1-Phase BLDC Motor OTP MCU with Gate-Driver

BD66RM2541G

Revision: V1.20 Date: April 23, 2025

www.holtek.com

Features

CPU Features

- Operating voltage
  - f_{SYS}=32kHz~20MHz: 4.5V~5.5V
  - V_{CC}=6V~40V
  - Maximum motor sustainable voltage up to 48V
- Up to 0.2μs 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 16/20MHz 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

- OTP Program Memory: 4K×16
- RAM Data Memory: 384×8
- OTP ROM parameter program – ORPP
- Watchdog Timer function
- 11 bidirectional I/O lines
- 3 pin-shared external interrupt inputs – H1, NFIN and INT
- 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 generator provides 3 groups of PWM outputs, supporting edge/center-aligned mode
- BLDC motor control
  - Hall noise filter for position detection
  - Motor control circuit
  - Motor protection circuit
- Up to two comparators
- Over current detection which contains an operational amplifier, a comparator 0 and an 8-bit D/A converter
- 8 external channel 12-bit resolution A/D converter

- Single Fully-duplex or Half-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- Integrated Multiplier/Divider Unit – MDU
- Single Time Base function for generation of fixed time interrupt signal
- Integrated 50mA LDO with 5.0V±2% accuracy
- Low voltage reset function
- Low voltage detect function
- Protection Features
  - Power Supply Input Under Voltage Lock-Out (VCC_UVLO)
  - Bootstrap Output Under Voltage Lock-Out (VBST_UVLO)
  - 12V LDO Output Under Voltage Lock-Out (V12P_UVLO)
  - 5V LDO Output Under Voltage Lock-Out (VREG_UVLO)
  - Over Temperature Protection (OTP)
- Package type: 24-pin QFN

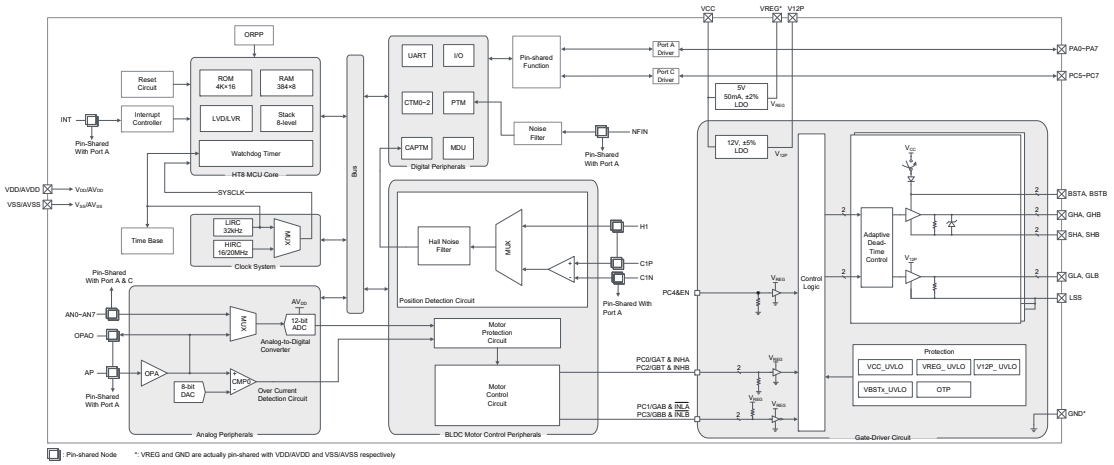
General Description

The BD66RM2541G is an OTP type 8-bit high performance RISC architecture microcontroller, which includes two groups of H-bridge gate-drivers for driving N/N MOS to control the brushless DC motor, and a host of fully integrated special features specifically designed for brushless DC motor applications.

