



Brushless DC MCU with Pre-driver

HT66FM5340

Revision: V1.10 Date: December 24, 2020

www.holtek.com

Features

CPU Features

- Operating voltage
 - ♦ $f_{SYS}=32kHz\sim 20MHz$: 4.5V~5.5V
- Up to 0.2 μs instruction cycle with 20MHz system clock at $V_{DD}=5V$
- Power down and wake-up functions to reduce power consumption
- Oscillator types
 - ♦ Internal High Speed 20MHz RC – HIRC
 - ♦ Internal Low Speed 32kHz RC – LIRC
- Internal oscillators require no external components
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 4K \times 16
- RAM Data Memory: 256 \times 8
- Watchdog Timer function
- 19 bidirectional I/O lines
- 6 bidirectional High Voltage I/O lines with pre-driver combinational circuit
- Five pin-shared external interrupt inputs – H1, H2, H3, NFIN and INT1
- Multiple Timer Modules for time measurement, input capture, compare match output, PWM output or single pulse output function
- Single 16-bit Capture Timer Module for motor protection
- Single 10-bit PWM generators, supporting edge/center aligned mode
- 6+2 channel 12-bit resolution A/D converter
- Over current detection – Operational Amplifier, Comparator 0 and 8-bit D/A converter
- Internal 5V LDO
- Four comparators
- Time-Base function for generation of fixed time interrupt signals
- Low voltage reset function
- Low voltage detect function
- Package type: 24-pin SSOP

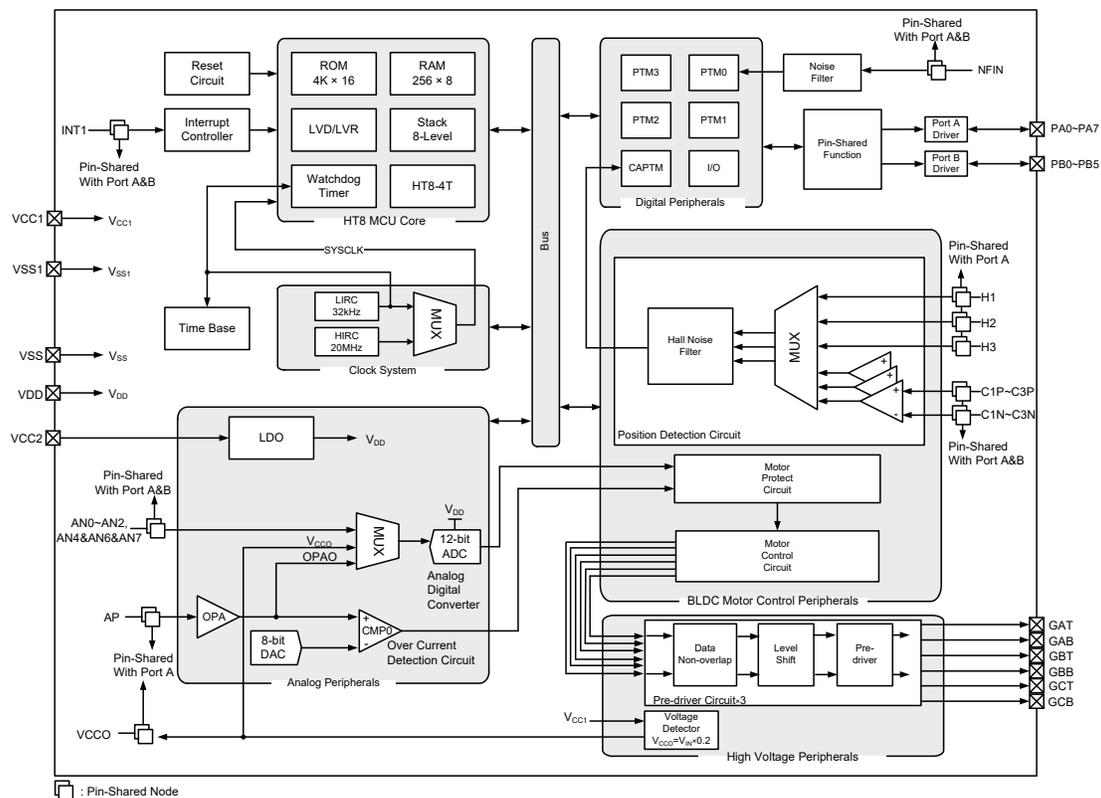
General Description

The HT66FM5340 is a Flash Memory 8-bit high performance RISC architecture microcontroller, which includes a host of fully integrated special features specifically designed for brushless DC motor applications.

The device includes an internal LDO (Low Dropout Regulator) for voltage regulation with a 12V input and 5V output as well as an internal pre-driver which provides a 12V high voltage output, which can directly drive a P-N MOSFET driver or a BLDC motor.

In addition to the advantages of low power consumption and I/O flexibility, the device also includes multiple and extremely flexible Timer Modules, internal oscillators, multi-channel A/D converter, D/A converter, Operational Amplifier, Pulse Width Modulation function, 16-bit Capture Timer Module function, Comparators, Motor Protect Module, Hall sensor position detect function, Time Base function, Low Voltage Reset, Low Voltage Detect, power-down and wake-up functions. Although especially designed for brushless DC motor applications, the enhanced versatility of this device also makes it applicable for use in a wide range of A/D application possibilities such as sensor signal processing, motor driving, industrial controllers, consumer products, subsystem controllers, ect.,.

Block Diagram



Pin Assignment

PA6/[NFIN]/[CTIN]/PTCK1/AN7/VCCO	□	1		24	□	PA5/H3/PTP3I/C3P/PTP2/PTP1
PA7/NFIN/INT1/AN6/HAO/OPAO	□	2		23	□	PA4/H2/PTP2I/C2P/PTP1/PTP2
PB1/CTIN/PTCK2/[INT1]/AN2/HBO/PTP2	□	3		22	□	PA3/H1/PTP1I/C1P
PB2/PTCK0/PTCK3/[NFIN]/AN1/HCO/PTP2	□	4		21	□	PB5/C1N/CPN/PTP0
PA1/PTP0I/[CTIN]/AN4/AP	□	5		20	□	PB4/C2N/PTP0/PTP3
VSS	□	6		19	□	PB3/C3N/AN0/PTP3
VDD	□	7		18	□	PA2/PTP1/PTP3/OCDSCK/ICPCK
VCC2	□	8		17	□	PA0/PTP3/PTP1/OCSDSA/ICPDA
GAT	□	9		16	□	GCB
GAB	□	10		15	□	GCT
GBT	□	11		14	□	VSS1
GBB	□	12		13	□	VCC1

HT66FM5340/HT66VM5340
24 SSOP-A

- Note: 1. Bracketed pin names indicate non-default pinout remapping locations. The detailed information can be referenced to the relevant chapter.
2. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by its corresponding software control bits.
3. The OCSDSA and OCDSCK pins are supplied as dedicated OCDS pins and as such only available for the HT66VM5340 device which is the OCDS EV chip for the HT66FM5340 device.
4. 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 Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/PTP3/PTP1/ OCSDSA/ICPDA	PA0	PAPU PAWU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP3	PAPS0	—	CMOS	PTM3 output
	PTP1	PAPS0	—	CMOS	PTM1 output
	OCSDSA	—	ST	CMOS	OCDS data/address pin, for EV chip only
	ICPDA	—	ST	CMOS	ICP data/address pin
PA1/PTP0I/[CTIN]/ AN4/AP	PA1	PAPU PAWU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP0I	PAPS0	ST	—	PTM0 input
	CTIN	PRM PAPS0	ST	—	CAPTM input
	AN4	PAPS0	AN	—	A/D Converter external input channel 4
	AP	PAPS0	AN	—	Positive input of OPA
PA2/PTP1/PTP3/ OCDSCK/ICPCK	PA2	PAPU PAWU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP1	PAPS0	—	CMOS	PTM1 output
	PTP3	PAPS0	—	CMOS	PTM3 output
	ICPCK	—	ST	—	ICP clock bus
	OCDSCK	—	ST	—	OCDS clock bus, for EV chip only

Pin Name	Function	OPT	I/T	O/T	Description
PA3/H1/PTP1I/C1P	PA3	PAPU PAWU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	H1	PAPS0	ST	—	Hall sensor input
	PTP1I	PAPS0	ST	—	PTM1 input
	C1P	PAPS0	ST	—	Comparator 1 positive input
PA4/H2/PTP2I/C2P/ PTP1/PTP2	PA4	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	H2	PAPS1	ST	—	Hall sensor input
	PTP2I	PAPS1	ST	—	PTM2 input
	C2P	PAPS1	ST	—	Comparator 2 positive input
	PTP1	PAPS1	—	CMOS	PTM1 output
	PTP2	PAPS1	—	CMOS	PTM2 output
PA5/H3/PTP3I/C3P/ PTP2/PTP1	PA5	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	H3	PAPS1	ST	—	Hall sensor input
	PTP3I	PAPS1	ST	—	PTM3 input
	C3P	PAPS1	ST	—	Comparator 3 positive input
	PTP2	PAPS1	—	CMOS	PTM2 output
	PTP1	PAPS1	—	CMOS	PTM1 output
PA6/[NFIN]/[CTIN]/ PTCK1/AN7/VCCO	PA6	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	NFIN	PRM PAPS1	ST	—	Noise Filter input
	CTIN	PRM PAPS1	ST	—	CAPTm input
	PTCK1	PAPS1	ST	—	PTM1 clock input
	AN7	PAPS1	AN	—	A/D Converter external input channel 7
	VCCO	PAPS1	—	CMOS	VCC DET output
PA7/NFIN/INT1/ AN6/HAO/OPAO	PA7	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	NFIN	PRM PAPS1	ST	—	External Noise Filter input
	INT1	PRM INTEG PAPS1	ST	—	External interrupt 1 input
	AN6	PAPS1	AN	—	A/D Converter external input channel 6
	HAO	PAPS1	—	CMOS	Test pin for SA
	OPAO	PAPS1	—	AN	OPA output
PB1/CTIN/PTCK2/ [INT1]/AN2/HBO/PTP2	PB1	PBPU PBPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTIN	PRM PBPS0	ST	—	CAPTm input
	PTCK2	PBPS0	ST	—	PTM2 clock input
	INT1	PRM INTEG	ST	—	External interrupt 1 input
	AN2	PBPS0	AN	—	A/D Converter external input channel 2
	HBO	PBPS0	—	CMOS	Test pin for SB
	PTP2	PBPS0	—	CMOS	PTM2 output

Pin Name	Function	OPT	I/T	O/T	Description
PB2/PTCK0/PTCK3/ [NFIN]/AN1/HCO/PTP2	PB2	PBPU PBPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK0	PBPS0	ST	—	PTM0 clock input
	PTCK3	PBPS0	ST	—	PTM3 clock input
	NFIN	PRM PBPS0	ST	—	External Noise Filter input
	AN1	PBPS0	AN	—	A/D Converter external input channel 1
	HCO	PBPS0	—	CMOS	Test pin for SC
	PTP2	PBPS0	—	CMOS	PTM2 output
PB3/C3N/AN0/PTP3	PB3	PBPU PBPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	C3N	PBPS0	ST	—	Comparator 3 negative input
	AN0	PBPS0	AN	—	A/D Converter external input channel 0
	PTP3	PBPS0	—	CMOS	PTM3 output
PB4/C2N/PTP0/PTP3	PB4	PBPU PBPS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	C2N	PBPS1	ST	—	Comparator 2 negative input
	PTP0	PBPS1	—	CMOS	PTM0 output
	PTP3	PBPS1	—	CMOS	PTM3 output
PB5/C1N/CPN/PTP0	PB5	PBPU PBPS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	C1N	PBPS1	ST	—	Comparator 1 negative input
	CPN	PBPS1	ST	—	Comparator 1, 2, 3 negative input
	PTP0	PBPS1	—	CMOS	PTM0 output
GAT	GAT	GATEC	—	AN	Phase A high side P-MOSFET driver
GAB	GAB	GATEC	—	AN	Phase A low side N-MOSFET driver
GBT	GBT	GATEC	—	AN	Phase B high side P-MOSFET driver
GBB	GBB	GATEC	—	AN	Phase B low side N-MOSFET driver
GCT	GCT	GATEC	—	AN	Phase C high side P-MOSFET driver
GCB	GCB	GATEC	—	AN	Phase C low side N-MOSFET driver
VDD	VDD	—	PWR	—	Digital positive power supply
VCC1	VCC1	—	PWR	—	High voltage driver positive power supply
VCC2	VCC2	—	PWR	—	LDO input
VSS	VSS	—	PWR	—	Digital negative power supply, ground
VSS1	VSS1	—	PWR	—	High voltage driver negative power supply, ground

Legend: I/T: Input type;

O/T: Output type;

OPT: Optional by register option;

PWR: Power;

ST: Schmitt Trigger input;

CMOS: CMOS output;

AN: Analog signal.

Absolute Maximum Ratings

Supply Voltage (V_{CC1}, V_{CC2})	$V_{SS}-0.3V$ to 18V
Supply Voltage (V_{DD})	$V_{SS}-0.3V$ to 6.0V
High Voltage Input Voltage	$V_{SS}-0.3V$ to $V_{CC1}+0.3V$
Input Voltage	$V_{SS}-0.3V$ to $V_{DD}+0.3V$
Storage Temperature.....	-50°C to 125°C
Operating Temperature.....	-40°C to 85°C
I_{OH} Total	-80mA
I_{OL} Total	80mA
Total Power Dissipation	500mW

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

D.C. Characteristics

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

Operating Voltage Characteristics

$T_a=-40^{\circ}C-85^{\circ}C$

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	$f_{SYS}=f_{HIRC}=20MHz$	4.5	—	5.5	V
		$f_{SYS}=f_{HIRC}/64=312.5kHz$	3.15	—	5.5	
	Operating Voltage – LIRC	$f_{SYS}=f_{LIRC}=32kHz$	3.15	—	5.5	V

Standby Current Characteristics

$T_a=-40^{\circ}C-85^{\circ}C$

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB}	SLEEP Mode	5V	LIRC & LVR on, LVD off, WDT on	—	60	100	μA
	IDLE0 Mode – LIRC	5V	f_{SUB} on	—	5	10	μA
	IDLE1 Mode – HIRC	5V	f_{SUB} on, $f_{SYS}=f_{HIRC}=20MHz$	—	400	500	μA

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

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

Operating Current Characteristics

 $T_a = -40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
I _{DD}	Operating Current – HIRC	5V	f _{sys} =f _{HIRC} =20MHz	—	9	12	mA
I _{HIRC}	Additional Current for HIRC Enable	5V	f _{HIRC} =20MHz	—	570	860	μA

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

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

A.C. Characteristics

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

High Speed Internal Oscillator – HIRC – Frequency Accuracy

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Temp.				
f _{HIRC}	20MHz Writer Trimmed HIRC Frequency	5V	Ta=25°C	-1%	20	+1%	MHz
			Ta=-40°C~85°C	-2%	20	+2%	
		4.5V~5.5V	Ta=25°C	-2.5%	20	+2.5%	
			Ta=-40°C~85°C	-3%	20	+3%	

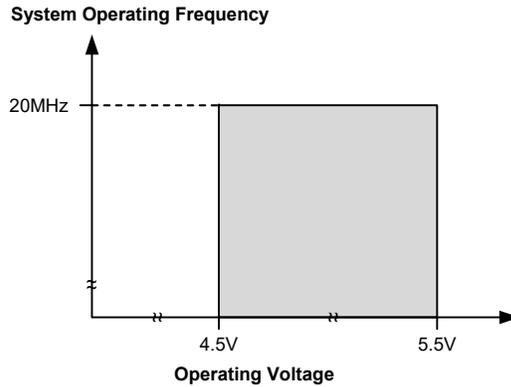
Note: 1. The 5V value for V_{DD} is provided as this is the fixed voltage at which the HIRC frequency is trimmed by the writer.

2. The row below the 5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage.

Low Speed Internal Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Temp.				
f _{LIRC}	LIRC Frequency	5V	Ta=25°C	-10%	32	+10%	kHz
		5V ± 0.5V	Ta=-40°C~85°C	-40%	32	+40%	

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
t _{SST}	System Start-up Time Wake-up from condition where f _{SYS} is off	—	f _{SYS} =f _H ~ f _H /64, f _H =f _{HIRC}	—	16	—	t _{HIRC}
		—	f _{SYS} =f _{SUB} =f _{LIRC}	—	2	—	t _{LIRC}
	System Start-up Time Wake-up from condition where f _{SYS} is on	—	f _{SYS} =f _H ~ f _H /64, f _H =f _{HIRC}	—	2	—	t _H
		—	f _{SYS} =f _{SUB} =f _{LIRC}	—	2	—	t _{SUB}
	System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode	—	f _{HIRC} switches from off → on (HTO=1)	—	16	—	t _{HIRC}
	System start-up Time (WDT time-out hardware cold reset)	—	—	—	0	—	t _H
t _{RSTD}	System Reset Delay Time Reset source from Power-on reset, LVR hardware reset or LVRC/WBTC software reset	—	—	25	50	100	ms
	System Reset Delay Time WDT time-out hardware cold reset	—	—	8.3	16.7	33.3	ms
t _{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

- Note: 1. For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.
2. The time units, shown by the symbols t_{HIRC}, t_{SYS} etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example t_{HIRC}=1/f_{HIRC}, t_{SYS}=1/f_{SYS} etc.
3. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

 $T_a = -40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{IL}	Input Low Voltage for I/O Ports	5V	—	0	—	0.3V _{DD}	V
V _{IH}	Input High Voltage for I/O Ports	5V	—	0.7V _{DD}	—	V _{DD}	V
I _{OL}	Sink Current for I/O Ports	5V	V _{OL} =0.1V _{DD}	32	65	—	mA
I _{OH}	Source Current for I/O Ports	5V	V _{OH} =0.9V _{DD}	-8	-16	—	mA
R _{PH}	Pull-high Resistance for I/O Ports ^(Note)	5V	—	10	30	50	kΩ
t _{TCK}	PTCKn Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t _{TPI}	PTP0I Input Pin Minimum Pulse Width	—	—	0.1	—	—	μs
	PTP1I~PTP3I Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t _{INT}	External Interrupt Minimum Pulse Width	—	—	1	5.0	10	μs
f _{TMCLK}	PTMn Maximum Timer Clock Source Frequency	5V	—	—	—	1	f _{sys}

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

LVR/LVD Electrical Characteristics

 $T_a = -40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

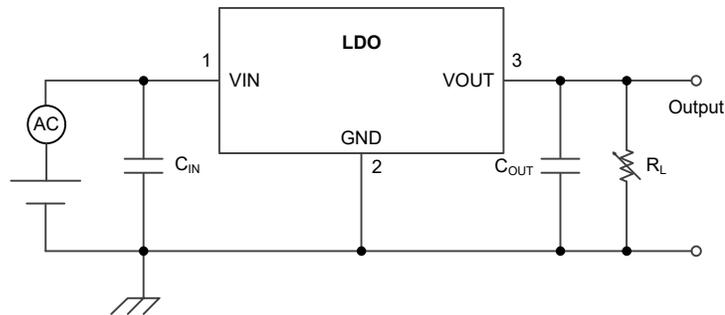
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 3.15V	-5%	3.15	+5%	V
V _{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 3.6V	-5%	3.6	+5%	V
I _{LVD}	Additional Current for LVD Enable	5V	—	—	—	70	μA
t _{LVDS}	LVDO Stable Time	—	For LVR enable, LVD off → on	—	—	15	μs
t _{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t _{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs

Low Dropout Regulator Electrical Characteristics

$V_{IN}=V_{OUT}+1V$, $T_a=-40^{\circ}C\sim 85^{\circ}C$, unless otherwise specified
 $C_{LOAD}=10\mu F+0.1\mu F$

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{IN}	Conditions				
V_{IN}	Input Voltage	—	—	$V_{OUT}+0.1$	6	15	V
V_{OUT}	Output Voltage	—	$T_a=25^{\circ}C$, $I_{LOAD}=1mA$, $V_{OUT}=5.0V$	-2%	5.0	2%	V
		—	$I_{LOAD}=1mA$, $V_{OUT}=5.0V$	-3%	5.0	3%	
ΔV_{LOAD}	Load Regulation ⁽¹⁾	—	$1mA \leq I_{LOAD} \leq 70mA$,	—	0.02	0.05	%/mA
V_{DROP}	Dropout Voltage ⁽²⁾	—	$V_{OUT}=5V$, $\Delta V_{OUT}=-2\%$ $I_{LOAD}=50mA$	—	150	300	mV
		—	$3V < V_{IN} < 5.1V$, $I_{LOAD}=50mA$	—	300	600	
I_{OUT}	Output Current	—	$\Delta V_{OUT}=-3\%$	70	—	—	mA
		—	$V_{IN}=V_{OUT}+2V$, $\Delta V_{OUT}=-5\%$	100	—	—	
I_Q	Quiescent Current	12V	No load	—	3	8	μA
ΔV_{LINE}	Line Regulation	—	$V_{OUT}+1V \leq V_{IN} \leq 12V$, $I_{LOAD}=1mA$	—	—	0.5	%/V
TC	Temperature Coefficient	—	$I_{LOAD}=10mA$	—	± 1.5	± 2	mV/ $^{\circ}C$
ΔV_{OUT_RIPPLE}	Output Voltage Ripple	6V	$I_{LOAD}=10mA$	—	—	10	mV
RR	Ripple Rejection ⁽³⁾	—	$V_{IN}=10V_{DC}+2V_{P-P(AC)}$, $I_{LOAD} \leq 50mA$, $f=120Hz$	35	—	—	dB
I_{LIMIT}	Current Limit	6V	$\Delta V_{OUT}=-10\%$	150	—	—	mA
t_{START}	LDO Startup Time	6V	$I_{LOAD}=1mA$, $V_{OUT}=5V \pm 5\%$	—	—	10	ms

- Note: 1. Load regulation is measured at a constant junction temperature, using pulse testing with a low ON time and is guaranteed up to the maximum power dissipation. Power dissipation is determined by the input/output differential voltage and the output current. Guaranteed maximum power dissipation will not be available over the full input/output range. The maximum allowable power dissipation at any ambient temperature is $P_D=(T_{J(MAX)}-T_a)/\theta_{JA}$.
2. Dropout voltage is defined as the input voltage minus the output voltage that produces a 2% change in the output voltage from the value at appointed V_{IN} .
3. Ripple rejection ratio measurement circuit. $RR=20 \times \log(\Delta V_{IN}/\Delta V_{OUT})$. $C_{IN}=10\mu F$, $C_{OUT}=10\mu F+0.1\mu F$.



4. Application information for LDO load capacitor selection for stability.

A/D Converter Electrical Characteristics

 $T_a = -40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{DD}	A/D Converter Operating Voltage	—	—	V _{LVR}	5.0	5.5	V
I _{OP}	Additional Current for A/D Converter Enable	5V	—	—	0.6	—	mA
I _{STB}	A/D Converter Standby Current	—	Digital input no change	—	—	4	μA
V _{REF}	A/D Converter Reference Voltage	—	—	2	V _{DD}	V _{DD} +0.1	V
t _{CONV}	A/D Conversion Time (Including Sample and Hold time)	—	—	—	16	—	t _{ADCK}
DNL	A/D Differential Non-linearity	4.5V	V _{REF} =V _{DD} , no load, t _{ADCK} =0.8μs	-2	—	+3	LSB
		5.5V					
		4.5V	V _{REF} =V _{DD} , no load, t _{ADCK} =12.8μs				
		5.5V					
INL	A/D Integral Non-linearity	4.5V	V _{REF} =V _{DD} , no load, t _{ADCK} =0.8μs	-4	—	+4	LSB
		5.5V					
		4.5V	V _{REF} =V _{DD} , no load, t _{ADCK} =12.8μs				
		5.5V					
t _{ADCK}	A/D Converter Clock Period	—	—	0.8	—	12.8	μs
t _{CKH}	A/D Converter Clock High Pulse	—	—	225	—	—	ns
t _{CKL}	A/D Converter Clock Low Pulse	—	—	225	—	—	ns
t _{ST}	A/D Converter Turn-on Setup Time	—	—	2	—	—	ns
	A/D Converter Start Bit Setup Time	—	—	2	—	—	
t _{STH}	A/D Converter Start Bit High Pulse	—	—	25	—	—	ns
t _{DEOC}	End of Conversion Output Delay	—	V _{DD} =5V	—	3	—	ns
t _{DOUT}	Conversion Data Output Delay	—	V _{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 Electrical Characteristics

 $T_a = -40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{DD}	D/A Converter Operating Voltage	—	—	V _{LVR}	—	5.5	V
V _{DA}	D/A Converter Output Voltage	—	OPCM[7:0]=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 Electrical Characteristics

 $T_a = -40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{OPR}	OPA Operating Voltage	5V	—	V _{LVR}	5.0	5.5	V
I _{OFF}	Power Down Current	5V	—	—	—	0.1	μA
V _{OPOS}	Input Offset Voltage	5V	Without calibration, AOF[4:0]=10000B	-15	—	+15	mV
		5V	With calibration	-2	—	+2	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{RI}	PGA Input Voltage Range	3.3V	Gain=1, Ga<±5%	V _{SS} +0.1	—	V _{DD} -1.4	V
		5V		V _{SS} +0.1	—	V _{DD} -1.4	
		3.3V	Gain=5, 10, 20 Ga<±5%	V _{SS} +0.1	—	V _{OR} (Max)/Gain	
		5V		V _{SS} +0.1	—	V _{OR} (Max)/Gain	
V _{OR}	OPA Maximum Output Voltage Range	3.3V	—	V _{SS} +0.1	—	V _{DD} -0.1	V
		5V	—	V _{SS} +0.1	—	V _{DD} -0.1	
PSRR	Power Supply Rejection Ratio	5V	—	90	—	96	dB
CMRR	Common Mode Rejection Ratio	5V	V _{CM} =0~(V _{DD} -1.4V)	—	106	—	dB
SR	Slew Rate+, Slew Rate-	5V	No load	1.29	2.18	—	V/μs
GBW	Gain Bandwidth	5V	R _L =600Ω, C _L =100pF	2.05	3.70	7.16	MHz
A _{OL}	Open Loop Gain	5V	R _L =600Ω, C _L =100pF	—	96	—	dB
PM	Phase Margin	5V	R _L =600Ω, C _L =100pF	—	90	—	Deg

Comparators Electrical Characteristics

T_a=-40°C~85°C

Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{CMP}	Comparator Operating Voltage	5V	—	V _{LVR}	5.0	5.5	V
I _{CMP}	Comparator Operating Current	5V	—	—	300	450	μA
I _{OFF}	Comparator Power Down Current	5V	Comparator disabled	—	—	0.1	μA
V _{CMPOS}	Comparator Input Offset Voltage	5V	—	-10	—	+10	mV
V _{HYS}	Hysteresis	5V	Comparator 0	50	100	150	mV
		5V	Comparator 1, 2, 3	20	40	60	
V _{CM}	Input Common Mode Voltage Range	—	—	V _{SS}	—	V _{DD} -1.4	V
A _{OL}	Comparator Open Loop Gain	—	—	100	120	—	dB
t _{PD}	Comparator Response Time	5V	V _M =0~(V _{DD} -1.4)V With 10mV overdrive	—	—	1	μs
			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).

High Voltage I/O Electrical Characteristics

T_a=-40°C to 85°C

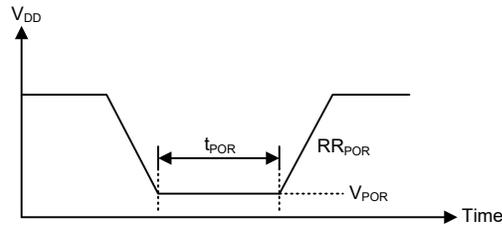
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{IN}	Input Voltage	—	—	6	—	15	V
I _{OH}	Source Current for HVO Pins	—	V _{OH} =0.9×V _{IN} , V _{IN} =12V	-90	-160	—	mA
		—	V _{OH} =0.9×V _{IN} , V _{IN} =6V	-40	-80	—	
I _{OL}	Sink Current for HVO Pins	—	V _{OL} =0.1×V _{IN} , V _{IN} =12V	90	160	—	mA
		—	V _{OL} =0.1×V _{IN} , V _{IN} =6V	40	80	—	
V _{CCO}	VCC DET Output	—	V _{IN} =12V	-5	—	+5	%

Note: V_{CCO}=0.2×V_{IN}.

Power-on Reset Characteristics

$T_a = -40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

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



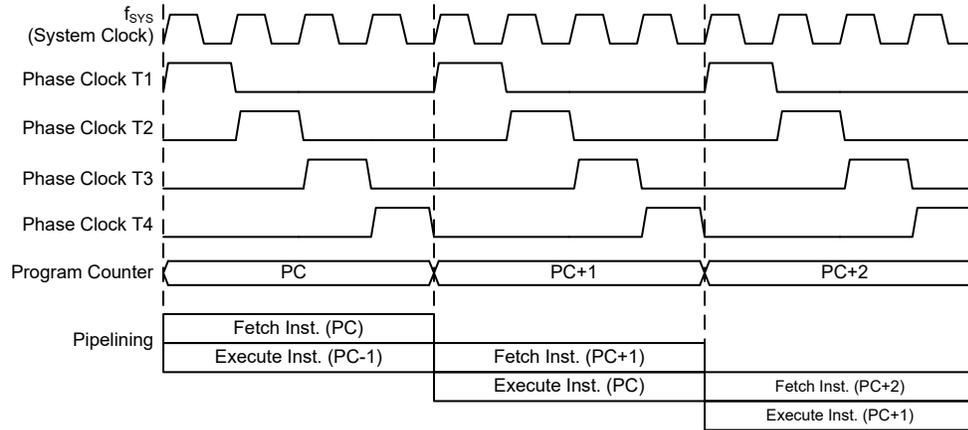
System Architecture

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

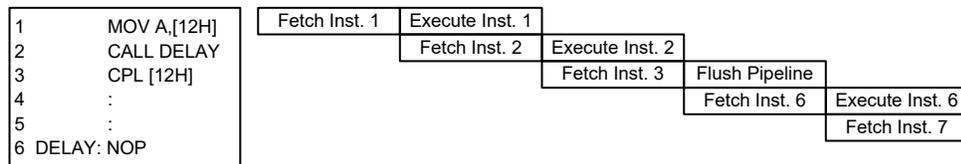
Clocking and Pipelining

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

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



System Clock and Pipelining



Instruction Fetching

Program Counter – PC

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

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

Program Counter	
High Byte	Low Byte (PCL)
PC11~PC8	PCL7~PCL0

Program Counter

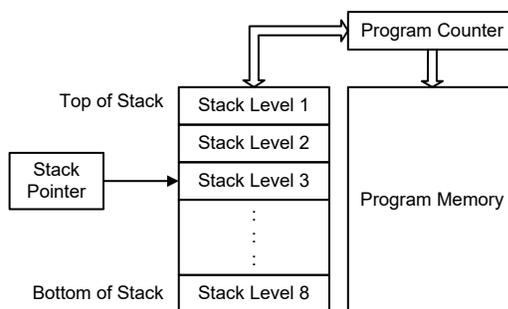
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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. For this device the stack has 8 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



Arithmetic and Logic Unit – ALU

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

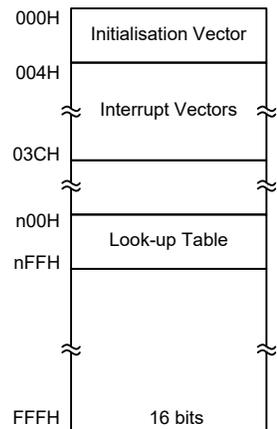
- Arithmetic operations:
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
LRR, LRRCA, LRRCA, LRRCA, LRRCA, LRRCA, LRRCA, LRRCA
- Increment and Decrement:
INCA, INC, DECA, DEC,
LINCA, LINC, LDECA, LDEC
- Branch decision:
JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI,
LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

Flash Program Memory

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

Structure

The Program Memory has a capacity of 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 pair of data table pointer registers.



Program Memory Structure

Special Vectors

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

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be 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 corresponding table read instruction such as “TABRD [m]” or “TABRDL [m]” respectively when the data table is located in sector 0. If the data table is located in other sectors, the data can be retrieved from the program memory using the corresponding extended table read instruction such as “LTABRD [m]” or “LTABRDL [m]” 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.

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

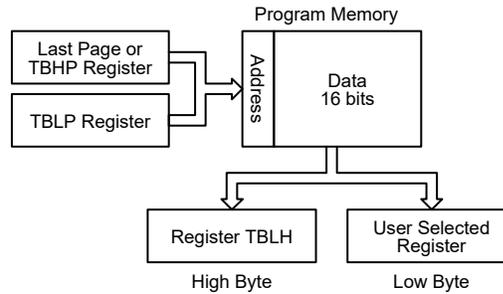


Table Program Example

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 low byte register is setup here to have an initial value of “06H”. This will ensure that the first data read from the data table will be at the Program Memory address “0F06H” or 6 locations after the start of the last page if using the the “TABRDL [m]” or “LTABRDL [m]” instruction. Note that the value for the table pointer is referenced to the specified address pointed by the TBLP and TBHP registers if the “TABRD [m]” or “LTABRD [m]” instruction is being used. In this case the high byte of the table data which is equal to zero will be transferred to the TBLH register automatically when the “TABRD [m]” or “LTABRD [m]” instruction is executed.

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

Table Read Program Example

```

tempreg1 db?      ; temporary register #1
tempreg2 db?      ; temporary register #2
:
:
mov a,06h         ; initialise low table pointer - note that this address is referenced
mov tblp,a        ; to the last page or the page that tbhp pointed
mov a,0fh         ; initialise high table pointer
mov tbhp,a        ; it is not necessary to set tbhp if executing tabrdl or ltabrdl
:
:
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 tempreg2
; the value "00H" will be transferred to the high byte register TBLH
:
:
org 0F00h ; set initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh

```

In Circuit Programming – ICP

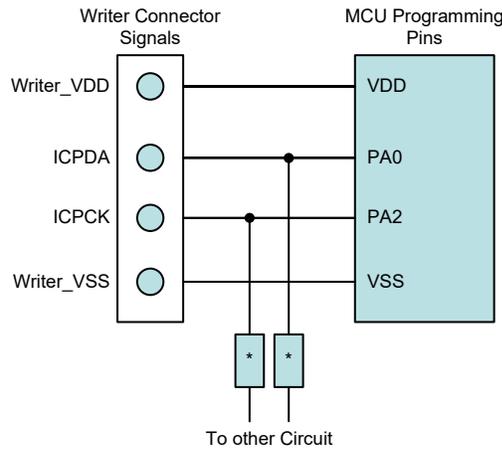
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VDD	VDD	Power Supply (5V)
VSS	VSS	Ground

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

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



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

On-Chip Debug Support – OCDS

An EV chip exists for the purposes of device emulation. This EV chip device also provides an “On-Chip Debug” function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for “On-Chip Debug” function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCSDA 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 OCSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named “Holtek e-Link for 8-bit MCU OCDS User’s Guide”.

