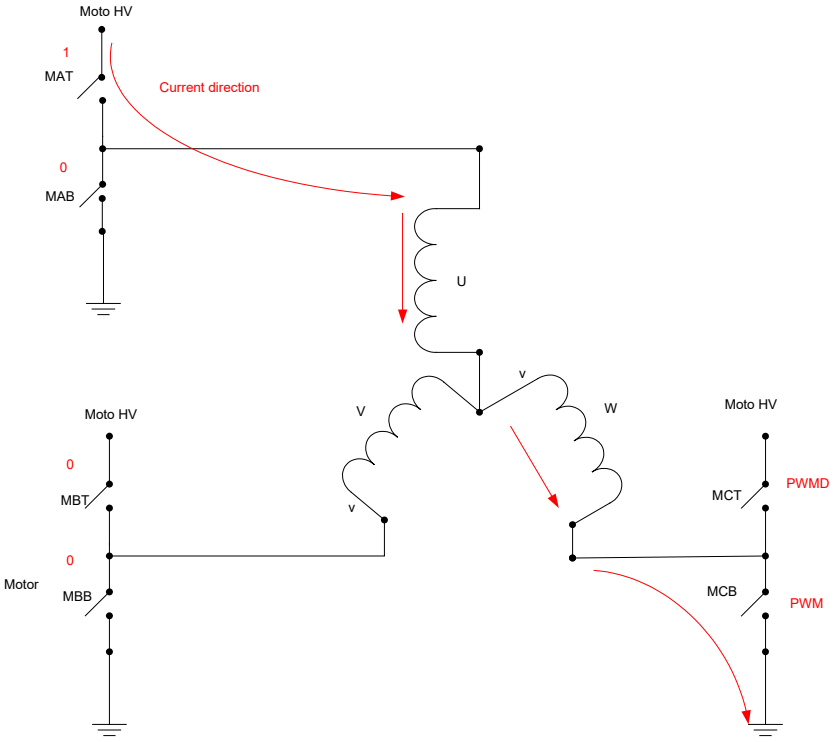
When the Motor Protect Mode is activated, the external Gate Driver Transistor Pair can select the brake, where the top arm is off and the bottom arm is on, or select free running where the top and bottom arm are both off. The protection decode table is shown below.

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	MSKMS	—	—	—	MPWMS	MPWE	FMOS	PWMS
R	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	1	0	0

bit 7 **MSKMS**: Mask Mode Select
0: H/W Mask Mode
1: S/W Mask Mode
bit 6~4 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	FHC	FHB	FHA
R/W	—	—	R	R	R	R	R	R
POR	—	—	0	0	0	0	0	0

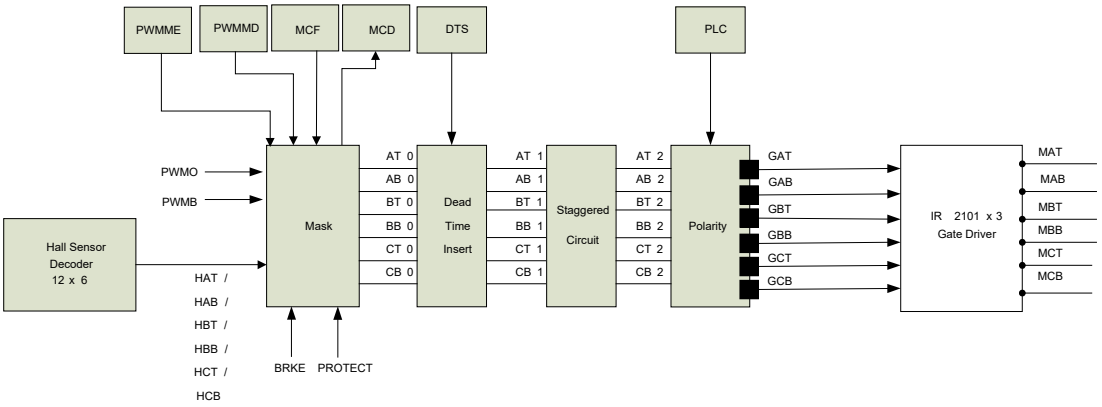
bit 7~6 unimplemented, read as"0"

Bit 5~3 **GAT/GAB/GBT**: Gate diver output monitor

Bit 2~0 **FHC/FHB/FHA**: HC/HB/HA filtered outputs
These signals are derived from the HC/HB/HA signals and filtered by the Hall Noise Filter.

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

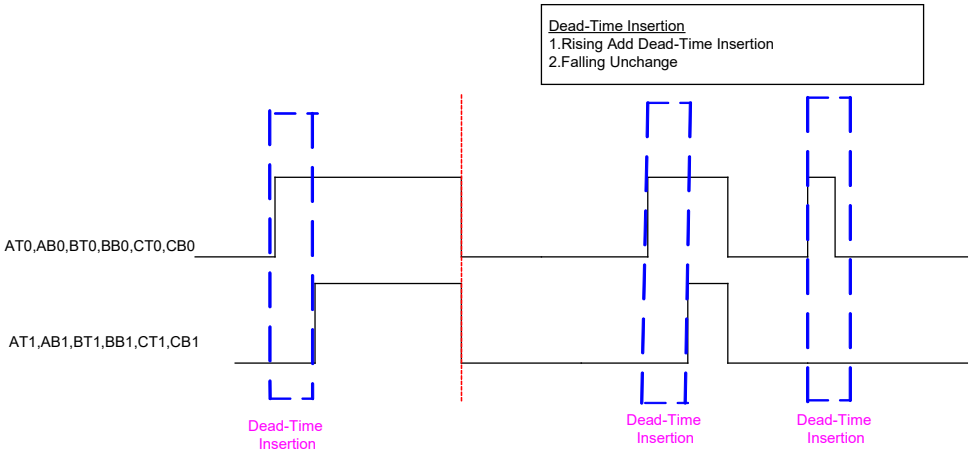
During transistor pair switching, the Dead Time function is used to prevent both upper and lower transistor pairs from conducting at the same time thus preventing a virtual short circuit condition from occurring. The actual dead time value can be setup to be within a value from 0.3μs to 5μs which is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

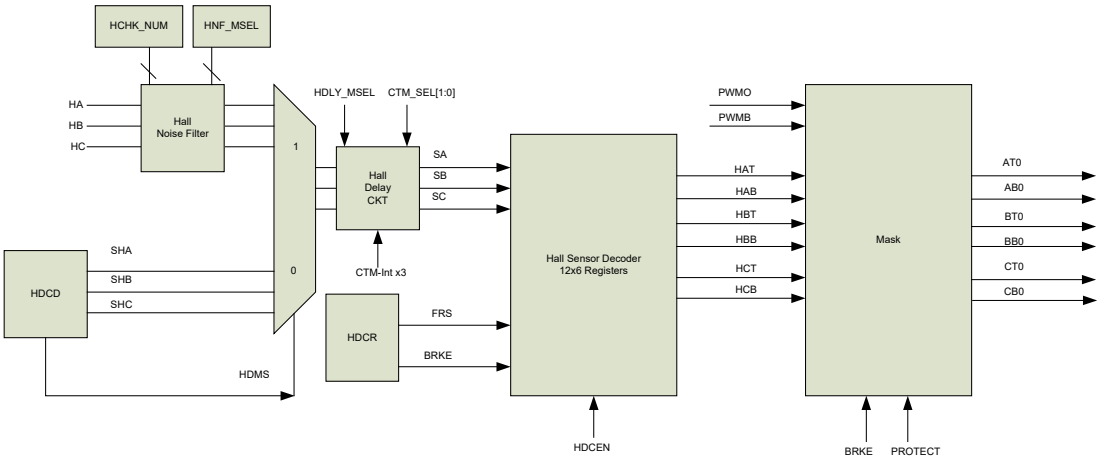
Bit Value	Status
0	Output not inverted
1	Output inverted

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.

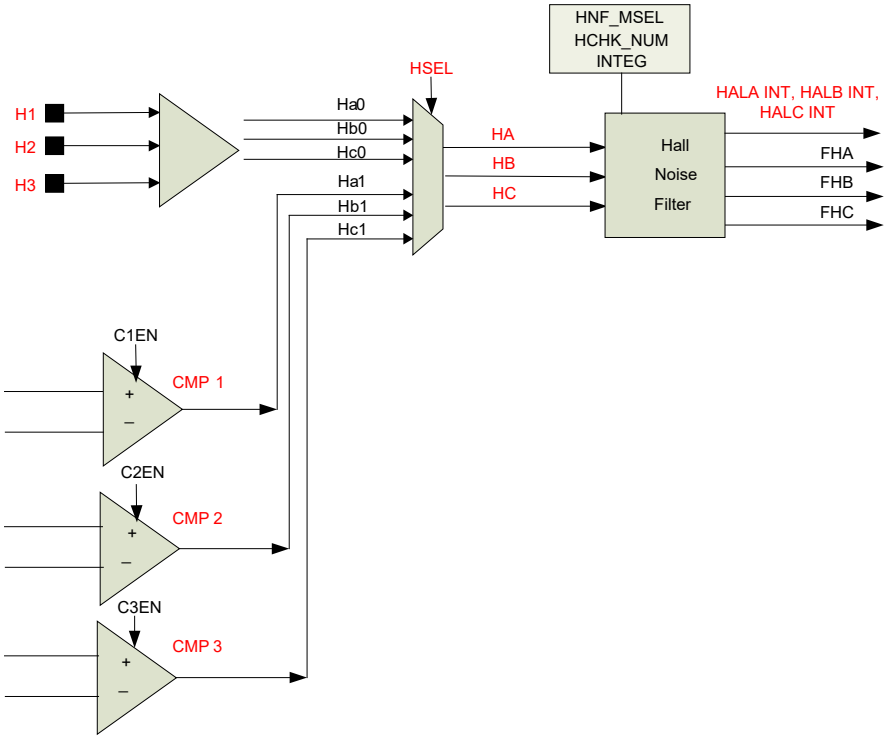


Hall Sensor Decoder Block Diagram

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 (H1/H2/H3), which in turn may result in incorrect Hall Sensor output decoding.



Hall Sensor Noise Fliter Blick Diagram

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. It is necessary to enable CMP1, CMP2 and CMP3 hysteries function before the camparators is used during motor control sensorless applications.

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

HCHK_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 TM is used to generate the Delay Time and set the selected TM 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.

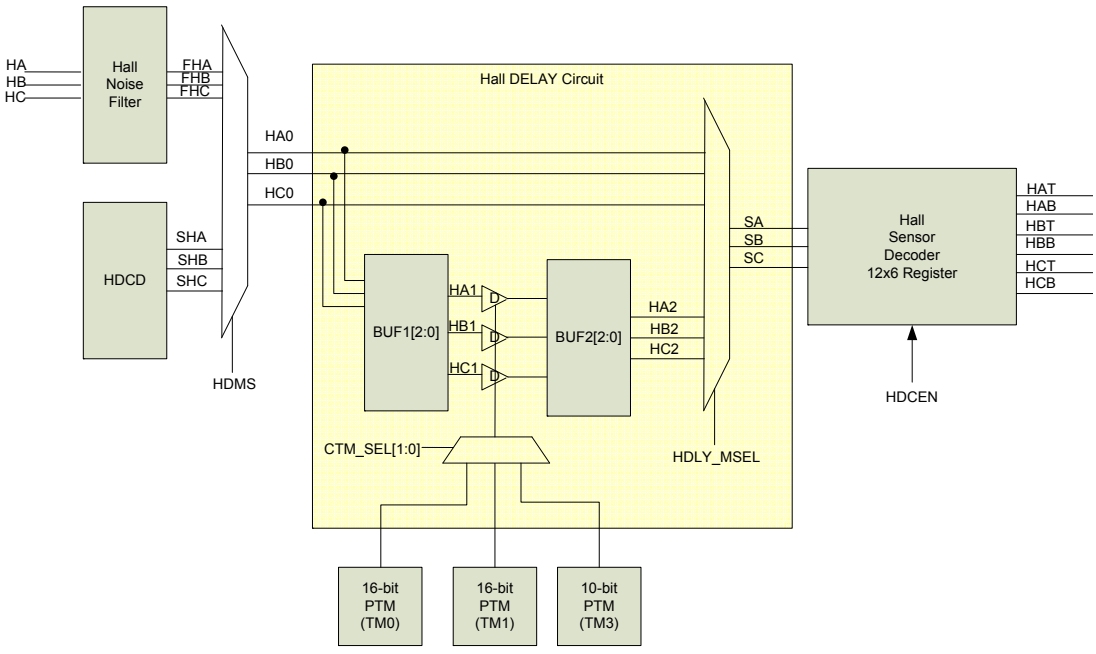
- When this bit is low, BUF1[2:0] and BUF2[2:0] will be cleared to zero.
- When this bit is low, TM0/TM1/TM2 retain their original TM functions.
- When the bit is high, the TM which is selected by the Delay function will be dedicated for use by the Hall Delay circuit.