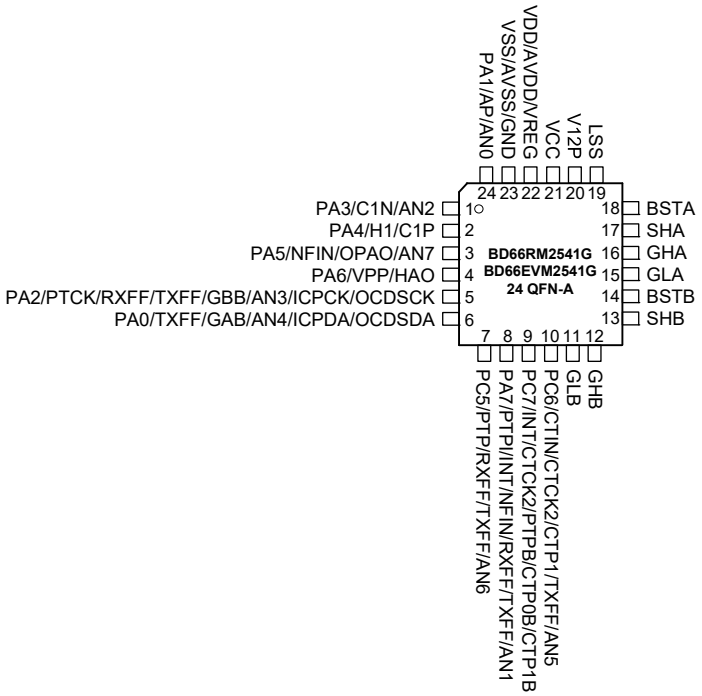
For memory features, the device is supplied with One-Time Programmable, OTP memory. Other memory includes an area of RAM Data Memory. By using the ORPP function, users have a convenient means to directly store their measured data in the unprogrammed Flash Program Memory.

In addition to the advantages of low power consumption and I/O flexibility, the device also includes multiple and extremely flexibile 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 Protection Module, Hall sensor position detection function, UART interface function, 16-bit MDU, Time Base function, 5V and 12V LDOs, Low Voltage Reset, Low Voltage Detection, 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 electrical tools application possibilities such as 1-phase cooling fans, etc.

Block Diagram



Pin Assignment



Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by its corresponding software control bits.

2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the BD66EVM2541G device (Flash type) which is the OCDS EV chip for the BD66RM2541G device (OTP type).

3. The VPP pin is the High Voltage input for OTP programming and only available for the BD66RM2541G device.

Pin Description

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

Pin Name	Function	OPT	I/T	O/T	Description
PA0/TXFF/GAB/AN4/ OCDSDA/ICPDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	TXFF	PAS0	—	CMOS	UART serial data output
	GAB	PAS0	—	CMOS	Pulse width modulation complementary output
	AN4	PAS0	AN	—	A/D converter external input channel 4
	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
	ICPDA	—	ST	CMOS	ICP data/address pin
PA1/AP/AN0	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	AP	PAS0	AN	—	Positive input of OPA
	AN0	PAS0	AN	—	A/D converter external input channel 0
PA2/PTCK/RXFF/TXFF/ GBB/AN3/OCDSCK/ ICPCK/	PA2	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	PTCK	PAS0	ST	—	PTM clock input or capture input
	RXFF/ TXFF	PAS0 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input /output in Single Wire Mode communication
	GBB	PAS0	—	CMOS	Pulse width modulation complementary output
	AN3	PAS0	AN	—	A/D converter external input channel 3
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
	ICPCK	—	ST	—	ICP clock pin
PA3/C1N/AN2	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	C1N	PAS0	AN	—	Comparator 1 negative input
	AN2	PAS0	AN	—	A/D converter external input channel 2
PA4/H1/C1P	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	H1	PAS1 IFS	ST	—	Hall sensor input
	C1P	PAS1	AN	—	Comparator 1 positive input
PA5/NFIN/OPAO/AN7	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	NFIN	PAS1 IFS	ST	—	Noise filter input
	OPAO	PAS1	—	AN	OPA output
	AN7	PAS1	AN	—	A/D converter external input channel 7
PA6/VPP/HAO	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up. This I/O is pin-shared with VPP and could result in increased current consumption if it is set to output high.
	VPP	PAS1	PWR	—	High Voltage input for OTP programming pin, not available for EV chip
	HAO	PAS1	—	CMOS	Test pin for Hall sensor