Holtek e-Link Pins	EV Chip Pins	Pin Description
OCSDA	OCSDA	On-Chip Debug Support Data/Address input/output
OCDSCK	OCDSCK	On-Chip Debug Support Clock input
VDD	VDD	Power Supply (5V)
VSS	VSS	Ground

Data Memory

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

Categorized into two types, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.

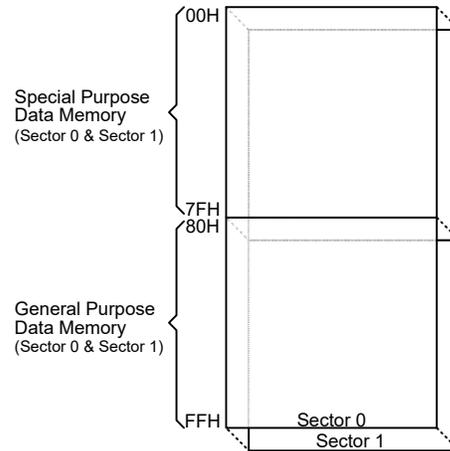
Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value.

Structure

The Data Memory is subdivided into two sectors, all of which are implemented in 8-bit wide RAM. In Special Purpose Data Memory most of the registers are accessible in all sectors while several registers are only accessible in Sector 0 or Sector 1. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH.

Special Purpose Data Memory	General Purpose Data Memory	
Located Sectors	Capacity	Sector: Address
0, 1	256×8	0: 80H~FFH 1: 80H~FFH

Data Memory Summary



Data Memory Structure

Data Memory Addressing

For the device that supports the extended instructions, the desired Data Memory Sector is selected by the MP1H or MP2H register and the certain Data Memory address in the pointed sector is specified by the MP1L or MP2L register when using indirect addressing access.

Direct Addressing can be used in all sectors using the corresponding instructions which can address all available data memory space. For the accessed data memory which is located in any data memory sector except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address “m” in the extended instructions has 9 valid bits for this device, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user programing for both reading and writing operations. By using the bit operation instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and writeable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section. Note that for locations that are unused, any read instruction to these addresses will return the value “00H”.

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH		
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH		
10H	SMOD	
11H	LVDC	
12H	LVRC	
13H		
14H	PAWU	
15H	PAPU	
16H	PA	
17H	PAC	
18H	WDTC	
19H	TBC	
1AH	INTEG1	
1BH	INTC0	
1CH	INTC1	
1DH	INTC2	
1EH	INTC3	
1FH	PWMCS	
20H	PBPU	
21H	PB	
22H	PBC	
23H		
24H	HVRC	
25H	GATEC	
26H	PTM0C2	
27H	PTM0BL	
28H	PTM0BH	
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		
3EH	INTEG0	
3FH	CTRL	

□ : Unused, read as 00H

	Sector 0	Sector 1
40H		
41H		
42H		OCPS
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	MF10	OPACAL
71H	MF11	PTM3C0
72H	MF12	PTM3C1
73H	MF13	PTM3DL
74H	MF14	PTM3DH
75H	MF15	PTM3AL
76H	MF16	PTM3AH
77H		PTM3RPL
78H		PTM3RPH
79H		
7AH		PAPS0
7BH	CMPC	PAPS1
7CH	HCHK_NUM	
7DH	HNF_MSEL	
7EH	NF_VIH	
7FH	NF_VIL	PRM

Special Purpose Data Memory

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, IAR2

The Indirect Addressing Registers, IAR0, IAR1 and IAR2, 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, IAR1 and IAR2 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, MP1L/MP1H or MP2L/MP2H. Acting as a pair, IAR0 and MP0 can together access data only from Sector 0 while the IAR1 register together with the MP1L/MP1H register pair and IAR2 register together with the MP2L/MP2H register pair can access data from any Data Memory Sector. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers will return a result of “00H” and writing to the registers will result in no operation.

Memory Pointers – MP0, MP1L, MP1H, MP2L, MP2H

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L, MP2H, 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 Sector 0, while MP1L/MP1H together with IAR1 and MP2L/MP2H together with IAR2 are used to access data from all sectors according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all sectors using the corresponding extended instructions which can address all available data memory space.

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 1

```
data .section 'data'
adres1  db ?
adres2  db ?
adres3  db ?
adres4  db ?
block   db ?
code .section at 0 'code'
org 00h
start:
    mov a, 04h                ; setup size of block
    mov block, a
    mov a, offset adres1     ; Accumulator loaded with first RAM address
    mov mp0, a               ; setup memory pointer with first RAM address
loop:
    clr IAR0                 ; clear the data at address defined by MP0
    inc mp0                  ; increment memory pointer
    sdz block                 ; check if last memory location has been cleared
    jmp loop
continue:
```

Indirect Addressing Program Example 2

```

data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block db ?
code .section at 0 'code'
org 00h
start:
    mov a, 04h                ; setup size of block
    mov block, a
    mov a, 01h                ; setup the memory sector
    mov mplh, a
    mov a, offset adres1      ; Accumulator loaded with first RAM address
    mov mpll, a               ; setup memory pointer with first RAM address
loop:
    clr IAR1                  ; clear the data at address defined by MP1L
    inc mpll                   ; increment memory pointer MP1L
    sdz block                  ; check if last memory location has been cleared
    jmp loop
continue:

```

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

Direct Addressing Program Example using extended instructions

```

data .section 'data'
temp db ?
code .section at 0 'code'
org 00h
start:
    lmov a, [m]                ; move [m] data to acc
    lsub a, [m+1]              ; compare [m] and [m+1] data
    snz c                       ; [m]>[m+1]?
    jmp continue               ; no
    lmov a, [m]                ; yes, exchange [m] and [m+1] data
    mov temp, a
    lmov a, [m+1]
    lmov [m], a
    mov a, temp
    lmov [m+1], a
continue:

```

Note: Here “m” is a data memory address located in any data memory sectors. For example, m=1F0H, it indicates address 0F0H in Sector 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 SC flag, CZ flag, 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, C, SC and CZ 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.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.
- SC is the result of the “XOR” operation which is performed by the OV flag and the MSB of the current instruction operation result.

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	SC	CZ	TO	PDF	OV	Z	AC	C
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
POR	x	x	0	0	x	x	x	x

“x”: unknown

- Bit 7 **SC**: The result of the “XOR” operation which is performed by the OV flag and the MSB of the instruction operation result.
- Bit 6 **CZ**: The operational result of different flags for different instructions.
For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.
For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the “AND” operation result which is performed by the previous operation CZ flag and current operation Z flag.
For other instructions, the CZ flag will not be affected.
- Bit 5 **TO**: Watchdog Time-out flag
0: After power up or executing the “CLR WDT” or “HALT” instruction
1: A watchdog time-out occurred
- Bit 4 **PDF**: Power down flag
0: After power up or executing the “CLR WDT” instruction
1: By executing the “HALT” instruction
- Bit 3 **OV**: Overflow flag
0: No overflow
1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa
- Bit 2 **Z**: Zero flag
0: The result of an arithmetic or logical operation is not zero
1: The result of an arithmetic or logical operation is zero
- Bit 1 **AC**: Auxiliary flag
0: No auxiliary carry
1: An operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction
- Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation
The “C” flag is also affected by a rotate through carry instruction.

Oscillators

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

Oscillator Overview

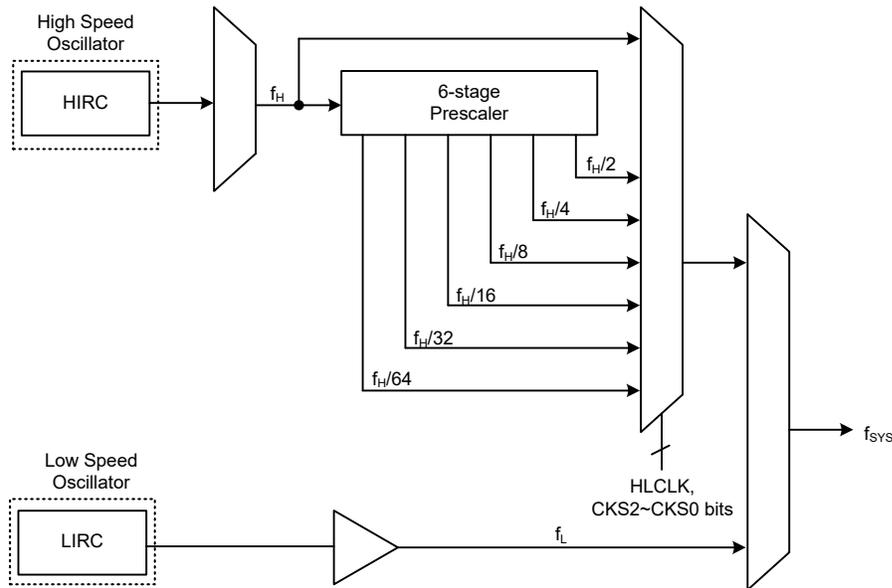
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The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

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

Oscillator Types

System Clock Configurations

There are two oscillator sources, one high speed oscillator and one low speed oscillator. The high speed system clock is sourced from the internal 20MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system clock is implemented using the HLCLK bit and CKS2~CKS0 bits in the SMOD register and as the system clock can be dynamically selected. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators.



Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 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.

Internal 32kHz Oscillator – LIRC

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

The low speed oscillator, in addition to providing a system clock source, is 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

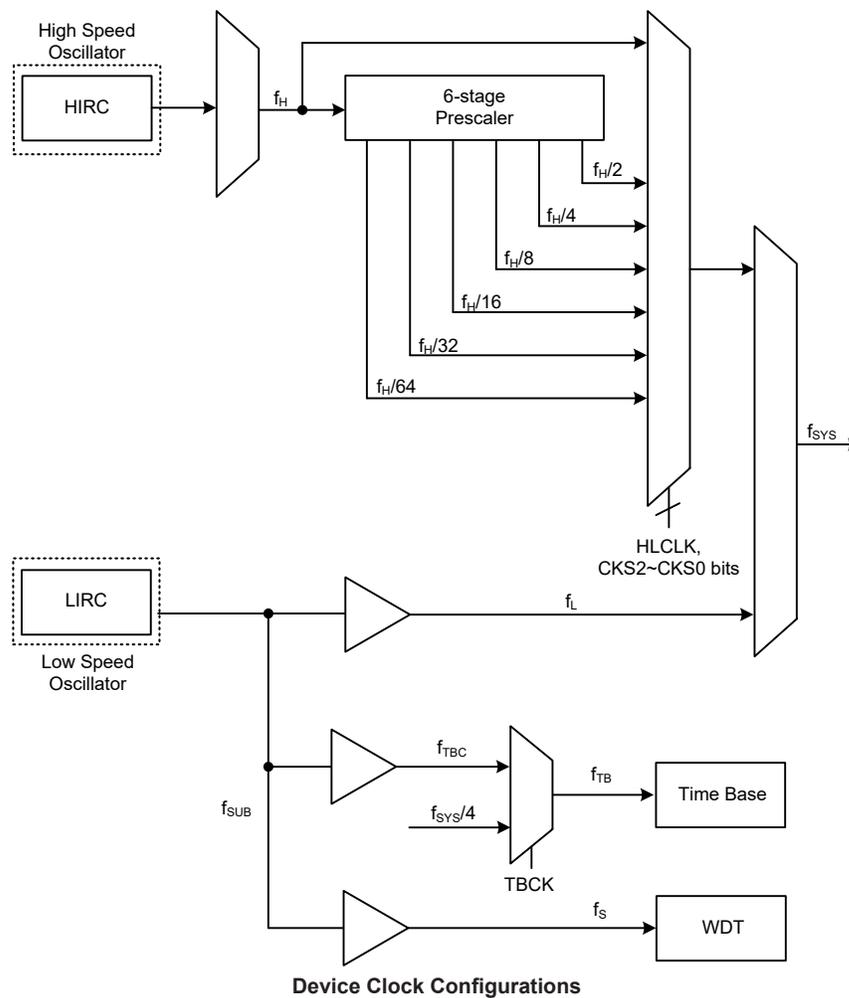
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their 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 register programming, a clock system can be configured to obtain maximum application performance.

The main system clock, can come from 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 is sourced from the HIRC oscillator. The low speed system clock source is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2 \sim f_H/64$.

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 from the LIRC oscillator.



Note: When the system clock source f_{SYS} is switched to f_L from f_H , the high speed oscillator will be turned off to conserve the power. The internal clocks, $f_H/2 \sim f_H/64$, for peripheral circuits will also stop running.

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 FAST 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}
FAST Mode	On	f _H ~f _H /64	On	On	On
SLOW Mode	On	f _L	On	On	On
IDLE1 Mode	Off	On	On	On	On
IDLE0 Mode	Off	Off	On	On	On
SLEEP Mode	Off	Off	On	On	Off

FAST Mode

This is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source coming from the HIRC oscillator. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the HLCLK bit and CKS2~CKS0 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_s clock will continue to operate.

IDLE0 Mode

The IDLE0 Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL register is 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 Time Base. In the IDLE0 Mode, the system oscillator will be stopped, the f_{sub}, f_s and f_{TBC} clocks will be on.

IDLE1 Mode

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

Control Registers

The SMOD register and the FSYSON bit in the CTRL register are 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**: System clock selection when HLCLK is “0”

000: f_L (LIRC)
 001: f_L (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 directly derived from f_L or f_H , a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as “0”

Bit 3 **LTO**: Low speed oscillator ready flag

0: Not ready
 1: Ready

This is the low speed system oscillator ready flag which indicates when the low speed oscillator is stable after power on reset or a wake-up has occurred.

Bit 2 **HTO**: High speed oscillator ready flag

0: Not ready
 1: Ready

This is the high speed oscillator ready flag which indicates when the high speed system oscillator is stable. This flag is cleared to “0” by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable. Therefore this flag will always be read as “1” by the application program after device power-on.

Bit 1 **IDLEN**: IDLE Mode control

0: Disable
 1: Enable

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

Bit 0 **HLCLK**: System clock selection

0: $f_H/2 \sim f_H/64$ or f_L
 1: f_H

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

• **CTRL Register**

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

“x”: unknown

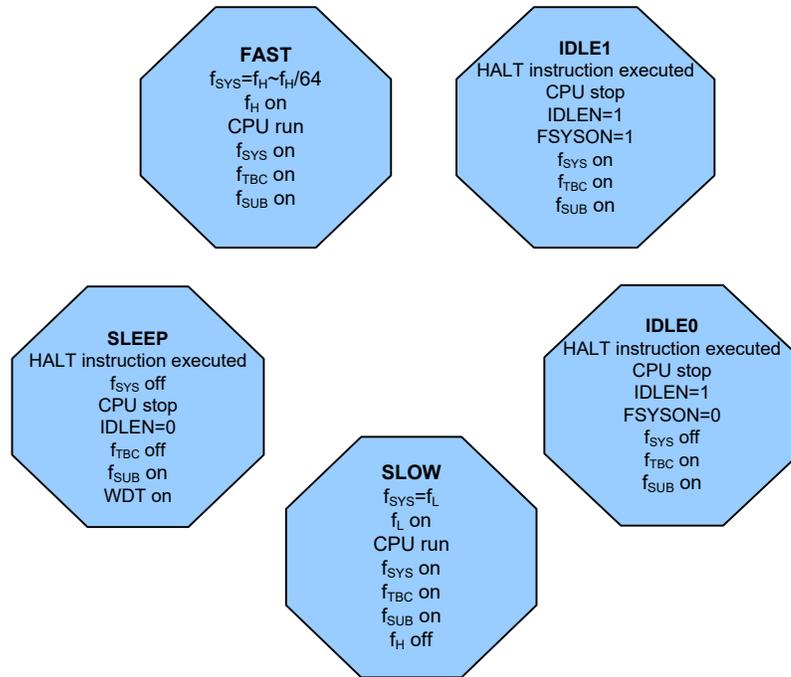
- Bit 7 **FSYSON**: f_{SYS} on/off control in IDLE Mode
 0: Disable
 1: Enable
 This bit is used to control whether the system clock is switched on or not in the IDLE Mode. If this bit is set to “0”, the system clock will be switched off in the IDLE Mode. However, the system clock will be switched on in the IDLE Mode when the FSYSON bit is set to “1”.
- Bit 6~3 Unimplemented, read as “0”
- Bit 2 **LVRF**: LVR function reset flag
 Described elsewhere
- Bit 1 **LRF**: LVR control register software reset flag
 Described elsewhere
- Bit 0 **WRF**: WDT Control register software reset flag
 Described elsewhere

Operating Mode Switching

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

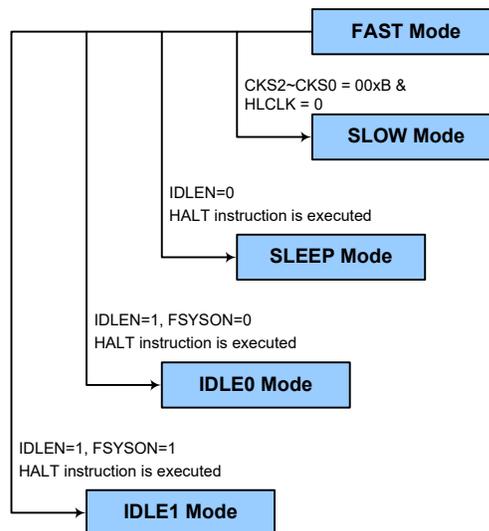
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the condition of the IDLEN bit in the SMOD register and the FSYSON bit 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, $f_H/2 \sim f_H/64$ 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.



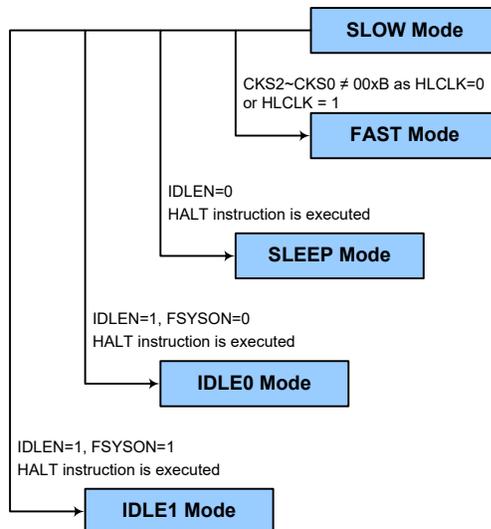
FAST Mode to SLOW Mode Switching

When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to “0” and setting 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 FAST Mode Switching

In SLOW mode the system clock uses LIRC low speed system oscillator. To switch back to the FAST 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 stabilize, the status of the HTO bit is checked.



Entering the SLEEP Mode

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

- The system clock and the Time Base clock will be stopped and the application program will stop at the “HALT” instruction, but the WDT will remain with the clock source coming from the f_s clock.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting since the WDT function is always enabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the “HALT” instruction in the application program with the IDLEN bit in the SMOD register equal to “1” and the FSYSN bit in the 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 clock f_{SUB} and the Watchdog Timer clock f_s will be on.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting since the WDT function is always enabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE1 Mode

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

- The system clock, Time Base clock f_{TBC} , Watchdog Timer clock f_s and the low frequency clock f_{SUB} will be on but the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting since the WDT function is always enabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

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 the device which has different package types, as there may be unbonded pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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

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

Wake-up

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

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

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

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 to zero by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the “HALT” instruction. The TO flag is set if a WDT time-out occurs and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_s , which is supplied by the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with V_{DD} , temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

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

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/reset MCU operation. The WDTC register is initiated to 01010011B at any reset and keeps unchanged at the WDT time-out occurrence a power down state. The WRF software reset flag will be indicated in the CTRL register.

• 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 software control

01010 or 10101: Enable

Other values: Reset MCU

When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time, t_{SRESET} and the WRF bit in the CTRL register will be set high.

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	—	—	—	—	x	0	0

“x”: unknown

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

Bit 6~3 Unimplemented, read as “0”

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

Bit 1 **LRF**: LVR control register software reset flag
 Described elsewhere

Bit 0 **WRF**: WDT control register software reset flag
 0: Not occurred
 1: Occurred

This bit is set to 1 by the WDT Control register software reset and cleared to zero by the application program. Note that this bit can only be cleared to 0 by the application program.

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, this clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to offer the enable control and reset control of the Watchdog Timer. The WDT function will be enabled if the WE4~WE0 bits are equal to 01010B or 10101B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after a delay time, t_{SRESET} .

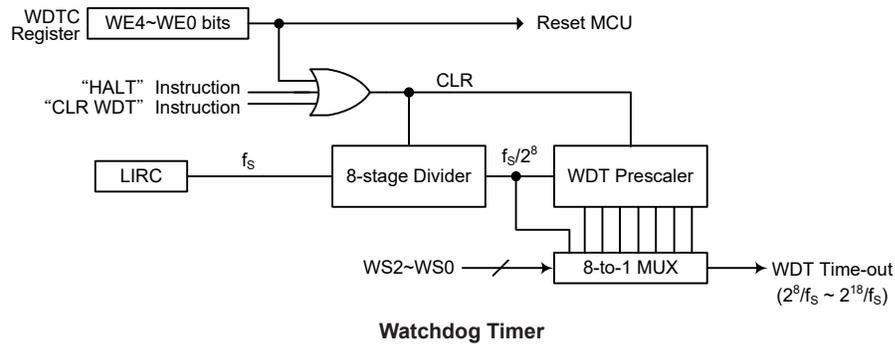
WE4~WE0 Bits	WDT Function
01010B or 10101B	Enable
Any other values	Reset MCU

Watchdog Timer Enable/Reset 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, in which mode the PDF bit is high, 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 software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bits, the second is using the Watchdog Timer software clear instruction, the third is via a HALT instruction.

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

The maximum time-out period is when the 2^{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 8s for the 2^{18} division ratio, and a minimum timeout of 8ms for the 2^8 division ratio.



Reset and Initialisation

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

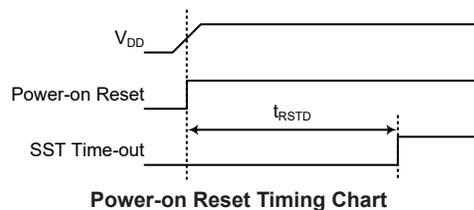
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

Reset Functions

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

Power-on Reset

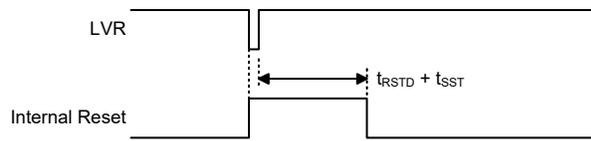
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions.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



Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level.

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 $0.9V \sim V_{LVR}$ such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the CTRL register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between $0.9V \sim V_{LVR}$ must exist for a time greater than the specified value by t_{LVR} in the LVR/LVD Electrical Characteristics table. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value is fixed at 3.15V by the LVS bit field in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by environmental noise, the LVR will reset the device after a delay time, t_{SRESET} . 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.



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								
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 – register is reset to POR value

When an actual low voltage condition occurs, as specified by the defined LVR voltage value above, and the low voltage condition keeps more than a t_{LVR} time, an MCU reset will be generated. In this situation the register contents will remain the same after such a reset occurs. The LVRF bit in the CTRL register will be set high after the low voltage reset.

Any register value, other than the four values above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time, t_{SRESET} . However in this situation the register contents will be reset to the POR value. The LRF bit in the CTRL register will be set high after this reset.

• CTRL Register

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

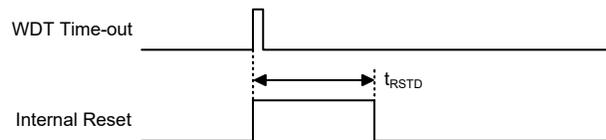
“x”: unknown

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

Watchdog Time-out Reset during Normal Operation

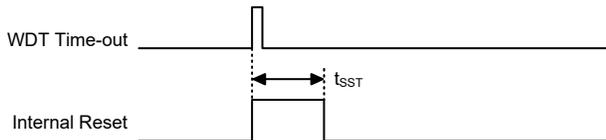
The Watchdog time-out flag TO will be set to “1” when Watchdog time-out Reset during normal operation.



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

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

“u” stands for unchanged

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

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

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

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	uuuu uuuu
ACC	x xxx x xxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	x xxx x xxx	uuuu uuuu	uuuu uuuu
TBLH	x xxx x xxx	uuuu uuuu	uuuu uuuu
TBHP	- - - - x xxx	- - - - uuuu	- - - - uuuu
STATUS	x x 0 0 x xxx	x x 1 u uuuu	u u 1 1 uuuu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
SMOD	0 0 0 - 0 0 1 1	0 0 0 - 0 0 1 1	u u u - u u u u
LVDC	- - 0 0 - 0 0 0	- - 0 0 - 0 0 0	- - u u - u u u
LVRC	0 1 0 1 0 1 0 1	0 1 0 1 0 1 0 1	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PA	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	uuuu uuuu
PAC	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	uuuu uuuu
WDTC	0 1 0 1 0 0 1 1	0 1 0 1 0 0 1 1	uuuu uuuu
TBC	0 0 1 1 - - - -	0 0 1 1 - - - -	uuuu - - - -
INTEG1	- - - - - 0 0	- - - - - 0 0	- - - - - u u
INTC0	- 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0	- u u u u u u u
INTC1	- 0 0 0 - 0 0 0	- 0 0 0 - 0 0 0	- u u u - u u u
INTC2	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	uuuu uuuu
INTC3	0 0 - - 0 0 - -	0 0 - - 0 0 - -	u u - - u u - -
PWMCS	- - - - 0 0 0 0	- - - - 0 0 0 0	- - - - u u u u
PBPU	- - 0 0 0 0 -	- - 0 0 0 0 -	- - u u u u -
PB	- - 1 1 1 1 -	- - 1 1 1 1 -	- - u u u u -
PBC	- - 1 1 1 1 -	- - 1 1 1 1 -	- - u u u u -

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
HVRC	---- --00	---- --00	---- --uu
GATEC	--00 0000	--00 0000	--uu uuuu
PTM0C2	---- -000	---- -000	---- -uuu
PTM0BL	0000 0000	0000 0000	uuuu uuuu
PTM0BH	0000 0000	0000 0000	uuuu uuuu
CAPTC0	0000 0-00	0000 0-00	uuuu u-uu
CAPTC1	0000 0000	0000 0000	uuuu uuuu
CAPTMDL	0000 0000	0000 0000	uuuu uuuu
CAPTMDH	0000 0000	0000 0000	uuuu uuuu
CAPTMAL	0000 0000	0000 0000	uuuu uuuu
CAPTMAH	0000 0000	0000 0000	uuuu uuuu
CAPTMCL	xxxx xxxx	xxxx xxxx	uuuu uuuu
CAPTMCH	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	xxxx ----	xxxx ----	uuuu ---- (ADRF=0)
			uuuu uuuu (ADRF=1)
ADRH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRF=0)
			---- uuuu (ADRF=1)
ADCR0	0110 0000	0110 0000	uuuu uuuu
ADCR1	0000 0000	0000 0000	uuuu uuuu
ADCR2	---- --00	---- --00	---- --uu
ADDL	0000 0000	0000 0000	uuuu uuuu
ADLVDL	0000 ----	0000 ----	uuuu ---- (ADRF=0)
			uuuu uuuu (ADRF=1)
ADLVDH	0000 0000	0000 0000	uuuu uuuu (ADRF=0)
			---- uuuu (ADRF=1)
ADHVDL	0000 ----	0000 ----	uuuu ---- (ADRF=0)
			uuuu uuuu (ADRF=1)
ADHVDH	0000 0000	0000 0000	uuuu uuuu (ADRF=0)
			---- uuuu (ADRF=1)
PBPS0	0000 00--	0000 00--	uuuu uu--
PBPS1	---- 0000	---- 0000	---- uuuu
INTEG0	-000 0000	-000 0000	-uuu uuuu
CTRL	0--- -x00	0--- -x00	u--- -xuu
OCPS	--00 0000	--00 0000	--uu uuuu
HDCR	0001 0000	0001 0000	uuuu uuuu
HDCD	---- -000	---- -000	---- -uuu
MPTC1	0000 0000	0000 0000	uuuu uuuu
MPTC2	---0 0000	---0 0000	---u uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PTM1C0	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	uuuu uuuu
PTM1DL	0000 0000	0000 0000	uuuu uuuu
PTM1DH	0000 0000	0000 0000	uuuu uuuu
PTM1AL	0000 0000	0000 0000	uuuu uuuu
PTM1AH	0000 0000	0000 0000	uuuu uuuu
PTM1RPL	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	0000 0000	0000 0000	uuuu uuuu
PTM0C0	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	uuuu uuuu
PTM0DL	0000 0000	0000 0000	uuuu uuuu
PTM0DH	0000 0000	0000 0000	uuuu uuuu
PTM0AL	0000 0000	0000 0000	uuuu uuuu
PTM0AH	0000 0000	0000 0000	uuuu uuuu
PTM0RPL	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	0000 0000	0000 0000	uuuu uuuu
PTM2C0	0000 0---	0000 0---	uuuu u---
PTM2C1	0000 0000	0000 0000	uuuu uuuu
PTM2DL	0000 0000	0000 0000	uuuu uuuu
PTM2DH	---- --00	---- --00	---- --uu
PTM2AL	0000 0000	0000 0000	uuuu uuuu
PTM2AH	---- --00	---- --00	---- --uu
PTM2RPL	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	---- --00	---- --00	---- --uu
PWMC	0000 0000	0000 0000	uuuu uuuu
DUTR0L	0000 0000	0000 0000	uuuu uuuu
DUTR0H	---- --00	---- --00	---- --uu
DUTR1L	0000 0000	0000 0000	uuuu uuuu
DUTR1H	---- --00	---- --00	---- --uu
DUTR2L	0000 0000	0000 0000	uuuu uuuu
DUTR2H	---- --00	---- --00	---- --uu
PRDRL	0000 0000	0000 0000	uuuu uuuu
PRDRH	---- --00	---- --00	---- --uu
PWMRL	0000 0000	0000 0000	uuuu uuuu
PWMRH	---- --00	---- --00	---- --uu
PWMME	--00 0000	--00 0000	--uu uuuu
PWMMD	--00 0000	--00 0000	--uu uuuu
MCF	0--- 0100	0--- 0100	u--- uuuu
MCD	--00 0xxx	--00 0xxx	--uu uuuu
DTS	0000 0000	0000 0000	uuuu uuuu
PLC	--00 0000	--00 0000	--uu uuuu
MFIO	-000 -000	-000 -000	-uuu -uuu
MF11	0000 0000	0000 0000	uuuu uuuu
MF12	0000 0000	0000 0000	uuuu uuuu
MF13	--00 --00	--00 --00	--uu --uu
MF14	--00 --00	--00 --00	--uu --uu
MF15	--00 --00	--00 --00	--uu --uu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
MF16	--00 --00	--00 --00	--uu --uu
CMPC	1111 0000	1111 0000	uuuu uuuu
HCHK_NUM	---0 0000	---0 0000	---u uuuu
HNF_MSEL	---- 0000	---- 0000	---- uuuu
NF_VIH	00-1 1001	00-1 1001	uu-u uuuu
NF_VIL	00-0 1010	00-0 1010	uu-u uuuu
HDCT0	--00 0000	--00 0000	--uu uuuu
HDCT1	--00 0000	--00 0000	--uu uuuu
HDCT2	--00 0000	--00 0000	--uu uuuu
HDCT3	--00 0000	--00 0000	--uu uuuu
HDCT4	--00 0000	--00 0000	--uu uuuu
HDCT5	--00 0000	--00 0000	--uu uuuu
HDCT6	--00 0000	--00 0000	--uu uuuu
HDCT7	--00 0000	--00 0000	--uu uuuu
HDCT8	--00 0000	--00 0000	--uu uuuu
HDCT9	--00 0000	--00 0000	--uu uuuu
HDCT10	--00 0000	--00 0000	--uu uuuu
HDCT11	--00 0000	--00 0000	--uu uuuu
OPOMS	00-- -010	00-- -010	uu-- -uuu
OPCM	0000 0000	0000 0000	uuuu uuuu
OPACAL	-001 0000	-001 0000	-uuu uuuu
PTM3C0	0000 0---	0000 0---	uuuu u---
PTM3C1	0000 0000	0000 0000	uuuu uuuu
PTM3DL	0000 0000	0000 0000	uuuu uuuu
PTM3DH	---- --00	---- --00	---- --uu
PTM3AL	0000 0000	0000 0000	uuuu uuuu
PTM3AH	---- --00	---- --00	---- --uu
PTM3RPL	0000 0000	0000 0000	uuuu uuuu
PTM3RPH	---- --00	---- --00	---- --uu
PAPS0	0000 0000	0000 0000	uuuu uuuu
PAPS1	0000 0000	0000 0000	uuuu uuuu
PRM	-000 00--	-000 00--	-uuu uu--

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

Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA and PB. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction “MOV A, [m]”, where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	PB5	PB4	PB3	PB2	PB1	—
PBC	—	—	PBC5	PBC4	PBC3	PBC2	PBC1	—
PBPU	—	—	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	—

“—”: Unimplemented, read as “0”

I/O Logic Function 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, namely PAPU and PBPU, and are implemented using weak PMOS transistors.

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

• PxPU Register

Bit	7	6	5	4	3	2	1	0
Name	PxPU7	PxPU6	PxPU5	PxPU4	PxPU3	PxPU2	PxPU1	PxPU0
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

PxPUn: I/O Port x pin pull-high function control

0: Disable

1: Enable

The PxPUn bit is used to control the pin pull-high function. Here the “x” can be A and B. However, the actual available bits for each I/O Port may be different.

Port A Wake-up

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

Note that the wake-up function can be controlled by the wake-up control registers only when the pin-shared functional pin is selected as general purpose input/output and the MCU enters any of the IDLE or SLEEP modes.

• PAWU Register

Bit	7	6	5	4	3	2	1	0
Name	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PAWU7~PAWU0**: PA7~PA0 pin wake-up function control

0: Disable

1: Enable

I/O Port Control Registers

Each I/O port has its own control register known as PAC and PBC, 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.

• PxC Register

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

PxCn: I/O Port x pin input/output type selection

0: Output

1: Input

The PxCn bit is used to control the pin type selection. Here the “x” can be A and B. However, the actual available bits for each I/O Port may be different.

Pin-shared Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For these pins, the desired function of the multi-function I/O pins is selected by a series of registers via the application program control.