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

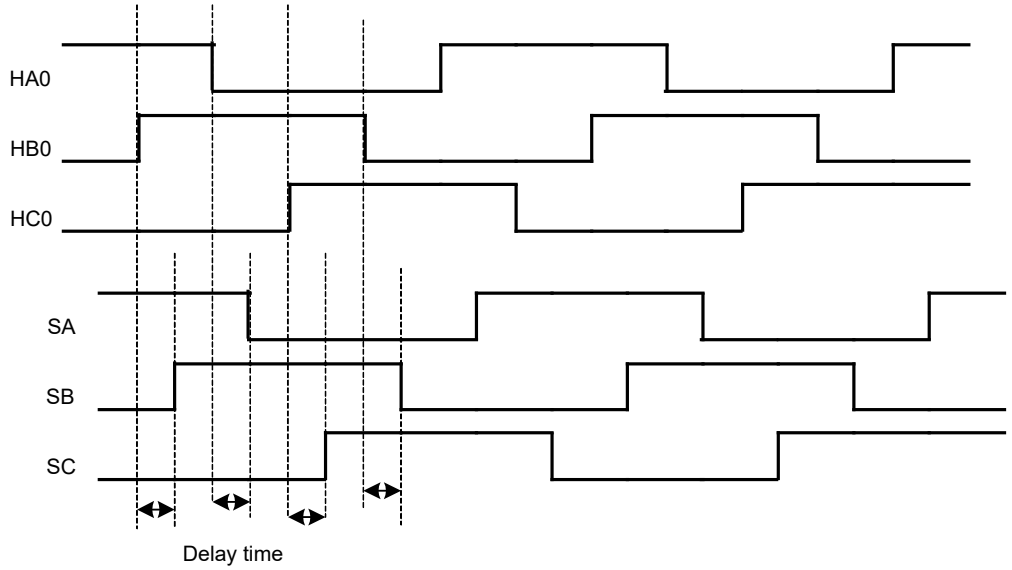
With regard to the TM functions the following steps should be taken before the Delay function is enabled.

- Keep TnON and TnPAU = 0
- The TM should be setup in the Compare Match Mode
- TnCCLR=1, therefore the TM is cleared with a comparator A match condition.
- 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 series of 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.

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

Brake Truth Table

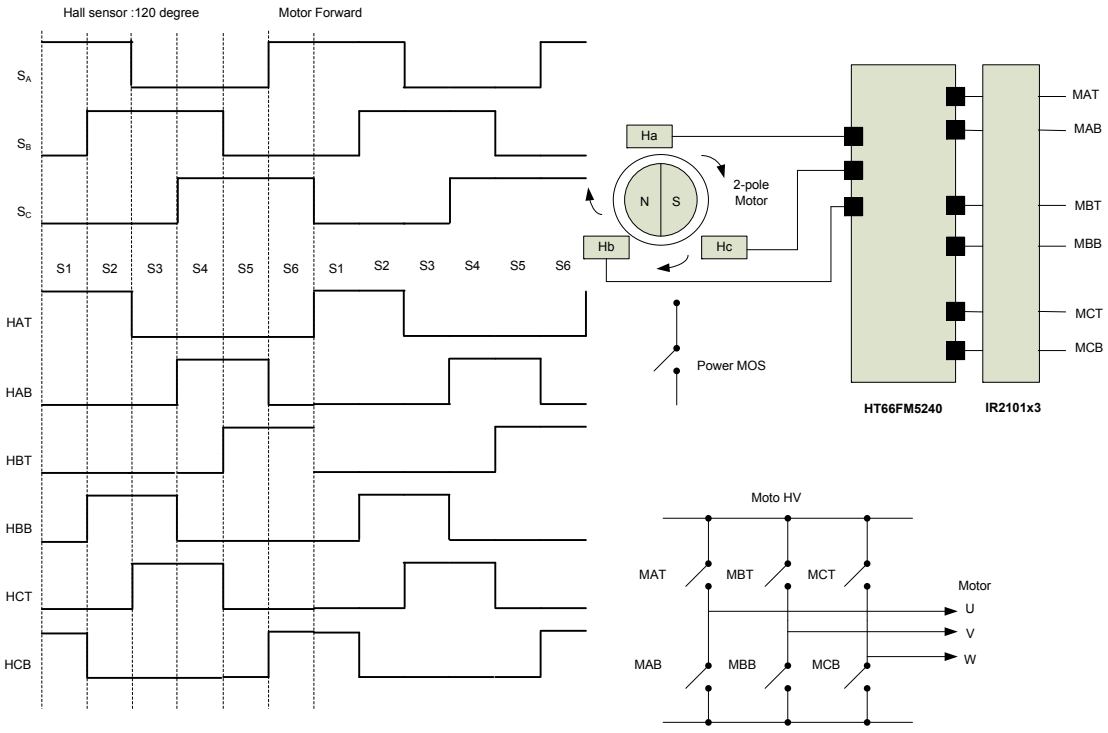
Hall Decoder disable (HDCEN=0)	60 degree			120 degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
	V	V	V	V	V	V	0	0	0	0	0	0

Hall Decoder Disable Truth Table

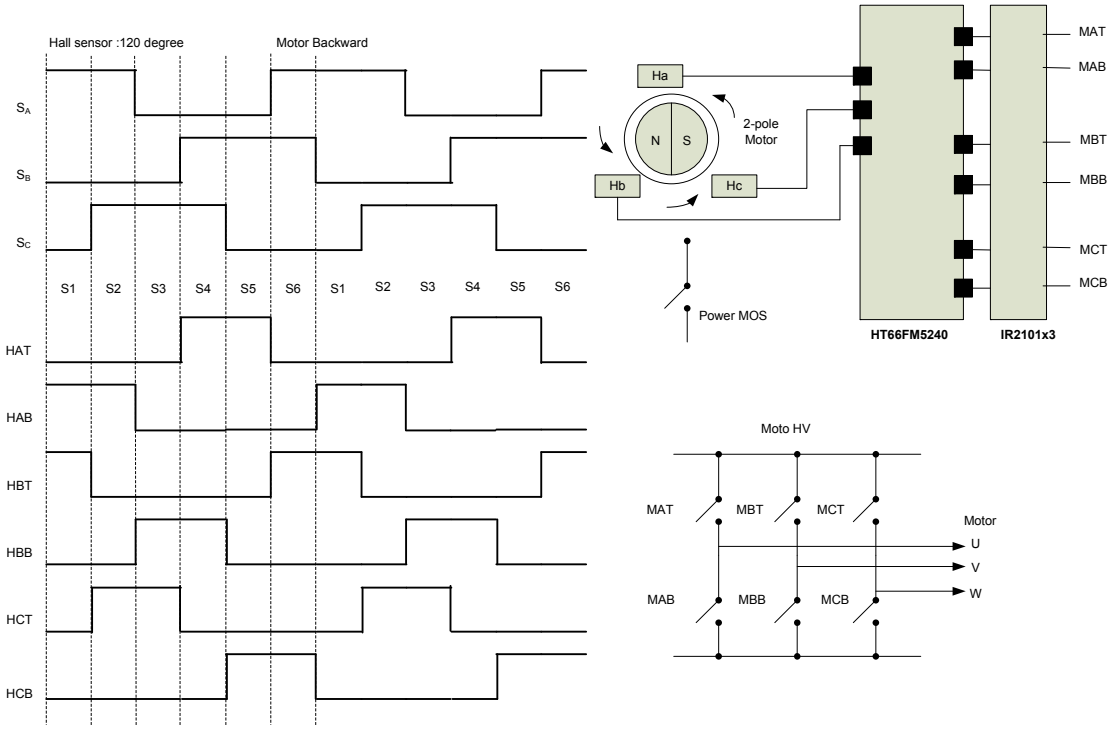
Hall Decoder error (HDCEN=X)	60 degree			120 degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
	1	0	1	1	1	1	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0

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.

- **INTEG0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	HSEL	INTCS1	INTCS0	INTBS1	INTBS0	INTAS1	INTAS0
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 **HSEL**: HA/HB/HC source select
0: H1/H2/H3
1: CMP1/CMP2/CMP3 output

bit 5~4 **INTCS1, INTCS0:** FHC Interrupt edge control for INTC
00 : disable
01 : rising edge trigger
10 : falling edge trigger
11 : dual edge trigger

bit 3~2 **INTBS1, INTBS0:** FHB Interrupt edge control for INTB
00 : disable
01 : rising edge trigger
10 : falling edge trigger
11 : dual edge trigger

bit 1~0 **INTAS1, INTAS0:** FHA Interrupt edge control for INTA
00 : disable
01 : rising edge trigger
10 : falling edge trigger
11 : dual edge trigger

• **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**: TM select of the Hall Delay Circuit
00:TM0 (16-bit PTM)
01:TM1 (16-bit PTM)
10:TM3 (10-bit PTM)
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: AT/BT/CT/AB/BB/CB=V
1: AT/BT/CT=0, AB/BB/CB=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

• **HDCTn Register n=0~11**

Bit	7	6	5	4	3	2	1	0
Name	—	—	HATDn	HABDn	HBTDn	HBBDn	HCTDn	HCBDn
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 **HATDn**: GAT output state control

Bit 4 **HABDn**: GAB output state control

Bit 3 **HBTDn**: GBT output state control

Bit 2 **HBBDn**: GBB output state control

Bit 1 **HCTDn**: GCT output state control

Bit 0 **HCBDn**: 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

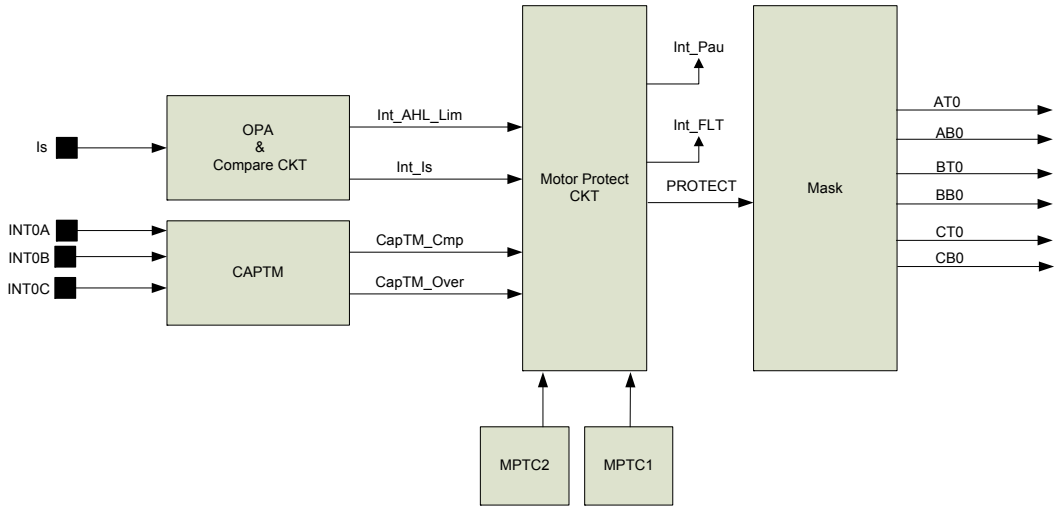
Bit 7~4 Unimplemented, read as "0"