Pin Name	Function	OPT	I/T	O/T	Description
PA7/PTPI/INT/NFIN/ RXFF/TXFF/AN1	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	PTPI	PAS1	ST	—	PTM capture input
	INT	PAS1 IFS INTEG INTC0	ST	—	External interrupt input
	NFIN	PAS1 IFS	ST	—	Noise filter input
	RXFF/ TXFF	PAS1 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input /output in Single Wire Mode communication
	AN1	PAS1	AN	—	A/D converter external input channel 1
PC5/PTP/RXFF/TXFF/ AN6	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP	PCS1	—	CMOS	PTM output
	RXFF/ TXFF	PCS1 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input /output in Single Wire Mode communication
	AN6	PCS1	AN	—	A/D converter external input channel 6
PC6/CTIN/CTCK2/CTP1/ TXFF/AN5	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTIN	PCS1	ST	—	CAPTM input
	CTCK2	PCS1 IFS	ST	—	CTM2 clock input
	CTP1	PCS1	—	CMOS	CTM1 output
	TXFF	PCS1	—	CMOS	UART serial data output
	AN5	PCS1	AN	—	A/D converter external input channel 5
PC7/INT/CTCK2/PTPB/ CTP0B/CTP1B	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT	PCS1 IFS INTEG INTC0	ST	—	External interrupt input
	CTCK2	PCS1 IFS	ST	—	CTM2 clock input
	PTPB	PCS1	—	CMOS	PTM inverted output
	CTP0B	PCS1	—	CMOS	CTM0 inverted output
	CTP1B	PCS1	—	CMOS	CTM1 inverted output
GHA	GHA	—	—	AN	High-side gate drive phase A
GHB	GHB	—	—	AN	High-side gate drive phase B
GLA	GLA	—	—	AN	Low-side gate drive phase A
GLB	GLB	—	—	AN	Low-side gate drive phase B
BSTA	BSTA	—	—	AN	Bootstrap output phase A
BSTB	BSTB	—	—	AN	Bootstrap output phase B
SHA	SHA	—	AN	—	High-side source connection phase A
SHB	SHB	—	AN	—	High-side source connection phase B
LSS	LSS	—	AN	—	Low-side source connection for phase A and B. Connect to ground of power stage.
VCC	VCC	—	PWR	—	Gate-driver power supply input and LDO input
V12P	V12P	—	—	AN	Regulated 12V output; supplied from VCC

Pin Name	Function	OPT	I/T	O/T	Description
VDD/AVDD/VREG	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
	VREG	—	—	AN	Regulated 5V output; supplied from VCC
VSS/AVSS/GND	VSS	—	PWR	—	Digital negative power supply, ground
	AVSS	—	PWR	—	Analog negative power supply, ground
	GND	—	PWR	—	Gate-driver circuit 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.

Internally Connected Signal

Several signals are not connected to external package pins. These signals are interconnection lines between the MCU and the Gate-Driver and are listed in the following table.

MCU Signal	Gate-Driver Signal	Function	Description
PC0/GAT	INHA	GAT	Pulse width modulation complementary output Internally connected to the gate-driver input INHA
		INHA	High-side gate-driver phase A control input, high active Internally connected to the MCU GAT output
PC1/GAB	$\overline{\text{INLA}}$	GAB	Pulse width modulation complementary output Internally connected to the gate-driver input $\overline{\text{INLA}}$
		$\overline{\text{INLA}}$	Low-side gate-driver phase A control input, low active Internally connected to the MCU GAB output
PC2/GBT	INHB	GBT	Pulse width modulation complementary output Internally connected to the gate-driver input INHB
		INHB	High-side gate-driver phase B control input, high active Internally connected to the MCU GBT output
PC3/GBB	$\overline{\text{INLB}}$	GBB	Pulse width modulation complementary output Internally connected to the gate-driver input $\overline{\text{INLB}}$
		$\overline{\text{INLB}}$	Low-side gate-driver phase B control input, low active Internally connected to the MCU GBB output
PC4	EN	PC4	General purpose I/O Internally connected to the gate-driver input EN
		EN	Gate-driver enable pin, when EN="0", only VREG keeps active Internally connected to the MCU I/O line PC4

Note: The internal signals, PC0/GAT~PC3/$\overline{\text{GBB}}$ and PC4, are internally connected to the gate-driver input or output, INHA~INHB, $\overline{\text{INLA}}$~$\overline{\text{INLB}}$ and EN respectively, the relevant pin-shared control bits and I/O control bits should be properly configured to implement correct interconnection.

Absolute Maximum Ratings

MCU Supply Voltage ..... V_{SS}-0.3V to 5.5V
MCU Input Voltage ..... V_{SS}-0.3V to V_{DD}+0.3V
VCC ..... -0.3V to 48V
SHx ..... -6(<150ns) to 48V
BSTx, GHx ..... -0.3V to 60V
$V_{(GHx,SHx)}$, $V_{(BSTx,SHx)}$ ..... -0.3V to 20V
V12P, GLx ..... -0.3V to 20V
Storage Temperature ..... -60°C to 150°C
Operating Temperature ..... -40°C to 105°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

Ta=-40°C~105°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	f_{SYS}=f_{HIRC}=16MHz	4.5	—	5.5	V
		f_{SYS}=f_{HIRC}=20MHz	4.5	—	5.5	
	Operating Voltage – LIRC	f_{SYS}=f_{LIRC}=32kHz	4.5	—	5.5	

Operating Current Characteristics

Ta=-40°C~105°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	Operating Current – LIRC	5V	f_{SYS}=32kHz	—	30	50	μA
	Operating Current – HIRC	4.5V	f_{SYS}=16MHz	—	1.5	3.0	mA
		5V		—	2.5	5.0	
		4.5V	f_{SYS}=20MHz	—	5.4	7.2	mA
		5V		—	9	12	

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 and the I/O pin-shared with VPP is not setup in an output high 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.