Pin-shared Function Selection Registers

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

When the pin-shared input function is selected to be used, the corresponding input and output functions selection should be properly managed. For example, if the Capture Timer CTIN pin is used, the corresponding pin-shared function should be configured as the CTIN function by configuring the PxPSn register and the CTIN signal input should be properly selected using the PRM register. If the external interrupt function is selected to be used, the relevant pin-shared function should be selected as an I/O function and the interrupt input signal active edge should be selected.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as INT1, PTCKn, PTPnI, NFIN, etc, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bit fields. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be setup as an input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PRM	—	INT1S	CTINPS1	CTINPS0	NFINPS1	NFINPS0	—	—
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBPS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	—	—
PBPS1	—	—	—	—	PBS13	PBS12	PBS11	PBS10

“—”: Unimplemented, read as “0”

Pin-shared Function Selection Register List

• PRM Register

Bit	7	6	5	4	3	2	1	0
Name	—	INT1S	CTINPS1	CTINPS0	NFINPS1	NFINPS0	—	—
R/W	—	R/W	R/W	R/W	R/W	R/W	—	—
POR	—	0	0	0	0	0	—	—

Bit 7 Unimplemented, read as “0”

Bit 6 **INT1S**: INT1 input source pin selection
 0: PA7
 1: PB1

Bit 5~4 **CTINPS1~CTINPS0**: CTIN input source pin selection
 00: PB1
 01: PA6
 10: PA1
 11: PB1

- Bit 3~2 **NFINPS1~NFINPS0**: NFIN input source pin selection
 00: PA7
 01: PA6
 10: PB2
 11: PA7
- Bit 1~0 Unimplemented, read as “0”

• **PAPS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
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 **PAS07~PAS06**: PA3 Pin-Shared function selection
 00: PA3/H1/PTP1I
 01: C1P
 10: PA3/H1/PTP1I
 11: PA3/H1/PTP1I
- Bit 5~4 **PAS05~PAS04**: PA2 Pin-Shared function selection
 00: PA2
 01: PTP1
 10: PTP3
 11: PA2
- Bit 3~2 **PAS03~PAS02**: PA1 Pin-Shared function selection
 00: PA1/PTP0I/[CTIN]
 01: AN4
 10: PA1/PTP0I/[CTIN]
 11: AP

Note that before using these two bits to select CTIN function as PA1 pin function, the CTIN input source should be managed at PA1 pin by programming the PRM register correctly.

- Bit 1~0 **PAS01~PAS00**: PA0 Pin-Shared function selection
 00: PA0
 01: PTP1
 10: PTP3
 11: PA0

• **PAPS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
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 **PAS17~PAS16**: PA7 Pin-Shared function selection
 00: PA7/NFIN/INT1
 01: AN6
 10: HAO
 11: OPAO
- Bit 5~4 **PAS15~PAS14**: PA6 Pin-Shared function selection
 00: PA6/[NFIN]/[CTIN]/PTCK1
 01: AN7
 10: PA6/[NFIN]/[CTIN]/PTCK1
 11: VCCO

Note that before using these two bits to select CTIN or NFIN function as PA6 pin function, the CTIN or NFIN input source should be managed at PA6 pin by programming the PRM register correctly.

- Bit 3~2 **PAS13~PAS12:** PA5 Pin-Shared function selection
 00: PA5/H3/PTP3I
 01: C3P
 10: PTP2
 11: PTP1
- Bit 1~0 **PAS11~PAS10:** PA4 Pin-Shared function selection
 00: PA4/H2/PTP2I
 01: C2P
 10: PTP1
 11: PTP2

• **PBPS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	—	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	—
POR	0	0	0	0	0	0	—	—

- Bit 7~6 **PBS07~PBS06:** PB3 Pin-Shared function selection
 00: PB3
 01: C3N
 10: AN0
 11: PTP3

- Bit 5~4 **PBS05~PBS04:** PB2 Pin-Shared function selection
 00: PB2/PTCK0/PTCK3/[NFIN]
 01: AN1
 10: HCO
 11: PTP2

Note that before using these two bits to select NFIN function as PB2 pin function, the NFIN input source should be managed at PB2 pin by programming the PRM register correctly.

- Bit 3~2 **PBS03~PBS02:** PB1 Pin-Shared function selection
 00: PB1/CTIN/PTCK2
 01: AN2
 10: HBO
 11: PTP2

- Bit 1~0 Unimplemented, read as “0”

• **PBPS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PBS13	PBS12	PBS11	PBS10
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

- Bit 7~4 Unimplemented, read as “0”

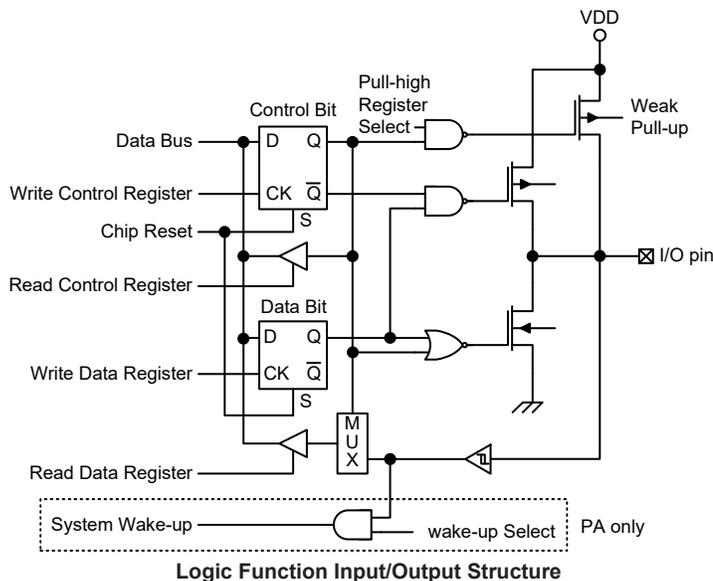
- Bit 3~2 **PBS13~PBS12:** PB5 Pin-Shared function selection
 00: PB5
 01: C1N
 10: CPN
 11: PTP0

When these bits are set to 10B, the C1N, C2N and C3N will be shorted internally and the CPN pin function is selected. Refer to the Comparators chapter for more details about this option.

Bit 1~0 **PBS11~PBS10:** PB4 Pin-Shared function selection
 00: PB4
 01: C2N
 10: PTP0
 11: PTP3

I/O Pin Structures

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



Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate I/O Port Control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

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

Timer Modules – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions each device includes several Timer Modules, abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has 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.

Introduction

The device contains four Periodic TMs with each TM having a reference name of PTM_n (n=0~3). The PTM0 and PTM1 are 16-bit Periodic Type TMs while the PTM2 and PTM3 are 10-bit Periodic Type TMs. 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 the Periodic Type TMs section. The main features of TMs are summarised in the accompanying table.

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

TM Function Summary

PTM0	PTM1	PTM2	PTM3
16-bit PTM	16-bit PTM	10-bit PTM	10-bit PTM

TM Name/Type Reference

TM Operation

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

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the PTnCK2~PTnCK0 bits in the PTM_n control registers, where “n” stands for the specific TM serial number. The clock source can be a ratio of the system clock f_{SYS} or the internal high clock f_{IH} , the f_{SUB} clock source or the external PTCK_n pin. The PTCK_n pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

The TMs each has two internal interrupts, the internal comparator A and 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

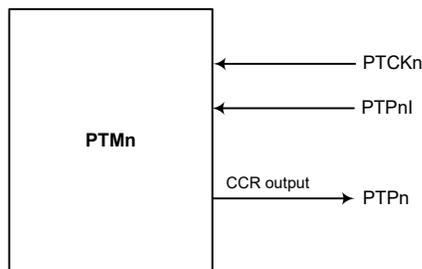
Each of the TMs has two TM input pins, with the label PTCKn and PTPnI. These two pins can be used as the capture input source pin whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTnIO1~PTnIO0 bits in the PTMnC1 register. The capture input comes from PTCKn or PTPnI is determined by the PTnCAPTS bit in the PTMnC1 register. The PTCKn input pin is also a clock source for the PTMn and is selected using the PTnCK2~PTnCK0 bits in the PTMnC0 register. This PTCKn input pin allows an external clock source to drive the internal TM. The PTCKn input pin can be chosen to have either a rising or falling active edge. The PTCKn pin is also used as the external trigger input pin in single pulse output mode.

The TMs each has one output with the label PTPn. When the TM is in the Compare Match Output Mode, the output 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 output pins are also the pins where the TM generates the PWM output waveform.

As the TM input and output pins are pin-shared with other functions, the TM input and output function must first be setup using relevant pin-shared function selection bits described in the Pin-shared Functions section. The details of the pin-shared function selection are described in the pin-shared function section.

PTM0		PTM1		PTM2		PTM3	
Input	Output	Input	Output	Input	Output	Input	Output
PTCK0, PTP0I	PTP0	PTCK1, PTP1I	PTP1	PTCK2, PTP2I	PTP2	PTCK3, PTP3I	PTP3

TM External Pins

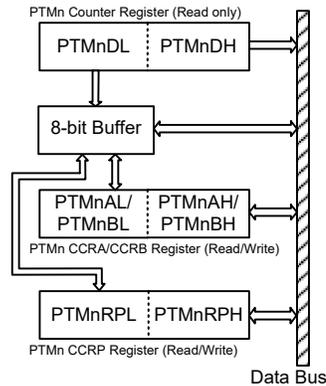


PTMn Function Pin Block Diagram (n=0~3)

Programming Considerations

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

As the CCRA, CCRB 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 as described above, it is recommended to use the “MOV” instruction to access the CCRA, CCRB and CCRP low byte registers, named PTMnAL, PTMnBL and PTMnRPL, using the following access procedures. Accessing the CCRA, CCRB or CCRP low byte registers without following these access procedures will result in unpredictable values.



The following steps show the read and write procedures:

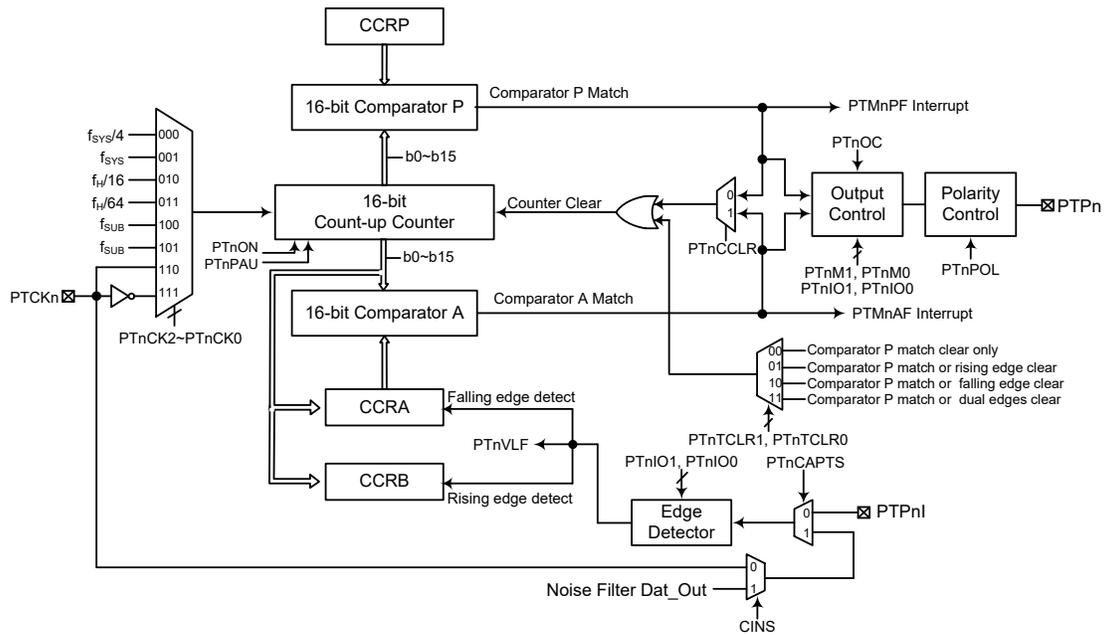
- Writing Data to CCRA, CCRB or CCRP
 - ♦ Step 1. Write data to Low Byte PTMnAL, PTMnBL or PTMnRPL
 - Note that here data is only written to the 8-bit buffer.
 - ♦ Step 2. Write data to High Byte PTMnAH, PTMnBH or PTMnRPH
 - Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA, CCRB or CCRP
 - ♦ Step 1. Read data from the High Byte PTMnDH, PTMnAH, PTMnBH 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, PTMnBL or PTMnRPL
 - This step reads data from the 8-bit buffer.

Periodic Type TM – PTM

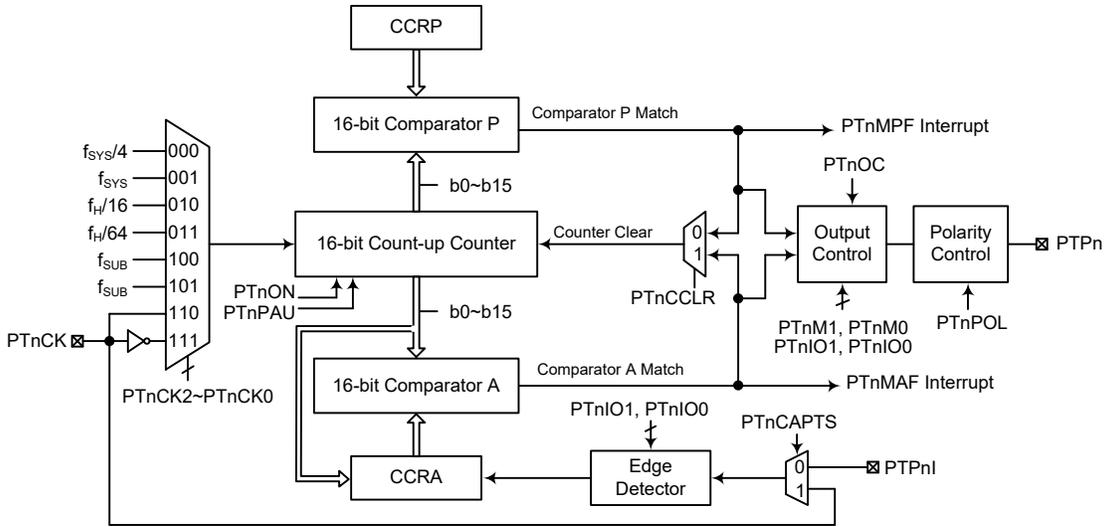
The Periodic Type TM contains five operating modes, which are Compare Match Output, Timer/Event Counter, Capture Input, Single Pulse Output and PWM Output modes. The Periodic TM can be controlled with two external input pins and can drive one external output pin.

Note that the PTMn external pins are pin-shared with other functions and can input or output on several pins, so before using the PTMn functions, the pin-shared function registers must be set properly to enable the PTMn pin function.

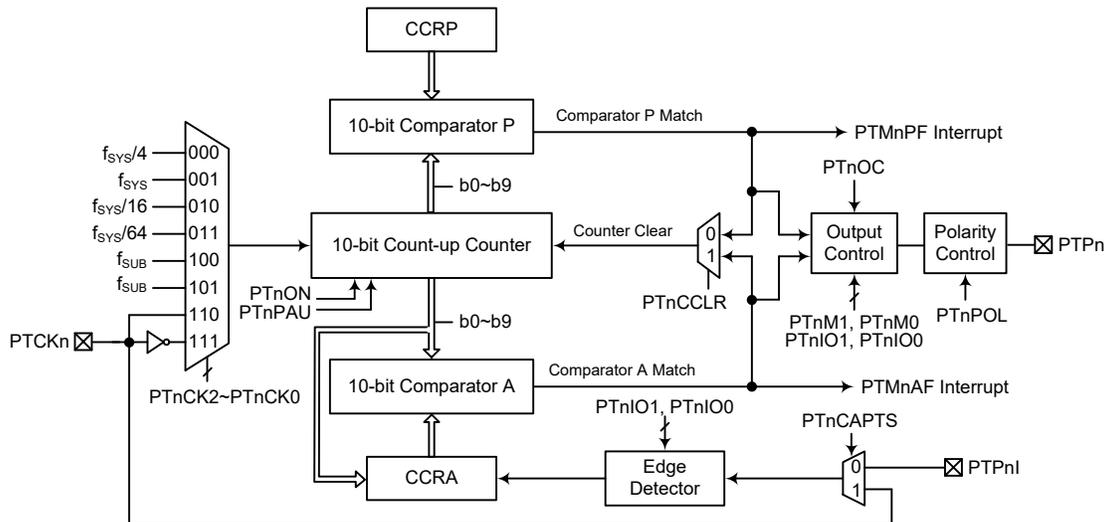
Type	Name	TM Input Pin	TM Output Pin
16-bit PTM	PTM0, PTM1	PTCK0, PTP0I; PTCK1, PTP1I	PTP0; PTP1
10-bit PTM	PTM2, PTM3	PTCK2, PTP2I; PTCK3, PTP3I	PTP2; PTP3



16-bit Periodic Type TM Block Diagram (n=0)



16-bit Periodic Type TM Block Diagram (n=1)



10-bit Periodic Type TM Block Diagram (n=2~3)

Periodic TM Operation

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

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

Periodic Type TM Register Description

Overall operation of the Periodic Type TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10-bit or 16-bit value, while two read/write register pairs exist to store the internal 10-bit or 16-bit CCRA value and CCRP value. The remaining two registers are control registers which setup the different operating and control modes. For the PTM0, there are another read/write register pair used to store the 16-bit CCRB value, and another control register is provided for its capture input operation.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnC2*	—	—	—	—	—	PTnTCLR1	PTnTCLR0	PTnVLF
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
PTMnBL*	D7	D6	D5	D4	D3	D2	D1	D0
PTMnBH*	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

Note: The Registers with * symbol are only available for PTM0.

16-bit Periodic TM Registers List (n=0~1)

Register Name	Bit							
	7	6	5	4	3	2	1	0
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

10-bit Periodic TM Registers List (n=2~3)

• PTMnC0 Register (n=0~3)

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**: PTMn 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 PTMn 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~PTnCK0**: Select PTMn Counter clock
 000: $f_{SYS}/4$
 001: f_{SYS}
 010: $f_H/16$
 011: $f_H/64$
 100: f_{SUB}
 101: f_{SUB}
 110: PTCKn rising edge clock
 111: PTCKn falling edge clock

These three bits are used to select the clock source for the PTMn. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the oscillator section.

- Bit 3 **PTnON**: PTMn Counter on/off control
 0: Off
 1: On

This bit controls the overall on/off function of the PTMn. Setting the bit high enables the counter to run, clearing the bit disables the PTMn. Clearing this bit to zero will stop the counter from counting and turn off the PTMn 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 PTMn is in the Compare Match Output Mode, PWM output Mode or Single Pulse Output Mode then the PTMn 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 (n=0)**

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCPTS	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~PTnM0**: Select PTMn Operating Mode
 00: Compare Match Output Mode
 01: Capture Input Mode
 10: PWM Output Mode or Single Pulse Output Mode
 11: Timer/Counter Mode

These bits setup the required operating mode for the PTMn. To ensure reliable operation the PTMn should be switched off before any changes are made to the PTnM1 and PTnM0 bits. In the Timer/Counter Mode, the PTMn output pin state is undefined.

- Bit 5~4 **PTnIO1~PTnIO0**: Select PTMn pin function
 Compare Match Output Mode
 00: No change
 01: Output low
 10: Output high
 11: Toggle output
 PWM Output 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

PTnTCLR[1:0]=00B:

00: Input capture at rising edge of input signal and the counter value will be latched into CCRA

01: Input capture at falling edge of input signal and the counter value will be latched into CCRA

10: Input capture at falling/rising edge of input signal and the counter value will be latched into CCRA

11: Input capture disabled

PTnTCLR[1:0]=01B or 10B or 11B:

00: Input capture at rising edge of input signal and the counter value will be latched into CCRB

01: Input capture at falling edge of input signal and the counter value will be latched into CCRA

10: Input capture at falling/rising edge of input signal and the counter value will be latched into CCRA at falling edge and into CCRB at rising edge

11: Input capture disabled

Timer/Counter Mode

Unused

These two bits are used to determine how the PTMn operates when a certain condition is reached. The function that these bits select depends upon in which mode the PTMn is running.

In the Compare Match Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn output pin changes state when a compare match occurs from the Comparator A. The PTMn 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 PTMn 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 PTMn output pin when a compare match occurs. After the PTMn 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 Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn 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 after the PTMn has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the PTMn is running.

In the Capture Input Mode, the PTnIO1 and PTnIO0 bits are used to select the active trigger edge on the selected input signal.

Bit 3

PTnOC: PTMn Output control bit

Compare Match Output Mode

0: Initial low

1: Initial high

PWM Output Mode/Single Pulse Output Mode

0: Active low

1: Active high

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

- Bit 2 **PTnPOL**: PTMn Output polarity control
 0: Non-invert
 1: Invert
 This bit controls the polarity of the output pins. When the bit is set high the PTMn output pin will be inverted and not inverted when the bit is zero. It has no effect if the PTMn is in the Timer/Counter Mode.
- Bit 1 **PTnCAPTS**: PTMn Capture Trigger Source selection
 0: From PTPnI pin
 1: From PTCKn pin (with 150ns filter) or Noise Filter Dat_Out
 This bit is used to select the PTMn capture input trigger source. When the PTnCAPTS bit is set to 1, the capture input trigger source can be PTCKn or noise filtered Dat_Out signal determined using the CINS bit in the NF_VIH register.
- Bit 0 **PTnCCLR**: Select PTMn Counter clear condition
 0: PTMn Comparator P match
 1: PTMn Comparator A match
 This bit is used to select the method which clears the counter. Remember that the Periodic TM 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 Output Mode, Single Pulse or Capture Input Mode or Capture Input Mode.

• **PTMnC1 Register (n=1~3)**

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~PTnM0**: Select PTMn Operating Mode
 00: Compare Match Output Mode
 01: Capture Input Mode
 10: PWM Output Mode or Single Pulse Output Mode
 11: Timer/Counter Mode
 These bits setup the required operating mode for the PTMn. To ensure reliable operation the PTMn should be switched off before any changes are made to the PTnM1 and PTnM0 bits. In the Timer/Counter Mode, the PTMn output pin state is undefined.
- Bit 5~4 **PTnIO1~PTnIO0**: Select PTMn pin function
 Compare Match Output Mode
 00: No change
 01: Output low
 10: Output high
 11: Toggle output
 PWM Output 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 input signal
 01: Input capture at falling edge of input signal
 10: Input capture at falling/rising edge of input signal
 11: Input capture disabled

Timer/Counter Mode

Unused

These two bits are used to determine how the PTMn operates when a certain condition is reached. The function that these bits select depends upon in which mode the PTMn is running.

In the Compare Match Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn output pin changes state when a compare match occurs from the Comparator A. The PTMn 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 PTMn 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 PTMn output pin when a compare match occurs. After the PTMn 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 Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn 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 TM has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the PTMn is running.

In the Capture Input Mode, the PTnIO1 and PTnIO0 bits are used to select the active trigger edge on the selected input source pin.

Bit 3

PTnOC: PTMn Output control bit

Compare Match Output Mod

0: Initial low

1: Initial high

PWM Output Mode/Single Pulse Output Mode

0: Active low

1: Active high

This is the output control bit for the PTMn output pin. Its operation depends upon whether PTMn is being used in the Compare Match Output Mode or in the PWM Output Mode/Single Pulse Output Mode.

It has no effect if the PTMn is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the PTMn output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low. In the Single Pulse Output Mode it determines the logic level of the PTMn output pin when the PTnON bit changes from low to high.

Bit 2

PTnPOL: PTPn Output polarity control

0: Non-invert

1: Invert

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

Bit 1

PTnCAPTS: PTMn Capture Trigger Source selection

0: From PTPnI pin

1: From PTCKn pin

Bit 0

PTnCCLR: Select PTMn Counter clear condition

0: PTMn Comparator P match

1: PTMn Comparator A match

This bit is used to select the method which clears the counter. Remember that the Periodic TM 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 Output Mode, Single Pulse Output Mode or Capture Input Mode.

• **PTMnC2 Register (n=0 only)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PTnTCLR1	PTnTCLR0	PTnVLF
R/W	—	—	—	—	—	R/W	R/W	R
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as “0”

Bit 2~1 **PTnTCLR1~PTnTCLR0**: Select PTMn Timer clear condition (for capture input mode only)

- 00: Comparator P match clear only
- 01: Comparator P match clear or rising edge clear
- 10: Comparator P match clear or falling edge clear
- 11: Comparator P match clear or dual edges clear

Bit 0 **PTnVLF**: PTMn Counter Value Latch trigger edge flag

- 0: Falling edge trigger the counter value latch
- 1: Rising edge trigger the counter value latch

When setting PTnTCLR1~PTnTCLR0 bits equal to 00B, ignore this flag status.

• **PTMnBL Register (n=0 only)**

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

Bit 7~0 **D7~D0**: PTM0 CCRB Low Byte Register bit 7 ~ bit 0
 PTM0 16-bit CCRB bit 7 ~ bit 0

• **PTMnBH Register (n=0 only)**

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 **D15~D8**: PTM0 CCRB High Byte Register bit 7 ~ bit 0
 PTM0 16-bit CCRB bit 15 ~ bit 8

• **PTMnDL Register (n=0~3)**

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 **D7~D0**: PTMn Counter Low Byte Register bit 7 ~ bit 0
 PTMn 10-bit/16-bit Counter bit 7 ~ bit 0

• **PTMnDH Register (n=0~1)**

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 **D15~D8**: PTMn Counter High Byte Register bit 7 ~ bit 0
PTMn 16-bit Counter bit 15 ~ bit 8

• **PTMnDH Register (n=2~3)**

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 **D9~D8**: PTMn Counter High Byte Register bit 1 ~ bit 0
PTMn 10-bit Counter bit 9 ~ bit 8

• **PTMnAL Register (n=0~3)**

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

Bit 7~0 **D7~D0**: PTMn CCRA Low Byte Register bit 7 ~ bit 0
PTMn 10-bit/16-bit CCRA bit 7 ~ bit 0

• **PTMnAH Register (n=0~1)**

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 **D15~D8**: PTMn CCRA High Byte Register bit 7 ~ bit 0
PTMn 16-bit CCRA bit 15 ~ bit 8

• **PTMnAH Register (n=2~3)**

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 1~0 **D9~D8**: PTMn CCRA High Byte Register bit 1 ~ bit 0
PTMn 10-bit CCRA bit 9 ~ bit 8

• **PTMnRPL Register (n=0~3)**

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

Bit 7~0 **D7~D0**: PTMn CCRP Low Byte Register bit 7 ~ bit 0
 PTMn 10-bit/16-bit CCRP bit 7 ~ bit 0

• **PTMnRPH Register (n=0~1)**

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 **D15~D8**: PTMn CCRP High Byte Register bit 7 ~ bit 0
 PTMn 16-bit CCRP bit 15 ~ bit 8

• **PTMnRPH Register (n=2~3)**

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 1~0 **D9~D8**: PTMn CCRP High Byte Register bit 1 ~ bit 0
 PTMn 10-bit CCRP bit 9 ~ bit 8

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.

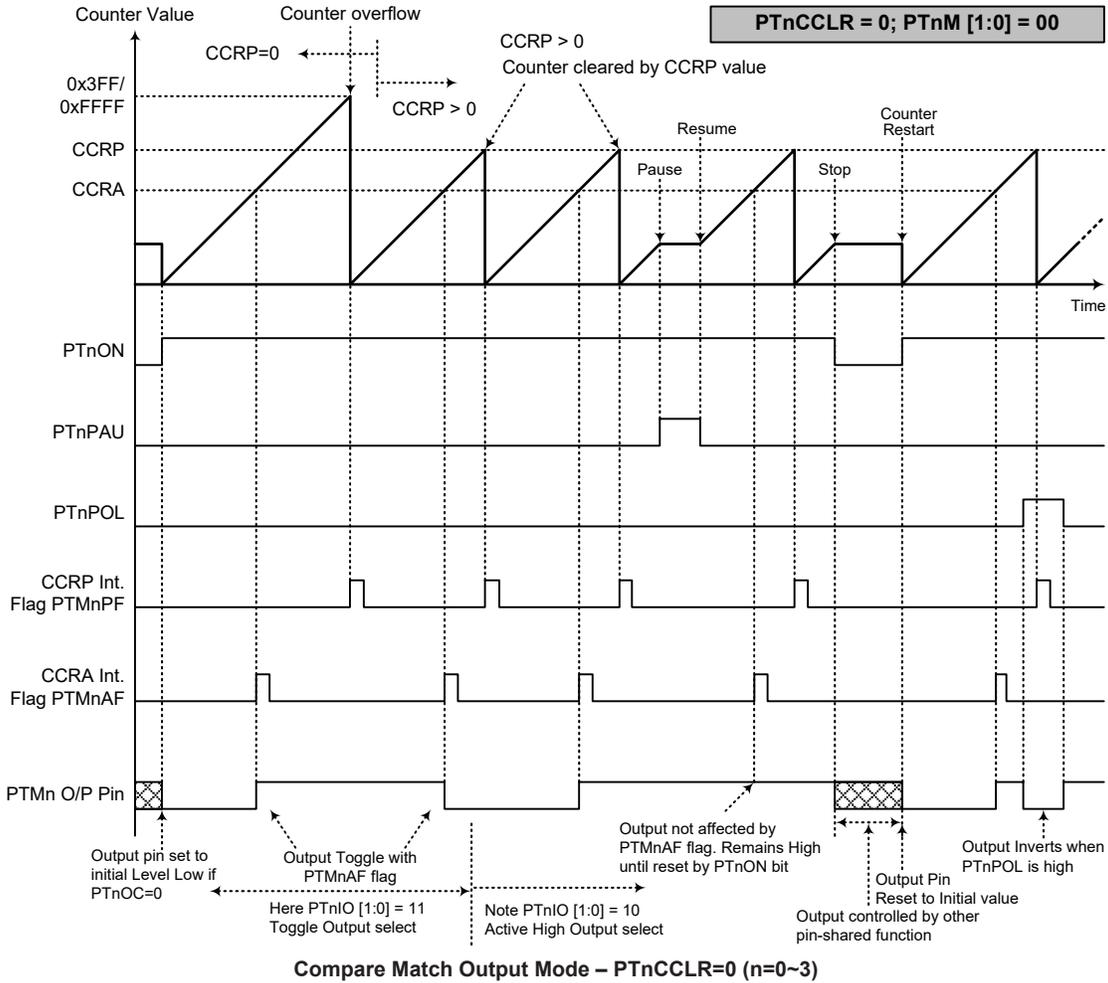
Compare Match Output Mode

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.

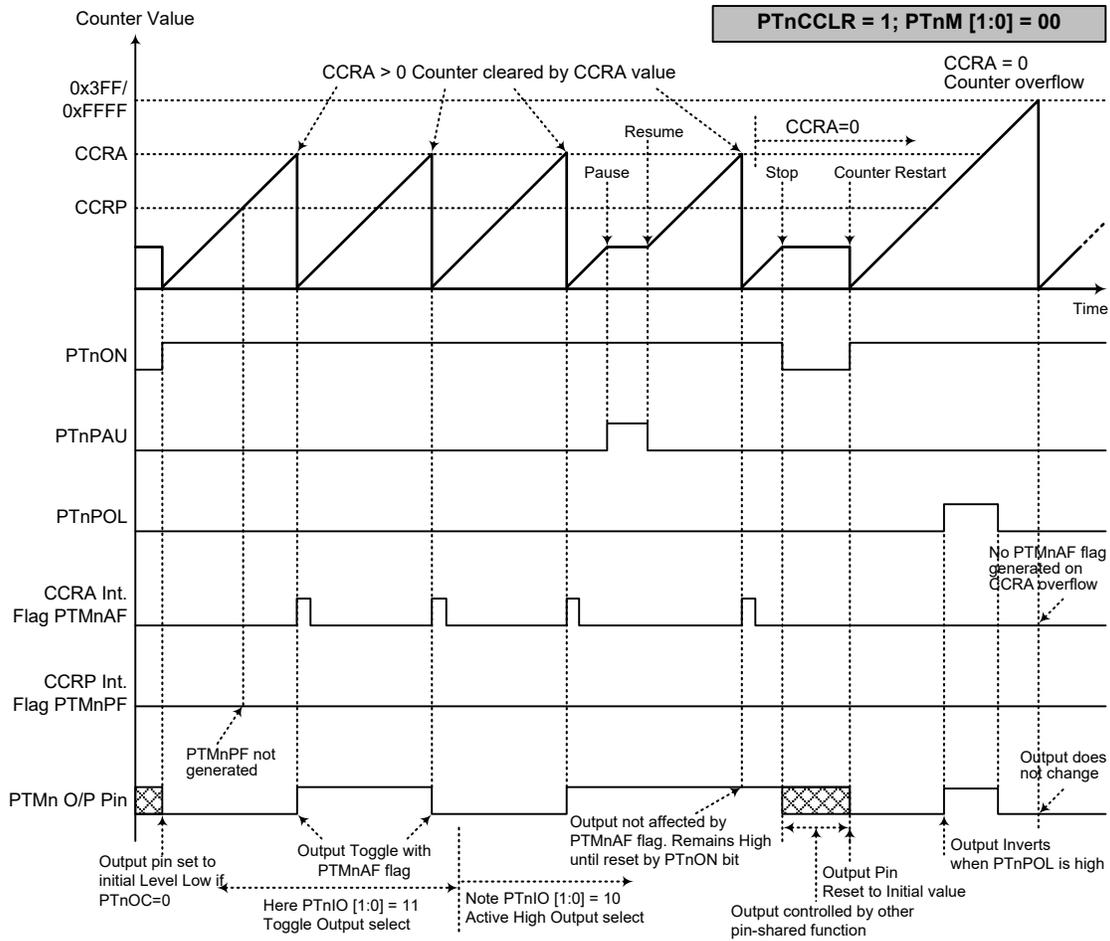
If the PTnCCLR bit in the PTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is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

If the CCRA bits are all zero, the counter will overflow when its reaches its maximum value 3FFH for 10-bit or FFFFH for 16-bit, however here the PTMnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is matched, the PTMn output pin will change state. The PTMn 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 PTMn output pin. The way in which the PTMn output pin changes state are determined by the condition of the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The PTMn 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 PTMn 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.