Bit 3 **HNF_EN**: Hall noise filter enable
0: Disable(bypass)
1: Enable

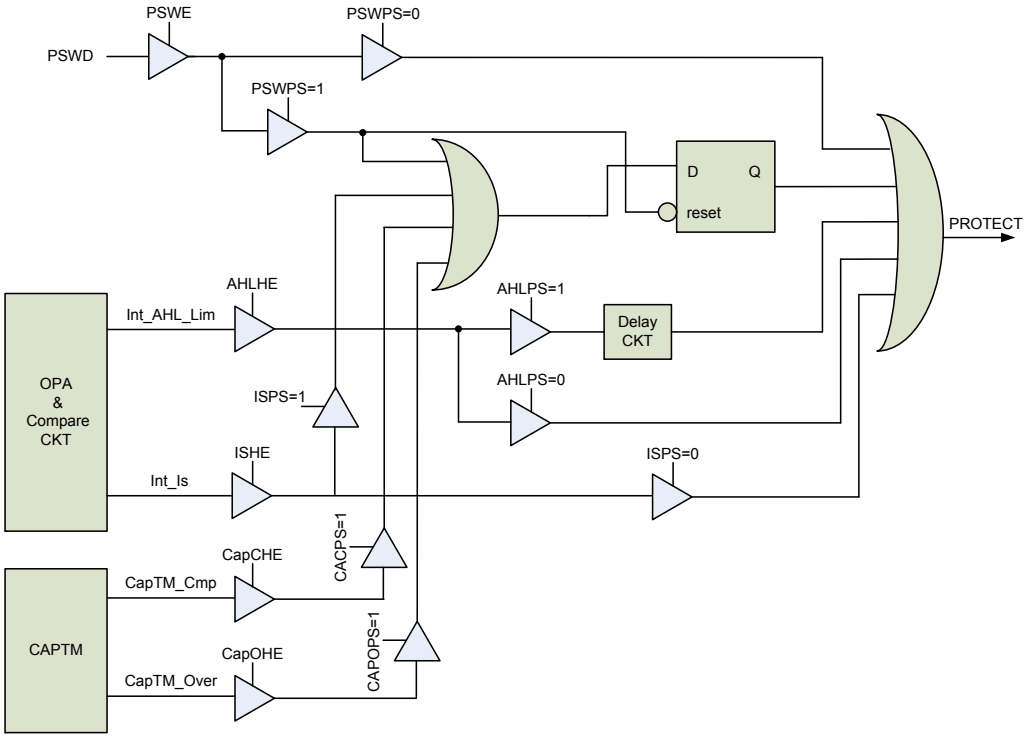
Bit 2~0 **HFR_SEL2 ~ HFR_SEL0**: Hall noise filter clock source select
000:f_{SYS}/2
001:f_{SYS}/4
010:f_{SYS}/8
011:f_{SYS}/16
100:f_{SYS}/32
101:f_{SYS}/64
110:f_{SYS}/128
111:Unused

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

The protection features are:

- Stall detection function
- Over current protection
- 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.

Current Protection Function

The device uses an internal OPA with a gain of 10, a high speed (2μs) 10-bitA/D Converter, an 8-bit D/A Converter and a comparator to measure the motor current and to detect for excessive current values. If an over current situation should occur, then the external drive circuit must be shut down immediately to prevent motor damage.

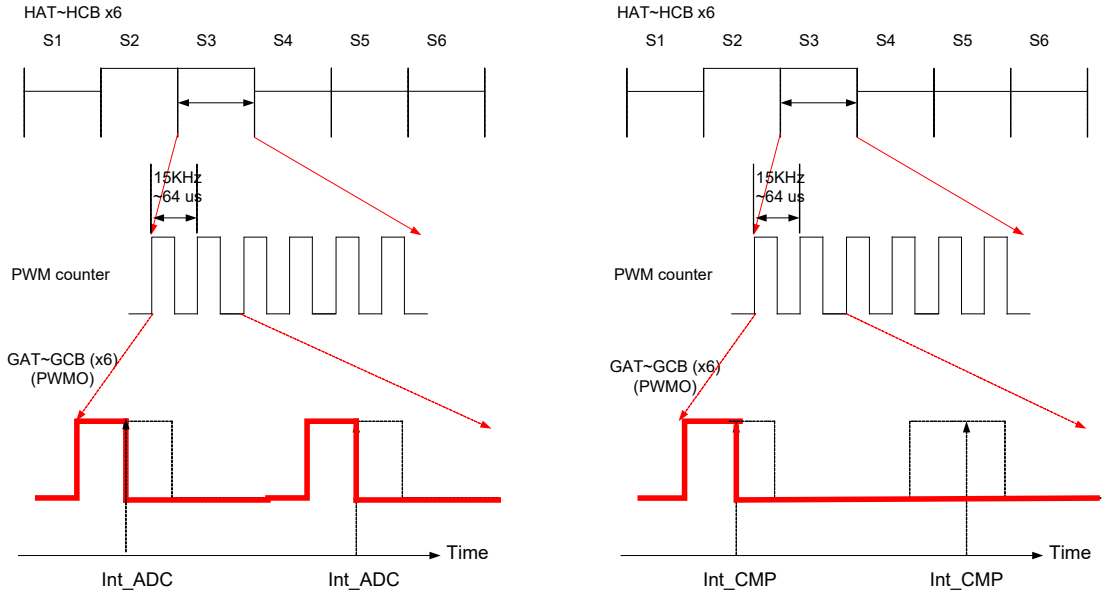
As the motor driver PCB will have rather large amounts of noise, and as this noise will be amplified by the OPA, this can easily lead to false triggering. For this reason the fault mode must be used.

For the MOS current limiting mechanism Int_AHL_Li: When AHLHE=0 then the hardware mode is disabled, and when AHLHE=1 the hardware is enabled. The current limiting circuit is a hardware circuit, for which the A/D converter channel must select the operation amplifier if it is to be effective.

AHLPS=0 → The protection circuit will allow the PWM output to immediately restart once the Int_AHL_Lim interrupt has been reset.

AHLPS=1 → The protection circuit will only allow the PWM output to restart on the next PWM period once the Int_AHL_Lim interrupt has been reset.

MOS over-current mechanism Int_Is: when ISHE=0 the hardware mode is disabled and when ISHE=1 the hardware mode is enabled. ISPS=0 then select the Fault Mode



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)
Restart 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 INT0A, INT0B and INT0C for rotor speed detection. The software will setup the CAPTMAH and CAPTMAL registers to monitor the Hall sensor inputs INT0A, INT0B and INT0C for motor speed control. If an abnormal situation exists, then a CapTM_Cmp or CapTM_Over interrupt will be generated.

Stall Detect Mechanism CapTM_Cmp: when CapCHE=0 disable the hardware mode and when CapCHE =1 enable the hardware mode. The stall detect mechanism must use the Pause Mode.

CAPCPS=1. then select the Pause Mode.

Stall Detect Mechanism CapTM_Over: when CapOHE=0 disable the hardware mode and when CapOHE=1, enable the hardware mode.

CAPOPS=1, then select the Pause Mode.

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	—	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	—
POR	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 1~0 Unimplemented, read as "0"

- **MPTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	PSWPS	AHLPS	ISPS	CAPCPS	CAPOPS
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	1	0	0	1	1

Bit 7~5 Unimplemented, read as "0"

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

Bit 3 **AHLPS**: Int_AHL_Lim Pause/Fault Mode Selection
0: Protection circuit allows immediate restart of PWM output when the Int_AHL_Lim interrupt has been reset.
1: Protection circuit only allows restart of PWM output when on the next PWM period when the Int_AHL_Lim interrupt has been reset.

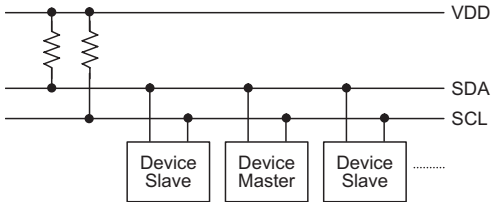
Bit 2 **ISPS**: Int_Is Pause/Fault Mode select
0: undefined, cannot be selected
1: Select Pause Mode

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

Bit 0 **CAPOPS**: CapTM_Over Pause Mode select
0: undefined, cannot be selected
1: Select Pause Mode

I²C Interface

The I²C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication protocol and the ability to accommodate multiple devices on the same bus has made it an extremely popular interface type for many applications.



I²C Master/Slave Bus Connection

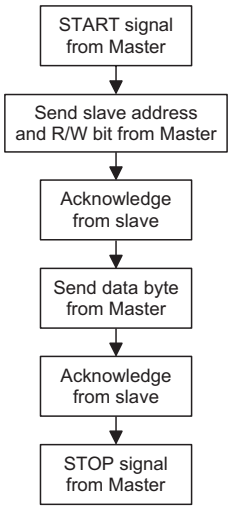
I²C Interface Operation

The I²C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I²C bus is identified by a unique address which will be transmitted and received on the I²C bus.

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For this device, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode.

It is suggested that the user shall not enter the micro processor to HALT mode by application program during processing I²C communication.

If the pin is configured to SDA or SCL function of I²C interface, the pin is configured to open-collect Input/Output port and its pull-up function can be enabled by programming the related Generic Pull-up Control Register.



I²C Registers

There are four control registers associated with the I²C bus, IICC0, IICC1, IICA and I2CTOC and one data register, IICD. The IICD register, is used to store the data being transmitted and received on the I²C bus. Before the microcontroller writes data to the I²C bus, the actual data to be transmitted must be placed in the IICD register. After the data is received from the I²C bus, the microcontroller can read it from the IICD register. Any transmission or reception of data from the I²C bus must be made via the IICD register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	I2CDBNC1	I2CDBNC0	I2CEN	—
IICC1	IICHCF	IICHAAS	IICHBB	IICHTX	IICTXAK	IICSRW	IICRNIC	IICRXAK
IICD	IICDD7	IICDD6	IICDD5	IICDD4	IICDD3	IICDD2	IICDD1	IICDD0
IICA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
I2CTOC	I2CTOEN	I2CTOF	I2CTOS5	I2CTOS4	I2CTOS3	I2CTOS2	I2CTOS1	I2CTOS0

I²C Register List

• IICC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	I2CDBNC1	I2CDBNC0	I2CEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 unimplemented, read as "0"

Bit 3~2 **I2CDBNC1~I2CDBNC0**: I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
10: 4 system clock debounce
11: 4 system clock debounce

Bit 1 **I2CEN**: I²C enable
0: Disable
1: Enable

Bit 0 Unimplemented, read as "0"

SPI function could be turned off or turned on by controlling the related pin-sharing control bit which decides the function of the IO ports pin-shared the pins SDA and SCL. When the IO ports pin-shared the pins SDA and SCL are chosen to the functions other than SDA and SCL by pin-sharing control bit, SPI function is turned off and its operating current will be reduced to a minimum value. In contrary, SPI function is turned on when the IO ports pin-shared the pins SDA and SCL are chosen to the pins SDA and SCL by controlling pin-sharing control bit.

• IICC1 Register

Bit	7	6	5	4	3	2	1	0
Name	IICHCF	IICHAAS	IICHBB	IICHTX	IICTXAK	IICSRW	IICRNIC	IICRXAK
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **IICHCF**: I²C Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer
The IICHCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Below is an example of the flow of a two-byte IIC data transfer.
First, IIC slave device receive a start signal from IIC master and then IICHCF bit is automatically cleared to zero.
Second, IIC slave device finish receiving the 1st data byte and then IICHCF bit is automatically set to one.
Third, user read the 1st data byte from IICD register by the application program and then IICHCF bit is automatically cleared to zero.
Fourth, IIC slave device finish receiving the 2nd data byte and then IICHCF bit is automatically set to one and so on.
Finally, IIC slave device receive a stop signal from IIC master and then IICHCF bit is automatically set to one.

Bit 6 **IICHAAS**: I²C Bus address match flag

0: Not address match
1: Address match

The IICHAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **IICHBB**: I²C Bus busy flag

0: I²C Bus is not busy
1: I²C Bus is busy

The IICHBB flag is the I²C busy flag. This flag will be "1" when the I²C bus is busy which will occur when a START signal is detected. The flag will be set to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **IICHTX**: Select I²C slave device is transmitter or receiver

0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **IICTXAK**: I²C Bus transmit acknowledge flag

0: Slave send acknowledge flag
1: Slave do not send acknowledge flag

The IICTXAK bit is the transmit acknowledge flag. After the slave device receipt of 8-bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set IICTXAK bit to "0" before further data is received.