Standby Current Characteristics

Ta=25°C, unless otherwise specifed

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @105°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	5V	WDT off	—	0.5	5.0	10.0	μA
		5V	WDT on	—	3	7	15	
	IDLE0 Mode – LIRC	5V	f_{SUB} on	—	5	10	15	
	IDLE1 Mode – HIRC	4.5V	f_{SUB} on, $f_{SYS}=f_{HIRC}$=16MHz	—	0.80	1.20	1.44	mA
		5V		—	1.4	2.0	2.4	
		4.5V	f_{SUB} on, $f_{SYS}=f_{HIRC}$=20MHz	—	1.32	1.98	2.37	mA
		5V		—	2.50	3.30	3.96	

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 and the I/O pin-shared with VPP is not setup in an output high 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.

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 the user selected HIRC frequency and the fixed voltage of 5V.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	16MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	16	+1%	MHz
			-20°C~105°C	-2%	16	+2%	
			-40°C~-20°C	-3.5%	16	+3.5%	
		4.5V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C~105°C	-3.5%	16	+3.5%	
	20MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	20	+1%	MHz
			-20°C~105°C	-2%	20	+2%	
			-40°C~-20°C	-3.5%	20	+3.5%	
		4.5V~5.5V	25°C	-2.5%	20	+2.5%	
			-40°C~105°C	-3.5%	20	+3.5%	

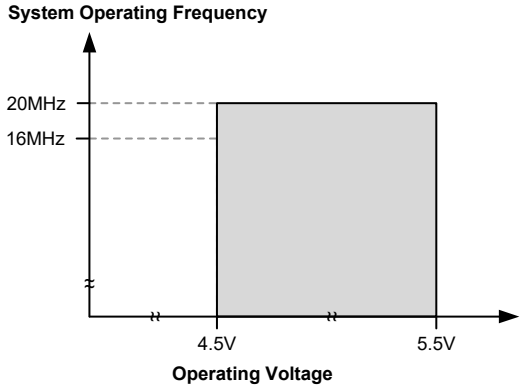
Note: 1. The 5V value for V_{DD} is provided as this is the fixed voltage at which the HIRC frequency is trimmed by the writer.

2. The row below the 5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage.
3. The minum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

Low Speed Internal Oscillator – LIRC – Frequency Accuracy

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	4.5V~5.5V	25°C	-20%	32	+20%	kHz
			-40°C~105°C	-50%	32	+60%	
t_{START}	LIRC Start Up Time	—	—	—	—	500	μs

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~105°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)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	System Start-up Time (Wake-up from Condition where f_{SYS} is on)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	System Speed Switch Time (FAST to SLOW Mode or SLOW to FAST Mode)	—	f_{HIRC} off → on	—	16	—	t_{HIRC}
t_{RSTD}	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR_{POR}=5V/ms	10	16	24	ms
	System Reset Delay Time (LVRC/WDTC Software Reset)	—	—				
	System Reset Delay Time (WDT Time-out Reset)	—	—				

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

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

Input/Output Characteristics

Ta=-40°C~105°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports	5V	INHLVC[m]=0 (m=0, 1, 2, 3)	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	
		5V	INHLVC[m]=1 (m=0, 1, 2, 3)	0	—	0.8	
V_{IH}	Input High Voltage for I/O Ports	5V	INHLVC[m]=0 (m=0, 1, 2, 3)	3.5	—	5.0	V
		—		$0.8V_{DD}$	—	V_{DD}	
		5V	INHLVC[m]=1 (m=0, 1, 2, 3)	2.5	—	5.0	
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(1)	5V	—	10	30	50	kΩ
I_{LEAK}	Input Leakage Current	5V	V_{IN}=V_{DD} or V_{IN}=V_{SS}	—	—	±1	μA
t_{INT}	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs
t_{TCK}	CTCK2, PTCK Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TPI}	PTPI Input Pin Minimum Pulse Width	—	—	50	—	—	ns
f_{TMCLK}	PTM Maximum Timer Clock Source Frequency	5V	—	—	—	1	f_{SYS}
t_{CPW}	PTM minimum capture pulse width	—	—	t_{CPW}(2)	—	—	t_{TMCLK}

Note: 1. The R_{PH} internal pull-high resistance value is calculated by connecting to ground and enabling the input pin with a pull-high resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R_{PH} value.