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



Compare Match Output Mode – PTnCCR=1 (n=0~3)

- Note:
1. With PTnCCR=1 a Comparator A match clear the counter
 2. The PTMn output pin controlled only by the PTMnAF flag
 3. The output pin reset to its initial state by a PTnON bit rising edge
 4. A PTMnPF flag is not generated when PTnCCR=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 PTMn output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the TM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits 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 PTMn 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 PTMn 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 Output Mode, the PTnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, CCRP register is used to clear the internal counter and thus control the PWM waveform frequency, while CCRA register 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 PTMn 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 PTMn, PWM Output Mode, Edge-aligned Mode**

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

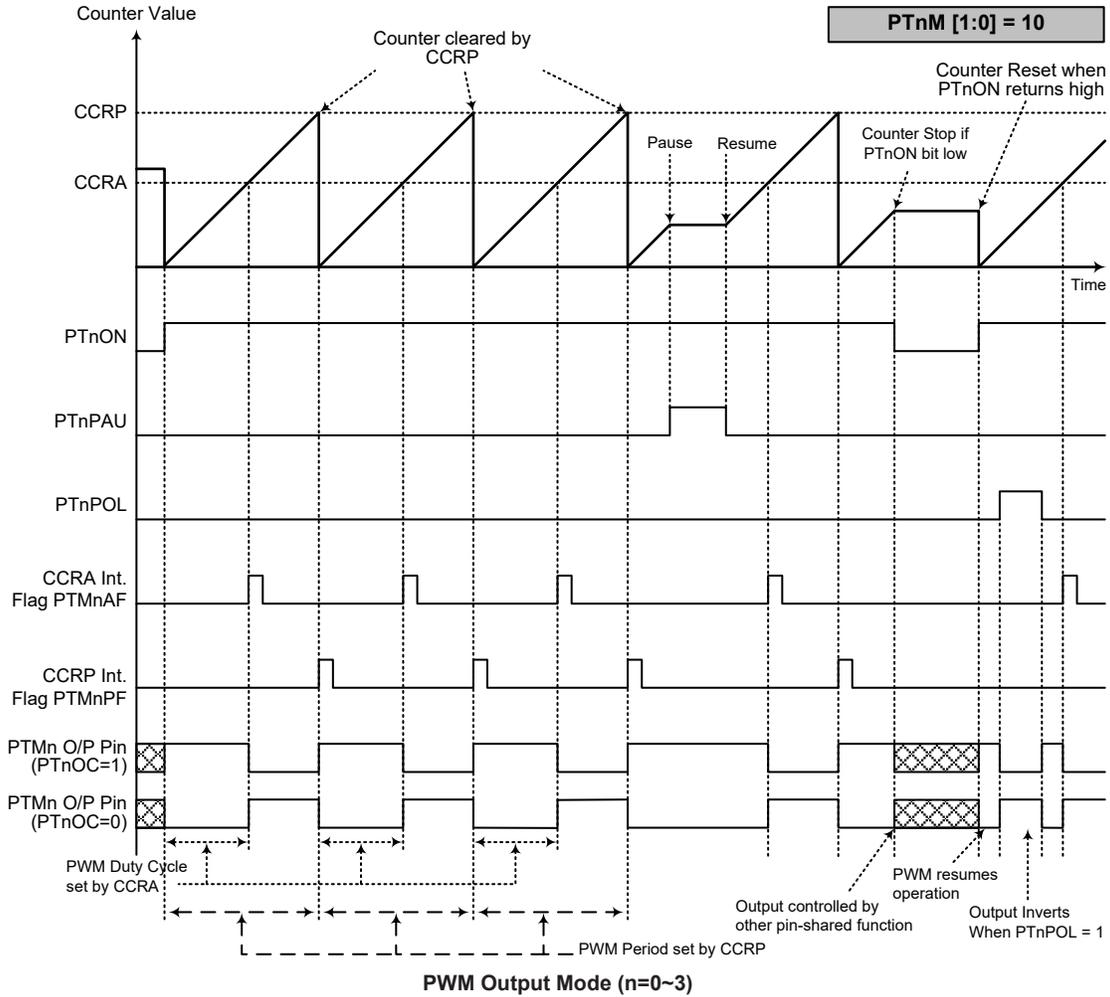
• **10-bit PTMn, PWM Output Mode, Edge-aligned Mode**

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

If $f_{SYS}=4\text{MHz}$, PTMn clock source select $f_{SYS}/4$, CCRP=512 and CCRA=128,

The PTMn PWM output frequency= $(f_{SYS}/4)/512=f_{SYS}/2048=2\text{kHz}$, duty= $128/512=25\%$.

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



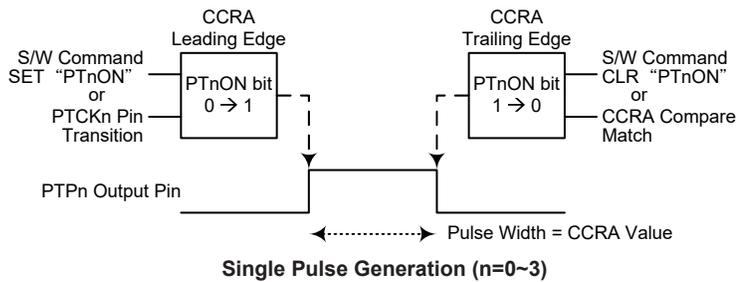
- Note:
1. Counter cleared by CCRP
 2. A counter clear sets the PWM Period
 3. The internal PWM function continues running even when PTnIO[1:0]=00 or 01
 4. The PTnCCLR bit has no influence on PWM operation

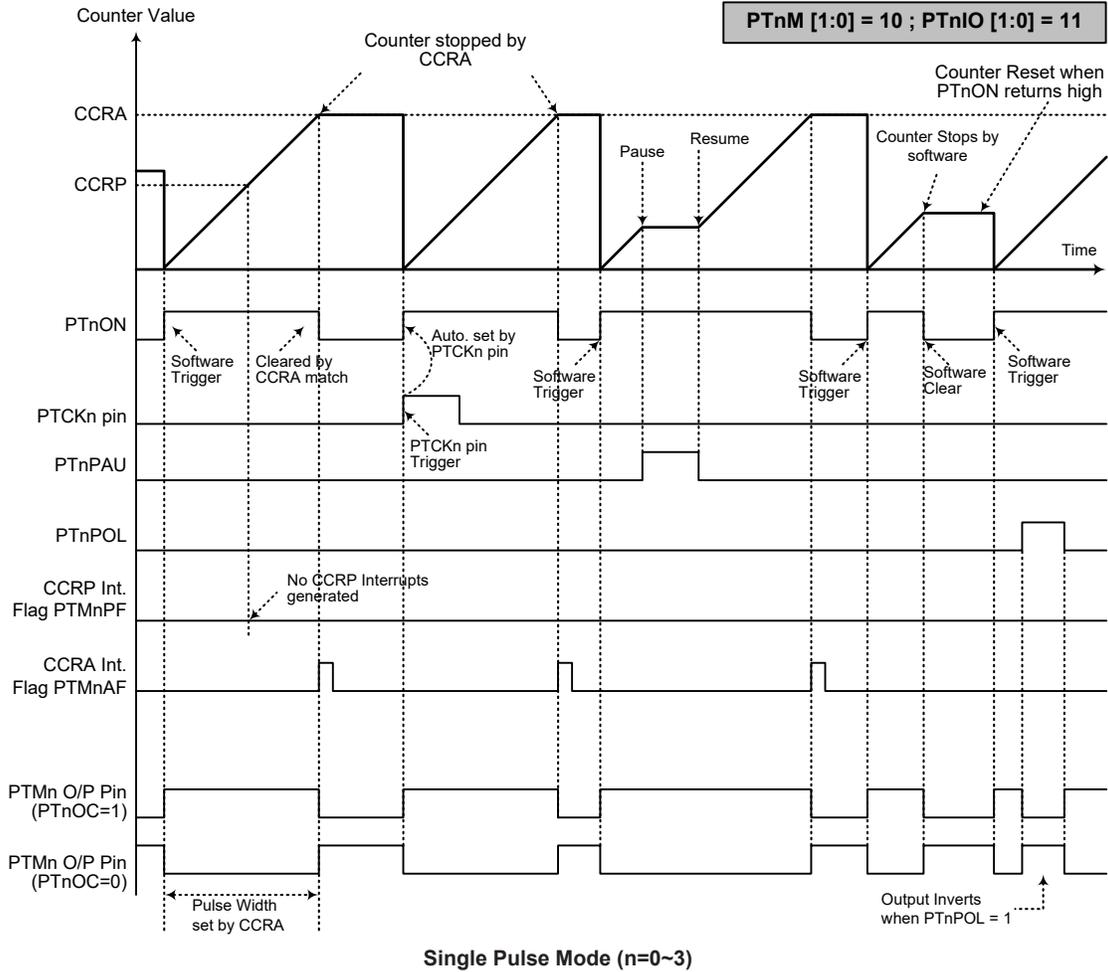
Single Pulse Output Mode

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 PTMn 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 PTCKn 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 PTMn 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 bit is not used in this Mode.





- Note: 1. Counter stopped by CCRA
 2. CCRP is not used
 3. The pulse is triggered by the PTCKn pin or by setting the PTnON bit high
 4. A PTCKn 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 cannot be changed

Capture Input Mode (PTM0)

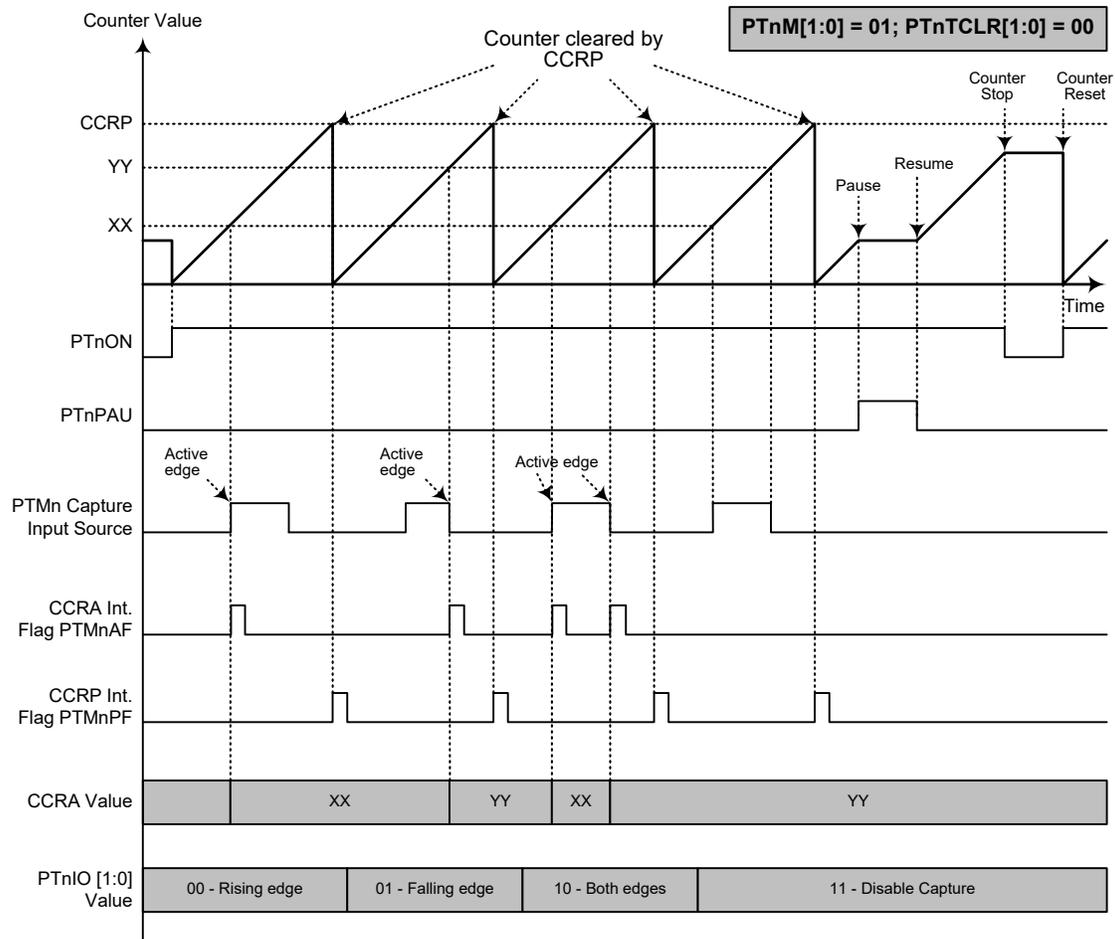
To select this mode bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPnI or PTCKn pin which is selected using the PTnCAPTS bit in the PTMnC1 register. The signal sourced from the Noise Filter Dat_Out which is a filtered signal of the NFIN pin input is also can be selected as the capture input signal. The input signal 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.

The PTnIO1 and PTnIO0 bits decide which active edge transition type to be latched and to generate an interrupt. The PTnTCLR1 and PTnTCLR0 bits decide the condition that the counter reset back to zero. The present counter value latched into CCRA or CCRB is decided by PTnIO1~PTnIO0 together with PTnTCLR1~PTnTCLR0 setting. The PTnIO1~PTnIO0 and PTnTCLR1~PTnTCLR0 bits are setup independently on each other.

When the required edge transition appears on the PTPnI or PTCKn pin or Dat_Out signal the present value in the counter will be latched into the CCRA or CCRB registers and a PTMn interrupt generated. Irrespective of what events occur on the PTPnI or PTCKn pin or Dat_Out signal, 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 PTMn 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 PTPnI or PTCKn pin or Dat_Out signal 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 PTPnI or PTCKn pin or Dat_Out signal, however it must be noted that the counter will continue to run.

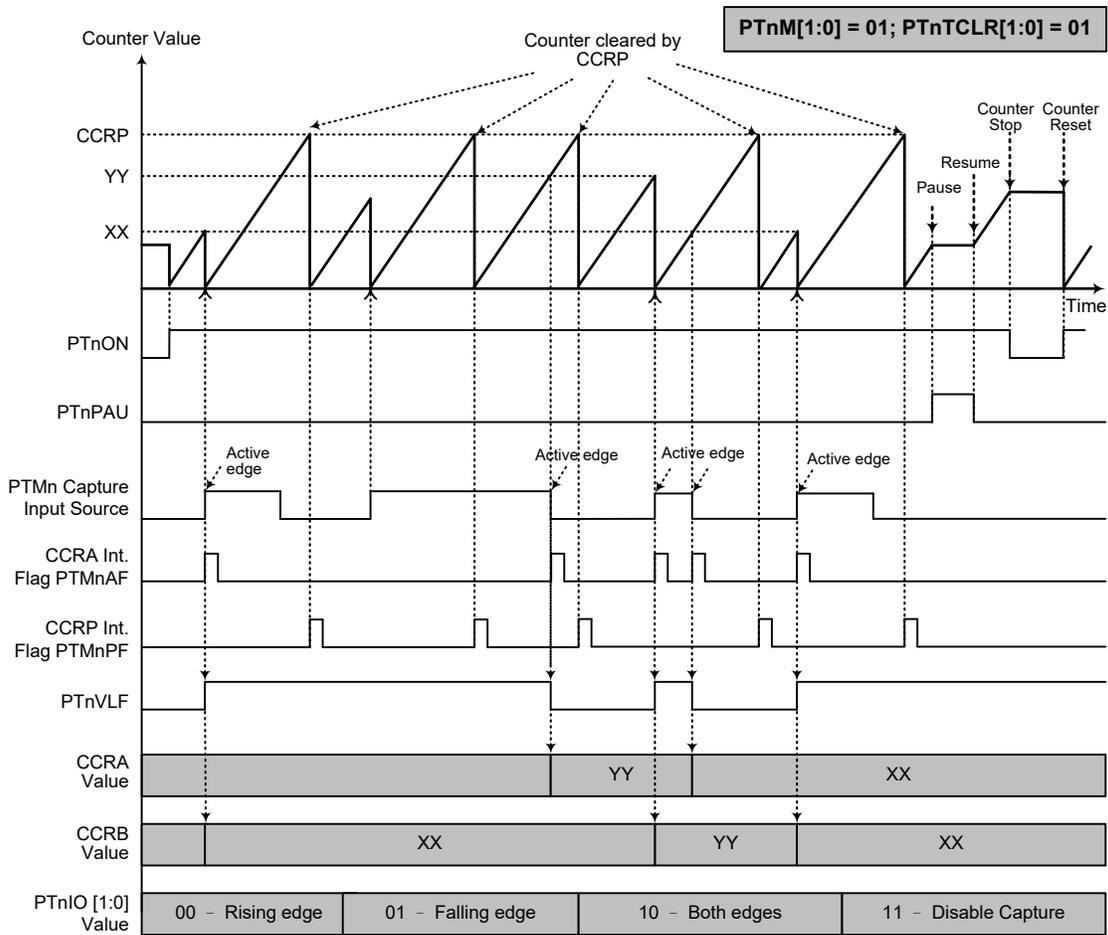
If the capture pulse width is less than two timer clock cycles, it may be ignored by hardware. The timer clock source must be equal to or less than 50MHz, otherwise the counter may fail to count.

As the PTPnI or PTCKn pin is pin-shared with other functions, care must be taken if the PTMn is in the Capture Input 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.



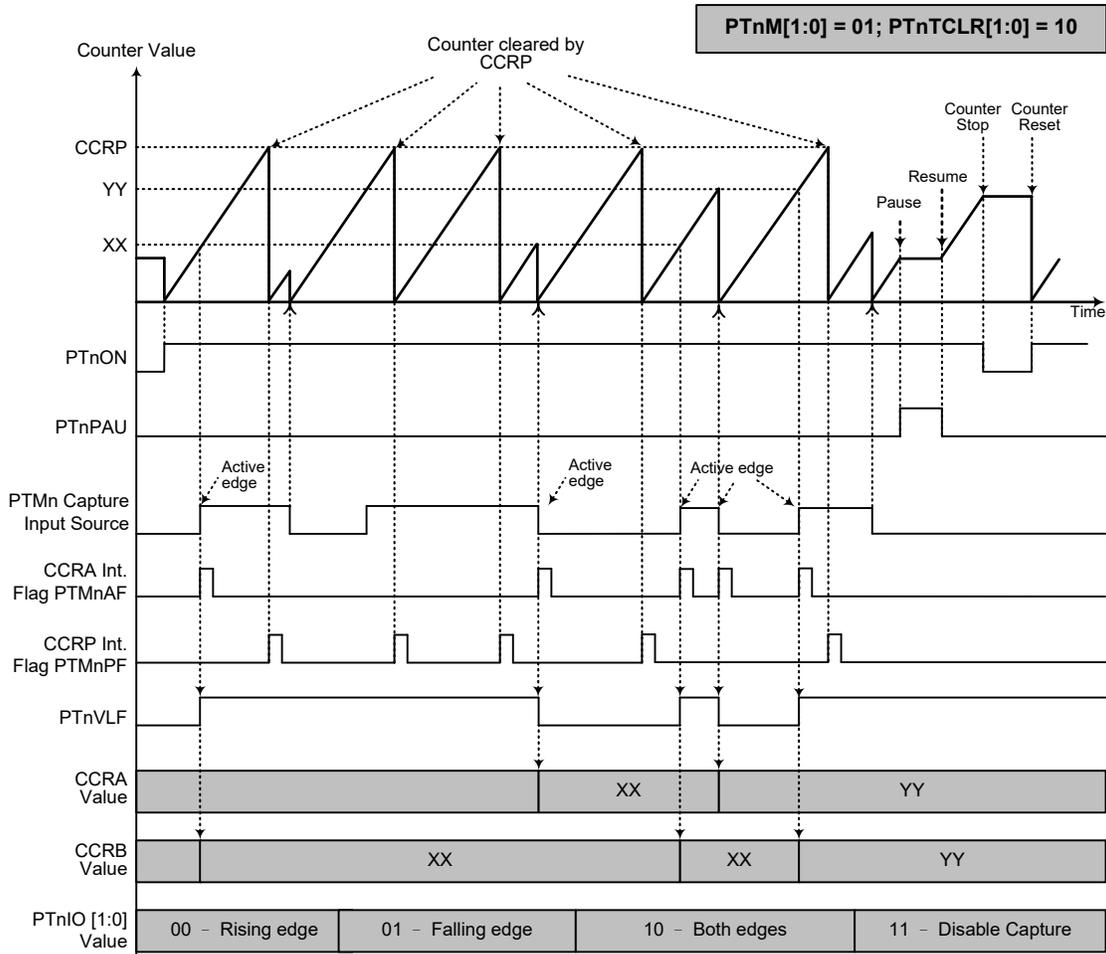
Capture Input Mode – PTnTCLR[1:0]=00 (n=0)

- Note: 1. PTnM[1:0]=01, PTnTCLR[1:0]=00 and active edge set by the PTnIO[1:0] bits
 2. A PTMn Capture input pin active edge transfers the counter value to CCRA
 3. Comparator P match will clear the counter
 4. PTnCCLR bit not used
 5. No output function – PTnOC and PTnPOL bits are not used
 6. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero.
 7. Ignore the PTnVLF bit status when PTnTCLR[1:0]=00



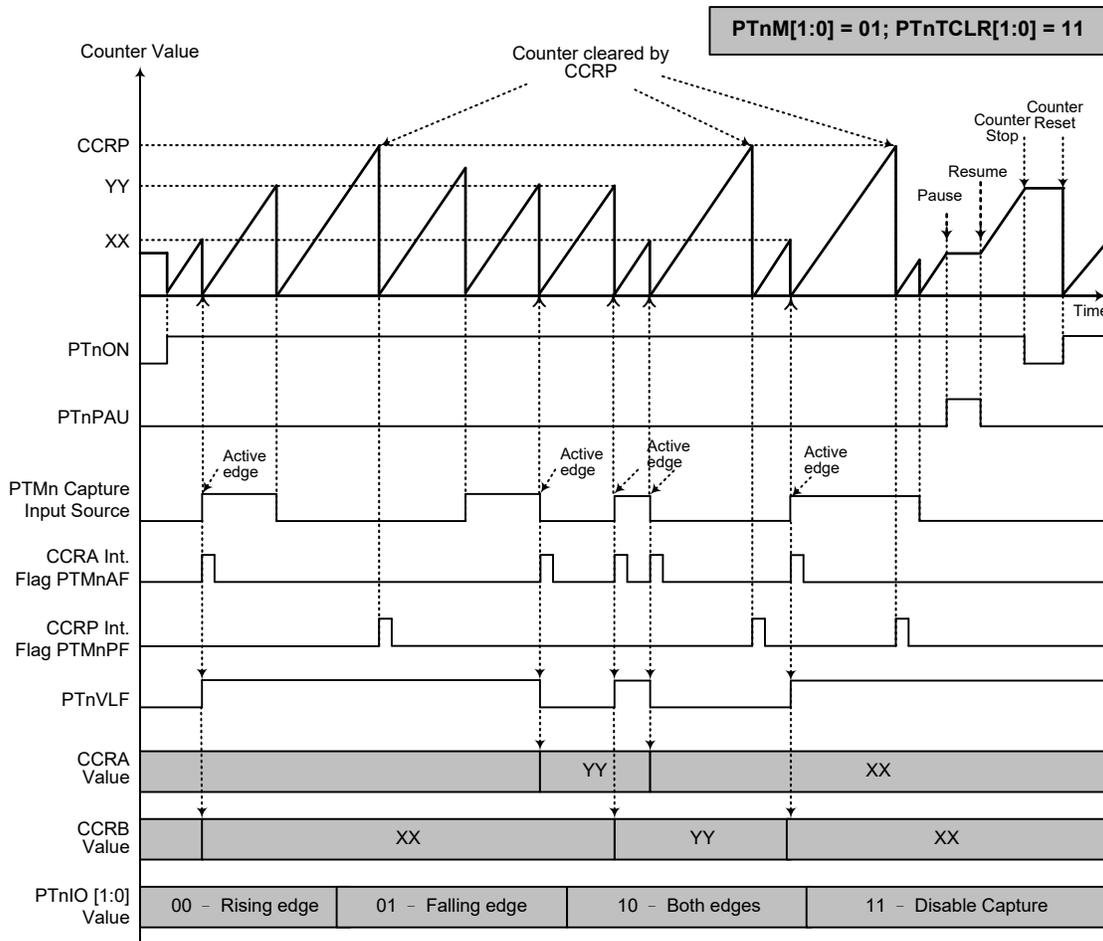
Capture Input Mode – PTnTCLR[1:0]=01 (n=0)

- Note: 1. PTnM[1:0]=01, PTnTCLR[1:0]=01 and active edge set by the PTnIO[1:0] bits
 2. A PTMn Capture input pin active edge transfers the counter value to CCRA or CCRB
 3. Comparator P match or PTMn capture input rising edge will clear the counter
 4. PTnCCLR bit is not used
 5. No output function – PTnOC and PTnPOL bits are not used
 6. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero



Capture Input Mode – PTnTCLR[1:0]=10 (n=0)

- Note: 1. PTnM[1:0]=01, PTnTCLR[1:0]=10 and active edge set by the PTnIO[1:0] bits
 2. A PTMn Capture input pin active edge transfers the counter value to CCRA or CCRB
 3. Comparator P match or PTMn capture input falling edge will clear the counter
 4. PTnCCLR bit is not used
 5. No output function – PTnOC and PTnPOL bits are not used
 6. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero



Capture Input Mode – PTnTCLR[1:0]=11 (n=0)

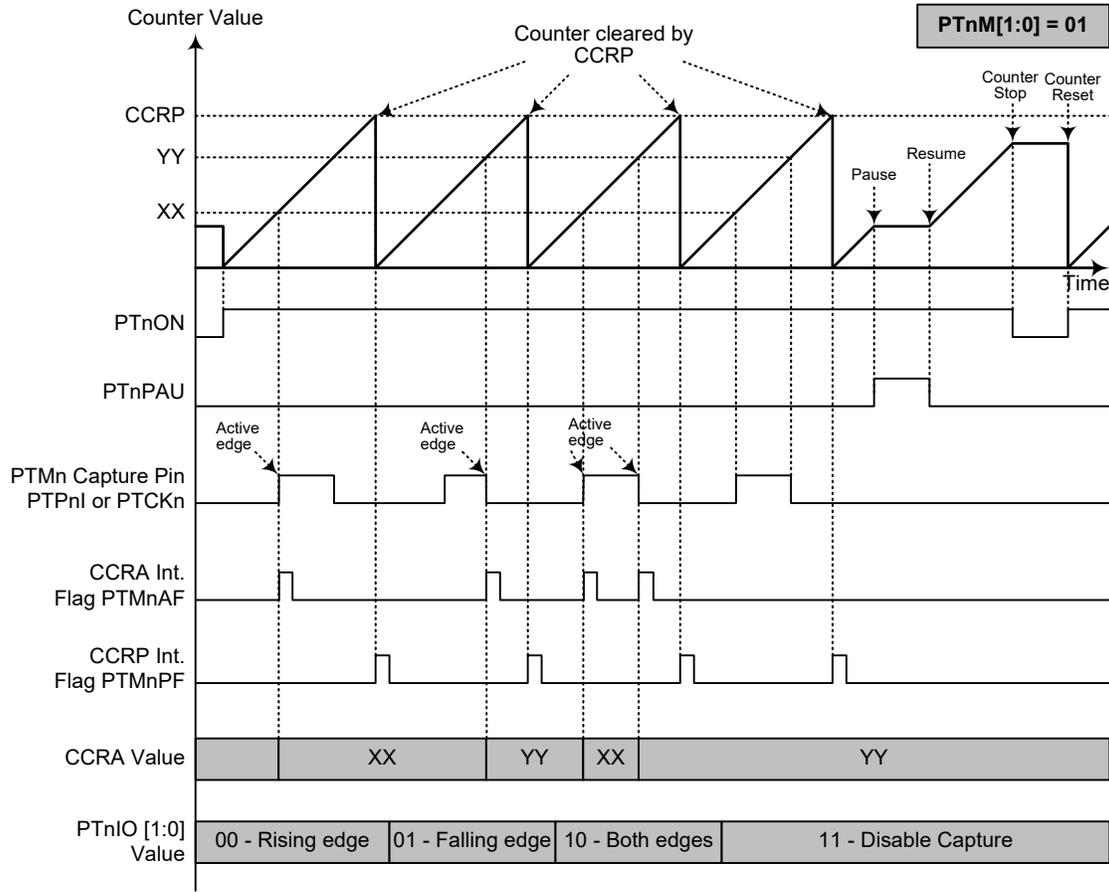
- Note: 1. PTnM[1:0]=01, PTnTCLR[1:0]=11 and active edge set by the PTnIO[1:0] bits
2. A PTMn Capture input pin active edge transfers the counter value to CCRA or CCRB
3. Comparator P match or PTMn capture input pin rising or falling edge will clear the counter
4. PTnCCLR bit is not used
5. No output function, PTnOC and PTnPOL bits not used
6. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero

Capture Input Mode (PTM1~PTM3)

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 PTPnI or PTCKn pin which is selected using the PTnCPTS bit in the PTMnC1 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 PTPnI or PTCKn pin the present value in the counter will be latched into the CCRA registers and a PTMn interrupt generated. Irrespective of what events occur on the PTPnI or PTCKn 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 PTMn 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 PTPnI or PTCKn 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 PTPnI or PTCKn pin, however it must be noted that the counter will continue to run.

As the PTPnI or PTCKn pin is pin shared with other functions, care must be taken if the PTMn is in the Capture Input 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 (n=1~3)

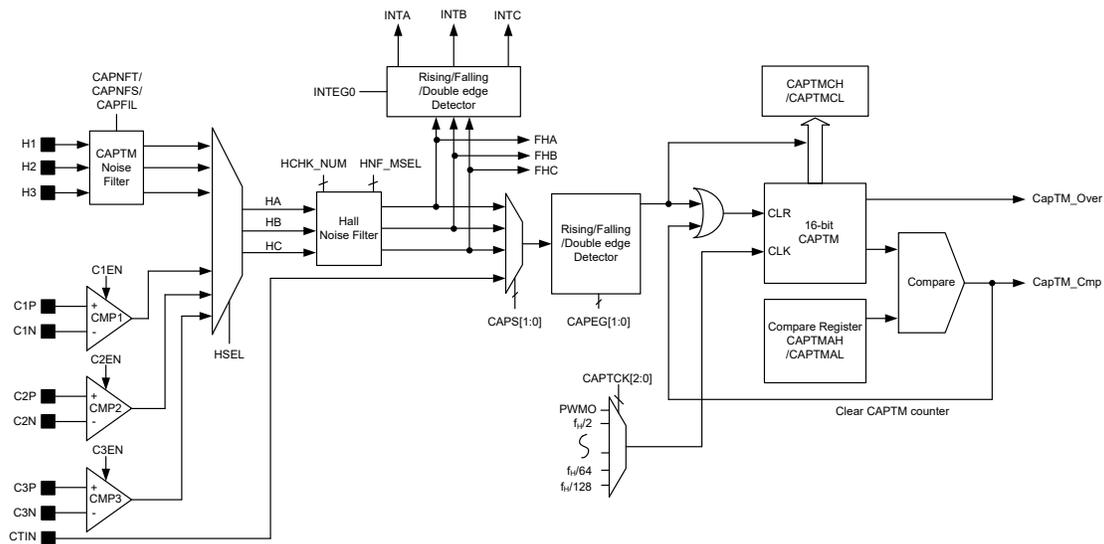
- Note: 1. PTnM[1:0]=01 and active edge set by the PTnIO[1:0] bits
 2. A PTMn Capture input pin active edge transfers the counter value to CCRA
 3. PTnCCLR bit not used
 4. No output function – PTnOC and PTnPOL bits are not used
 5. CCRP determines the counter value and the counter has a maximum count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

Capture Timer Overview

At the core of the Capture Timer Module 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 PWMO. 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 occurs a signal will be generated to reset the internal counter. The counter can also be cleared when a capture trigger is generated by one of four sources, FHA, FHB, FHC and CTIN.



Note: The detailed control and input selection for the Hall noise filter is described in the Hall Sensor Noise Filter section.

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. Another 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.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CAPTC0	CAPTPAU	CAPTCK2	CAPTCK1	CAPTCK0	CAPTON	—	CAPS1	CAPS0
CAPTC1	CAPEG1	CAPEG0	CAPEN	CAPNFT	CAPNFS	CAPFIL	CAPCLR	CAMCLR
CAPTMLD	D7	D6	D5	D4	D3	D2	D1	D0
CAPTMDDH	D15	D14	D13	D12	D11	D10	D9	D8
CAPTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CAPTMALH	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 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 **CAPTCK2~CAPTCK0**: Select CAPTM Counter clock
 000: PWMO periodic signal
 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 clock.
- 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**: CAPTM Capture Source selection
 00: FHA
 01: FHB
 10: FHC
 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**: CAPTM capture active edge selection
 00: CAPTM capture disabled
 01: Rising edge capture
 10: Falling edge capture
 11: Dual edges capture
- Bit 5 **CAPEN**: CAPTM capture input control
 0: Disable
 1: Enable

- Bit 4 **CAPNFT**: CAPTM Noise Filter sampling times definition
 0: 2 times
 1: 4 times
 The CAPTM Noise Filter circuit requires sampling the signal twice or 4 times continuously, when the sampled signals are all the same, the signal will be acknowledged. The sample frequency is decided by the CAPNFS bit.
- Bit 3 **CAPNFS**: CAPTM Noise Filter clock source selection
 0: t_H
 1: $4 \times t_H$
 The clock source for the Capture Timer Module Counter is provided by f_H or $f_H/4$.
- Bit 2 **CAPFIL**: CAPTM capture input noise filter control
 0: Disable
 1: Enable
- Bit 1 **CAPCLR**: CAPTM Counter capture auto-reset control
 0: Disable
 1: Enable
 If this bit is set high, when FHA/FHB/FHC/CTIN generates the required capture edge, the hardware will automatically transfer the value in the CAPTMDL and CAPTMDH register to the capture register pair CAPTMCL and CAPTMCH, and then reset the CAPTM counter.
- Bit 0 **CAMCLR**: CAPTM Counter compare match auto-reset control
 0: Disable
 1: Enable
 If this bit is set high, when a compare match condition occurs, the hardware will automatically reset the CAPTM counter.

• **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 **D7~D0**: 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 **D15~D8**: 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 **D7~D0**: 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 **D15~D8**: CAPTM Compare High Byte Register bit 7 ~ bit 0
CAPTM 16-bit Compare Register bit 15 ~ bit 8

• **CAPTACL 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 **D7~D0**: CAPTM Capture Low Byte Register bit 7 ~ bit 0
CAPTM 16-bit Capture Register bit 7 ~ bit 0

• **CAPTACH 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

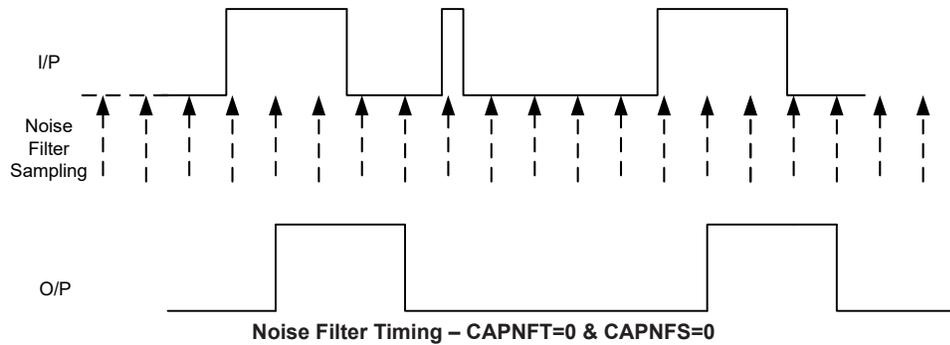
Bit 7~0 **D15~D8**: CAPTM Capture High Byte Register bit 7 ~ bit 0
CAPTM 16-bit Capture Register bit 15 ~ bit 8

Capture Timer Operation

The Capture Timer is used to detect and measure input signal pulse widths and periods. It operates in Capture or Compare Mode. There are four timer capture inputs, FHA, FHB, FHC and CTIN. Each of these capture inputs has its own edge detector selection, it can be either a rising edge, a falling edge or both rising and falling edges.

The CAPTON bit is used to control the overall Capture Timer module enable/disable. 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 capture timer also includes an internal noise filter which is used to filter out unwanted glitches or pulses on the H1, H2 and H3 input pin. 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 recognise an edge as a valid capture event. The sampling 2 or 4 time units are based on either t_H or $4 \times t_H$ determined using the CAPNFS bit.