Bit 2 **IICSRW**: I²C Slave Read/Write flag

0: Slave device should be in receive mode
1: Slave device should be in transmit mode

The IICSRW flag is the I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C bus. When the transmitted address and slave address is match, that is when the IICHAAS flag is set high, the slave device will check the IICSRW flag to determine whether it should be in transmit mode or receive mode. If the IICSRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the IICSRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

Bit 1 **IICRNIC**: I²C running using Internal Clock Control

0: I²C running using internal clock
1: I²C running not using Internal Clock

The I²C module can run without using internal clock, and generate an interrupt if the IIC interrupt is enabled, which can be used in SLEEP Mode, IDLE(SLOW) Mode, NORMAL(SLOW) Mode.

Bit 0 **IICRXAK**: I²C Bus Receive acknowledge flag

0: Slave receive acknowledge flag
1: Slave do not receive acknowledge flag

The IICRXAK flag is the receiver acknowledge flag. When the IICRXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the IICRXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the IICRXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

The IICD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C functions. Before the device writes data to the I²C bus, the actual data to be transmitted must be placed in the IICD register. After the data is received from the I²C bus, the device can read it from the IICD register. Any transmission or reception of data from the I²C bus must be made via the IICD register.

• IICD Register

Bit	7	6	5	4	3	2	1	0
Name	IICDD7	IICDD6	IICDD5	IICDD4	IICDD3	IICDD2	IICDD1	IICDD0
R/W	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

"x" unknown

Bit 7~0 **IICDD7~IICDD0**: IIC Data Buffer bit 7 ~ bit 0

• IICA Register

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
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

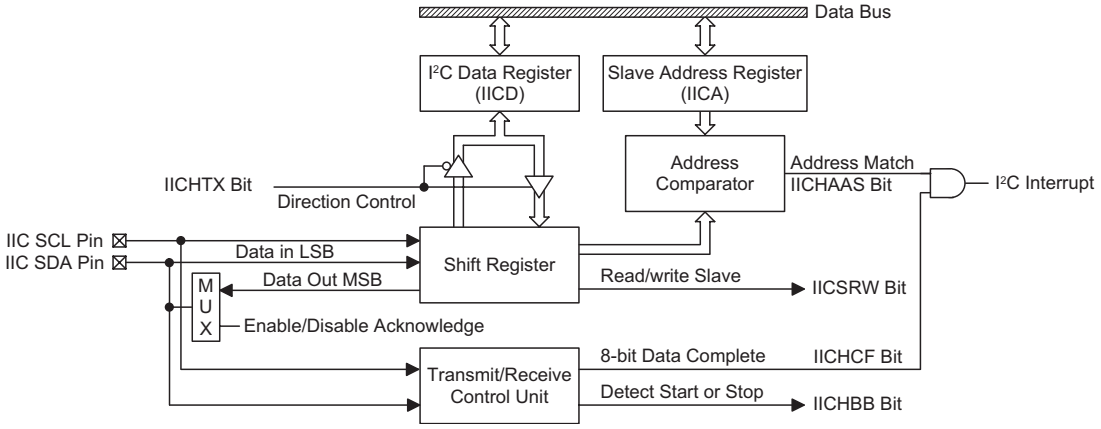
Bit 7~1 **IICA6~IICA0**: I²C slave address

IICA6~ IICA0 is the I²C slave address bit 6 ~ bit 0.

The IICA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~ 1 of the IICA register define the device slave address. Bit 0 is not defined.

When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the IICA register, the slave device will be selected.

Bit 0 Unimplemented, read as "0"

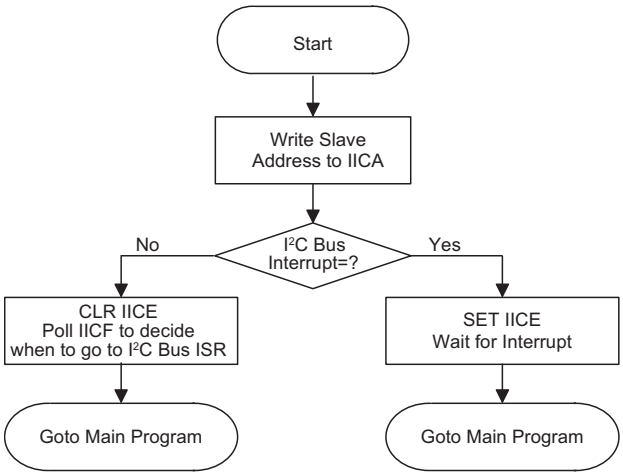


I²C Block Diagram

I²C Bus Communication

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the IICHAAS bit in the IICC1 register will be set and an I²C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the IICHAAS bit to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
  Set Configure the pin-shared I/O ports to I²C pin function. (SCL and SAD).
- Step 2
  Set I2CEN bit in the IICC0 register to "1" to enable the I²C bus.
- Step 3
  Write the slave address of the device to the I²C bus address register IICA.
- Step 4
  Set the IICE interrupt enable bit of the interrupt control register to enable the I²C interrupt and Multi-function interrupt.



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

The START signal can only be generated by the master device connected to the I²C bus and not by the slave device. This START signal will be detected by all devices connected to the I²C bus. When detected, this indicates that the I²C bus is busy and therefore the IICHBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.

Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the IICSRW bit of the IICC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag IICHAAS when the addresses match.

As an I²C bus interrupt can come from two sources, when the program enters the interrupt subroutine, the IICHAAS bit should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the SCL line.

I²C Bus Read/Write Signal

The IICSRW bit in the IICC1 register defines whether the slave device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the IICSRW flag is "1" then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the IICSRW flag is "0" then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

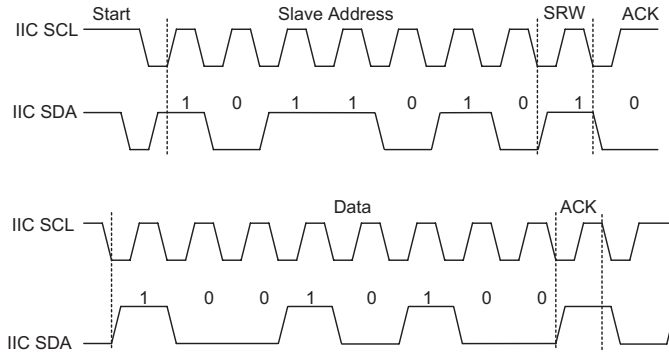
I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the IICHAAS flag is high, the addresses have matched and the slave device must check the IICSRW flag to determine if it is to be a transmitter or a receiver. If the IICSRW flag is high, the slave device should be setup to be a transmitter so the IICHTX bit in the IICC1 register should be set to "1". If the IICSRW flag is low, then the microcontroller slave device should be setup as a receiver and the IICHTX bit in the IICC1 register should be set to "0".

I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the IICD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the IICD register. If setup as a receiver, the slave device must read the transmitted data from the IICD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as IICTXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the IICRXAK bit in the IICC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.

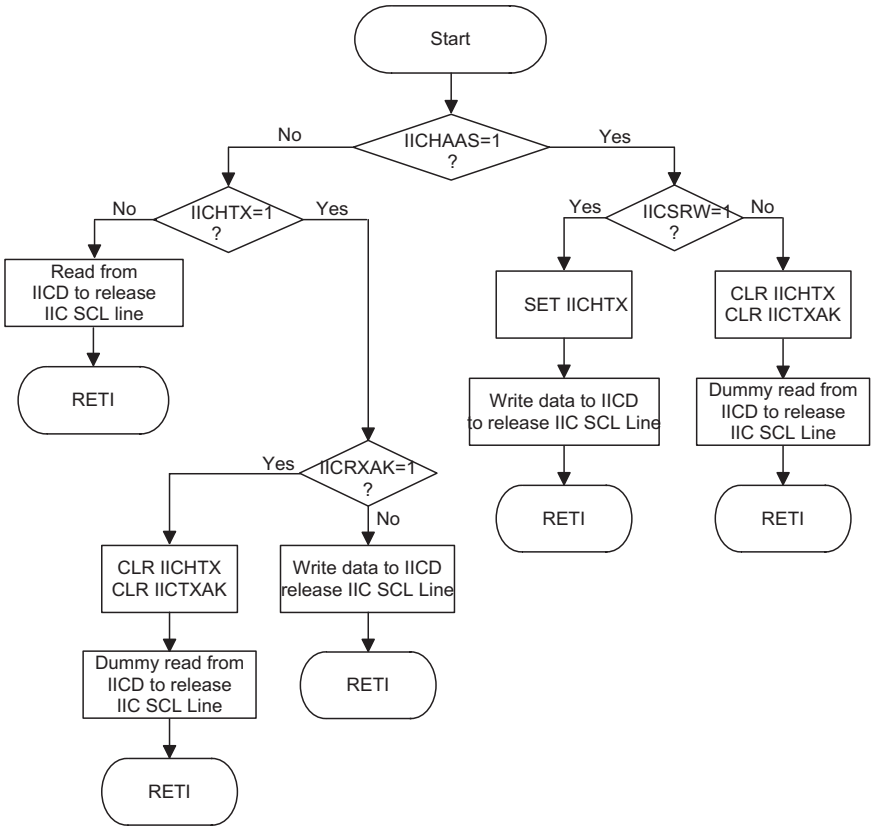


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=IICSRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (IICRXAK bit for transmitter, IICTXAK bit for receiver 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	--------	S	SA	SR	M	D	A	D	A	--------	P

I²C Communication Timing Diagram

Note: *When a slave address is matched, the device must be placed in either the transmit mode and then write data to the IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the I²C SCL line.



I²C Bus ISR Flow Chart

I²C Time-out Control

In order to reduce the problem of I²C lockup due to reception of erroneous clock sources, a time-out function is provided. If the clock source to the I²C is not received then after a fixed time period, the I²C circuitry and registers will be reset.

The time-out counter starts counting on an I²C bus "START" & "address match" condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out setup by the I2CTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.

When an I²C time-out counter overflow occurs, the counter will stop and the I2CTOEN bit will be cleared to zero and the I2CTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the I²C interrupt vector. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

Register	After I²C Time-out
IICD, IICA, IICC0	No change
IICC1	Reset to POR condition

I²C Registers After Time-out

The I2CTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using bits in the I2CTOC register. The time-out time is given by the formula:

$$((1\sim64) \times 32) / f_{SUB}.$$

This gives a range of about 1ms to 64ms. Note also that the LIRC oscillator is continuously enabled.

- **I2CTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	I2CTOEN	I2CTOF	I2CTOS5	I2CTOS4	I2CTOS3	I2CTOS2	I2CTOS1	I2CTOS0
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 **I2CTOEN**: I²C Time-out Control
0: disable
1: enable

Bit 6 **I2CTOF**: Time-out flag (set by time-out and clear by software)
0: no time-out
1: time-out occurred

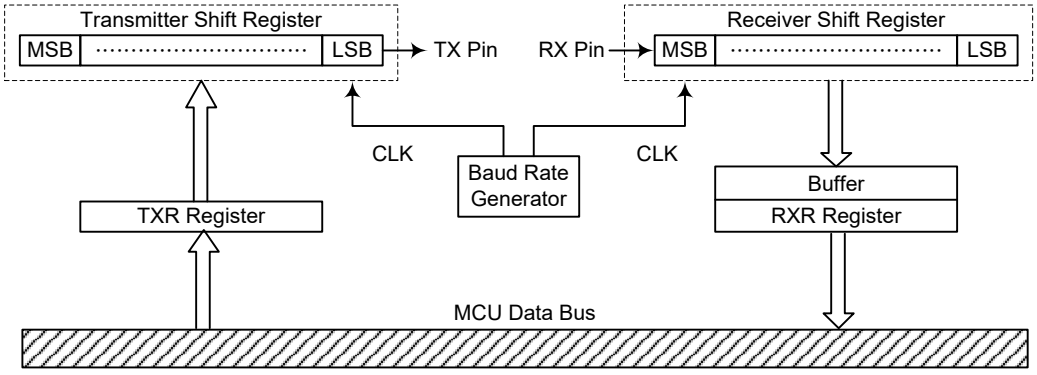
Bit 5~0 **I2CTOS5~I2CTOS0**: Time-out Definition
I²C time-out clock source is $f_{SUB}/32$.
I²C time-out time is given by: ([I2CTOS5 : I2CTOS0]+1) × $(32/f_{SUB})$

UART Interface

The device contains an integrated full-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UART function contains the following features:

- Full-duplex, asynchronous communication
- 8 or 9 bits character length
- Even, odd or no parity options
- One or two stop bits
- Baud rate generator with 8-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 2-byte Deep FIFO Receive Data Buffer
- Transmit and receive interrupts
- Interrupts can be initialized by the following conditions:
  - Transmitter Empty
  - Transmitter Idle
  - Receiver Full
  - Receiver Overrun
  - Address Mode Detect



UART Data Transfer Scheme

UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as TX and RX. The TX pin is the UART transmitter pin, which can be used as a general purpose I/O or other pin-shared functional pin if the pin is not configured as a UART transmitter pin, which occurs when the TXEN bit in the UCR2 control register is equal to zero. Similarly, the RX pin is the UART receiver pin, which can also be used as a general purpose I/O pin, if the pin is not configured as a receiver pin, which occurs if the RXEN bit in the UCR2 register is equal to zero. Along with

the UARTEN bit, the TXEN and RXEN bits, if set, will setup these UART interface pins to their respective TX output and RX input conditions if the corresponding pin-shared control is selected these pins as UART interface pins.