2. If PTCAPTS=0, then t_{CPW}=max(2×t_{TMCLK}, t_{TPI}). If PTCAPTS=1, then t_{CPW}=max(2×t_{TMCLK}, t_{TCK}).
t_{TMCLK}=1/f_{TMCLK}
Example 1: If PTCAPTS=0, f_{TMCLK}=16MHz, t_{TPI}=50ns=0.05μs, then t_{CPW}=max(0.125μs, 0.05μs)=0.125μs
Example 2: If PTCAPTS=1, f_{TMCLK}=16MHz, t_{TCK}=0.3μs, then t_{CPW}=max(0.125μs, 0.3μs)=0.3μs
Example 3: If PTCAPTS=0, f_{TMCLK}=8MHz, t_{TPI}=50ns=0.05μs, then t_{CPW}=max(0.25μs, 0.05μs)=0.25μs
Example 4: If PTCAPTS=0, f_{TMCLK}=4MHz, t_{TPI}=50ns=0.05μs, then t_{CPW}=max(0.5μs, 0.05μs)=0.5μs

Memory Electrical Characteristics

Ta=-40°C~105°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
OTP Program Memory							
V_{DD}	Operating Voltage for Read – ORPP	—	—	V_{LVR}	5.0	5.5	V
	Operating Voltage for Write – ORPP	—	—	4.5	5.0	5.5	
V_{PP}	V_{PP} for Write – ORPP	—	—	8.25	8.50	8.75	V
t_{WR}	Write Cycle Time – ORPP	—	—	—	300	450	μs
E_P	Cell Endurance	—	—	1	—	—	W
t_{RETD}	ROM Data Retention time	—	Ta=25°C	—	40	—	Year
Flash Program Memory – for BD66EVM2541G only							
V_{DD}	Operating Voltage for Read & Write	—	—	4.5	—	5.5	V
t_{WR}	Write Time	—	—	—	2.2	2.7	ms
E_P	Cell Endurance	—	—	100K	—	—	W

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{RETD}	ROM Data Retention time	—	Ta=25°C	—	40	—	Year
t_{ACTV}	ROM Activation Time	—	Wake-up from Power Down Mode	32	—	64	μs
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

Note: 1. The ROM activation time t_{ACTV} should be added when calculating the total system start-up time of a wake-up from the IDLE/SLEEP mode.
2. "E/W" means Erase/Write times.

LVR/LVD Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 3.15V	-3%	3.15	+3%	V
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 3.3V	-5%	3.3	+5%	V
		—	LVD enable, voltage select 3.6V	-5%	3.6	+5%	
		—	LVD enable, voltage select 4.0V	-5%	4.0	+5%	
t_{LVDS}	LVDO Stable Time	—	For LVR enable, LVD off → on	—	—	18	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	TLVR[1:0]=00B	120	240	480	μs
		—	TLVR[1:0]=01B	0.5	1.0	2.0	ms
		—	TLVR[1:0]=10B	1	2	4	
		—	TLVR[1:0]=11B	2	4	8	
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	TLVD[1:0]=00B/11B	60	140	220	μs
		—	TLVD[1:0]=01B	90	200	340	
		—	TLVD[1:0]=10B	150	320	580	
I_{LVR}	Additional Current for LVR Enable	—	LVD disable	—	—	14	μA

A/D Converter Electrical Characteristics

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{ADI}	A/D Converter Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	A/D Converter Reference Voltage	—	—	2	—	V_{DD}	V
N_R	Resolution	—	—	—	—	12	Bit
DNL	Differential Non-linearity	—	V_{REF}=V_{DD}, no load, t_{ADCK}=0.8μs, Ta=-40°C~85°C	-3	—	3	LSB
INL	Integral Non-linearity	—	V_{REF}=V_{DD}, no load, t_{ADCK}=0.8μs, Ta=-40°C~85°C	-4	—	4	LSB
I_{ADC}	Additional Current for A/D Converter Enable	5V	No load (t_{ADCK}=0.8μs)	—	850	1000	μA
t_{ADCK}	Clock Period	—	—	0.8	—	12.8	μs
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t_{ADS}	Sampling Time	—	—	—	4	—	t_{ADCK}
t_{ADC}	A/D Conversion Time (Include ADC Sample and Hold Time)	—	—	—	16	—	t_{ADCK}