Capture Mode Operation

The capture timer module contains two capture registers, CAPTMCL and CAPTMCH, which are used to store the present value in the counter. When the Capture Module is enabled, then each capture input source receives a valid trigger signal, the content of the free running counter-up 16-bit counter, which is contained in the CAPTMDL and CAPTMDH registers, will be transferred into the capture registers, CAPTMCL and CAPTMCH. When this occurs, the CAPOF interrupt flag bit in the interrupt control register will be set. As the capture interrupt is contained within a Multi-function Interrupt, if this interrupt is enabled by setting the interrupt enable bit CAPOE and the global interrupt, its associated multi-function interrupt are enabled, 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 compare value in these compare registers, the CAPCF interrupt flag bit in the interrupt control register will be set. If the CAMCLR bit is set high, then the counter will be reset to zero automatically when a compare match condition occurs.

For motor applications, 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.

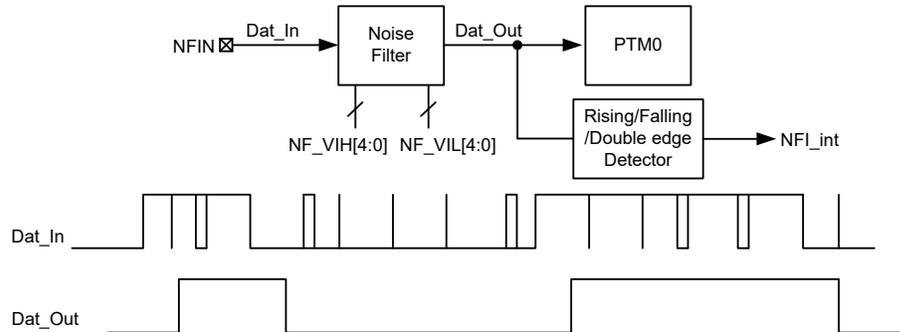
Noise Filter

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 outputs to the 16-bit PTM0 capture circuit in order to ensure that the motor control circuit works normally.

The noise filter circuit is an I/O surge filtering analog circuit which can filter micro-second grade sharp-noise.

Antinoise pulse width maximum: $(NF_VIH[4:0]-NF_VIL[4:0]) \times (1/f_{sys}) \times 100$

Where $(NF_VIH[4:0]-NF_VIL[4:0]) > 1$



Noise Filter Register 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 control

- 0: Noise Filter used
- 1: Noise Filter Bypass, Dat_Out=Dat_In

Bit 6 **CINS**: PTM0 capture input source selection

- 0: Noise Filter Dat_Out not selected (remains original PTM0 capture path)
- 1: Noise Filter Dat_Out selected

Bit 5 Unimplemented, read as “0”

Bit 4~0 **D4~D0**: NF_VIH[4:0] data

• 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 Unimplemented, read as “0”

Bit 4~0 **D4~D0**: NF_VIL[4:0] data

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.

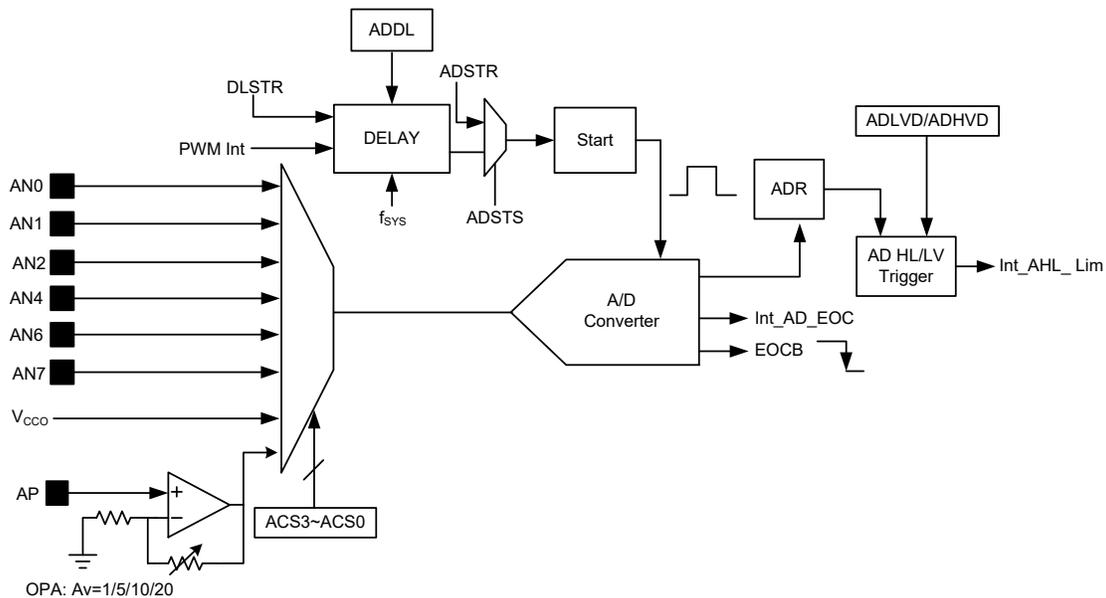
A/D Converter Overview

This device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into 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. It also can convert the voltage detection circuit output, V_{CC0} , into a 12-bit digital value. More detailed information about the A/D input signal is described in the “A/D Converter Control Registers” section.

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 Signals
6+2	ACS3~ACS0	AN0~AN2, AN4, AN6~AN7; OPAO, V_{CC0}

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 a series of registers. A read only register pair, ADRH and ADRL, exists to store the A/D converter data 12-bit value. Two register pairs, ADLVDH/ADLVDL and ADHVDH/ADHVDL, are used to store the boundary limit values of the A/D interrupt trigger. 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 (ADRF=0)	D3	D2	D1	D0	—	—	—	—
ADRL (ADRF=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADRH (ADRF=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADRH (ADRF=1)	—	—	—	—	D11	D10	D9	D8
ADLVDL (ADRF=0)	D3	D2	D1	D0	—	—	—	—
ADLVDL (ADRF=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADLVDH (ADRF=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADLVDH (ADRF=1)	—	—	—	—	D11	D10	D9	D8
ADHVDL (ADRF=0)	D3	D2	D1	D0	—	—	—	—
ADHVDL (ADRF=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADHVDH (ADRF=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADHVDH (ADRF=1)	—	—	—	—	D11	D10	D9	D8
ADCR0	ADSTR	EOCB	ADOFF	ADRF	ACS3	ACS2	ACS1	ACS0
ADCR1	ADSTS	DLSTR	PWIS	ADCHVE	ADCLVE	ADCK2	ADCK1	ADCK0
ADCR2	—	—	—	—	—	—	PWDIS1	PWDIS0
ADDL	D7	D6	D5	D4	D3	D2	D1	D0

A/D Converter Register List

A/D Converter Data Registers

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.

ADRF	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

To control the function and operation of the A/D converter, several 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 A/D converter input channel. As the device contains only one actual analog to digital converter hardware circuit, at a time only one of the external or internal analog signal inputs can be routed to the converter. It is the function of the ACS3~ACS0 bits to determine which analog channel input pins, V_{CCO} or OPA output signal is actually connected to the internal A/D converter.

The ADDL register is used to store the A/D conversion start delay time if the Delay trigger circuit is selected.

The relevant pin-shared function selection bits determine which pins on I/O Ports are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

• ADCR0 Register

Bit	7	6	5	4	3	2	1	0
Name	ADSTR	EOCB	ADOFF	ADRF5	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
 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.
- Bit 6 EOCB:** End of A/D conversion flag
 0: A/D conversion end
 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 is high.
- Bit 5 ADOFF:** A/D converter module power on/off control
 0: A/D converter module power on
 1: A/D converter 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. It is recommended to set ADOFF bit high before entering IDLE/SLEEP Mode for saving power.
- Bit 4 ADRF5:** A/D converter 12-bit data format select
 0: ADRH/ADLVDH/ADHVDH=D[11:4]; ADRL/ADLVDL/ADHVDL=D[3:0]
 1: ADRH/ADLVDH/ADHVDH=D[11:8]; ADRL/ADLVDL/ADHVDL=D[7:0]
 This bit controls the format of the 12-bit A/D converted data or boundary limit data of the A/D interrupt trigger in the corresponding high byte and low byte registers. Details are respectively provided in the A/D data registers and A/D boundary registers sections.

- Bit 3~0 **ACS3~ACS0:** A/D converter channel selection
 0000: AN0
 0001: AN1
 0010: AN2
 0011: Undefined
 0100: AN4
 0101: Voltage detection circuit output, V_{CCO}
 0110: OPA output
 0111: AN6
 1000: AN7
 1001~1111: Undefined

• **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:** A/D conversion start trigger circuit selection
 0: Select ADSTR bit trigger circuit
 1: Select DELAY start
- Bit 6 **DLSTR:** DELAY start function control
 0: Disable
 1: Enable
- Bit 5 **PWIS:** PWM Module interrupt source selection
 0: Select PWM period match interrupt (PWMP_Int)
 1: Select PWM duty match interrupt (PWMD0~2_Int)
- Bit 4~3 **ADCHVE~ADCLVE:** Configuration A/D converter interrupt trigger condition
 00: $ADLVD[11:0] < ADR[11:0] < ADHVD[11:0]$
 01: $ADR[11:0] \leq ADLVD[11:0]$
 10: $ADR[11:0] \geq ADHVD[11:0]$
 11: $ADR[11:0] \leq ADLVD[11:0]$ or $ADR[11:0] \geq ADHVD[11:0]$
- Bit 2~0 **ADCK2~ADCK0:** A/D converter clock source select
 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:** PWMn duty match interrupt source selection when PWIS=1
 00: PWM0 duty match interrupt to trigger PWM interrupt
 01: PWM1 duty match interrupt to trigger PWM interrupt
 10: PWM2 duty match interrupt to trigger PWM interrupt
 11: Reserved (select PWM2 duty match interrupt to trigger PWM interrupt)

• **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 **D7~D0**: Control code for A/D converter DELAY circuit delay time (count based on system clock)

A/D Converter Boundary Registers

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

ADRF5	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

ADRF5	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 Converter 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 to start the A/D converter. When the microcontroller 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 the PWM duty interrupt signal, selected using the PWIS bit. If selects PWM duty interrupt signal, the actual PWM duty interrupt trigger source can be selected by the PWDIS1 and PWDIS0 bits 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 delay time being setup using the ADDL register. The actual delay time is calculated by the ADDL register content multiplied by the system clock period. The delay time 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 swithing by the motor drive tansistors. 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 cleared to zero by the microcontroller after an A/D 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 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 clock f_{SYS} and by bits ADCK2~ADCK0, there are some limitations on the maximum and minimum A/D clock source speeds that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK} , is from 0.8 μ s to 12.8 μ s, care must be taken for system clock frequencies. Refer to the following table for examples, where values marked with an asterisk * show where, special care must be taken, as the values may exceed the specified A/D Clock Period range.

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	1.6 μ s	3.2 μ s	6.4 μ s	12.8 μ s	Undefined
10MHz	100ns *	200ns*	400ns*	800ns	1.6 μ s	3.2 μ s	6.4 μ s	Undefined
20MHz	50ns *	100ns *	200ns*	400ns*	800ns	1.6 μ s	3.2 μ s	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 cleared to power on the A/D converter. When the ADOFF bit is cleared to power on the A/D converter internal circuitry, even if no pins are selected for use as A/D inputs, 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, ADHVDH/ADHVDL and ADLVDH/ADLVDL contain preset values which can be compared with the converted values in ADRH/ADRL registers. Various types of comparisons can be made as defined by the ADCLVE and ADCHVE bits and an interrupt will be 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

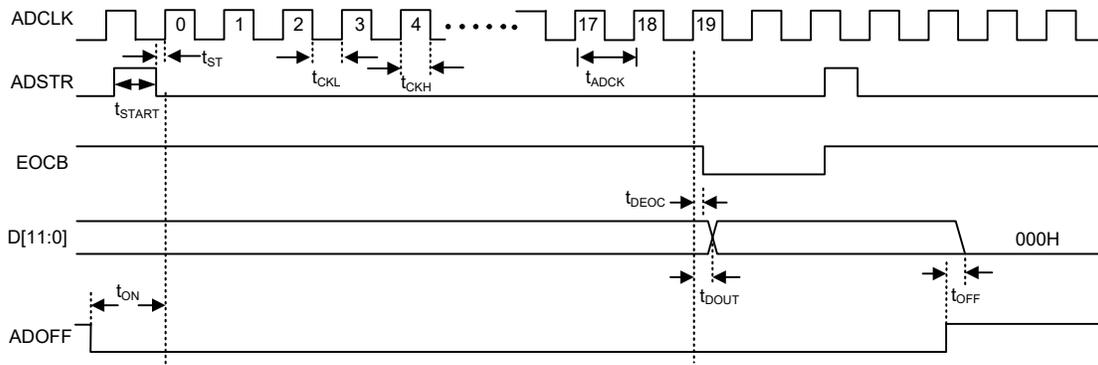
The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

- 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 converter module by clearing the ADOFF bit in the ADCR0 register to zero.
- Step 3
Select which signal is to be connected to the internal A/D converter by correctly configuring the ACS3~ACS0 bits which are also in the ADCR0 register.
- Step 4
Select which pin is to be used as A/D input and configure it by correctly programming the corresponding bit in the pin-shared control register.

- Step 5
 Select which trigger circuit is to be used to start an A/D conversion by correctly programming the ADSTS bit in the ADCR1 register.
- Step 6
 Select A/D converter output data format by setting the ADRFS bit in the ADCR0 register.
- Step 7
 If A/D conversion interrupt is 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, the A/D converter interrupt control bit, AEOCE, and its corresponding multi-function interrupt control bit must all be set high.
- Step 8
 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 circuit, the DELAY start function should be enabled by setting the DLSTR bit in the ADCR1 register and setup the ADDL register to achieve a delay time.
- Step 9
 To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR0 register can be polled. If A/D conversion is in progress, the EOCB flag will be set high. After the A/D conversion process is complete, the EOCB flag will go low and then the output data can be read from ADRH and ADRL registers to obtain the conversion value. As an alternative method, if the interrupt 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 Conversion Function

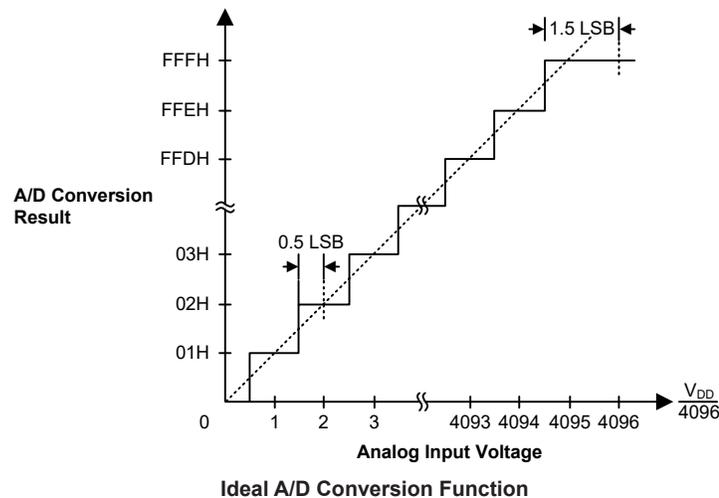
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to V_{DD} , this gives a single bit analog input value of V_{DD} divided by 4096.

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

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

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

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



A/D Conversion 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 A/D converter interrupt
mov a,03H
mov ADCR1,a         ; select fsys/8 as A/D clock
clr ADOFF
mov a,80h           ; setup PBPS0 to configure pin AN0
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
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
```

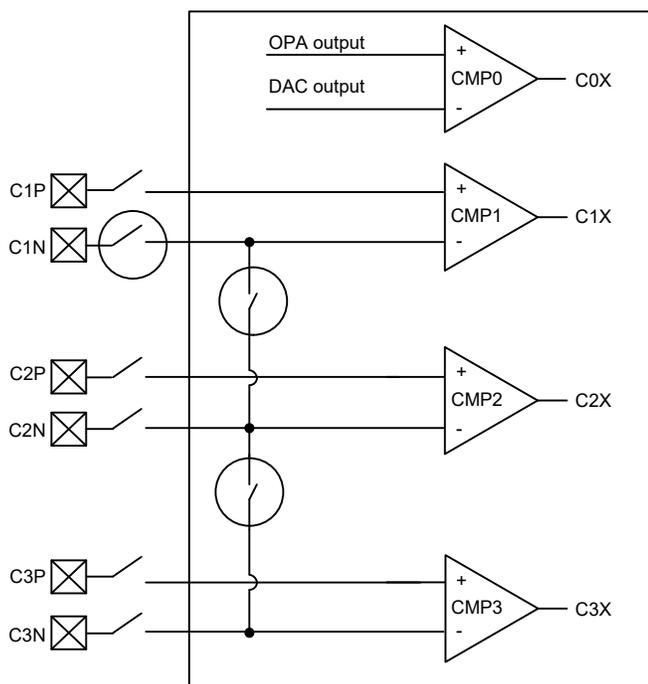
Example: using the interrupt method to detect the end of conversion