UART Data Transfer Scheme

The block diagram shows the overall data transfer structure arrangement for the UART interface. The actual data to be transmitted from the MCU is first transferred to the TXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate Generator. Only the TXR register is mapped onto the MCU Data Memory, the Transmit Shift Register is not mapped and is therefore inaccessible to the application program.

Data to be received by the UART is accepted on the external RX pin, from where it is shifted in, LSB first, to the Receiver Shift Register at a rate controlled by the Baud Rate Generator. When the shift register is full, the data will then be transferred from the shift register to the internal RXR register, where it is buffered and can be manipulated by the application program. Only the RXR register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.

It should be noted that the actual register for data transmission and reception, although referred to in the text, and in application programs, as separate TXR and RXR registers, only exists as a single shared register in the Data Memory. This shared register known as the TXR/RXR register is used for both data transmission and data reception.

UART Status and Control Registers

There are five control registers associated with the UART function. The USR, UCR1 and UCR2 registers control the overall function of the UART, while the BRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXR data register.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0

UART Register List

• TXR_RXR Register

The TXR_RXR register is the data register which is used to store the data to be transmitted on the TX pin or being received from the RX pin.

Bit	7	6	5	4	3	2	1	0
Name	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
R/W	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

"x": unknown

Bit 7~0 **TXRX7~TXRX0**: UART Transmit/Receive Data bits

• **USR Register**

The USR register is the status register for the UART, which can be read by the program to determine the present status of the UART. All flags within the USR register are read only and further explanations are given below:

Bit	7	6	5	4	3	2	1	0
Name	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

Bit 7 **PERR**: Parity error flag
0: No parity error is detected
1: Parity error is detected
The PERR flag is the parity error flag. When this read only flag is "0", it indicates a parity error has not been detected. When the flag is "1", it indicates that the parity of the received word is incorrect. This error flag is applicable only if Parity mode (odd or even) is selected. The flag can also be cleared by a software sequence which involves a read to the status register USR followed by an access to the RXR data register.

Bit 6 **NF**: Noise flag
0: No noise is detected
1: Noise is detected
The NF flag is the noise flag. When this read only flag is "0", it indicates no noise condition. When the flag is "1", it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of as overrun. The NF flag can be cleared by a software sequence which will involve a read to the status register USR followed by an access to the RXR data register.

Bit 5 **FERR**: Framing error flag
0: No framing error is detected
1: Framing error is detected
The FERR flag is the framing error flag. When this read only flag is "0", it indicates that there is no framing error. When the flag is "1", it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register USR followed by an access to the RXR data register.

Bit 4 **OERR**: Overrun error flag
0: No overrun error is detected
1: Overrun error is detected
The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is "0", it indicates that there is no overrun error. When the flag is "1", it indicates that an overrun error occurs which will inhibit further transfers to the RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register USR followed by an access to the RXR data register.

Bit 3 **RIDLE**: Receiver status
0: data reception is in progress (data being received)
1: no data reception is in progress (receiver is idle)
The RIDLE flag is the receiver status flag. When this read only flag is "0", it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is "1", it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is "1" indicating that the UART receiver is idle and the RX pin stays in logic high condition.

Bit 2 **RXIF**: Receive RXR data register status
0: RXR data register is empty
1: RXR data register has available data
The RXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the RXR read data register is empty. When the flag is "1", it indicates that the RXR read data register contains new data. When the contents of the shift register are transferred to the RXR register, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag is cleared when the USR register is read with RXIF set, followed by a read from the RXR register, and if the RXR register has no data available.

Bit 1 **TIDLE**: Transmission status
0: data transmission is in progress (data being transmitted)
1: no data transmission is in progress (transmitter is idle)
The TIDLE flag is known as the transmission complete flag. When this read only flag is "0", it indicates that a transmission is in progress. This flag will be set to "1" when the TXIF flag is "1" and when there is no transmit data or break character being transmitted. When TIDLE is equal to 1, the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared by reading the USR register with TIDLE set and then writing to the TXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 **TXIF**: Transmit TXR data register status
0: character is not transferred to the transmit shift register
1: character has transferred to the transmit shift register (TXR data register is empty)
The TXIF flag is the transmit data register empty flag. When this read only flag is "0", it indicates that the character is not transferred to the transmitter shift register. When the flag is "1", it indicates that the transmitter shift register has received a character from the TXR data register. The TXIF flag is cleared by reading the UART status register (USR) with TXIF set and then writing to the TXR data register. Note that when the TXEN bit is set, the TXIF flag bit will also be set since the transmit data register is not yet full.

• UCR1 Register

The UCR1 register together with the UCR2 register are the two UART control registers that are used to set the various options for the UART function such as overall on/off control, parity control, data transfer bit length, etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

"x": unknown

Bit 7 **UARTEN**: UART function enable control
0: disable UART. TX and RX pins are in a floating state
1: enable UART. TX and RX pins function as UART pins
The UARTEN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX pin as well as the TX pin will be in a floating state. When the bit is equal to "1", the UART will be enabled and the TX and RX pins will function as defined by the TXEN and RXEN enable control bits. When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2 and BRG registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, all pending transmissions and

receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6 **BNO**: Number of data transfer bits selection
0: 8-bit data transfer
1: 9-bit data transfer
This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to "1", a 9-bit data length format will be selected. If the bit is equal to "0", then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits RX8 and TX8 will be used to store the 9th bit of the received and transmitted data respectively.

Bit 5 **PREN**: Parity function enable control
0: Parity function is disabled
1: Parity function is enabled
This bit is the parity function enable bit. When this bit is equal to 1, the parity function will be enabled. If the bit is equal to 0, then the parity function will be disabled.

Bit 4 **PRT**: Parity type selection bit
0: Even parity for parity generator
1: Odd parity for parity generator
This bit is the parity type selection bit. When this bit is equal to 1, odd parity type will be selected. If the bit is equal to 0, then even parity type will be selected.

Bit 3 **STOPS**: Number of stop bits selection
0: One stop bit format is used
1: Two stop bits format is used
This bit determines if one or two stop bits are to be used. When this bit is equal to "1", two stop bits format are used. If the bit is equal to "0", then only one stop bit format is used.

Bit 2 **TXBRK**: Transmit break character
0: No break character is transmitted
1: Break characters transmit
The TXBRK bit is the Transmit Break Character bit. When this bit is equal to "0", there are no break characters and the TX pin operates normally. When the bit is equal to "1", there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to "1", after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.

Bit 1 **RX8**: Receive data bit 8 for 9-bit data transfer format (read only)
This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **TX8**: Transmit data bit 8 for 9-bit data transfer format (write only)
This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UCR2 Register**

The UCR2 register is the second of the UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation if the UART Transmitter and Receiver as well as enabling the various UART interrupt sources. The register also serves to control the baud rate speed, receiver wake-up function enable and the address detect function enable. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
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 **TXEN**: UART Transmitter enable control
0: UART Transmitter is disabled
1: UART Transmitter is enabled
The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to "0", the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TX pin will be in a floating state. If the TXEN bit is equal to "1" and the UARTEN bit is also equal to 1, the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be in a floating state.

Bit 6 **RXEN**: UART Receiver enable control
0: UART Receiver is disabled
1: UART Receiver is enabled
The bit named RXEN is the Receiver Enable Bit. When this bit is equal to "0", the receiver will be disabled with any pending data receptions being aborted. In addition the receiver buffers will be reset. In this situation the RX pin will be in a floating state. If the RXEN bit is equal to "1" and the UARTEN bit is also equal to 1, the receiver will be enabled and the RX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX pin will be in a floating state.

Bit 5 **BRGH**: Baud Rate speed selection
0: Low speed baud rate
1: High speed baud rate
The bit named BRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register, BRG, controls the baud rate of the UART. If the bit is equal to 0, the low speed mode is selected.

Bit 4 **ADDEN**: Address detect function enable control
0: Address detection function is disabled
1: Address detection function is enabled
The bit named ADDEN is the address detection function enable control bit. When this bit is equal to 1, the address detection function is enabled. When it occurs, if the 8th bit, which corresponds to RX7 if BNO=0, or the 9th bit, which corresponds to RX8 if BNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of the BNO bit. If the address bit known as the 8th or 9th bit of the received word is "0" with the address detection function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **WAKE**: RX pin falling edge wake-up function enable control
0: RX pin wake-up function is disabled
1: RX pin wake-up function is enabled
The bit enables or disables the receiver wake-up function. If this bit is equal to 1 and the device is in IDLE or SLEEP mode, a falling edge on the RX pin will wake up the device. If this bit is equal to 0 and the device is in IDLE or SLEEP mode, any edge transitions on the RX pin will not wake up the device.

Bit 2 **RIE**: Receiver interrupt enable control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled
The bit enables or disables the receiver interrupt. If this bit is equal to 1 and when the receiver overrun flag OERR or received data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to 0, the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.

Bit 1 **TIIE**: Transmitter Idle interrupt enable control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled
The bit enables or disables the transmitter idle interrupt. If this bit is equal to 1 and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to 0, the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

Bit 0 **TEIE**: Transmitter Empty interrupt enable control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled
The bit enables or disables the transmitter empty interrupt. If this bit is equal to 1 and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to 0, the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRG register and the second is the value of the BRGH bit within the UCR2 control register. The BRGH bit decides, if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value in the BRG register, N, which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the BRG register and has a range of between 0 and 255.

UCR2 BRGH Bit	0	1
Baud Rate (BR)	$\frac{f_{SYS}}{[64(N+1)]}$	$\frac{f_{SYS}}{[16(N+1)]}$

By programming the BRGH bit which allows selection of the related formula and programming the required value in the BRG register, the required baud rate can be setup. Note that because the actual baud rate is determined using a discrete value, N, placed in the BRG register, there will be an error associated between the actual and requested value. The following example shows how the BRG register value N and the error value can be calculated.

• **BRG Register**

Bit	7	6	5	4	3	2	1	0
Name	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
R/W	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

"x": unknown

Bit 7~0 **BRG7~BRG0**: Baud Rate values

By programming the BRGH bit in the UCR2 register which allows selection of the related formula described above and programming the required value in the BRG register, the required baud rate can be setup.

Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, and with BRGH set to 0 determine the BRG register value N, the actual baud rate and the error value for a desired baud rate of 4800.

From the above table the desired baud rate $BR = \frac{f_{SYS}}{[64(N+1)]}$

Re-arranging this equation gives $N = \frac{f_{SYS}}{(BR \times 64)} - 1$

Giving a value for $N = \frac{4000000}{(4800 \times 64)} - 1 = 12.0208$

To obtain the closest value, a decimal value of 12 should be placed into the BRG register. This gives an actual or calculated baud rate value of $BR = \frac{4000000}{[64(12+1)]} = 4808$

Therefore the error is equal to $\frac{4808 - 4800}{4800} = 0.16\%$

The following tables show the actual values of baud rate and error values for the two value of BRGH.