OCP Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OCP}	Operating Current	5V	C0HYEN=0 OPAVS[2:0]=110B OPCM[7:0]=10000000B	—	550	800	μA
V_{REF}	DAC Reference Voltage	5V	—	2.2	—	V_{DD}	V
V_{OS_CMP}	Comparator 0 Input Offset Voltage	5V	Without calibration	-15	—	15	mV
V_{HYS0}	Comparator Hysteresis	5V	—	50	100	150	mV
V_{CM_CMP}	Comparator Common Mode Voltage Range	5V	—	0.1	—	V_{DD} -1.4	V
V_{OS_OPA}	OPA Input Offset Voltage	5V	Without calibration (AOF[4:0]=10000B)	-15	—	15	mV
		5V	With calibration	-2	—	2	mV
V_{CM_OPA}	OPA Common Mode Voltage Range	5V	—	0.1	—	V_{DD} -1.4	V
V_{OR}	OPA Maximum Output Voltage Range	5V	—	V_{SS} +0.1	—	V_{DD} -0.1	V
Ga	PGA Gain Accuracy	5V	All gain	-5	—	5	%
DNL	Differential Nonlinearity	5V	DAC V_{REF}=V_{DD}	—	—	±1.5	LSB
INL	Integral Nonlinearity	5V	DAC V_{REF}=V_{DD}	—	—	±2	LSB

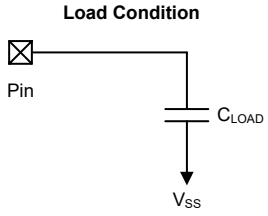
Comparator Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{CMP}	Additional Current for Comparator Enable	5V	—	—	—	200	μA
I_{CSTB}	Comparator Power Down Current	5V	Comparator disable	—	—	0.1	μA
V_{HYS}	Comparator Hysteresis	5V	CnHYEN[1:0]=00B (n=1)	0	0	5	mV
			CnHYEN[1:0]=01B (n=1)	—	10	25	
			CnHYEN[1:0]=10B (n=1)	—	20	35	
			CnHYEN[1:0]=11B (n=1)	—	40	60	
V_{CM}	Input Common Mode Voltage Range	5V	—	0.1	—	V_{DD} -1.4	V
A_{OL}	Comparator Open Loop Gain	5V	—	60	80	—	dB
t_{RP}	Comparator Response Time	5V	With 100mV overdrive(1) (2)	—	200	400	ns

Note: 1. Measured with comparator one input pin at V_{CM}=(0.1V+V_{DD}-1.4V)/2 while the other pin input transition from V_{SS} to (V_{CM}+100mV) or from (V_{DD}-1.4V) to (V_{CM}-100mV).

2. Load Condition: C_{LOAD}=50pF



DNF Electrical Characteristics

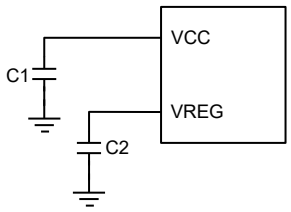
Ta=-40°C~105°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{NFIN}	Digital Noise Filter Capture Pulse Width	—	$f_{SYS}=f_{HIRC}$	8	—	128	$1/f_{SYS}$

LDO Electrical Characteristics

V_{CC}=24V, C1=4.7μF, C2=2.2μF and Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{CC}	Supply Voltage	—	6	—	40	V
I_{CC}	Supply Standby Current	VREG without extra load	—	300	400	μA
V_{REG}	Regulator Output Voltage	VREG with 1mA load	4.9	5.0	5.1	V
I_{LOAD}	VREG Output Current	V_{CC}=6V~40V(without thermal limited)	50	—	—	mA



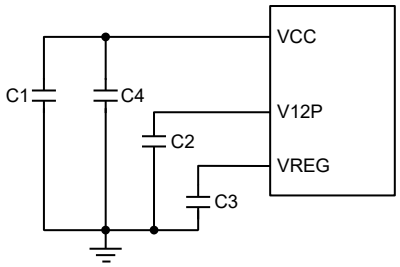
Gate-driver Electrical Characteristics

V_{CC}=24V, C1=4.7μF, C2=2.2μF, C3=2.2μF, C4=22μF, Ta=25˚C, unless otherwise specified