```
clr AEOCE          ; disable A/D interrupt
mov a,03H
mov ADCR1,a        ; select fsys/8 as A/D clock
clr ADOFF
mov a,80h          ; setup PBPS0 to configure pin AN0
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 A/D interrupt request flag
set AEOCE          ; enable A/D interrupt
clr MF1F           ; clear Multi-function interrupt 1 request flag
set MF1E           ; enable Multi-function interrupt 1
set EMI            ; enable global interrupt
:
:
; A/D 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
```

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.

As the comparator pins are pin-shared with other functions, the comparator functional pins must first be setup using relevant pin-shared function selection register described in the Pin-shared Function section.



Note: When the PBS13~PBS12 bits in the PBPS1 register are configured to 10B to select the CPN pin function, the switches circled in the block diagram will be turned on, in which case C1N, C2N and C3N is shorted internally and connected to the CPN pin.

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. 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.

Comparator Register

The CMPC control register controls the hysteresis functions and on/off control of the four comparators. The CnHYEN bit is the comparator n 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. The CnEN bit is the comparator n 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.

As the comparator 0 is used in the over current detector, there will be more description about it in the associated chapter.

• CMPC 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	1	1	1	1	0	0	0	0

Bit 7 **C3HYEN**: Comparator 3 Hysteresis control
 0: Off
 1: On

Bit 6 **C2HYEN**: Comparator 2 Hysteresis control
 0: Off
 1: On

Bit 5 **C1HYEN**: Comparator 1 Hysteresis control
 0: Off
 1: On

Bit 4 **C0HYEN**: Comparator 0 Hysteresis control
 0: Off
 1: On

Bit 3 **C3EN**: Comparator 3 On/Off control
 0: Off
 1: On

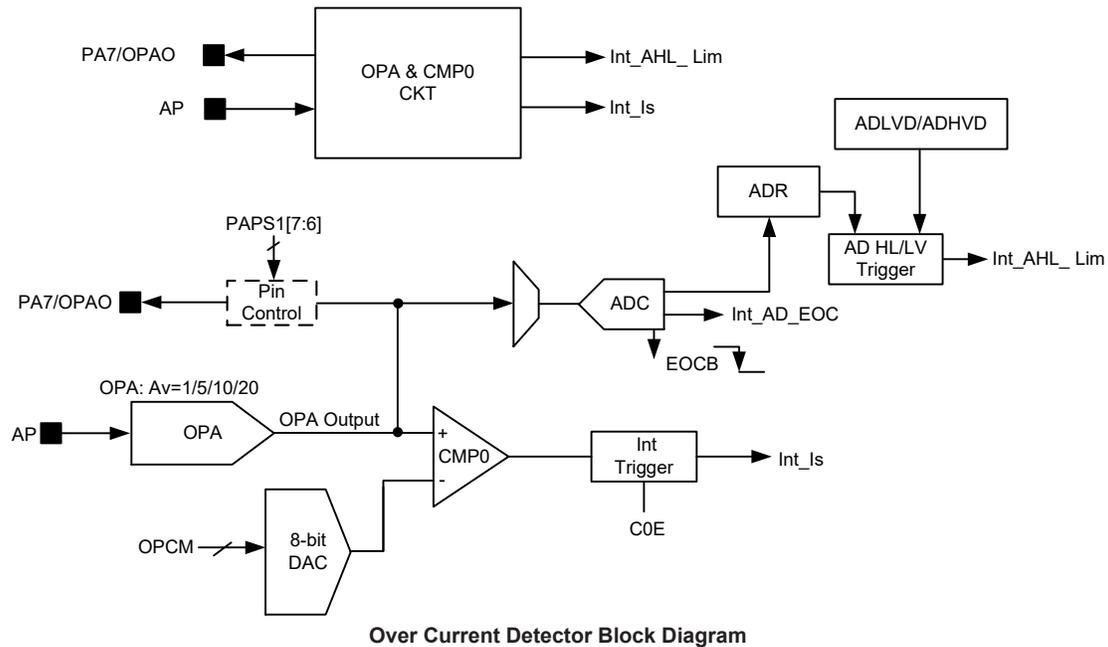
Bit 2 **C2EN**: Comparator 2 On/Off control
 0: Off
 1: On

Bit 1 **C1EN**: Comparator 1 On/Off control
 0: Off
 1: On

Bit 0 **C0EN**: Comparator 0 On/Off control
 0: Off
 1: On

Over Current Detection

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



Over Current Detect Functional Description

The over current detection function for motor protection can be achieved through a set of circuits which include an operational amplifier OPA, an A/D Converter, an 8-bit D/A Converter and comparator 0. 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 can be generated used for current detection.

1. A/D Converter compare result interrupt – Int_AHL_Lim
2. Comparator 0 interrupt – Int_Is

Over Current Detect Registers

There are four registers to control the function and operation of the over current detection circuits, known as OPOMS, OPCM, OPACAL and CMPC. These 8-bit registers define functions such as comparator interrupt edge control, comparator enable/disable and hysteresis function control, OPA operation mode selection and OPA calibration. The OPCM register is the 8-bit D/A Converter output control register used for OPAO 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
CMPC	C3HYEN	C2HYEN	C1HYEN	C0HYEN	C3EN	C2EN	C1EN	C0EN

Over Current Detection 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**: Interrupt edge control for Comparator 0
 00: Comparator 0 and D/A converter both disabled
 01: Rising edge trigger
 10: Falling edge trigger
 11: Dual edge trigger

Bit 5~3 Unimplemented, read as “0”

Bit 2~0 **OPAVS2~OPAVS0**: OPA gain selection
 000: OPA disabled
 001: Av=5
 010: Av=10
 011: Av=20
 100~110: Undefined
 111: Av=1

Note that when the OPA is used, the corresponding pin-shared control bit should be properly configured to enable the AP pin function. It is also suggested for users to turn off the pull-high resistor connected to this pin by setting the PAPU register.

• **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 **D7~D0**: 8-bit D/A converter control code bit 7 ~ bit 0
 The D/A Converter can be enabled by setting the CMP0_EG[1:0] bits to any value except “00”, so the two bits should be properly set before writing into the OPCM register, to make sure the D/A converter can output successfully.

• **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**: Reference input selection for comparator offset calibration
 0: Inverting input of comparator
 1: Non-inverting input of comparator

Bit 5 **AOFM**: Normal or Calibration Mode selection
 0: Normal Mode
 1: Offset Calibration Mode

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

• **CMPC 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	1	1	1	1	0	0	0	0

- Bit 7 C3HYEN:** Comparator 3 Hysteresis control
 0: Off
 1: On
 This is the comparator 3 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 comparator 2 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 comparator 1 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 comparator 0 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 3 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 2 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 1 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 0 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.

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 which has an adjustable PWM Duty can be used 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 using both the high side, GAT/GBT/GCT and the low side, GAB/GBB/GCB, of the external Gate Driver Transistor Pairs under software control.

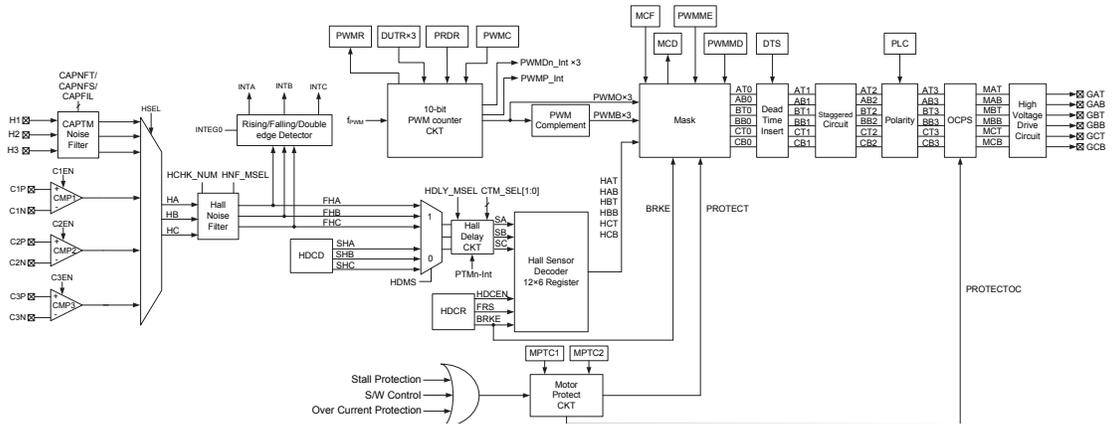
The Dead-Time Insertion circuit is used to ensure the high and low sides of a Gate Driver Transistor will not be turned on at the same time to prevent a virtual power short circuit. The dead time should be configured in the range of 0.3 μ s~5 μ s by related register under software control.

The Staggered circuit can force all the outputs to an off status if an error condition occurs which could be due to external factors such as ESD problems or external Gate Driver Transistor high and low sides both being on simultaneously.

The Polarity circuit can select the output polarity of the output signals to support several driving signal combinations for different external gate drive circuit.

The Motor Protection circuit includes many detection circuits for a motor stall condition or over current condition. If any of the above described situations is detected then the motor external drive circuit can be switched off immediately to prevent damage to the motor.

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 running functions are controlled by the HDCC/HDCR registers. The HA/HB/HC or SHA/SHB/SHC signals can be selected as the Hall Sensor Decoder circuit inputs.

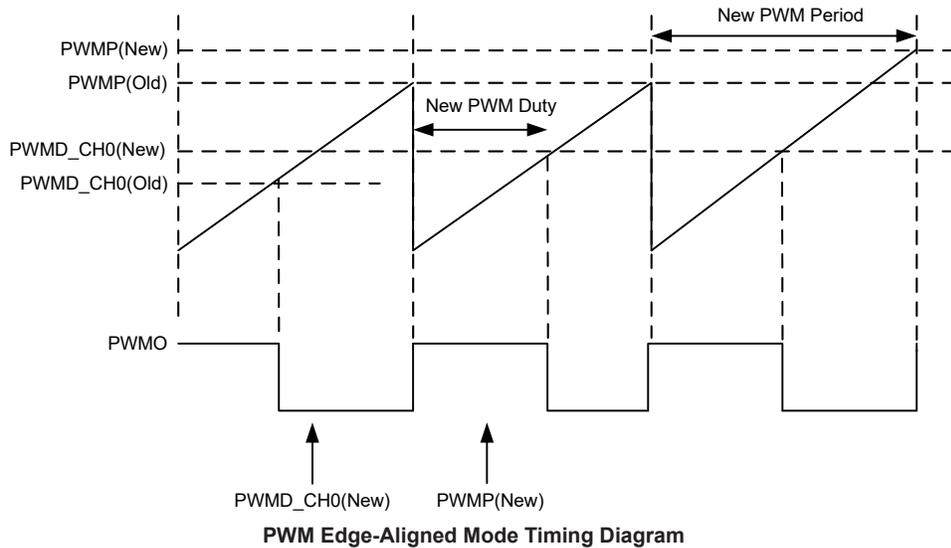
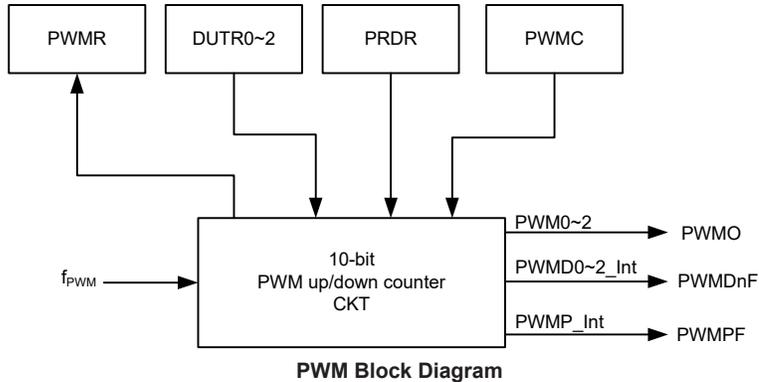


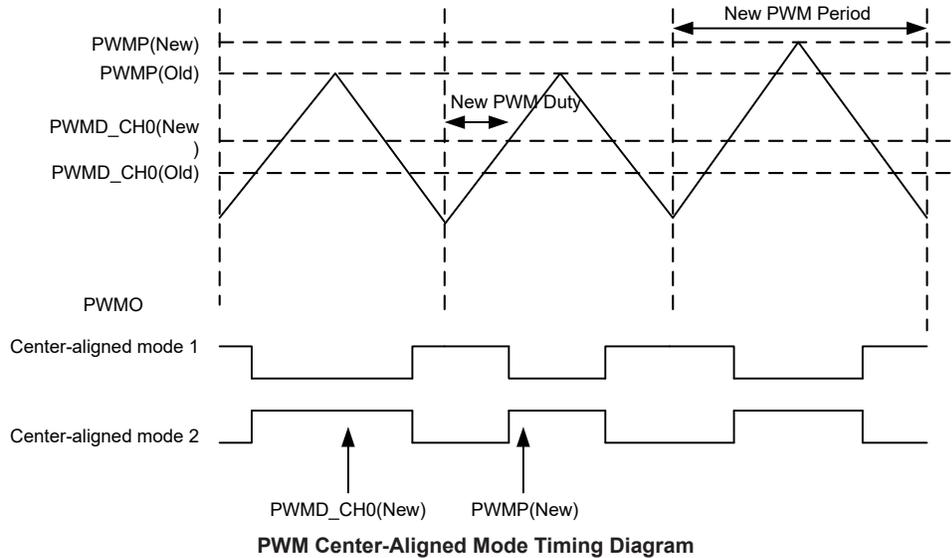
Note: GAT, GAB, GBT, GBB, GCT, GCB=PWM0H, PWM0L, PWM1H, PWM1L, PWM2H, PWM2L.

BLDC Motor Control Block Diagram

PWM Counter Control Circuit

The BLDC Motor control circuit includes a 10-bit PWM generator. The duty cycle and frequency of the PWM signal can be adjusted by programming 10-bit data into the corresponding PWM duty and period control registers.





PWM Duty Synchronous Update Modes

In high speed BLDC applications, using PWM interrupt to update the duty may result in asynchronous update for the three PWM duty values. This will generate undesired PWM duty outputs and lead to control errors. For this problem, two methods are provided for synchronous update of three PWM duty values.

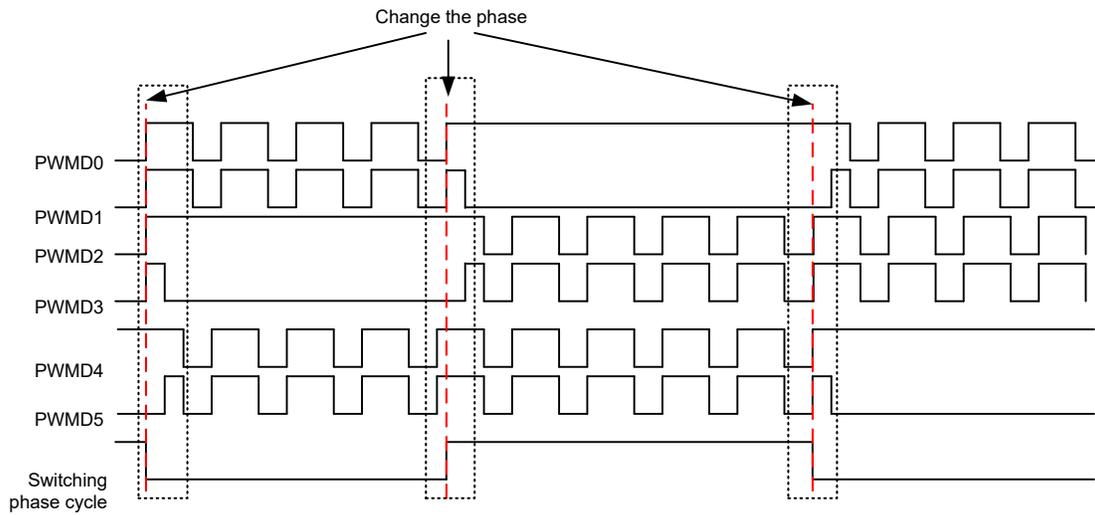
- DUTR0~DUTR2 PWM duty outputs are same
 Usually the three PWM duty outputs are same for square wave control. By setting the PWMSV bit high, the data written to the DUTR0H and DUTR0L registers will also be synchronously loaded to DUTR1H/DUTR1L and DUTR2H/DUTR2L. In this way the PWM duty synchronous update is implemented with reduced instructions.
- DUTR0~DUTR2 PWM duty outputs are not same
 If the three PWM duty outputs which require to be updated synchronously are not the same, set the PWMSU bit high to enable the PWM DUTR0~DUTR2 duty synchronous update function. When the PWM DUTR0~DUTR2 duty synchronous update request flag PWMSUF is set high, the hardware will synchronously update DUTR0, DUTR1 and DUTR2 values, after which the request flag will be automatically cleared to zero.

PWM Register Description

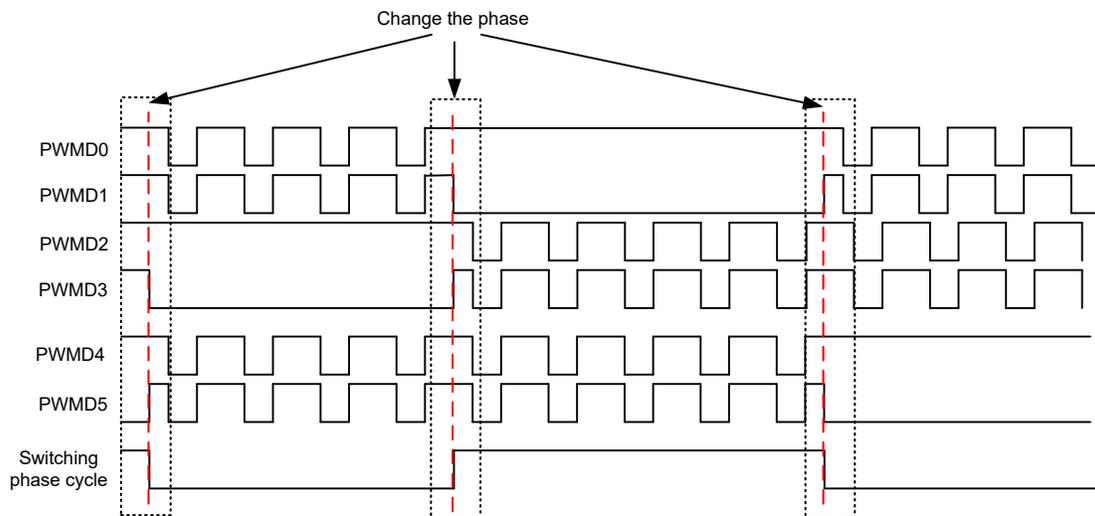
Overall PWM operation is controlled by a series of registers. The DUTRnL/DUTRnH register pair is used for PWM duty control for adjustment of the motor output power. The PRDRL/PRDRH register pair are used together to form a 10-bit value to setup the PWM period for PWM frequency adjustment. Being able to change the PWM frequency is useful for motor characteristic matching to reduce problems such as noise interference and resonance. The PWMRL/PWMRH registers are used to monitor the PWM counter dynamically. The PWMON bit in the PWMC register is on/off control bit for the 10-bit PWM counter. The PWM counter clock source can be selected by PCKS1~PCKS0 bits in the PWMC register. The PWMMS[1:0] bit field in the PWMC register determines the PWM alignment type, which can be either edge or center type.

The PWMCS register is used for PWM DUTR0~DUTR2 duty value synchronisation update control and the PWM output state quickly update control when the PWM phase changes. When the PWMMEDO bit is zero, if the PWM phase change occurs, the timer will continue to count until the

PWM period is complete, and then output to the next phase. However, when the PWMMEDO bit is high, the PWM will output to the next phase immediately once the phase is changed. The timings are shown as follows.



Change the Phase Timing – PWMMEDO=0



Change the Phase Timing – PWMMEDO=1

The DUTRn and PRDR registers are writable only not readable, the following steps show the write procedures:

- Writing Data to DUTRn or PRDR
 - ♦ Step 1. Write data to High Byte DUTRnH or PRDRH
 - Note that here data is only written to the 2-bit buffer.
 - ♦ Step 2. Write data to Low Byte DUTRnL or PRDRL
 - Here data is written directly to the Low byte registers and simultaneously data is latched from the 2-bit buffer to the High Byte registers.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PWMC	PWMS1	PWMS0	PCKS1	PCKS0	PWMON	ITCMS1	ITCMS0	PWMLD
PWMC	—	—	—	—	PWMMEDO	PWMSUF	PWMSU	PWMSV
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	PWMS1	PWMS0	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 **PWMS1~PWMS0**: PWM alignment mode selection

- 00: Edge-aligned mode
- 01: Edge-aligned mode
- 10: Center-aligned mode 1
- 11: Center-aligned mode 2

Bit 5~4 **PCKS1~PCKS0**: PWM counter clock source selection (based on $f_{PWM}=20\text{MHz}$)

- 00: f_{PWM} , PWM frequency (Min.)=20kHz
- 01: $f_{PWM}/2$, PWM frequency (Min.)=10kHz
- 10: $f_{PWM}/4$, PWM frequency (Min.)=5kHz
- 11: $f_{PWM}/8$, PWM frequency (Min.)=2.5kHz

Bit 3 **PWMON**: PWM circuit on/off control

- 0: Off
- 1: On

This bit controls the on/off of the overall PWM circuit. 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**: PWM center-aligned mode duty interrupt control

- 00: Disable center-aligned mode duty interrupt
- 01: Center-aligned mode duty interrupt only in count up condition
- 10: Center-aligned mode duty interrupt only in count down condition
- 11: Center-aligned mode duty interrupt both in count up and down conditions

Bit 0 **PWMLD**: PWM PRDR and DUTRn (n=0~2) register data update control

- 0: The register values of PRDR and DUTRn (n=0~2) are never loaded to counter and comparator registers
- 1: The PRDR register value will be loaded to counter register after counter underflow, and hardware will clear it by next clock cycle

• **PWMCS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PWMMEDO	PWMSUF	PWMSU	PWMSV
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

- Bit 7~4 Unimplemented, read as “0”
- Bit 3 **PWMMEDO**: PWM output state quickly update control bit when PWM change the phase
0: Disable
1: Enable
- Bit 2 **PWMSUF**: PWM DUTR0~DUTR2 duty synchronous update request flag (when PWMSU=1)
0: No request
1: DUTR0~DUTR2 duty synchronous update request
Setting this bit high will request to update DUTR0~DUTR2 duty data synchronously. When synchronous update has finished, this bit will be cleared to zero by hardware.
- Bit 1 **PWMSU**: PWM DUTR0~DUTR2 duty synchronous update control
0: Disable
1: Enable
- Bit 0 **PWMSV**: Synchronize DUTR0 content to DUTR1 and DUTR2 Control
0: Disable, DUTR0~DUTR2 should be configured separately
1: Enable, the PWM duty value written to DUTR0 is synchronised to DUTR1 and DUTR2

• **DUTRnL Register (n=0~2)**

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

- Bit 7~0 **D7~D0**: 10-bit PWMn Duty Low Byte Register bit 7 ~ bit 0
10-bit DUTRn Register bit 7 ~ bit 0

• **DUTRnH Register (n=0~2)**

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

- Bit 7~2 Unimplemented, read as “0”
- Bit 1~0 **D9~D8**: 10-bit PWMn Duty High Byte Register bit 1 ~ bit 0
10-bit DUTRn 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	W	W	W	W	W	W	W	W
POR	0	0	0	0	0	0	0	0

- Bit 7~0 **D7~D0**: 10-bit PWM Period Low Byte Register bit 7 ~ bit 0
10-bit PRDR Register bit 7 ~ bit 0

• **PRDRH Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **D9~D8**: 10-bit PWM Period High Byte Register bit 1 ~ bit 0
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 **D7~D0**: 10-bit PWM Counter Low Byte Register bit 7 ~ bit 0
10-bit PWM Counter bit 7 ~ bit 0

• **PWMRH Register**

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

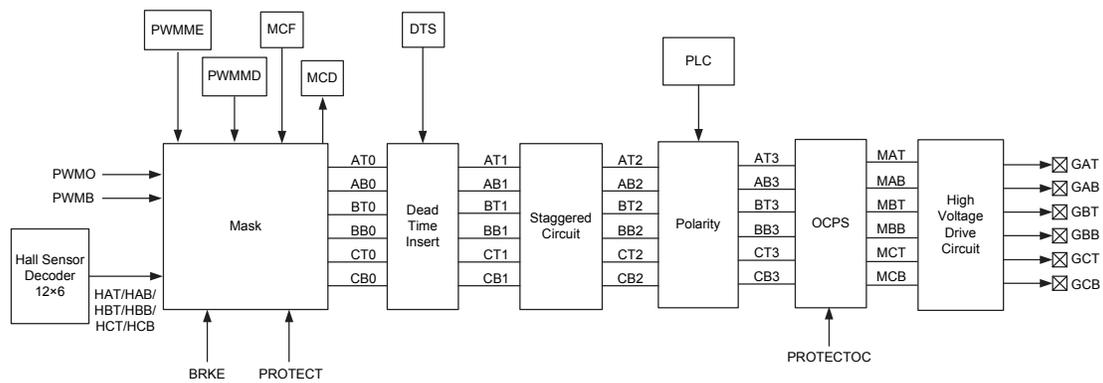
Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **D9~D8**: 10-bit PWM Counter High Byte Register bit 1 ~ bit 0
10-bit PWM Counter bit 9 ~ bit 8

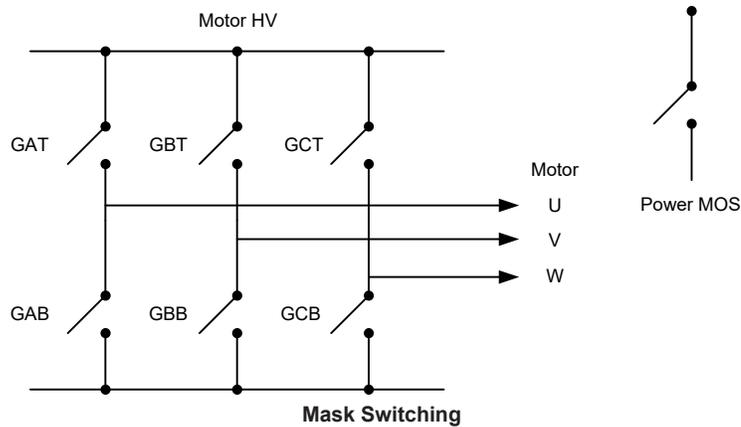
Mask Function

The device includes a Mask function for Motor control to provide its control flexibility.

The internal mask circuit has three operating modes, which are known as the Normal Mode, Brake Mode and Motor Protect Mode. The Normal Mode has two sub-modes, Hardware Mask Mode and Software Mask Mode.



Mask Function Block Diagram



Normal Mode

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

- When PWMS=0, Transistor pair low side GAB/GBB/GCB will output PWM.
- When PWMS=1, Transistor pair high side GAT/GBT/GCT will output PWM.
- 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.
- When MSKMS=0, Hardware Mask Mode is selected.
- When MSKMS=1, Software Mask Mode is selected.

• MCF Register

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

- Bit 7 **MSKMS**: Mask Mode selection
0: Hardware Mask Mode
1: Software Mask Mode
- Bit 6~4 Unimplemented, read as “0”
- Bit 3 **MPWMS**: Hardware Mask Mode PWM output selection
0: Complementary output
1: Non-complementary output
This bit selection is invalid when in the Software Mask Mode where the PWM mode selection is determined by the PWMME register.
- Bit 2 **MPWE**: PWM output control
0: PWM output disable (AT0/BT0/CT0/AB0/BB0/CB0 can not output PWM)
1: PWM output enable
- Bit 1 **FMOS**: Mask output control when PROTECT is triggered
0: AT0/BT0/CT0=0, AB0/BB0/CB0=0
1: AT0/BT0/CT0=0, AB0/BB0/CB0=1

Bit 0 **PWMS**: High sides or low sides output PWM selection for Hardware Mask mode
 0: Low sides output PWM
 1: High sides output PWM

• **MCD Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	MAT	MAB	MBT	FHC	FHB	FHA
R/W	—	—	R	R	R	R	R	R
POR	—	—	0	0	0	x	x	x

“x”: unknown

Bit 7~6 Unimplemented, read as “0”

Bit 5~3 **MAT/MAB/MBT**: Gate driver output

Bit 2~0 **FHA/FHB/FHC**: HA/HB/HC filtered outputs via Hall Noise Filter

These signals are derived from the HA/HB/HC signals and filtered by the Hall Noise Filter.

Hardware Mask Mode

Complementary control, MPWMS=0

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

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

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

Non-complementary control, MPWMS=1

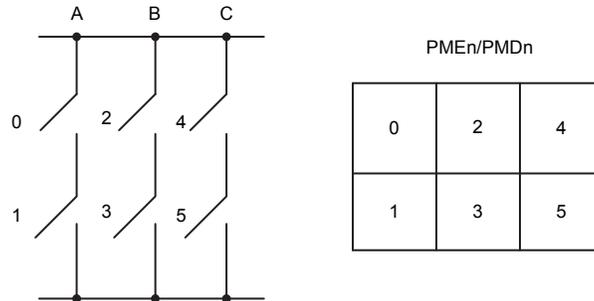
PWMS=0	HAT	HAB	AT0	AB0	PWMS=1	HAT	HAB	AT0	AB0	
	0	0	0	0		0	0	0	0	0
	0	1	0	PWMO		0	1	0	1	1
	1	0	1	0		1	0	PWMO	0	0
	1	1	0	0		1	1	0	0	0

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

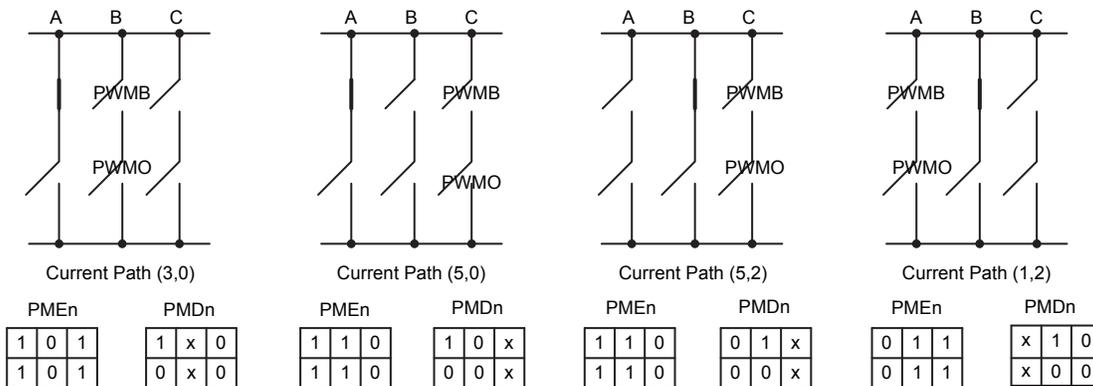
PWMS=0	HCT	HCB	CT0	CB0	PWMS=1	HCT	HCB	CT0	CB0	
	0	0	0	0		0	0	0	0	0
	0	1	0	PWMO		0	1	0	1	
	1	0	1	0		1	0	PWMO	0	
	1	1	0	0		1	1	0	0	

Software Mask Mode

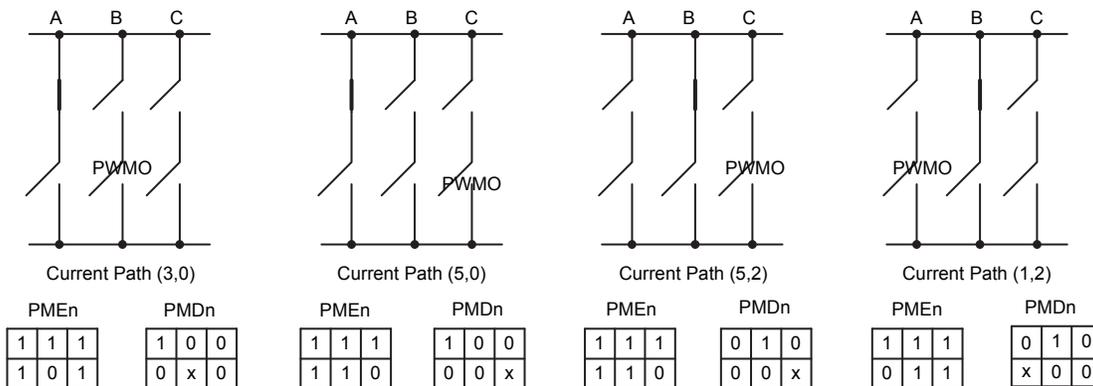
To control the software Mask circuit, two registers known as PWMME and PWMMD are provided. PWMME register is used to control PWM signal masked or not and PWMMD is used to determine the MOSFET Driver Circuit is on or off.



3-phase Inverter Symbol and Control Bits



Mask Complement Mode Examples



Mask Independent Mode Examples

- Note: 1. If the mask mode is enabled, when PWMxH and PWMxL are masked simultaneously, the two line data of each pair, PMD0 and PMD1, PMD2 and PMD3, PMD4 and PMD5, can not be both set to “1” simultaneously. If they are both set high in the same time, they will be forced to output “0” instead.
2. If PWM and complementary PWM are enabled simultaneously, one of the complementary channels outputs PWM and the other one can not be masked to “1” but outputs “0” automatically by hardware.
3. If the GxT and GxB (x=A, B or C) pins are configured as I/O function, then PWM Mask function will be invalid.

• **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 **PMEn**: PWM mask enable bit (n=0~5)

0: Disabled, PWM generator signal is output to next stage

1: Enabled, PWM generator signal is masked

The PWM generator signal will be masked when this bit is set high. Then the corresponding PWMn channel will output with the mask data PMDn.

• **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 **PMDn**: PWM mask data bit (n=0~5)

0: Output logic low

1: Output logic high

When the corresponding mask enable bit PMEn=1, this Mask data bit can control the output.

Brake Mode

The device provides a brake function. The brake mode has the highest priority. When this mode is activated, all the low sides of the external gate driver transistors are on and the high sides are turned off. When Brake mode is enabled, the truth table of the external gate driver transistor high/low side status are shown below.

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

Motor Protect Mode

When the motor PROTECT is activated, the external gate driver transistor pair can be forced into brake status, where the high sides are all off and the low sides are all on, or free running status where the high and low sides are all 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

Other Functions

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

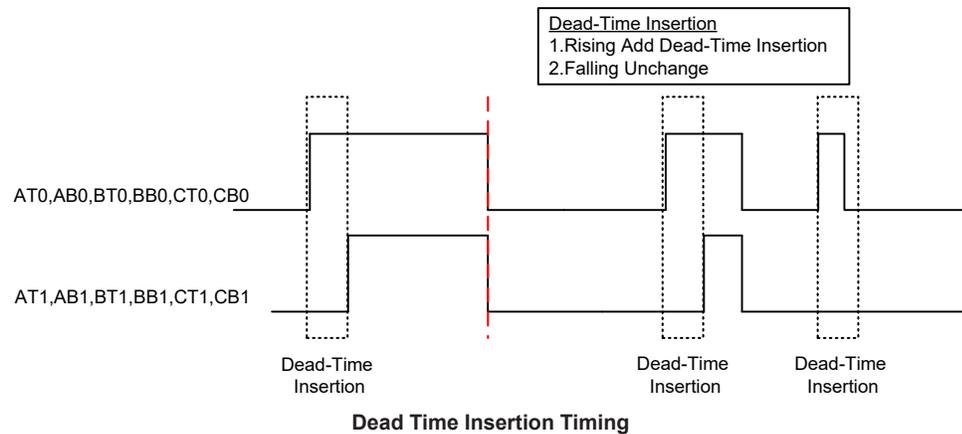
Dead Time Insertion Function

During a transistor pair switchover, a dead time should be introduced to prevent the transistor being turned on at the same time in both the high and low sides which will provide a direct short circuit. The actual dead time must be setup in the range of between 0.3 μ s and 5 μ s and is selected by the application program.

When the AT0/AB0/BT0/BB0/CT0/CB0 outputs at a rising edge, then a Dead Time is inserted.

When the AT0/AB0/BT0/BB0/CT0/CB0 outputs at a falling edge, then their outputs remain unchanged.

The Dead-Time Insertion Circuit is only used during motor control. The Dead Time function is controlled by DTS register.



• 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: $f_{DT} = f_H$ (based on $f_{SYS} = 20\text{MHz}$)

01: $f_{DT} = f_H / 2$

10: $f_{DT} = f_H / 4$

11: $f_{DT} = f_H / 8$

Bit 5 **DTE**: Dead Time function control

0: Disable

1: Enable

Bit 4~0 **D4~D0**: Dead Time Register bit 4 ~ bit 0

5-bit Dead Time counter.

Dead Time = $(DTS[4:0] + 1) / f_{DT}$

Staggered Function

The Staggered Function will force all output drive transistors to be off when a software error occurs or other errors 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

Note: 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 Control Function

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

• **PLC Register**

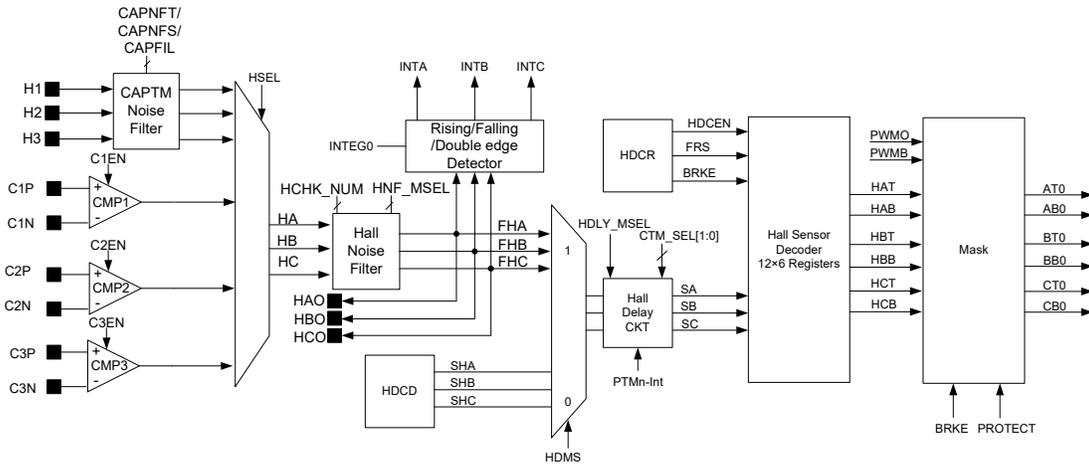
Bit	7	6	5	4	3	2	1	0
Name	—	—	PCBC	PCTC	PBBC	PBTC	PABC	PATC
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

- Bit 7~6 Unimplemented, read as “0”
- Bit 5 **PCBC**: GCB Output polarity Control
 0: Non-invert
 1: Invert
- Bit 4 **PCTC**: GCT Output polarity Control
 0: Non-invert
 1: Invert
- Bit 3 **PBBC**: GBB Output polarity Control
 0: Non-invert
 1: Invert
- Bit 2 **PBTC**: GBT Output polarity Control
 0: Non-invert
 1: Invert
- Bit 1 **PABC**: GAB Output polarity Control
 0: Non-invert
 1: Invert
- Bit 0 **PATC**: GAT Output polarity Control
 0: Non-invert output
 1: Invert output

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 motor direction and speed control.

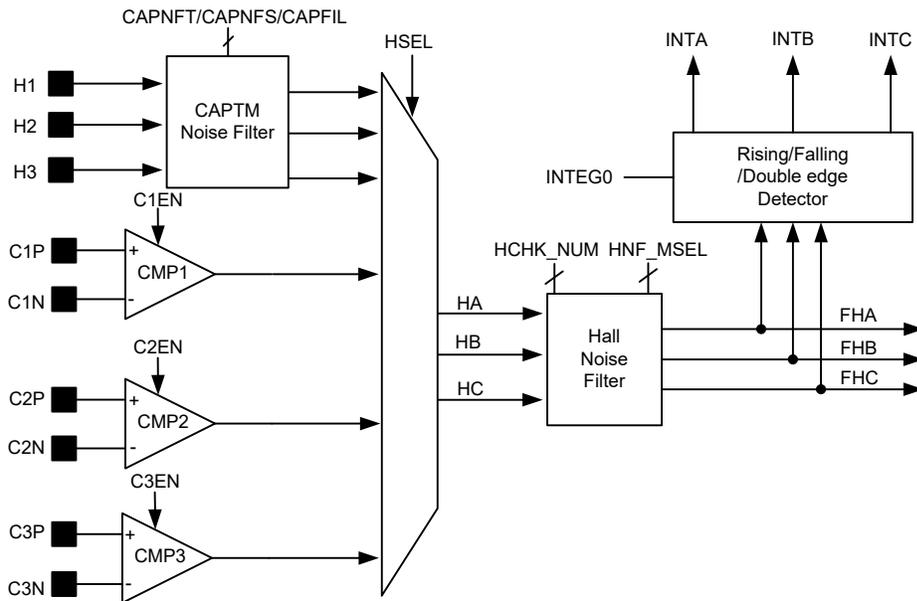


Hall Sensor Decoder Block Diagram

The Hall Sensor decoder input signals are selected by setting the HDMS bit high. If the HDMS bit is zero then SHA/SHB/SHC which are set in the HDCD register will be used instead of the actual Hall Sensor signals. If the HDMS bit is high, HA/HB/HC source is selected by the HSEL bit in the INTEG0 Register.

Hall Sensor Noise Filter

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



Note: The detailed control for the CAPTM noise filter is described in the Capture Timer Module section.

Hall Sensor Noise Filter Block 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 control for the noise filter.

HNF_EN bit	Status
0	Noise filter off – HA/HB/HC bypass the noise filter
1	Noise filter on

Hall Sensor Noise Filter Enable

The sampling frequency of the Hall noise filter is setup using the HFR_SEL [2:0] bits.

The HCK_N [4:0] bits in the HCHK_NUM register are used to setup the number of Hall Sensor input compare times.

$HCK_N [4:0] \times \text{Sampling space} = \text{Anti-noise ability} = \text{Hall Delay Time}$.

It should be noted that longer Hall delay time will result in worse 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 by programming the CTM_SEL1~CTM_SEL0 bits and set the selected TM to run in the Compare Match Output Mode.
- 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.

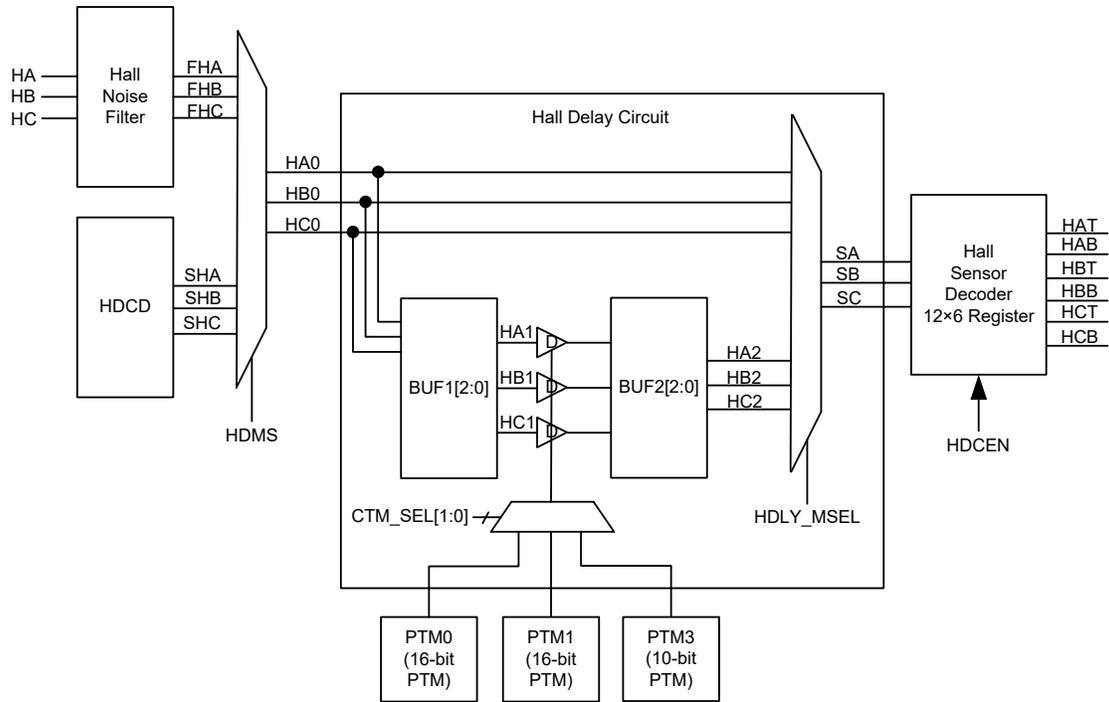
The following points should be noted regarding the HDLY_MSEL bit.

1. When this bit is low, BUF1[2:0] and BUF2[2:0] will be cleared to zero.
2. When this bit is low, PTM0, PTM1 and PTM2 remain their original TM functions.
3. When this bit is high, the TM which is selected by the Delay function will be dedicated for use by the Hall Delay circuit. In this case, the original TM functions will still remain active except for that the PTnON bit which will be controlled automatically by the hardware. And the selected PTM should be properly configured according to the requirement.

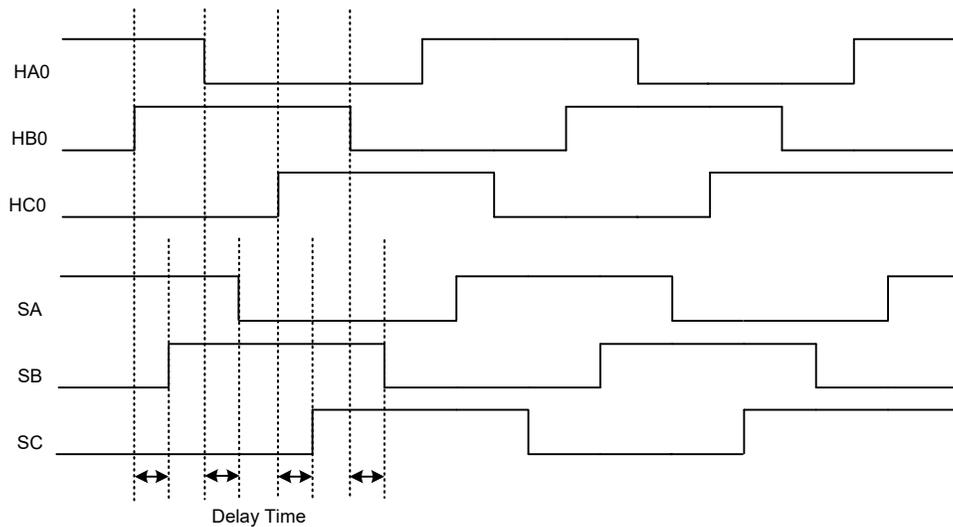
With regard to the selected PTM functions the following notes should be taken before the Delay function is enabled.

1. Remain PTnON=0 and PTnPAU=0.
2. The PTM should be set in the Compare Match Output Mode.
3. Set PTnCCLR=1, therefore the PTM counter is cleared with a comparator A match condition.
4. Setup the Delay time by properly setting the PTMn CCRA and selection its counter clock.

After the Delay function is enabled by setting the HDLY_MSEL bit from low to high, the Delay time must not be more than one step time of the Hall input, otherwise the output can not be anticipated and will drop out of step. One Hall input cycle includes six steps.



Delay Function Block Diagram



Delay Function Timing

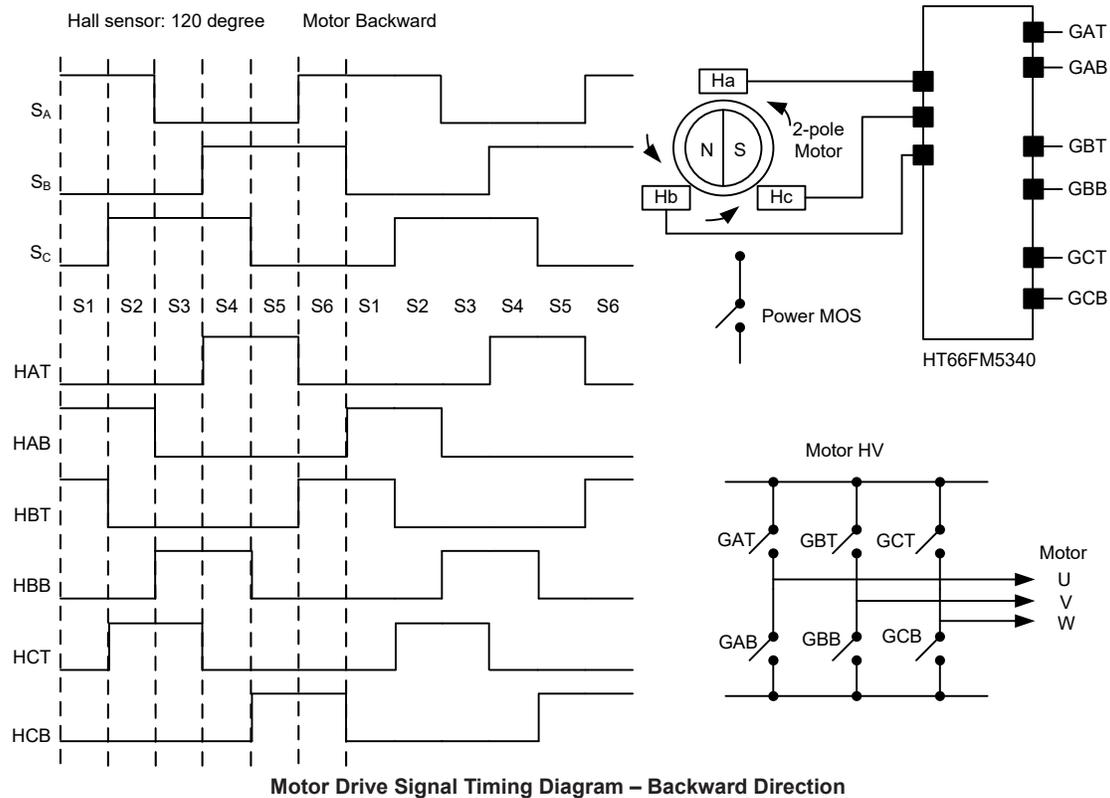
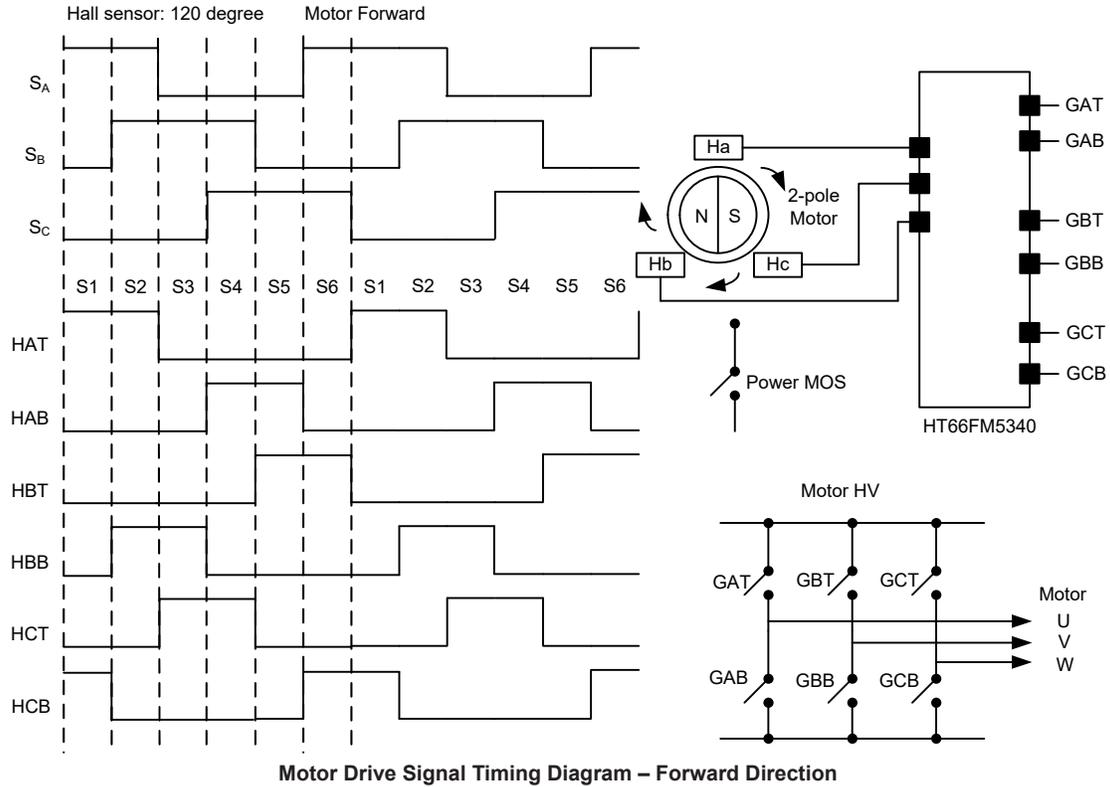
Motor Control Drive Signals

The direction of the BLDC motor is controlled using the HDCR and HDCD registers as well as a series of 6-bit HDCT registers, HDCT0~HDCT11. When using the Hall Sensor Decoder function, the direction can be determined using the FRS bit and the brake operation 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.

Forward (HDCEN=1, FRS=0, BRKE=0)	60 Degree			120 Degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
	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]						
Backword (HDCEN=1, FRS=1, BRKE=0)	60 Degree			120 Degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
	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]						
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
Hall Decoder Disable (HDCEN=0) Free Running	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 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

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



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 time number register and HNF_MSEL is the Hall Noise Filter Mode select register. The INTEG0 register is Hall Noise Filter input interrupt edge control 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 selection
 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**: Select TM used for Hall Delay Circuit
 00: PTM0 (16-bit PTM)
 01: PTM1 (16-bit PTM)
 10: PTM3 (10-bit PTM)
 11: Unused
- Bit 5 **HDLY_MSEL**: Hall Delay Circuit selection
 0: Original path (Bypass Delay Circuit)
 1: Select Hall Delay Circuit
- Bit 4 **HALS**: Hall Sensor angle configuration selection
 0: 60 degree
 1: 120 degree
- Bit 3 **HDMS**: Hall Sensor Decoding Mode selection
 0: Software Mode (SHA/SHB/SHC in the HDCD register)
 1: Hall Sensor Mode (FHA/FHB/FHC via noise filter)

- Bit 2 **BRKE**: Motor brake control
0: MAT/MBT/MCT/MAB/MBB/MCB=V
1: MAT/MBT/MCT=0, MAB/MBB/MCB=1
- Bit 1 **FRS**: Motor Forward/Backward selection
0: Forward
1: Backward
- Bit 0 **HDCEN**: Hall Sensor Decoder control
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**: Software Hall C
- Bit 1 **SHB**: Software Hall B
- Bit 0 **SHA**: Software 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**: HAT output state control
0: Output 0
1: Output 1
- Bit 4 **HABDn**: HAB output state control
0: Output 0
1: Output 1
- Bit 3 **HBTDn**: HBT output state control
0: Output 0
1: Output 1
- Bit 2 **HBBDn**: HBB output state control
0: Output 0
1: Output 1
- Bit 1 **HCTDn**: HCT output state control
0: Output 0
1: Output 1
- Bit 0 **HCBDn**: HCB output state control
0: Output 0
1: Output 1

• **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**: Number of Hall Noise Filter check times

• **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 control

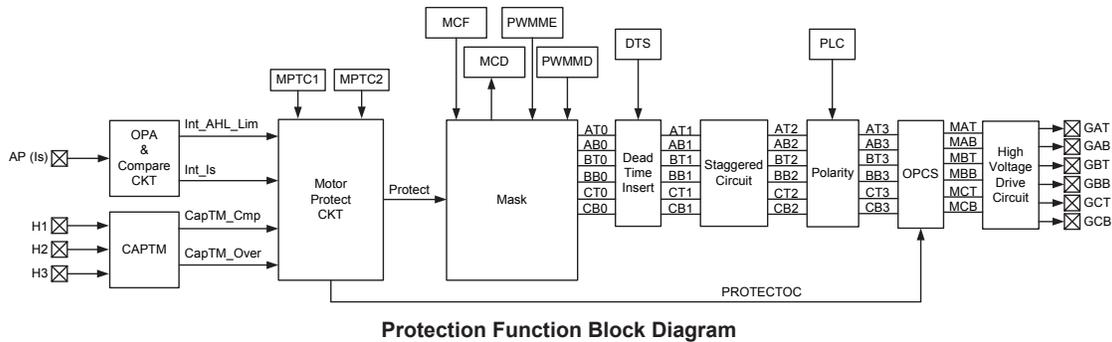
- 0: Disable (bypass)
- 1: Enable

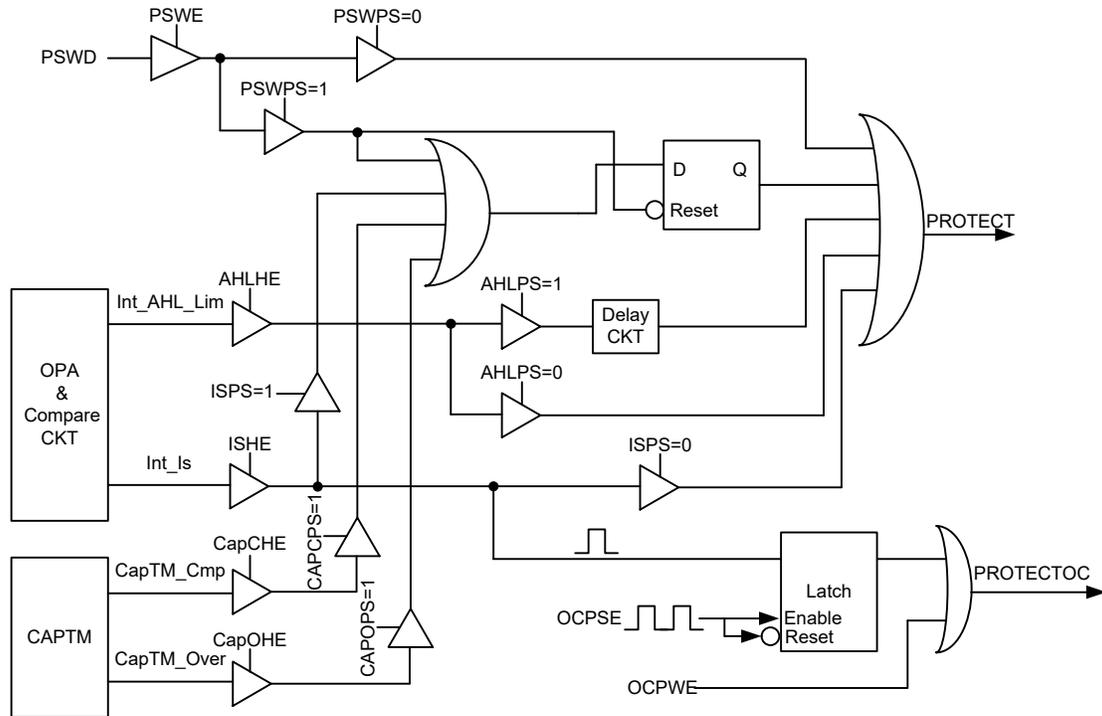
Bit 2~0 **HFR_SEL2~HFR_SEL0**: Hall Noise Filter clock source selection

- 000: $f_{ih}/2$
- 001: $f_{ih}/4$
- 010: $f_{ih}/8$
- 011: $f_{ih}/16$
- 100: $f_{ih}/32$
- 101: $f_{ih}/64$
- 110: $f_{ih}/128$
- 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 and motor stalling, etc., to reduce motor damage or for safety reasons. This device includes a range of protection and safety features.





Protection Function Control Logic

Motor Protection Function Description

The PROTECT mechanism provides three kinds of motor protection features to turn off motor immediately, 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 by software