Baud Rate K/BPS	f_{SYS}=8MHz					
	Baud Rates for BRGH=0			Baud Rates for BRGH=1		
	BRG	Kbaud	Error (%)	BRG	Kbaud	Error (%)
0.3	—	—	—	—	—	—
1.2	103	1.202	0.16	—	—	—
2.4	51	2.404	0.16	207	2.404	0.16
4.8	25	4.808	0.16	103	4.808	0.16
9.6	12	9.615	0.16	51	9.615	0.16
19.2	6	17.8857	-6.99	25	19.231	0.16
38.4	2	41.667	8.51	12	38.462	0.16
57.6	1	62.500	8.51	8	55.556	-3.55
115.2	0	125	8.51	3	125	8.51
250	—	—	—	1	250	0

Baud Rates and Error Values

UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ format. This is composed of one start bit, eight or nine data bits and one or two stop bits. Parity is supported by the UART hardware and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding BNO, PRT, PREN and STOPS bits in

the UCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the transmitter and receiver of the UART are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

The basic on/off function of the internal UART function is controlled using the UARTEN bit in the UCR1 register. If the UARTEN, TXEN and RXEN bits are set, then these two UART pins will act as normal TX output pin and RX input pin respectively. If no data is being transmitted on the TX pin, then it will default to a logic high value.

Clearing the UARTEN bit will disable the TX and RX pins and these two pins will be used as an I/O or other pin-shared functional pin. When the UART function is disabled, the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the enable control, the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2 and BRG registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

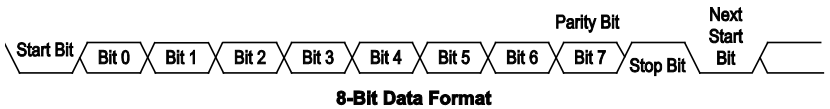
Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1 register. The BNO bit controls the number of data bits which can be set to either 8 or 9. The PRT bit controls the choice if odd or even parity. The PREN bit controls the parity on/off function. The STOPS bit decides whether one or two stop bits are to be used. The following table shows various formats for data transmission. The address detect mode control bit identifies the frame as an address character. The number of stop bits, which can be either one or two, is independent of the data length.

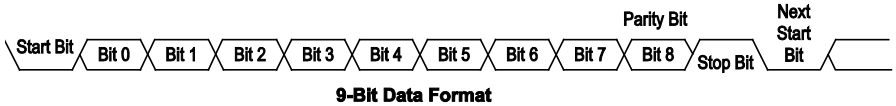
Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
Exmaple of 9-bit Data Formats				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

Transmitter Receiver Data Format

The following diagram shows the transmit and receive waveforms for both 8-bit and 9-bit data formats.



8-Bit Data Format



9-Bit Data Format

UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR register. The data to be transmitted is loaded into this TXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the TXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN bit is set, but the data will not be transmitted until the TXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin will then return to the I/O or other pin-shared function.

Transmitting Data

When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant bit LSB first. In the transmit mode, the TXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8 bit in the UCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT, PREN and STOPS bits to define the required word length, parity type and number of stop bits.
- Setup the BRG register to select the desired baud rate.
- Set the TXEN bit to ensure that the UART transmitter is enabled and the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR register. Note that this step will clear the TXIF bit.

This sequence of events can now be repeated to send additional data. It should be noted that when TXIF=0, data will be inhibited from being written to the TXR register. Clearing the TXIF flag is always achieved using the following software sequence:

1. A USR register access

2. A TXR register write execution

The read-only TXIF flag is set by the UART hardware and if set indicates that the TXR register is empty and that other data can now be written into the TXR register without overwriting the previous data. If the TEIE bit is set, then the TXIF flag will generate an interrupt. During a data transmission, a write instruction to the TXR register will place the data into the TXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLE bit the following software sequence is used:

1. A USR register access
2. A TXR register write execution

Note that both the TXIF and TIDLE bits are cleared by the same software sequence.

Transmitting Break

If the TXBRK bit is set, then the break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13×N "0" bits, where N=1, 2, etc. if a break character is to be transmitted, then the TXBRK bit must be first set by the application program and then cleared to generate the stop bits. Transmitting a break character will not generate a transmit interrupt. Note that a break condition length is at least 13 bits long. If the TXBRK bit is continually kept at a logic high level, then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the TXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out one or two stop bits. The automatic logic high at the end of the last break character will ensure that the start bit of the next frame is recognized.

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, will be stored in the RX8 bit in the UCR1 register. At the receiver core lies the Receiver Shift Register more commonly known as the RSR. The data which is received on the RX external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

Receiving Data

When the UART receiver is receiving data, the data is serially shifted in on the external RX input pin to the shift register, with the least significant bit LSB first. The RXR register is a four byte deep FIFO data buffer, where two bytes can be held in the FIFO while the 3rd byte can continue to be received. Note that the application program must ensure that the data is read from RXR before the 3rd byte has been completely shifted in, otherwise the 3rd byte will be discarded and an overrun error OERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT, PREN and STOPS bits to define the required word length, parity type and number of stop bits.
- Setup the BRG register to select the desired baud rate.
- Set the RXEN bit to ensure that the UART receiver is enabled and the RX pin is used as a UART receiver pin.

At this point the receiver will be enabled which will begin to look for a start bit.

When a character is received, the following sequence of events will occur:

- The RXIF bit in the USR register will be set then RXR register has data available, at least three more character can be read.

- When the contents of the shift register have been transferred to the RXR register and if the RIE bit is set, then an interrupt will be generated.
- If during reception, a frame error, noise error, parity error or an overrun error has been detected, then the error flags can be set.

The RXIF bit can be cleared using the following software sequence:

1. A USR register access
2. A RXR register read execution

Receiving Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the BNO and STOPS bits. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO and STOPS. The RXIF bit is set, FERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLE bit is set. If a long break signal has been detected and the receiver has received a start bit, the data bits and the invalid stop bit, which sets the FERR flag, the receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. A break is regarded as a character that contains only zeros with the FERR flag set. The break character will be loaded into the buffer and no further data will be received until stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

- The framing error flag, FERR, will be set.
- The receive data register, RXR, will be cleared.
- The OERR, NF, PERR, RIDLE or RXIF flags will possibly be set.

Idle Status

When the receiver is reading data, which means it will be in between the detection of a start bit and the reading of a stop bit, the receiver status flag in the USR register, otherwise known as the RIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the RIDLE flag will have a high value, which indicates the receiver is in an idle condition.

Receiver Interrupt

The read only receive interrupt flag RXIF in the USR register is set by an edge generated by the receiver. An interrupt is generated if RIE=1, when a word is transferred from the Receive Shift Register, RSR, to the Receive Data Register, RXR. An overrun error can also generate an interrupt if RIE=1.

Managing Receiver Errors

Several types of reception errors can occur within the UART module, the following section describes the various types and how they are managed by the UART.

Overrun Error – OERR

The RXR register is composed of a four byte deep FIFO data buffer, where four bytes can be held in the FIFO register, while a 5th byte can continue to be received. Before the 5th byte has been entirely shifted in, the data should be read from the RXR register. If this is not done, the overrun error flag OERR will be consequently indicated.

In the event of an overrun error occurring, the following will happen:

- The OERR flag in the USR register will be set.
- The RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIE bit is set.

The OERR flag can be cleared by an access to the USR register followed by a read to the RXR register.

Noise Error – NF

Over-sampling is used for data recovery to identify valid incoming data and noise. If noise is detected within a frame, the following will occur:

- The read only noise flag, NF, in the USR register will be set on the rising edge of the RXIF bit.
- Data will be transferred from the shift register to the RXR register.
- No interrupt will be generated. However this bit rises at the same time as the RXIF bit which itself generates an interrupt.

Note that the NF flag is reset by a USR register read operation followed by an RXR register read operation.

Framing Error – FERR

The read only framing error flag, FERR, in the USR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, both stop bits must be high. Otherwise the FERR flag will be set. The FERR flag is buffered along with the received data and is cleared in any reset.

Parity Error – PERR

The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity function is enabled, PREN=1, and if the parity type, odd or even, is selected. The read only PERR flag is buffered along with the received data bytes. It is cleared on any reset, it should be noted that the FERR and PERR flags are buffered along with the corresponding word and should be read before reading the data word.

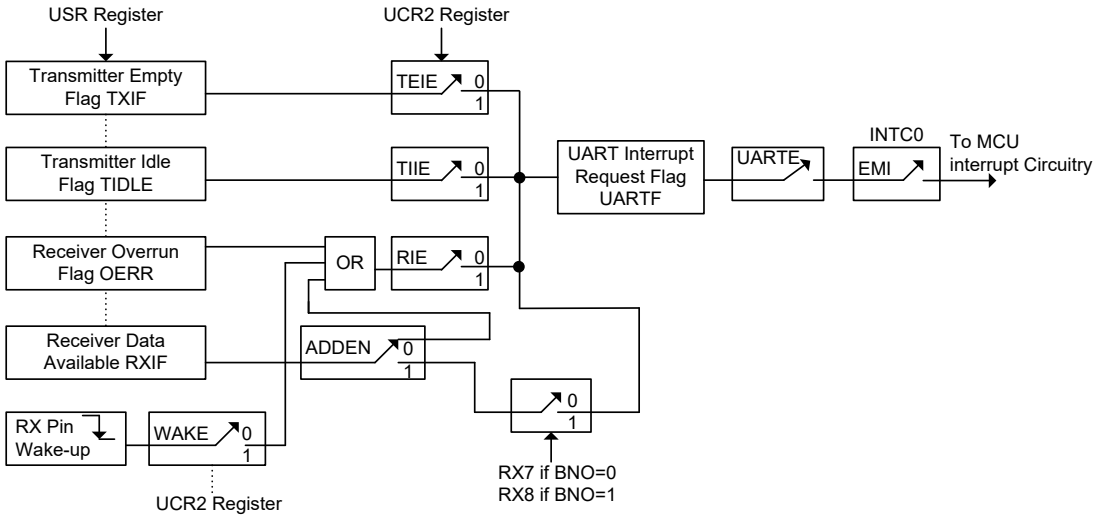
UART Interrupt Structure

Several individual UART conditions can generate a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. When any of these conditions are created, if its corresponding interrupt control is enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding USR register flags which will generate a UART interrupt if its associated interrupt enable control bit in the UCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual UART interrupt sources.

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the microcontroller is woken up by a falling edge on the RX pin, if the WAKE and RIE bits in the UCR2 register are set. Note that in the event of an RX wake-up interrupt occurring, there will be a

certain period of delay, commonly known as the System Start-up Time, for the oscillator to restart and stabilize before the system resumes normal operation.

Note that the USR register flags are read only and cannot be cleared or set by the application program, neither will they be cleared when the program jumps to the corresponding interrupt servicing routine, as is the case for some of the other interrupts. The flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART register section. The overall UART interrupt can be disabled or enabled by the related interrupt enable control bits in the interrupt control registers of the microcontroller to decide whether the interrupt requested by the UART module is masked out or allowed.



UART Interrupt Structure

Address Detect Mode

Setting the Address Detect function enable control bit, ADDEN, in the UCR2 register, enables this special function. If this bit is set to 1, then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is equal to 1, then when the data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the related interrupt enable control bit and the EMI bit of the microcontroller must also be enabled for correct interrupt generation. The highest address bit is the 9th bit if the bit BNO=1 or the 8th bit if the bit BNO=0. If the highest bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is equal to 0, then a Receive Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last but status. The address detection and parity functions are mutually exclusive functions. Therefore, if the address detect function is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity function enable bit PREN to zero.

ADDEN	Bit 9 if BNO=1 Bit 8 if BNO=0	UART Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

ADDEN Bit Function

UART Power Down Mode and Wake-up

When the MCU is in the Power Down Mode, the UART will cease to function. When the device enters the Power Down Mode, all clock sources to the module are shutdown. If the MCU enters the Power Down Mode while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the Power Down Mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the Power Down Mode, note that the USR, UCR1, UCR2, transmit and receive registers, as well as the BRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the Power Down mode.

The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set before the MCU enters the Power Down Mode, then a falling edge on the RX pin will wake up the MCU from the Power Down Mode. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX pin will be ignored.

For a UART wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UART interrupt enable bit, UARTE, must also be set. If these two bits are not set then only a wake up event will occur and no interrupt will be generated. Note also that as it takes certain system clock cycles after a wake-up before normal microcontroller resumes, the UART interrupt will not be generated until after this time has elapsed.