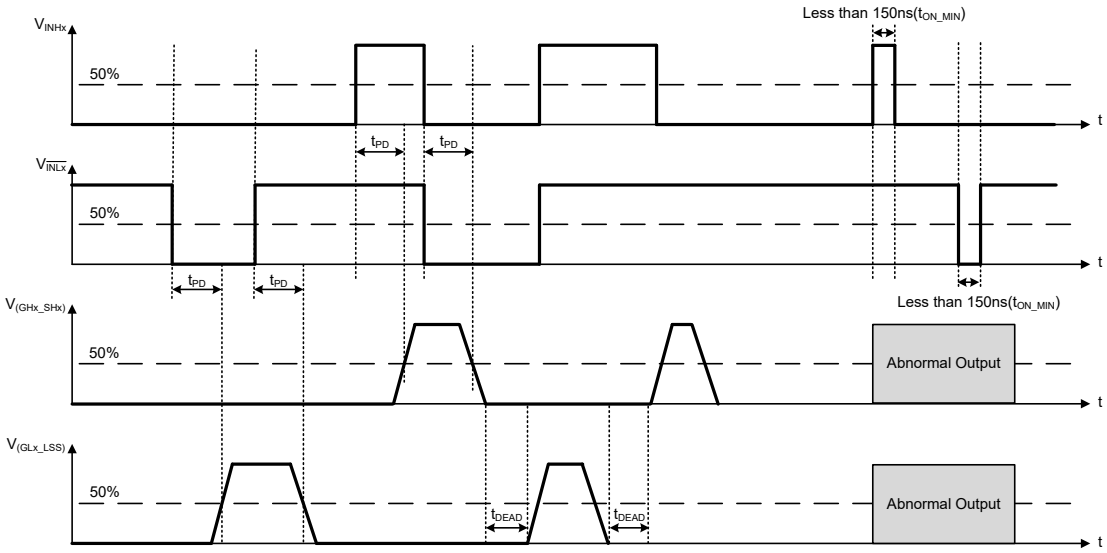
Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
Power Supply						
V_{CC}	Supply Voltage	—	6	—	40	V
Bootstrap						
I_{BST}	Current Consumption from BST	INHx='1' and $\overline{INLx}$='1'	—	80	100	μA
I_{BSTC}	Bootstrap Charging Current	INHx='0' and $\overline{INLx}$='1' (SHx=GND)	—	25	—	mA
Gate-driver (GHx, SHx, GLx)						
V_{GSH}	High-Side V_{GS} Gate Drive – $V_{(GHx,SHx)}$	V_{CC}=6V~13V, f_{PWM}=25kHz	V_{CC}-2	V_{CC}-1.5	—	V
		V_{CC}=13V~40V, f_{PWM}=25kHz	10	12	13	V
V_{GSL}	Low-Side V_{GS} Gate Drive – $V_{(GLx,LSS)}$	V_{CC}=6V~13V	V_{CC}-1	V_{CC}-0.5	—	V
		V_{CC}=13V~40V	10	12	—	V
I_{DRVP}	High-Side and Low-Side Gate Peak Source Current	R_{DRV}=open, C_{GS}=200nF	—	700	—	mA
I_{DRVN}	High-Side and Low-Side Gate Peak Sink Current	R_{DRV}=open, C_{GS}=200nF	—	1000	—	mA
t_{DEAD}	Dead Time	—	—	120	200	ns
t_{DEAD_MIS}	Dead Time Mismatch	Dead time difference between rising and falling edges	—	50	—	ns
t_{PD}	Propagation Delay	INHx to GHx and $\overline{INLx}$ to GLx transition (No connected capacitor with GHx/GLx)	—	40	200	ns
t_{PD_MIS}	High-Side/Low-Side Propagation Delay Mismatch	Propagation delay difference between different phases or different sides	—	20	—	ns

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
t_{ON_MIN}	Minimum Input Pulse Width(Note)	—	—	—	150	ns
R_{OFF1}	Low-Side Gate Hold-off Resistor	GLx to LSS	—	200	—	kΩ
R_{OFF2}	High-Side Gate Hold-off Resistor	GHx to SHx	—	400	—	kΩ
Protections						
V_{CC_UVLO+}	V_{CC} Turn On Level	V_{CC} rises	—	5.5	6	V
V_{CC_UVLO-}	V_{CC} Turn Off Level	V_{CC} falls	4.5	5.0	—	V
V_{REG_UVLO+}	V_{REG} Turn On Level	V_{REG} rises	—	—	4.0	V
V_{REG_UVLO-}	V_{REG} Turn Off Level	V_{REG} falls	3.0	—	—	V
V_{12P_UVLO+}	V_{12P} Turn On Level	V_{12P} rises, $\overline{INLx}$='0'	—	5.5	6	V
V_{12P_UVLO-}	V_{12P} Turn Off Level	V_{12P} falls, $\overline{INLx}$='0'	4.2	5.0	—	V
V_{BST_UVLO+}	$V_{(BSTx,SHx)}$ Turn On Level	$V_{(BSTx,SHx)}$ rises, INHx='1'	—	3.7	4.2	V
V_{BST_UVLO-}	$V_{(BSTx,SHx)}$ Turn Off Level	$V_{(BSTx,SHx)}$ falls, INHx='1'	2.2	2.6	—	V
T_{SHD}	Thermal Shutdown Threshold	—	—	160	—	˚C
T_{REC}	Thermal Recovery Threshold	—	—	120	—	˚C

Note: When the INHx or $\overline{INLx}$ input signal pulse width is less than t_{ON_MIN}, the output may malfunction.