When the motor protection circuit is on, i.e., PROTECT=1, the external Gate Drive transistor pair can be put into two different protection modes. The first is the Brake Mode where the high side is off and the low side is on, and the second is the Free Running Mode where both high and low sides are off. The FMOS bit in the MCF register determines which type is used.

The motor protection circuit can be triggered and released 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, the PROTECT signal is set by a trigger source event occurrence and the PROTECT signal is automatically released after the trigger source trigger event is resolved. For the Pause Mode, PROTECT signal setup is determined by a trigger source event occurrence while the PROTECT signal can only be released by software.

Additionally, the PROTECTOC mechanism is used to immediately switch off the driver signal output. If an over current condition was detected, immediately the preset OCP protection signals will be output ignoring the original signals to protect the power transistors from damage. The PROTECTOC signal can be setup by software or hardware trigger mechanism which is selected by programming the corresponding OCPSE (hardware) or OCPWE (software) bit in the MPTC1 register. Whatever the PROTECTOC signal triggered source was, it can only be released by software.

Current Protection Function

The device contains an internal OPA, a high speed 12-bit A/D Converter, an 8-bit D/A Converter and a comparator 0 to measure the motor current and to detect excessive current values. If an over current situation should occur, then the external drive circuit can be shut down immediately to prevent motor damage. More details are provided in the Over Current Detection chapter.

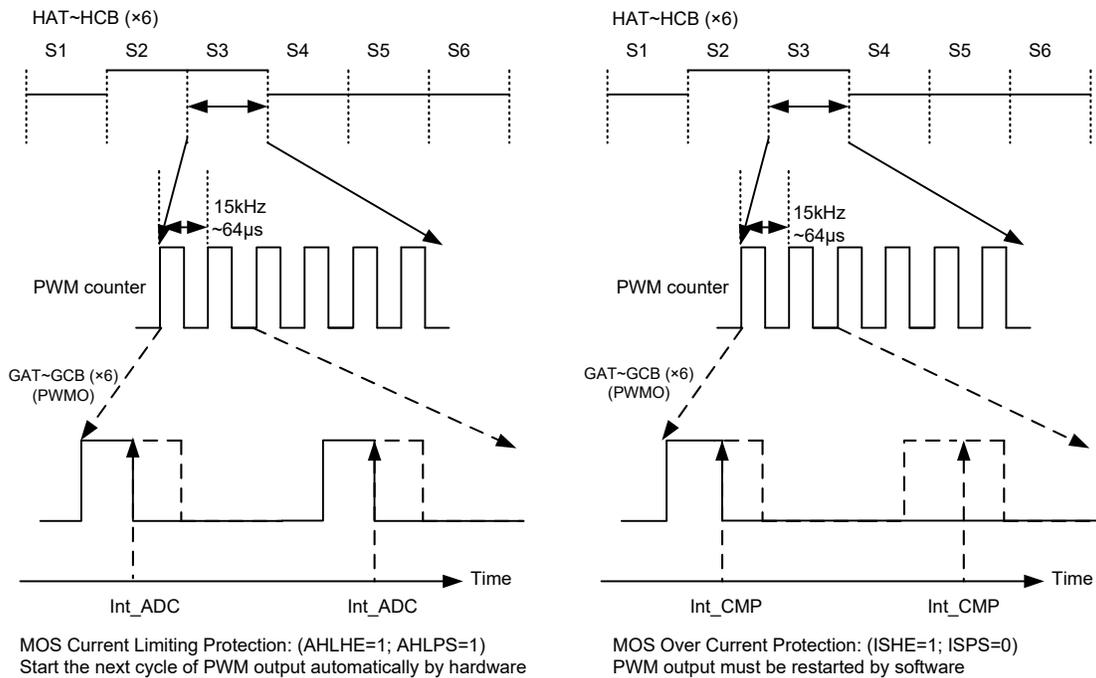
As the motor driver PCB will have rather large amounts of noise, and as this noise will be amplified by the OPA, this can easily lead to false triggering. For this reason the Fault Mode must be used.

For the MOS Limit current mechanism Int_AHL_Lim, when AHLHE=0 then the hardware protect mode is disabled, and when AHLHE=1 the hardware protect is enabled. The current limiting circuit is a hardware circuit, and the bit control can be valid when the A/D converter channel selected the operational amplifier output.

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

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

For the MOS over current mechanism Int_Is, when ISHE=0 the hardware trigger mode is disabled. When ISHE=1 the hardware trigger mode is enabled. When ISPS=0, the Fault Mode is selected, when ISPS=1, the Pause Mode is selected.



Current Protection Timing

The PROTECTOC signal can be triggered by software or hardware trigger mechanism which is selected by programming the corresponding OCPSE or OCPWE bit in the MPTC1 register.

OCPSE=1: H/W Direct Over Current trigger & S/W release

Ensure that the ISHE bit has been set to select the hardware Int_Is over current protection before setting the OCPSE bit high. Then the over current compare interrupt signal Int_Is can be used to directly switch off the drive signals. Since Int_Is is a pulse signal, the over current trigger signal must be latched. When the PROTECTOC signal is logic high, the drive signals at the polarity stage will be ignored and the over current protection logics in the OCPS register are used to immediately switch off drive signals to protect the power MOS. To release the PROTECTOC over current protection mechanism, set the OCPSE bit from low to high to trigger the software reset function thus pulling the PROTECTOC signal to low, after which the normal drive signals at the Polarity stage will be recovered.

OCPWE=1: S/W trigger & S/W release

It is a more immediate current protection mechanism. The PROTECTOC signal will be triggered directly by setting the OCPWE bit high. To release the over current protection, clear the OCPWE bit to reset the PROTECTOC signal, after which the normal Polarity drive signal will be recovered.

Motor Stall Detection Function

For 3-phase BLDC applications with Hall Sensors, the 16-bit CAPTM can be used to monitor the H1, H2 and H3 inputs for rotor speed detection. The software will setup the CAPTMAH and CAPTMAL registers to monitor the Hall sensor input pins H1, H2 and H3 for rotor speed control. If an abnormal situation exists, a CapTM_Cmp or CapTM_Over interrupt will be generated, which is described in the CAPTM section.

CapTM_Cmp Stall Detect Mechanism: when CapCHE=0 disable the hardware trigger mode, and when CapCHE=1 enable the hardware trigger mode. The stall detect mechanism must use the Pause Mode which is selected by setting the CAPCPS bit.

CapTM_Over Stall Detect Mechanism: when CapOHE=0 disable the hardware trigger mode, and when CapOHE=1 enable the hardware trigger Mode. CAPOPS=1, then select the Pause Mode.

Motor Protection Register Description

There are three registers, MPTC1, MPTC2 and OCPS, 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	OCPWE	OCPSE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PSWD:** Motor Protection Software Mode data
 0: PSWD=0
 1: PSWD=1

Bit 6 **PSWE:** Motor Protection Software Mode control
 0: Disable
 1: Enable

When the motor protection software mode has been enabled and triggered, it can be released by setting this bit from 1→0→1.

Bit 5 **CapOHE:** CapTM_Over Hardware trigger Mode control
 0: Disable
 1: Enable

- Bit 4 **CapCHE**: CapTM_Cmp Hardware trigger Mode control
 0: Disable
 1: Enable
- Bit 3 **ISHE**: Int_Is Hardware trigger Mode control
 0: Disable
 1: Enable
- Bit 2 **AHLHE**: Int_AHL_Lim Hardware trigger Mode control
 0: Disable
 1: Enable
- Bit 1 **OCPWE**: PROTECTOC software trigger & software release control
 0: Disable
 1: Enable
- Bit 0 **OCPSE**: PROTECTOC over current hardware trigger & software release control
 0: Disable
 1: Enable

When the PROTECTOC has been triggered, the PROTECTOC protection can only be released by setting this bit from high to low then to high.

• **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	—	—	—	0	0	0	0	0

- Bit 7~5 Unimplemented, read as “0”
- Bit 4 **PSWPS**: Pause/Fault Mode selection in Software Mode
 0: Fault Mode
 1: Pause Mode
- Bit 3 **AHLPS**: Int_AHL_Lim Mode selection
 0: Protection circuit allows immediate restart of PWM output when the Int_AHL_Lim interrupt has been reset.
 1: Protection circuit allows restart of PWM output on the next PWM period when the Int_AHL_Lim interrupt has been reset.
- Bit 2 **ISPS**: Int_Is Pause/Fault Mode selection
 0: Fault Mode
 1: Pause Mode
- Bit 1 **CAPCPS**: CapTM_Cmp Mode selection
 0: Undefined, cannot be selected
 1: Pause Mode
 Note if the CapTM_Cmp trigger protection is enabled, the mode must be selected as Pause Mode by setting the CAPCPS bit high.
- Bit 0 **CAPOPS**: CapTM_Over Pause Mode selection
 0: Undefined, cannot be selected
 1: Pause Mode
 Note if the CapTM_Over trigger protection is enabled, the mode must be selected as Pause Mode by setting the CAPOPS bit high.

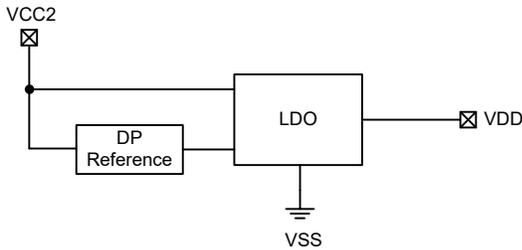
• **OCPS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	OCPCB	OCPCT	OCPBB	OCPBT	OCPAB	OCPAT
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 **OCPCB**: GCB over current protection output selection
 0: Output 0
 1: Output 1
- Bit 4 **OCPCT**: GCT over current protection output selection
 0: Output 0
 1: Output 1
- Bit 3 **OCPBB**: GBB over current protection output selection
 0: Output 0
 1: Output 1
- Bit 2 **OCPBT**: GBT over current protection output selection
 0: Output 0
 1: Output 1
- Bit 1 **OCPAB**: GAB over current protection output selection
 0: Output 0
 1: Output 1
- Bit 0 **OCPAT**: GAT over current protection output selection
 0: Output 0
 1: Output 1

Low Dropout Regulator – LDO

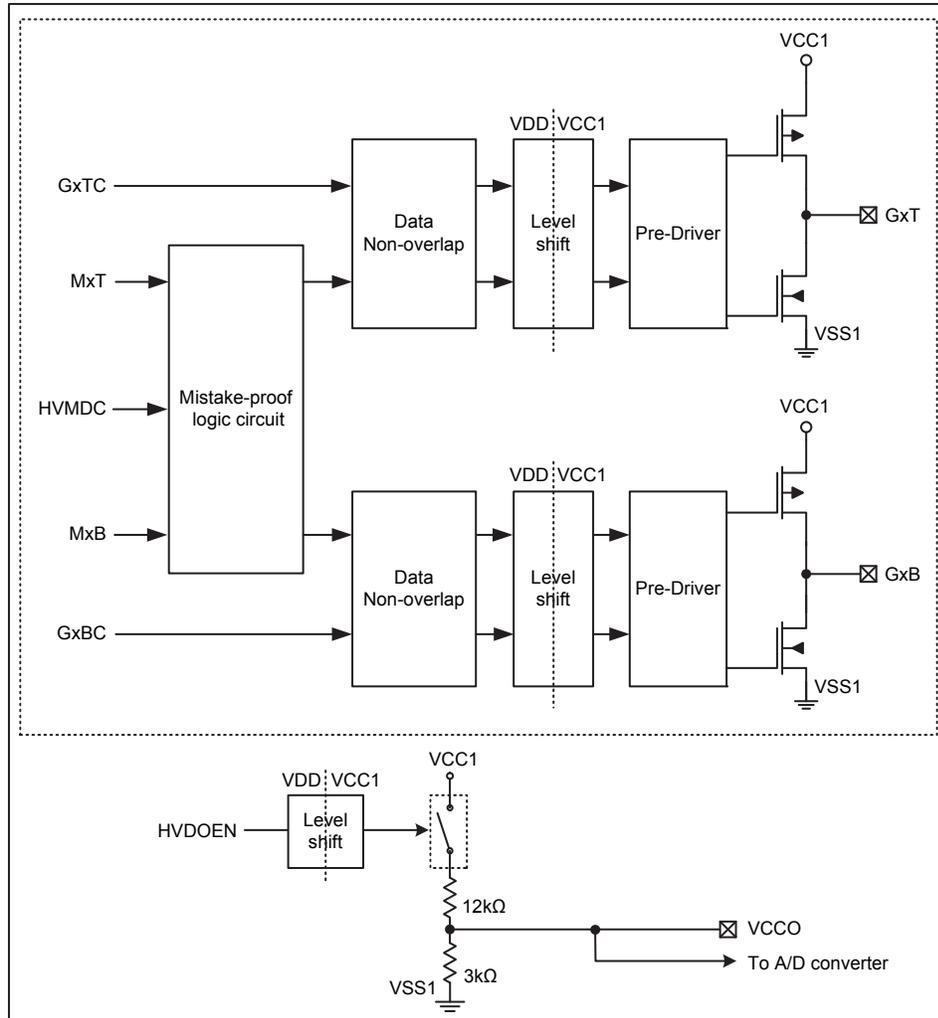
The device contains an internal Low Dropout Regulator, known as LDO. The LDO regulator can reduce a higher voltage approximately 5.1V to 15V on input pin VCC2 to a 5V level supplied on output pin VDD. When the LDO input voltage, V_{IN} , is ranged from 3V to 5.1V, the LDO output voltage, V_{OUT} , follows V_{IN} . This lower voltage level can provide a fixed power supply for internal or external circuits.



LDO Block Diagram

High Voltage I/O Ports

The device provides several 15V high voltage input/output lines, known as GxT and GxB, where “x” stands for A, B or C. These high voltage I/O ports can convert logic output signals to 15V voltage outputs.



High Voltage I/O Block Diagram

- Note: 1. The structure contained in the dash line is identical for group “x”, where “x” stands for A, B or C, and the structure contained in the solid line is shared by all HVIO lines.
 2. Refer to HVIO Output Mode Description section for the GxT and GxB truth table details.
 3. MAT/MAB, MBT/MBB and MCT/MCB signals come from the BLDC motor control circuit output.

High Voltage I/O Functional Description

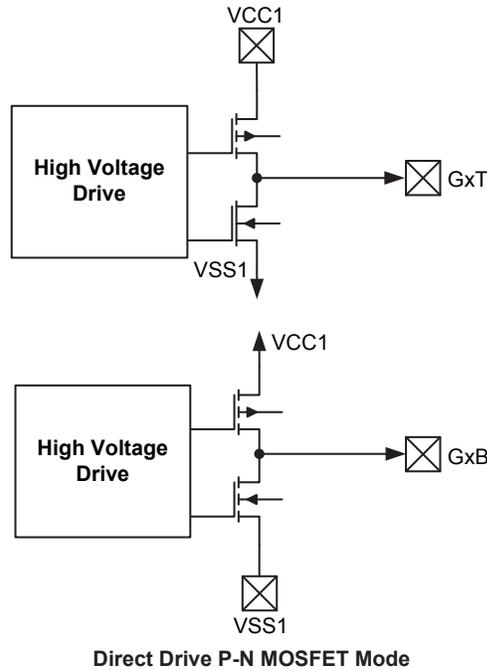
The High Voltage I/O driver circuit is composed of two sections, the pre-driver combinational circuit and a voltage detector circuit.

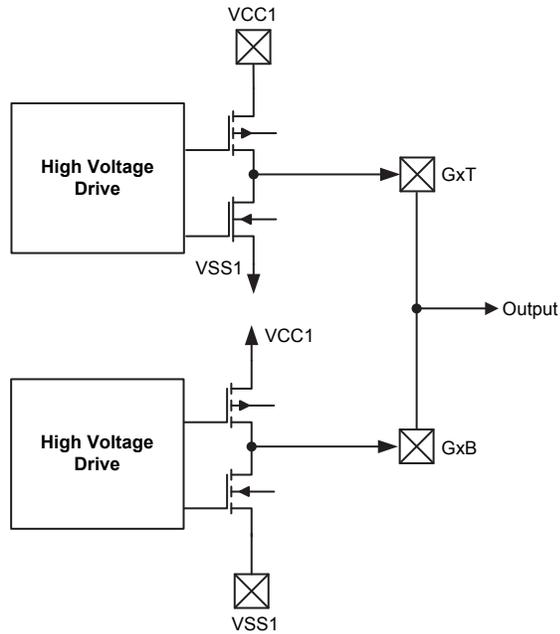
Pre-driver Combinational Circuit

The internal pre-driver combinational circuit consists of three groups of high voltage driver circuits. Each group is mainly composed of an ESD protection circuit, a level shift circuit, a mistake-proof logic circuit, a non-overlap circuit and a high voltage push-pull output circuit. The GxT pin can be used to drive the inverter's high side, P-MOSFET, while the GxB pin can be used to drive the inverter's low side, N-MOSFET.

High Voltage I/O Output Mode Description

The high voltage I/O output mode is selected using the HVMDM bit in the HVRC register. The device will directly drive the external P-N MOSFET when the HVMDM bit is zero. Otherwise, the device will directly drive the external motor when the HVMDM bit is high. The high voltage I/O output mode is shown in the following figures.





Direct Drive Motor MOS Mode

HVMDC=0: Direct Drive P-N MOSFET Mode

The GxT truth table is shown as follows:

GxTC	MxT	MxB	GxT Status
0	x	x	Floating
1	0	x	VCC1
1	1	0	VSS1
1	1	1	VCC1

“x”: Don't care

Note: 1. When the GxTC is zero, GxT pin will be floating state regardless of the MxT and MxB settings.

2. When the GxTC is high:

- (1) When the MxT is cleared to zero, GxT pin will drive to VCC1 regardless of the MxB setting.
- (2) When the MxT and MxB are both cleared to zero, GxT pin will drive to VSS1.
- (3) When the MxT and MxB are both set to high, GxT pin will drive to VCC1.

The GxB truth table is shown as follows:

GxBC	MxT	MxB	GxB Status
0	x	x	Floating
1	x	0	VSS1
1	0	1	VCC1
1	1	1	VSS1

“x”: Don't care

Note: 1. When the GxBC is zero, GxB pin will be floating state regardless of the MxT and MxB settings.

2. When the GxBC is high:

- (1) When the MxB is cleared to zero, GxB pin will drive to VSS1 regardless of the MxT setting.
- (2) When the MxB is set to high and xT is cleared to zero, GxB pin will drive to VCC1.
- (3) When the MxT and MxB are both set to high, GxT pin will drive to VSS1.

HVMDC=1: Direct Drive Motor MOS Mode

The GxT truth table is shown as follows:

GxTC	MxT	MxB	GxT Status
0	x	x	Floating
1	0	0	VSS1
1	0	1	Floating
1	1	0	Floating
1	1	1	VCC1

“x”: Don't care

Note: 1. When the GxTC is zero, GxT pin will be floating state regardless of the MxT and MxB settings.

2. When the GxTC is high:

- (1) When the MxT and MxB are both cleared to zero, GxT pin will drive to VSS1.
- (2) When the MxT and MxB are set to 10 or 01, GxT pin will be floating state.
- (3) When the MxT and MxB are both set to high, GxT pin will drive to VCC1.

The GxB truth table is shown as follows:

GxBC	MxT	MxB	GxB Status
0	x	x	Floating
1	0	0	VSS1
1	0	1	Floating
1	1	0	Floating
1	1	1	VCC1

“x”: Don't care

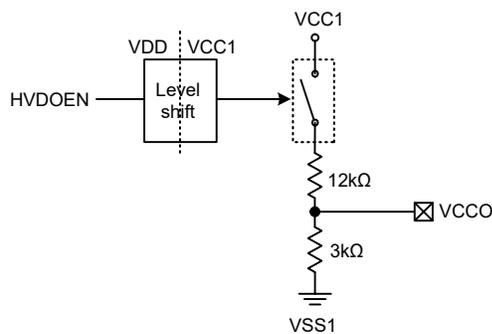
Note: 1. When the GxBC is zero, GxB pin will be floating state regardless of the MxT and MxB settings.

2. When the GxBC is high:

- (1) When the MxT and MxB are both cleared to zero, GxB pin will drive to VSS1.
- (2) When the MxB and MxT are set to 10 or 01, GxB pin will be floating state.
- (3) When the MxB and MxT are both set to high, GxT pin will drive to VCC1.

Voltage Detector Circuit

An internal voltage detector circuit is used to monitor the VCC1 voltage levels. This function enabled/disable is determined by the HVDOEN bit. Users can observe the voltage from the VCCO pin.



Voltage Detector Circuit

High Voltage I/O Registers

Overall operation of high voltage I/O ports is controlled using two registers. The GATEC register is used to control the GxB and GxT enable/disable. The HVRC register is used to select the high voltage mode and divider output enable/disable.

Register Name	Bit							
	7	6	5	4	3	2	1	0
GATEC	—	—	GCBC	GCTC	GBBC	GBTC	GABC	GATC
HVRC	—	—	—	—	—	—	HVMDC	HVDOEN

High Voltage I/O Register List

• GATEC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	GCBC	GCTC	GBBC	GBTC	GABC	GATC
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 **GCBC**: GCB output control bit
0: Disable
1: Enable
- Bit 4 **GCTC**: GCT output control bit
0: Disable
1: Enable
- Bit 3 **GBBC**: GBB output control bit
0: Disable
1: Enable
- Bit 2 **GBTC**: GBT output control bit
0: Disable
1: Enable
- Bit 1 **GABC**: GAB output control bit
0: Disable
1: Enable
- Bit 0 **GATC**: GAT output control bit
0: Disable
1: Enable

• HVRC Register

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

- Bit 7~2 Unimplemented, read as «0»
- Bit 1 **HVMDC**: High voltage mode selection
0: Direct drive P-N MOSFET mode
1: Direct drive Motor MOS mode
- Bit 0 **HVDOEN**: High voltage divider output enable control
0: Disable
1: Enable

Low Voltage Detector

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 determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

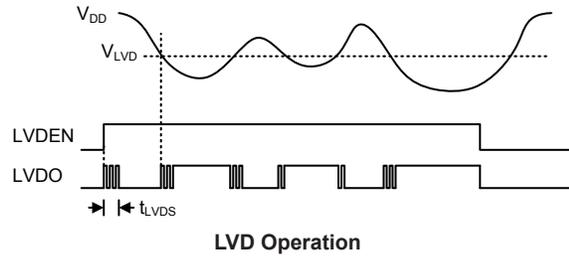
• LVDC Register

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

- Bit 7~6 Unimplemented, read as “0”
- Bit 5 **LVDO**: LVD Output flag
0: No Low Voltage Detected
1: Low Voltage Detected
- Bit 4 **LVDEN**: Low Voltage Detector Enable control
0: Disable
1: Enable
- Bit 3 Unimplemented, read as “0”
- Bit 2~0 **VLVD2~VLVD0**: LVD Voltage selection
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 pre-specified voltage level has a fixed value 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. When the device is in the SLEEP or IDLE mode, 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.



The Low Voltage Detector also has its own interrupt, 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. In this case, the LVF 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 LVF 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 interrupt functions. The external interrupts are generated by the action of the external H1, H2, H3, NFIN and INT1 pins, while the internal interrupts are generated by various internal functions including the TMs, the Comparator 0, 16-bit CAPTM, Time Base, PWM, LVD 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 falls into three categories. The first is the INTEG0 and INTEG1 registers which setup the interrupt trigger edge type for external input interrupts. The second is the INTC0~INTC3 registers which setup the primary interrupts, the third is the MFI0~MFI6 registers which setup the Multi-function interrupts.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
Hall sensor interrupts (External interrupt 0)	HALAE	HALAF	Hall noise filtered inputs
	HALBE	HALBF	
	HALCE	HALCF	
External interrupt 1	INT1E	INT1F	—
Comparator 0	C0E	C0F	—
Noise Filter input function	NFIE	NFIF	—

Function	Enable Bit	Request Flag	Notes
Low Voltage Detector	LVE	LVF	—
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
	PWMPE	PWMPF	—
Timer Module	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	—	LVF	MF1F	NFIF	—	LVE	MF1E	NFIE
INTC2	MF5F	MF4F	MF3F	MF2F	MF5E	MF4E	MF3E	MF2E
INTC3	TBF	MF6F	—	—	TBE	MF6E	—	—
MF10	—	HALCF	HALBF	HALAF	—	HALCE	HALBE	HALAE
MF11	CAPCF	CAPOF	ALIMF	AEOCF	CAPCE	CAPOE	ALIME	AEOCE
MF12	PWMPF	PWMD2F	PWMD1F	PWMD0F	PWMPE	PWMD2E	PWMD1E	PWMD0E
MF13	—	—	PTM2AF	PTM2PF	—	—	PTM2AE	PTM2PE
MF14	—	—	PTM0AF	PTM0PF	—	—	PTM0AE	PTM0PE
MF15	—	—	PTM1AF	PTM1PF	—	—	PTM1AE	PTM1PE
MF16	—	—	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 selection
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**: External Interrupt edge control for INT1 input
 00: Disable
 01: Rising edge trigger
 10: Falling edge trigger
 11: Dual edge trigger

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	C0F	INT1F	HALLF (MF0F)	C0E	INT1E	HALLE (MF0E)	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 interrupt 1 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 interrupt 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	—	LVF	MF1F	NFIF	—	LVE	MF1E	NFIE
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 **LVF**: 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 Filter NFIN input interrupt request flag
0: No request
1: Interrupt request
- Bit 3 Unimplemented, read as “0”
- Bit 2 **LVE**: LVD interrupt control
0: Disable
1: Enable
- Bit 1 **MF1E**: Multi-function interrupt 1 control
0: Disable
1: Enable
- Bit 0 **NFIE**: Noise Filter NFIN 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								
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	—	—	TBE	MF6E	—	—
R/W	R/W	R/W	—	—	R/W	R/W	—	—
POR	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 5~4 Unimplemented, read as “0”
- Bit 3 **TBE**: Time Base interrupt control
 0: Disable
 1: Enable
- Bit 2 **MF6E**: Multi-function interrupt 6 control
 0: Disable
 1: Enable
- Bit 1~0 Unimplemented, read as “0”

• **MFIO 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

• **MF1 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 conversion compare result interrupt request flag
0: No request
1: Interrupt request
- Bit 4 **AEOCF**: A/D conversion end 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 conversion compare result interrupt control
0: Disable
1: Enable
- Bit 0 **AEOCE**: A/D conversion end interrupt control
0: Disable
1: Enable

• **MF12 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 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**: PTM2 Comparator A match interrupt request flag
 0: No request
 1: Interrupt request
- Bit 4 **PTM2PF**: PTM2 Comparator P match interrupt request flag
 0: No request
 1: Interrupt request
- Bit 3~2 Unimplemented, read as “0”
- Bit 1 **PTM2AE**: PTM2 Comparator A match interrupt control
 0: Disable
 1: Enable
- Bit 0 **PTM2PE**: PTM2 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**: PTM0 Comparator A match interrupt request flag
 0: No request
 1: Interrupt request
- Bit 4 **PTM0PF**: PTM0 Comparator P match interrupt request flag
 0: No request
 1: Interrupt request
- Bit 3~2 Unimplemented, read as “0”
- Bit 1 **PTM0AE**: PTM0 Comparator A match interrupt control
 0: Disable
 1: Enable
- Bit 0 **PTM0PE**: PTM0 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**: PTM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request
- Bit 4 **PTM1PF**: PTM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request
- Bit 3~2 Unimplemented, read as “0”
- Bit 1 **PTM1AE**: PTM1 Comparator A match interrupt control
0: Disable
1: Enable
- Bit 0 **PTM1PE**: PTM1 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**: PTM3 Comparator A match interrupt request flag
0: No request
1: Interrupt request
- Bit 4 **PTM3PF**: PTM3 Comparator P match interrupt request flag
0: No request
1: Interrupt request
- Bit 3~2 Unimplemented, read as “0”
- Bit 1 **PTM3AE**: PTM3 Comparator A match interrupt control
0: Disable
1: Enable
- Bit 0 **PTM3PE**: PTM3 Comparator P match interrupt control
0: Disable
1: Enable

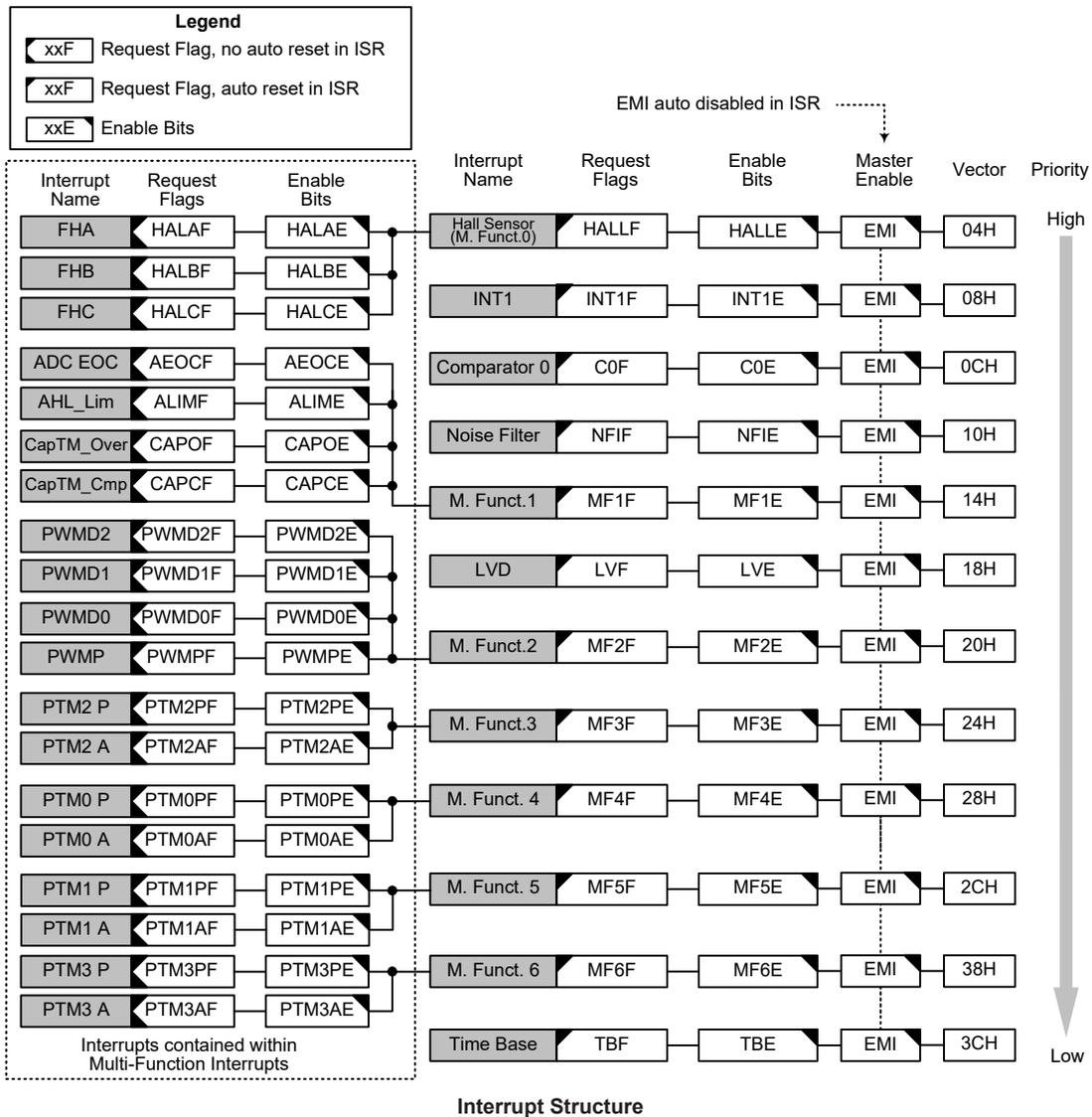
Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A 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 interrupt 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 to zero 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.



Hall Sensor Interrupts

The Hall sensor interrupt, also known as the external interrupt 0, is a multi-function interrupt. They are controlled by signal transitions on the Hall sensor noise filter output signals, FHA, FHB and FHC. A Hall sensor interrupt request will take place when the Hall sensor interrupt request flag, HALAF, HALBF or HALCF, is set, which will occur when a transition (rising edge, falling edge or both edges selected by the INTEG0 register) appears on the Hall sensor noise filter outputs. To allow the program to branch to its prespective interrupt vector address, the global interrupt enable bit, EMI, and the Hall sensor global interrupt control bit, HALLE, and the Hall Sensor interrupt enable bit, HALAE/HALBE/HALCE, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG0 register. If the Hall noise filter inputs are selected to come from the external input pins, H1, H2 and H3, which are pin-shared with I/O pins, they should be configured as an external Hall sensor input pins using the corresponding pin-shared control bits before the Hall sensor interrupt functions are enabled. When the interrupt is enabled, the stack is not full and either one of the Hall sensor interrupts occurs, a subroutine call to the Multi-function interrupt 0

vector will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the related Multi-function interrupt request flag, HALLF (MF10), will be automatically reset, but the Hall sensor interrupt request flags, HALAF, HALBF and HALCF, must be manually cleared to zero by the application program.

The INTEG0 register is used to select the type of active edge that will trigger the Hall sensor interrupt. A choice of either rising or falling or both edge types can be chosen to trigger a Hall sensor interrupt. Note that the INTEG0 register also can be used to disable the Hall sensor input interrupt functions and.

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 occur 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 1 function and to choose the trigger edge type. 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. As the external interrupt pin is pin-shared with I/O pin, it should be configured as external interrupt pin before the external pin interrupt function is enabled. The pin must also be setup as an input type by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, 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 pin will remain valid even if the pin is used as an external interrupt input.

The INTEG1 register is used to select the type of active edge that will trigger the external interrupt 1. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG1 register can also be used to disable the external interrupt 1 function.

Noise Filter Input Interrupt

The noise filter input external interrupt is controlled by signal transitions on the NFIN pin. An external noise filtered input interrupt request will take place when the noise filter input interrupt request flag, NFIF, is set, which will occur when a transition appears on the external Noise Filter input 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. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective NFIN input interrupt enable bit, NFIE, must first be set. As the external noise filter interrupt pin NFIN is pin-shared with I/O pin, it should be configured as the noise filter input pin by the corresponding pin-shared function selection bits before the Noise Filter external interrupt function is enabled. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the noise filter interrupt pin, a subroutine call to the noise filter interrupt vector will take place. When the interrupt is serviced, the noise filter input interrupt request flag, NFIF, 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 NFIN pin will remain valid even if the pin is used as a noise filter interrupt input.

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

Comparator 0 Interrupt

The comparator interrupt is controlled by the internal comparator 0. A comparator interrupt request will take place when the comparator 0 interrupt request flag, C0F, is set, a situation that will occur when the comparator 0 output changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and comparator 0 interrupt enable bit, C0E, must first be set. When the interrupt is enabled, the stack is not full and the comparator 0 input generates a comparator output transition, a subroutine call to the comparator 0 interrupt vector, will take place. When the interrupt is serviced, the comparator 0 interrupt request flag, C0F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Capture Timer Module Interrupts

The CAPTM has two interrupts which are contained within the Multi-function Interrupt and known as CapTM_Over and CapTM_Cmp. A CAPTM capture overflow interrupt request or compare match interrupt will take place when the relevant interrupt request flag, CAPOF or CAPCF, is set, which occurs when CAPTM capture overflows or compare matches. To allow the program to branch to their respective interrupt vector address, the global interrupt enable bit, EMI, and the relevant interrupt enable bit and Multi-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 to zero. As the CAPOF or CAPCF flag will not be automatically cleared to zero, it has to be cleared by the application program.

Multi-function Interrupts

Within the device there 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 input interrupts, PWM period and duty match interrupts, TM Interrupts, A/D Converter Interrupts, and CAPTM Interrupts.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags, HALLF (MF0F) and MFnF, are set. The Multi-function interrupt request flags will be set when any of its included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Multi-function interrupt enable bit must first be set. 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 interrupt 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 interrupt Multi-function request 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.

PWM Module Interrupts

The PWM module has four interrupts, one PWM Period match interrupt known as PWMP interrupt and three PWM Duty match interrupts known as PWMDn (n=0~2) interrupt. The PWMP and PWMDn interrupts are contained in multi-function interrupt 2. A PWM interrupt request will take place when the PWM interrupt request flag, PWMPF or PWMDnF, are set, which occurs when the

PWM Period or Duty n matches. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, the PWM Period or Duty match interrupt enable bit, PWMPE or PWMDnE, and the multi-function interrupt 2 enable bit, must first be set. When the interrupt is enabled, the stack is not full and the PWM Period or Duty matches, 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 multi-function interrupt 2 request flag will also be automatically reset, but the interrupt request flag, PWMDnF or PWMPF, must be manually cleared by the application program.

A/D Converter Interrupts