Low Voltage Detector – LVD

The device has a Low Voltage Detector function, also known as LVD. This enables the device to monitor the power supply voltage, V_{DD}, and provides 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 a fixed voltage below which a low voltage condition will be detemined. 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

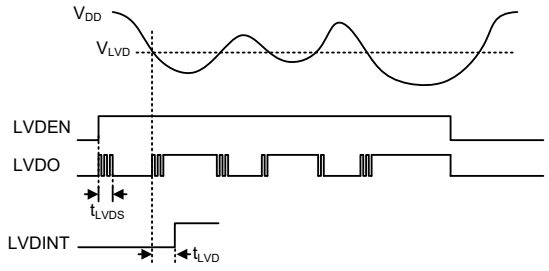
Bit 4 **LVDEN**: Low Voltage Detector Control
0: Disable
1: Enable

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: 3.6V
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 voltage of 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.

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. The device contains several external interrupt and internal interrupts functions. The external interrupt is generated by the action of the external H1, H2, H3, NFIN and INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, Comparators, 16-bit CAPTM, 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 depends upon the device chosen but fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI6 registers which setup the Multi-function interrupts. Finally there are two registers, INTEG0 and INTEG1, to set up the external interrupt trigger edge type.

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 interrupts)	HALAE	HALAF	Noise filtered inputs
	HALBE	HALBF	
	HALCE	HALCF	
External interrupt 1	INT1E	INT1F	—
Comparator 0	C0E	C0F	—
Noise filter input function	NFIE	NFIF	—
LVD	LVDE	LVDF	—
EEPROM write operation	EPWE	EPWF	—
I^2C interface	IICE	IICF	—
UART interface	UARTE	UARTF	—
Time Base	TBE	TBF	—
Multi-function	MFnE	MFnF	n = 0 ~ 6
A/D converter	AEOCE	AEOCF	—
	ALIME	ALIMF	—
16-bit Capture Timer Module	CAPOE	CAPOF	—
	CAPCE	CAPCF	
PWM function	PWMDnE	PWMDnF	n = 0 ~ 2
	PWMPnE	PWMPnF	—
TM	PTMnPE	PTMnPF	n = 0 ~ 3
	PTMnAE	PTMnAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG0	—	HSEL	INTCS1	INTCS0	INTBS1	INTBS0	INTAS1	INTAS0
INTEG1	—	—	—	—	—	—	INT1S1	INT1S0
INTC0	—	C0F	INT1F	HALLF (MF0F)	C0E	INT1E	HALLE (MF0E)	EMI
INTC1	EPWF	LVDF	MF1F	NFIF	EPWE	LVDE	MF1E	NFIE
INTC2	MF5F	MF4F	MF3F	MF2F	MF5E	MF4E	MF3E	MF2E
INTC3	TBF	MF6F	UARTF	IICF	TBE	MF6E	UARTE	IICE
MFI0	—	HALCF	HALBF	HALAF	—	HALCE	HALBE	HALAE
MFI1	CAPCF	CAPOF	ALIMF	AEOCF	CAPCE	CAPOE	ALIME	AEOCE
MFI2	PWMPF	PWMD2F	PWMD1F	PWMD0F	PWMPE	PWMD2E	PWMD1E	PWMD0E
MFI3	—	—	PTM2AF	PTM2PF	—	—	PTM2AE	PTM2PE
MFI4	—	—	PTM0AF	PTM0PF	—	—	PTM0AE	PTM0PE
MFI5	—	—	PTM1AF	PTM1PF	—	—	PTM1AE	PTM1PE
MFI6	—	—	PTM3AF	PTM3PF	—	—	PTM3AE	PTM3PE

Interrupt Register List

- **INTEG0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	HSEL	INTCS1	INTCS0	INTBS1	INTBS0	INTAS1	INTAS0
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 **HSEL**: HA/HB/HC source select
0: H1/H2/H3
1: CMP1/CMP2/CMP3 output

bit 5~4 **INTCS1, INTCS0:** FHC Interrupt edge control for INTC
00 : disable
01 : rising edge trigger
10 : falling edge trigger
11 : dual edge trigger

bit 3~2 **INTBS1, INTBS0:** FHB Interrupt edge control for INTB
00 : disable
01 : rising edge trigger
10 : falling edge trigger
11 : dual edge trigger

bit 1~0 **INTAS1, INTAS0:** FHA Interrupt edge control for INTA
00 : disable
01 : rising edge trigger
10 : falling edge trigger
11 : dual edge trigger

- **INTEG1 Register**

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

Bit 7~2 unimplemented, read as “0”

Bit 1~0 **INT1S1~INT1S0**: Define INT1 interrupt input active edge
00: Disabled interrupt
01: Rising edge interrupt
10: Falling edge interrupt
11: Dual edges interrupt

• **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 **C0F**: 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	EPWF	LVDF	MF1F	NFIF	EPWE	LVDE	MF1E	NFIE
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 **EPWF** : Data EEPROM interrupt request flag
0: No request
1: Interrupt request

Bit 6 **LVDF**: LVD 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 **NFIF**: Noise filtered input interrupt request flag
0: No request
1: Interrupt request

Bit 3 **EPWE** : Data EEPROM interrupt control
0: Disable
1: Enable

Bit 2 **LVDE**: LVD interrupt control
0: Disable
1: Enable
Bit 1 **MF1E**: Multi-function Interrupt 1 Control
0: Disable
1: Enable
Bit 0 **NFIE**: Noise filtered input interrupt control
0: Disable
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	MF5F	MF4F	MF3F	MF2F	MF5E	MF4E	MF3E	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 **MF5F**: Multi-function interrupt 5 request flag
0: No request
1: Interrupt request
Bit 6 **MF4F**: Multi-function interrupt 4 request flag
0: No request
1: Interrupt request
Bit 5 **MF3F**: Multi-function interrupt 3 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 **MF5E**: Multi-function interrupt 5 control
0: Disable
1: Enable
Bit 2 **MF4E**: Multi-function interrupt 4 control
0: Disable
1: Enable
Bit 1 **MF3E**: Multi-function interrupt 3 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	TBF	MF6F	UARTF	IICF	TBE	MF6E	UARTE	IICE
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 **TBF**: Time Base interrupt request flag
0: no request
1: interrupt request
Bit 6 **MF6F**: Multi-function interrupt 6 request flag
0: no request
1: interrupt request
Bit 5 **UARTF**: UART interface interrupt request flag
0: no request
1: interrupt request

Bit 4 **IICF**: I²C interface interrupt request flag
0: no request
1: interrupt request

Bit 3 **TBE**: Time Base interrupt control
0: disable
1: enable

Bit 2 **MF6E**: Multi-function interrupt 6 control
0: disable
1: enable

Bit 1 **UARTE**: UART interface interrupt control
0: disable
1: enable

Bit 0 **IICE**: I²C interface interrupt 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	CAPCF	CAPOF	ALIMF	AEOCF	CAPCE	CAPOE	ALIME	AEOCE
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 **CAPCF**: CAPTM compare match interrupt request flag
0: No request
1: Interrupt request
Bit 6 **CAPOF**: CAPTM capture overflow interrupt request flag
0: No request
1: Interrupt request

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 **CAPCE**: CAPTM compare match interrupt control
0: Disable
1: Enable
Bit 2 **CAPOE**: CAPTM capture overflow interrupt control
0: Disable
1: Enable
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	PWMD2F	PWMD1F	PWMD0F	PWMPE	PWMD2E	PWMD1E	PWMD0E
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 **PWMPF**: PWM Period match interrupt request flag
0: No request
1: Interrupt request
Bit 6 **PWMD2F**: PWM2 Duty match interrupt request flag
0: No request
1: Interrupt request
Bit 5 **PWMD1F**: PWM1 Duty match interrupt request flag
0: No request
1: Interrupt request
Bit 4 **PWMD0F**: PWM0 Duty match interrupt request flag
0: No request
1: Interrupt request
Bit 3 **PWMPE**: PWM Period match interrupt Interrupt Control
0: Disable
1: Enable
Bit 2 **PWMD2E**: PWM2 Duty match interrupt Control
0: Disable
1: Enable
Bit 1 **PWMD1E**: PWM1 Duty match interrupt Control
0: Disable
1: Enable
Bit 0 **PWMD0E**: PWM0 Duty match interrupt Control
0: Disable
1: Enable

• **MFI3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM2AF	PTM2PF	—	—	PTM2AE	PTM2PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 unimplemented, read as “0”

Bit 5 **PTM2AF**: TM2 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTM2PF**: TM2 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 unimplemented, read as "0"

Bit 1 **PTM2AE**: TM2 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTM2PE**: TM2 Comparator P match interrupt control
0: Disable
1: Enable

• **MFI4 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM0AF	PTM0PF	—	—	PTM0AE	PTM0PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7 ~6 Unimplemented, read as "0"

Bit 5 **PTM0AF**: TM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTM0PF**: TM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTM0AE**: TM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTM0PE**: TM0 Comparator P match interrupt control
0: Disable
1: Enable

• **MFI5 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM1AF	PTM1PF	—	—	PTM1AE	PTM1PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 unimplemented, read as "0"

Bit 5 **PTM1AF**: TM1 Comparator A match Interrupt request flag
0: no request
1: interrupt request

Bit 4 **PTM1PF**: TM1 Comparator P match Interrupt request flag
0: no request
1: interrupt request

Bit 3~2 unimplemented, read as "0"

Bit 1 **PTM1AE**: TM1 Comparator A match Interrupt control
0: disable
1: enable

Bit 0 **PTM1PE**: TM1 Comparator P match Interrupt control
0: disable
1: enable

• **MFI6 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM3AF	PTM3PF	—	—	PTM3AE	PTM3PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 unimplemented, read as "0"

Bit 5 **PTM3AF**: TM3 Comparator A match Interrupt request flag
0: no request
1: interrupt request

Bit 4 **PTM3PF**: TM3 Comparator P match Interrupt request flag
0: no request
1: interrupt request

Bit 3~2 unimplemented, read as "0"

Bit 1 **PTM3AE**: TM3 Comparator A match Interrupt control
0: disable
1: enable

Bit 0 **PTM3PE**: TM3 Comparator P match Interrupt control
0: disable
1: enable

Interrupt Operation

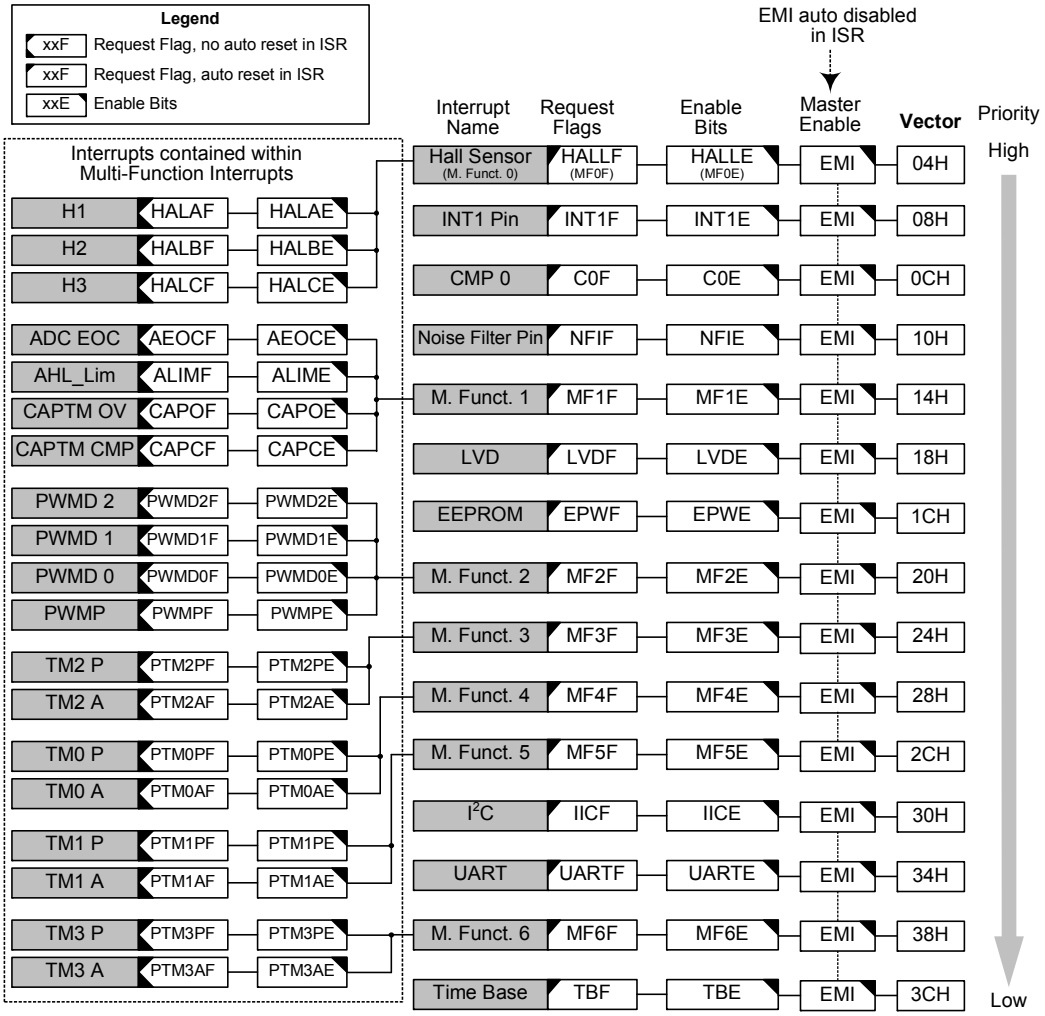
When the conditions for an interrupt event occur, such as a TM Compare P or Compare A match or A/D conversion completion etc, the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of 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 a Multi-function Interrupt. It is controlled by signal transitions on the pins, Hall Sensor input pins, H1, H2 and H3. An external interrupt request will take place when the external interrupt request flag, HALAF, HALBF or 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. Additionally the correct interrupt edge type must be selected using the INTEG0 register to enable the external Hall sensor input pin interrupt function and to choose the trigger edge type. As the external Hall sensor pins are pin-shared with I/O pins, they should be configured as an external Hall sensor input pins before the Hall sensor pin interrupt functions are enabled.