The A/D Converter has two interrupts which both are contained in Multi-function interrupt. One is the A/D converter compare result interrupt which is controlled by the comparison of the converted data registers (ADRH/ADRL) to the boundary register pairs (ADHVDH/ADHVDL and ADLVDH/ADLVDL) and the limiting condition set by ADCHVE and ADCLVE bits. An A/D converter compare result interrupt request will take place after the comparing and the preset Limit type defined in the ADCHVE and ADCLVE bits in the ADCR1 register takes place. The other is controlled by the end of an A/D conversion process.

An A/D converter interrupt request will take place when the A/D converter interrupt request flag, AEOCF or ALIMF, is set, which occurs when the A/D conversion process finishes or a compare result generation. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, the related Multi-function interrupt enable bit and the A/D converter interrupt enable bit, AEOCE or ALIME, must first be set. When the interrupt is enabled, the stack is not full and an A/D conversion compare result generates or an end of A/D conversion process occurs, a subroutine call to its interrupt vector will take place. When the A/D converter interrupt is serviced, the related Multi-function interrupt request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. However the AEOCF or ALIMF flag will not be automatically cleared to zero, they have to be cleared by the application program.

TM Interrupts

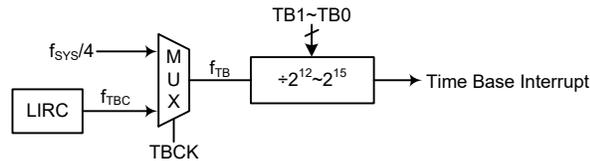
The Periodic Type TMs each have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the multi-function interrupts. For all of the TM types there are two interrupt request flags, PTMnPF and PTMnAF, and two enable control bits, PTMnPE and PTMnAE. A TM interrupt request will take place when any of the TM request flags together with the associated multi-function interrupt request flag are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant 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 multi-function interrupt flag will be automatically cleared to zero. As the TM interrupt request flags will not be automatically cleared to zero, they have to be cleared by the application program.

Time Base Interrupt

The function of the Time Base Interrupt is to provide regular time signal in the form of an internal interrupt. 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 overflows, 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, f_{TB} , originates from the internal clock source $f_{SYS}/4$ or f_{TBC} and then passes through a divider, the division ratio of which is selected by programming the TB1 and TB0 bits to obtain longer interrupt periods whose value ranges.



Time Base Interrupt

• 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**: Time Base 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: $f_{TB}/2^{12}$
 01: $f_{TB}/2^{13}$
 10: $f_{TB}/2^{14}$
 11: $f_{TB}/2^{15}$
- Bit 3~0 Unimplemented, read as “0”

LVD Interrupt

An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, 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, LVE, 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 Low Voltage Interrupt is serviced, the interrupt request flag, LVF, 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 or a low power supply voltage 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 will be automatically cleared to zero, 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 Description

Introduction

The device is specially directed at three-phase BLDC motor control applications as it includes a fully integrated LDO and pre-driver. The internal pre-driver can directly drive a P-N MOSFET driver or a BLDC motor.

The device forms the main power control core of BLDC motor driving systems. BLDC motor sensor driving solutions rely on Hall sensors to detect the rotor magnetic field and then drive the corresponding motor phases based on the Hall sensor signals thus implementing the phase changes required to power the motor. It is therefore necessary to first understand the relationship between the Hall sensor signals and motor driving phases. However, for sensorless driving solutions there is no such problem. Motor driving is divided into two types, square-wave control and sine-wave control, which are both supported by the device. In general driving systems, system voltage and current detection as well as speed commands are necessary. Power component temperature and motor temperature detection, etc., may also be needed in accordance with different product requirements. Basing on the basic functional requirements mentioned above, this chapter will introduce how to use the device to implement three-phase BLDC motor control.

BLDC motor systems are mainly composed of the power circuit, MCU control circuit, motor driving circuit and voltage/current detection circuit, which will be introduced in detail in the hardware block diagram section.

Functional Description

All necessary and important functions required for brushless motor driving are integrated in the device and are coordinated by the BLDC motor control circuit. These functions include a 10-bit motor dedicated PWM, Mask function, Hall sensor decoder and motor protection function, which when combined with the OCP, capture timer, etc., provide the advantage of fast system protection. From the perspective of a control system, the main points of each function will be explained in the following sections.

Hall Sensor/Sensorless

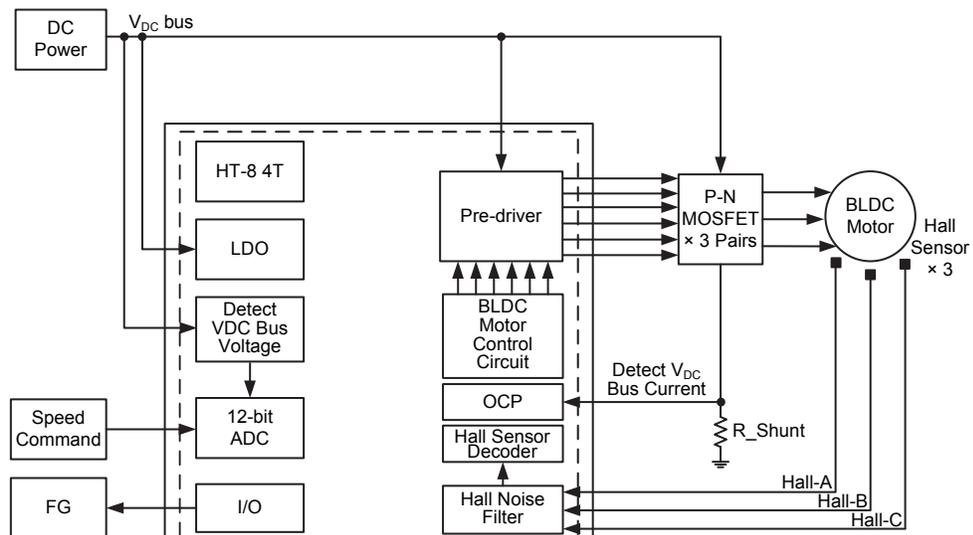
For brushless motor operation, it is necessary to know the rotor position in order to provide the correct operation phase. There are two methods to obtain the rotor position, which are determined by the actual solution: Hall sensor solutions (Hall IC & Hall component) or sensorless solutions. If using a Hall IC, the present Hall sensor information can be obtained from the H1~H3 pins. However for sensorless or Hall component solutions, the rotor position can only be obtained after necessary processing by the internal three comparators. Refer to the Hall Sensor Noise Filter Block Diagram for more details.

Square-wave/Sine-wave Driving

The square-wave driving and the sine-wave driving are different in the phase change algorithm and more importantly in the PWM configuration. The mask function is composed of mask circuit, dead-time circuit and polarity control circuit, which when combined with the motor dedicated 10-bit PWM can implement square-wave and sine-wave driving. When the hardware mask mode is selected, the square-wave driving phase change logic is implemented by the Hall sensor decoder. Users can detect the HA~HC signals for automatic phase change or use the program for phase change. When the software mask mode is selected, users can flexibly configure the PWM signals on the motor control dedicated I/O ports and use the PWM center-align mode and dead-time control to implement the PWM output signals required for sine-wave driving.

1. The DC power is a 12V system which provides power for the LDO, the V_{DC} bus voltage detection circuit and the pre-driver circuit.
2. The LDO outputs 5V as standard power for the MCU and Hall sensors.
3. The speed adjustment command depends on the clock cycle and the digital signal sources from the NFIN pin.
4. V_{DC} is first divided via the voltage detection circuit and then input to the MCU A/D channel. In this way the A/D converter can measure the current system voltage.
5. The pre-driver circuit input signals are from the BLDC motor control circuit. Its outputs GxT and GxB which are co-phases and in parallel to drive the BLDC motor.
6. The three Hall element signals should be first input to the internal comparators, CMP1~CMP3, and then they should be input to the Hall sensor noise filter. After being filtered, these signals will be transferred to the interrupt controller, capture timer and Hall sensor decoder. This can avoid noise interference when the motor is running.
7. Motor operating current sampling is achieved using the current sampling resistor, R_{Shunt} . When a current flows through this resistor, a small voltage signal will be generated. This signal will first be amplified by the OPA in the OCP circuit, and then read by the A/D Converter or checked by the DAC and CMP to detect if an over-current situation has occurred in which case it will trigger the corresponding over current protection measures.
8. Lastly, by coordinating all the relevant peripheral circuits the BLDC motor control circuit will configure the motor driving PWM signals and phase change logic according to the Hall sensor signals and protection signals to implement correct motor operating control.

Pre-driver Direct Drive P-N MOSFET Mode

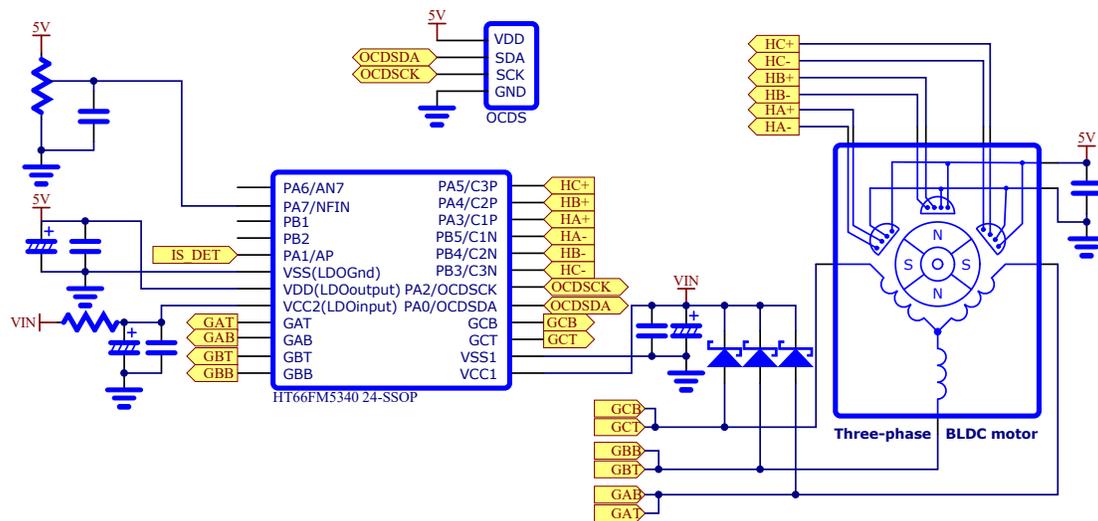


BLDC Motor Control with Hall Sensor – Direct Drive P-N MOSFET Mode

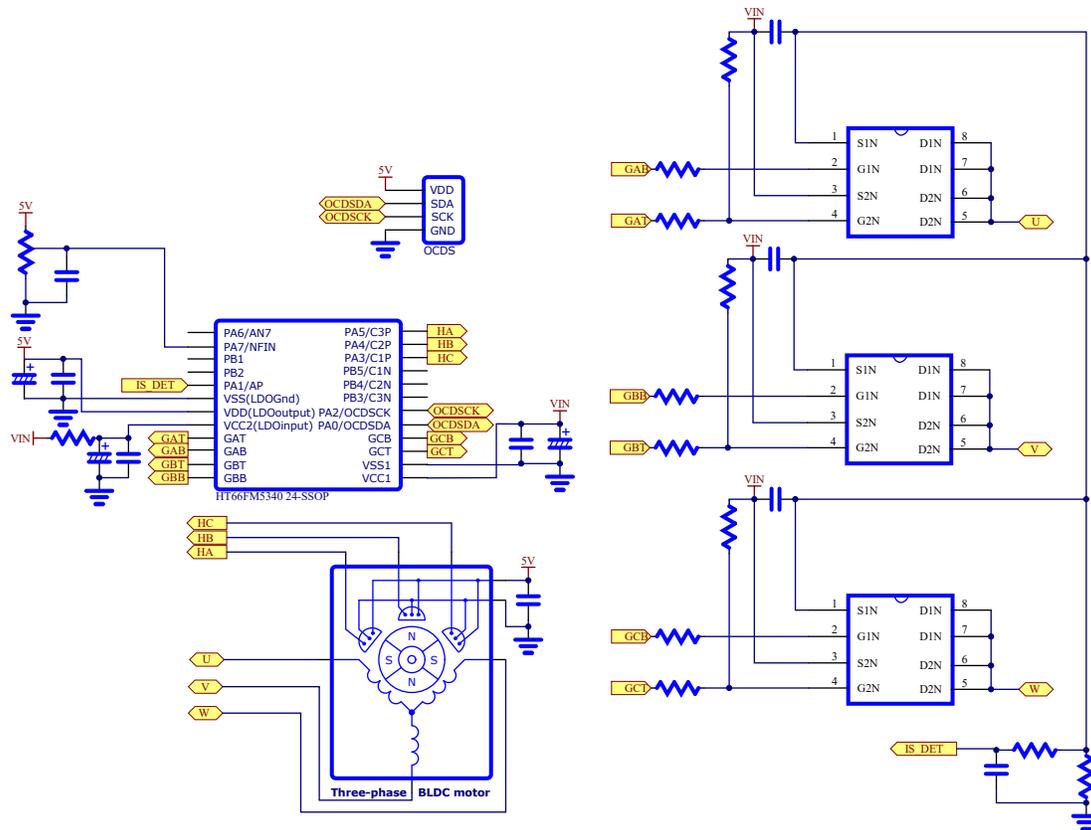
1. The DC power is a low voltage system which provides power for the LDO, the V_{DC} bus voltage detection circuit, the pre-driver circuit and the P-N MOSFET circuit.
2. The LDO outputs 5V standard power for the MCU, Hall sensor and VR speed command component.
3. The speed command component is a variable resistor. It outputs a voltage from 0V to 5V for the speed adjustment command.

4. V_{DC} is first divided via the voltage detection circuit and then input to the MCU A/D channel. In this way the A/D converter can measure the current system voltage.
5. The pre-driver circuit is controlled by the BLDC motor control circuit. The GxT and the GxB outputs are used to drive the external P-MOSFET and N-MOSFET respectively.
6. When the motor is loaded, the Hall signals are susceptible to noise interference. Therefore the three Hall sensors signals, Hall-A, Hall-B and Hall-C should first be input to the Hall sensor noise filter. After being filtered, these signals will be transferred to the interrupt controller, capture timer and Hall sensor decoder.
7. Motor operating current sampling is achieved using the current sampling resistor, R_Shunt. When a current flows through this resistor, a small voltage signal will be generated. This signal will first be amplified by the OPA in the OCP circuit and then read by the A/D converter or checked by the D/A converter and CMP to detect if an over-current situation has occurred, in which case it will trigger the corresponding over current protection measures.
8. Lastly, by coordinating all the relevant peripheral circuits the BLDC motor control circuit will configure the motor driving PWM signals and phase change logic according to the Hall sensor signals and protection signals to implement correct motor operating control.

Hardware Circuit



BLDC Motor Control with Hall Sensor – Direct Drive Motor Mode



BLDC Motor Control with Hall Sensor – Direct Drive P-N MOSFET Mode

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 0.5 μ s and branch or call instructions would be implemented within 1 μ s. 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 several 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 such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

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, SC
ADDM A,[m]	Add ACC to Data Memory	1 ^{Note}	Z, C, AC, OV, SC
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV, SC
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV, SC
ADCM A,[m]	Add ACC to Data memory with Carry	1 ^{Note}	Z, C, AC, OV, SC
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV, SC, CZ
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV, SC, CZ
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1 ^{Note}	Z, C, AC, OV, SC, CZ
SBC A,x	Subtract immediate data from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1 ^{Note}	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1 ^{Note}	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	1 ^{Note}	Z
ORM A,[m]	Logical OR ACC to Data Memory	1 ^{Note}	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1 ^{Note}	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	1 ^{Note}	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	1 ^{Note}	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1 ^{Note}	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1 ^{Note}	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1 ^{Note}	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1 ^{Note}	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1 ^{Note}	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	1 ^{Note}	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1 ^{Note}	None
SET [m].i	Set bit of Data Memory	1 ^{Note}	None
Branch Operation			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1 ^{Note}	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1 ^{Note}	None
SZ [m].i	Skip if bit i of Data Memory is zero	1 ^{Note}	None
SNZ [m]	Skip if Data Memory is not zero	1 ^{Note}	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1 ^{Note}	None
SIZ [m]	Skip if increment Data Memory is zero	1 ^{Note}	None
SDZ [m]	Skip if decrement Data Memory is zero	1 ^{Note}	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1 ^{Note}	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1 ^{Note}	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read Operation			
TABRD [m]	Read table (specific page) to TBLH and Data Memory	2 ^{Note}	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2 ^{Note}	None
ITABRD [m]	Increment table pointer TBLP first and Read table to TBLH and Data Memory	2 ^{Note}	None
ITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	2 ^{Note}	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1 ^{Note}	None
SET [m]	Set Data Memory	1 ^{Note}	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1 ^{Note}	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

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

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

3. For the “CLR WDT” instruction the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after the “CLR WDT” instructions is executed. Otherwise the TO and PDF flags remain unchanged.

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sections except sector 0, the extended instruction can be used to access the data memory instead of using the indirect addressing access to improve the CPU firmware performance.

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	2 ^{Note}	Z, C, AC, OV, SC
LADC A,[m]	Add Data Memory to ACC with Carry	2	Z, C, AC, OV, SC
LADCM A,[m]	Add ACC to Data memory with Carry	2 ^{Note}	Z, C, AC, OV, SC
LSUB A,[m]	Subtract Data Memory from ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	2 ^{Note}	Z, C, AC, OV, SC, CZ
LSBC A,[m]	Subtract Data Memory from ACC with Carry	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	2 ^{Note}	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2 ^{Note}	C
Logic Operation			
LAND A,[m]	Logical AND Data Memory to ACC	2	Z
LOR A,[m]	Logical OR Data Memory to ACC	2	Z
LXOR A,[m]	Logical XOR Data Memory to ACC	2	Z
LANDM A,[m]	Logical AND ACC to Data Memory	2 ^{Note}	Z
LORM A,[m]	Logical OR ACC to Data Memory	2 ^{Note}	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2 ^{Note}	Z
LCPL [m]	Complement Data Memory	2 ^{Note}	Z
LCPLA [m]	Complement Data Memory with result in ACC	2	Z
Increment & Decrement			
LINCA [m]	Increment Data Memory with result in ACC	2	Z
LINC [m]	Increment Data Memory	2 ^{Note}	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2 ^{Note}	Z
Rotate			
LRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2 ^{Note}	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2 ^{Note}	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2 ^{Note}	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2 ^{Note}	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2 ^{Note}	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2 ^{Note}	None
LSET [m].i	Set bit of Data Memory	2 ^{Note}	None

Mnemonic	Description	Cycles	Flag Affected
Branch			
LSZ [m]	Skip if Data Memory is zero	2 ^{Note}	None
LSZA [m]	Skip if Data Memory is zero with data movement to ACC	2 ^{Note}	None
LSNZ [m]	Skip if Data Memory is not zero	2 ^{Note}	None
LSZ [m].i	Skip if bit i of Data Memory is zero	2 ^{Note}	None
LSNZ [m].i	Skip if bit i of Data Memory is not zero	2 ^{Note}	None
LSIZ [m]	Skip if increment Data Memory is zero	2 ^{Note}	None
LSDZ [m]	Skip if decrement Data Memory is zero	2 ^{Note}	None
LSIZA [m]	Skip if increment Data Memory is zero with result in ACC	2 ^{Note}	None
LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC	2 ^{Note}	None
Table Read			
LTABRD [m]	Read table to TBLH and Data Memory	3 ^{Note}	None
LTABRDL [m]	Read table (last page) to TBLH and Data Memory	3 ^{Note}	None
LITABRD [m]	Increment table pointer TBLP first and Read table to TBLH and Data Memory	3 ^{Note}	None
LITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	3 ^{Note}	None
Miscellaneous			
LCLR [m]	Clear Data Memory	2 ^{Note}	None
LSET [m]	Set Data Memory	2 ^{Note}	None
LSWAP [m]	Swap nibbles of Data Memory	2 ^{Note}	None
LSWAPA [m]	Swap nibbles of Data Memory with result in ACC	2	None

Note: 1. For these extended skip instructions, if the result of the comparison involves a skip then up to four cycles are required, if no skip takes place two cycles is required.

2. Any extended instruction which changes the contents of the PCL register will also require three 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	$ACC \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C, SC
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	$[m] \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C, SC
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	$ACC \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C, SC
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	$ACC \leftarrow ACC + x$
Affected flag(s)	OV, Z, AC, C, SC
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	$[m] \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C, SC
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	$ACC \leftarrow ACC \text{ "AND" } [m]$
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	$ACC \leftarrow ACC \text{ "AND" } x$
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	$[m] \leftarrow ACC \text{ "AND" } [m]$
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	$[m] \leftarrow [m] - 1$
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	$ACC \leftarrow [m] - 1$
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	$TO \leftarrow 0$ $PDF \leftarrow 1$
Affected flag(s)	TO, PDF
INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	$[m] \leftarrow [m] + 1$
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	$ACC \leftarrow [m] + 1$
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 \leftarrow 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	$ACC \leftarrow [m]$
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	$ACC \leftarrow x$
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	$[m] \leftarrow ACC$
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	$ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
SBC A, x	Subtract immediate data from ACC with Carry
Description	The immediate data 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	$ACC \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
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	$[m] \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
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	$[m] \leftarrow [m] - 1$ 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	$ACC \leftarrow [m] - 1$ 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	$[m] \leftarrow FFH$
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	$[m].i \leftarrow 1$
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	$[m] \leftarrow [m] + 1$ 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	$ACC \leftarrow [m] + 1$ Skip if $ACC=0$
Affected flag(s)	None
SNZ [m].i	Skip if Data Memory is not 0
Description	If 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 $[m].i \neq 0$
Affected flag(s)	None
SNZ [m]	Skip if Data Memory is not 0
Description	If 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 $[m] \neq 0$
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	$ACC \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C, SC, CZ

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	$[m] \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C, SC, CZ
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	$ACC \leftarrow ACC - x$
Affected flag(s)	OV, Z, AC, C, SC, CZ
SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
Affected flag(s)	None
SZ [m]	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if $[m]=0$
Affected flag(s)	None
SZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m]$ Skip if $[m]=0$
Affected flag(s)	None
SZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if $[m].i=0$
Affected flag(s)	None

TABRD [m]	Read table (specific page) to TBLH and Data Memory
Description	The low byte of the program code (specific page) addressed by the table pointer pair (TBLP and 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
ITABRD [m]	Increment table pointer low byte first and read table to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
ITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then 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

Extended Instruction Definition

The extended instructions are used to directly access the data stored in any data memory sections.

LADC 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	$ACC \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C, SC
LADCM 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	$[m] \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C, SC
LADD 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	$ACC \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C, SC
LADDM 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	$[m] \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C, SC
LAND 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	$ACC \leftarrow ACC \text{ "AND" } [m]$
Affected flag(s)	Z
LANDM 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	$[m] \leftarrow ACC \text{ "AND" } [m]$
Affected flag(s)	Z
LCLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	$[m] \leftarrow 00H$
Affected flag(s)	None
LCLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	$[m].i \leftarrow 0$
Affected flag(s)	None

LCPL [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	$[m] \leftarrow \overline{[m]}$
Affected flag(s)	Z
LCPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow \overline{[m]}$
Affected flag(s)	Z
LDAA [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	$[m] \leftarrow ACC + 00H$ or $[m] \leftarrow ACC + 06H$ or $[m] \leftarrow ACC + 60H$ or $[m] \leftarrow ACC + 66H$
Affected flag(s)	C
LDEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	$[m] \leftarrow [m] - 1$
Affected flag(s)	Z
LDECA [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	$ACC \leftarrow [m] - 1$
Affected flag(s)	Z
LINC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	$[m] \leftarrow [m] + 1$
Affected flag(s)	Z
LINCA [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	$ACC \leftarrow [m] + 1$
Affected flag(s)	Z

LMOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	$ACC \leftarrow [m]$
Affected flag(s)	None
LMOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	$[m] \leftarrow ACC$
Affected flag(s)	None
LOR 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	$ACC \leftarrow ACC \text{ "OR" } [m]$
Affected flag(s)	Z
LORM 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	$[m] \leftarrow ACC \text{ "OR" } [m]$
Affected flag(s)	Z
LRL [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	$[m].(i+1) \leftarrow [m].i; (i=0\sim 6)$ $[m].0 \leftarrow [m].7$
Affected flag(s)	None
LRLA [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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim 6)$ $ACC.0 \leftarrow [m].7$
Affected flag(s)	None
LRLC [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	$[m].(i+1) \leftarrow [m].i; (i=0\sim 6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C
LRLCA [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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim 6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C

LRR [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	$[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow [m].0$
Affected flag(s)	None
LRRRA [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	$ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow [m].0$
Affected flag(s)	None
LRRC [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	$[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
LRRCA [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	$ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
LSBC 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	$ACC \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
LSBCM 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	$[m] \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSDZ [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	$[m] \leftarrow [m] - 1$ Skip if $[m]=0$
Affected flag(s)	None
LSDZA [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	$ACC \leftarrow [m] - 1$ Skip if $ACC=0$
Affected flag(s)	None
LSET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	$[m] \leftarrow FFH$
Affected flag(s)	None
LSET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	$[m].i \leftarrow 1$
Affected flag(s)	None
LSIZ [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	$[m] \leftarrow [m] + 1$ Skip if $[m]=0$
Affected flag(s)	None
LSIZA [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	$ACC \leftarrow [m] + 1$ Skip if $ACC=0$
Affected flag(s)	None
LSNZ [m].i	Skip if Data Memory is not 0
Description	If 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 $[m].i \neq 0$
Affected flag(s)	None

LSNZ [m]	Skip if Data Memory is not 0
Description	If the content 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] ≠ 0
Affected flag(s)	None
LSUB 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, SC, CZ
LSUBM 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, SC, CZ
LSWAP [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
LSWAPA [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
LSZ [m]	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m]=0
Affected flag(s)	None
LSZA [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

LSZ [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
LTABRD [m]	Read table (current page) to TBLH and Data Memory
Description	The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
LTABRDL [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
LITABRD [m]	Increment table pointer low byte first and read table to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBLP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
LITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then 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
LXOR 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
LXORM 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

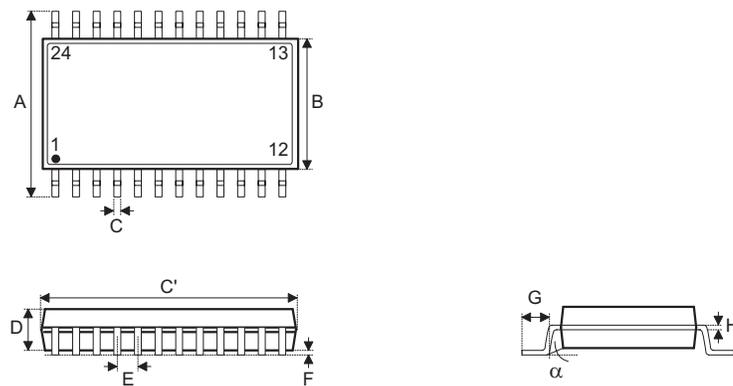
Package Information

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

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

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

24-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°

Copyright© 2018 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at <http://www.holtek.com/en/>.