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 INT1 pin. 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. Additionally the correct interrupt edge type must be selected using the INTEG1 register to enable the external interrupt function and to choose the trigger edge type. As this external interrupt pin is pin-shared with I/O pins, it should be configured as external interrupt pins before the external pin interrupt function is enabled. 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.

Noise Filtered Input Interrupt

The external noise filtered input interrupt is controlled by signal transitions on the NFIN pin. An external interrupt request will take place when the external interrupt request flag, NFIF, is set, which will occurs when a transition appears on the external interrupt pin. Additionally the correct interrupt edge type must be selected using the NFIS1 and NFIS0 bits in the NF_VIL register to enable the external interrupt function and to choose the trigger edge type. As this external noise filtered interrupt pin is pin-shared with I/O pins, it should be configured as the noise filtered interrupt pin before the external pin interrupt function is enabled. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, NFIE, 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, NFIF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that the pull-high resistor selection on the external NFIN pin will remain valid even if the pin is used as an external interrupt input.

Comparator Interrupt

The comparator interrupt is controlled by the internal comparator 0. A comparator interrupt request will take place when the comparator interrupt request flag, C0F, 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, 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 flag, C0F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Time Base Interrupt

The function of the Time Base Interrupt is to provide regular time signal in the form of an internal interrupt. It is controlled by the overflow signal 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} or $f_{SYS}/4$ and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges.

- **TBC Register**

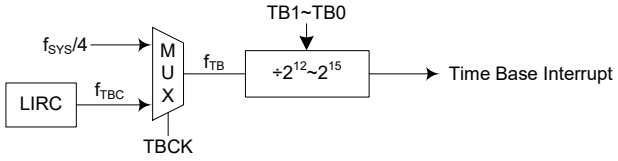
Bit	7	6	5	4	3	2	1	0
Name	TBON	TBCK	TB1	TB0	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	1	1	—	—	—	—

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: $2^{12}/f_{TB}$
01: $2^{13}/f_{TB}$
10: $2^{14}/f_{TB}$
11: $2^{15}/f_{TB}$

Bit 3~0 Unimplemented, read as "0"



Time Base Interrupt

Multi-function Interrupt

Within this device are seven 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.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags, HALLF and MFnF, 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 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.

PWM Module Interrupts

The PWM Module has four interrups. All of them are contained in Multi-function interrupt, which is known as PWMDn and PWMP. They are the Duty or the Period maching of the PWM Module. A PWM interrupt request will take place when the PWM interrupt request flag, PWMDnF 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, PWMDnF or PWMPF, must be manually cleared by the application program.

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 matches, 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 Module Interrupt

The Periodic Type TMs have two interrupts each. All of the TM interrupts are contained within the Multi-function Interrupts. For each of the Periodic Type TMs there are two interrupt request flags PTMnPF and PTMnAF and two enable bits PTMnPE and PTMnAE. 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.

EEPROM Interrupt

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

LVD 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, and Low Voltage Interrupt enable bit, LVDE, 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 LVD Interrupt vector, will take place. When the EEPROM interrupt is serviced, the interrupt request flag, LVDF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

I²C Interrupt

A I²C Interrupt request will take place when the I²C Interrupt request flag,IICF, is set, which occurs when a byte of data has been received or transmitted by the I²C interface. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, IICE, must first be set. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the I²C interface, a subroutine call to the respective Interrupt vector, will take place. When the I²C Interface Interrupt is serviced, the interrupt request flag, IICF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

UART Interrupt

A UART Interrupt request will take place when the UART Interrupt request flag, UARTF, is set, which occurs when any one of the UART interrupt trigger events occurs on the UART interface. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the UART Interface Interrupt enable bit, UARTE, must first be set. When the interrupt is enabled, the stack is not full and an UART interrupt trigger event occurs on the UART interface, a subroutine call to the respective Interrupt vector, will take place. When the UART Interface Interrupt is serviced, the interrupt request flag, UARTF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

Interrupt Wake-up Function

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

Programming Considerations

By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt 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~MF6F, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

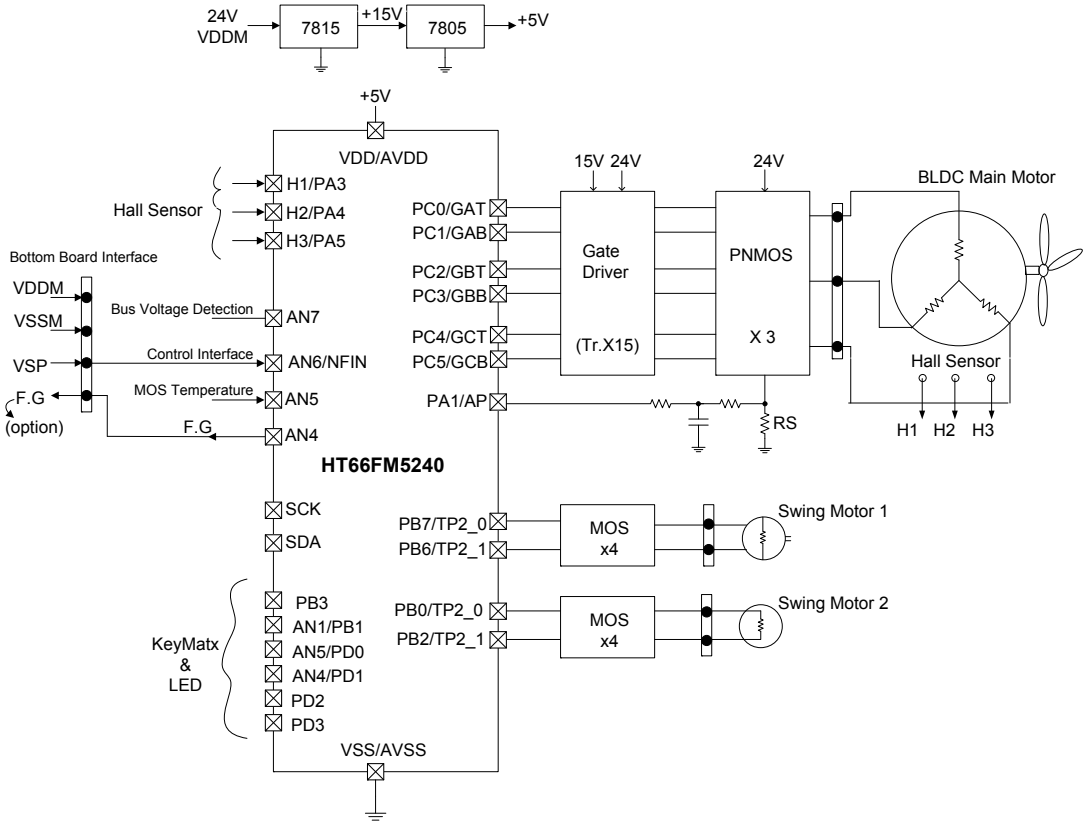
It is recommended that programs do not use the "CALL" instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged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

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

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

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

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

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

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

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

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

Instruction Definition

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

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

Operation: ACC ← ACC + [m] + C

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

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

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

Operation: [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation: ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation: ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation: [m] ← ACC + [m]

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

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

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

Operation: ACC ← ACC "AND" [m]

Affected flag(s): Z

AND A,x Logical AND immediate data to ACC

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

Operation: ACC ← ACC "AND" x

Affected flag(s): Z

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

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

Operation: [m] ← ACC "AND" [m]

Affected flag(s): Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s): C

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

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

SET [m] Set Data Memory

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

Operation: [m] ← FFH

Affected flag(s): None

SET [m].i Set bit of Data Memory

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

Operation: [m].i ← 1

Affected flag(s): None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

Description The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

Description The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

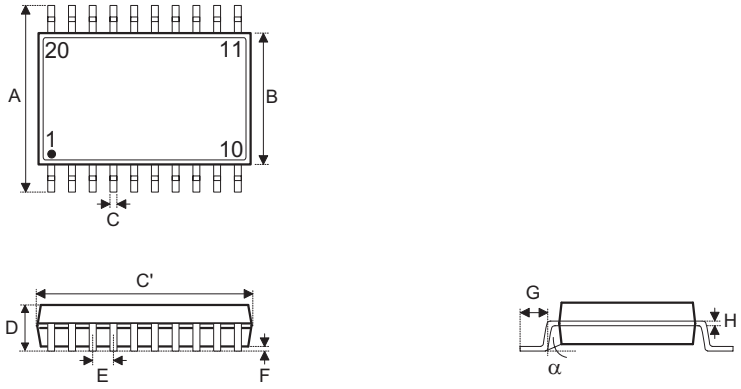
Package Information

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

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

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

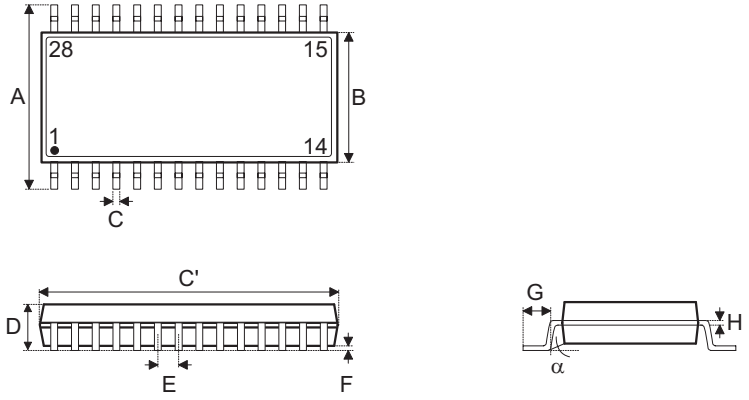
20-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.008	—	0.012
C’		0.341 BSC	
D	—	—	0.069
E		0.025 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		6.00 BSC	
B		3.90 BSC	
C	0.20	—	0.30
C’		8.66 BSC	
D	—	—	1.75
E		0.635 BSC	
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

28-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.008	—	0.012
C’		0.390 BSC	
D	—	—	0.069
E		0.025 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		6.00 BSC	
B		3.90 BSC	
C	0.20	—	0.30
C’		9.90 BSC	
D	—	—	1.75
E		0.635 BSC	
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

Copyright© 2023 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

The information provided in this document has been produced with reasonable care and attention before publication, however, HOLTEK does not guarantee that the information is completely accurate. The information contained in this publication is provided for reference only and may be superseded by updates. HOLTEK disclaims any expressed, implied or statutory warranties, including but not limited to suitability for commercialization, satisfactory quality, specifications, characteristics, functions, fitness for a particular purpose, and non-infringement of any third-party's rights. HOLTEK disclaims all liability arising from the information and its application. In addition, HOLTEK does not recommend the use of HOLTEK's products where there is a risk of personal hazard due to malfunction or other reasons. HOLTEK hereby declares that it does not authorise the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.