Timing Diagram

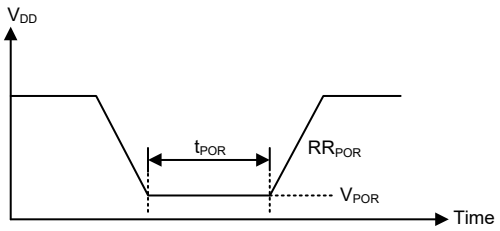


Gate Drive Timing

Power-on Reset Characteristics

Ta=25°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



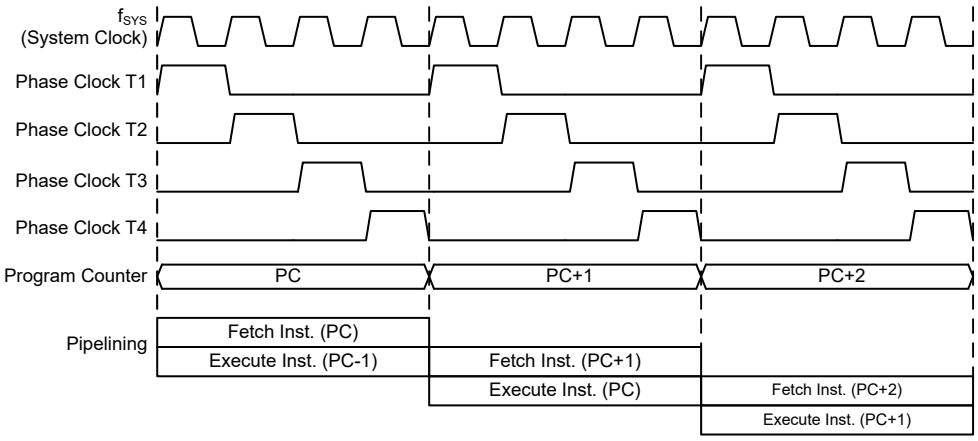
System Architecture

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 need one more cycle. An 8-bit wide ALU is used in practically all instruction set operations, which carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O and A/D control system with maximum reliability and flexibility. This makes the device suitable for affordable, high-volume production for controller applications.

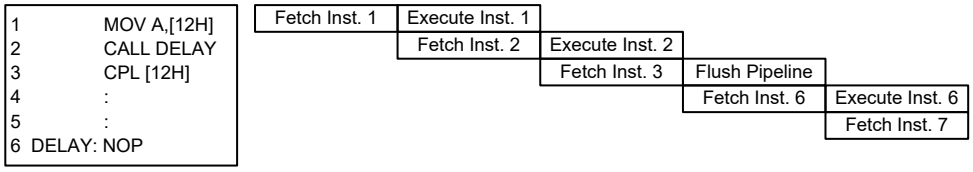
Clocking and Pipelining

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

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

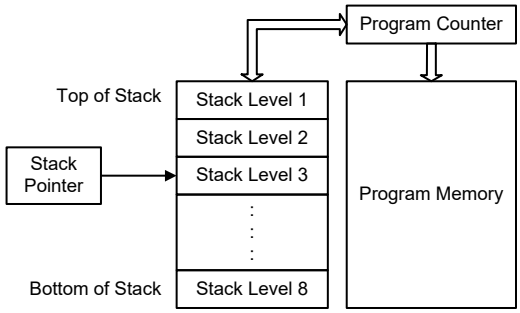
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. 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, LRRA, LRRCA, LRRC, LRLA, LRL, LRLCA, LRLC
- Increment and Decrement:
  INCA, INC, DECA, DEC,
  LINCA, LINC, LDECA, LDEC

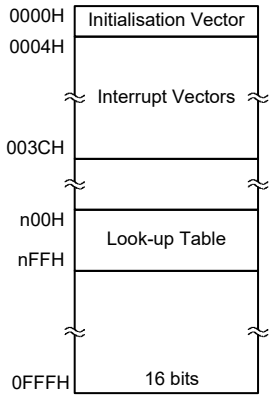
- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI,
  LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

OTP Program Memory

The Program Memory is the location where the user code or program is stored. The device is supplied with One-Time Programmable, OTP memory where users can program their application code into the device.

Structure

The Program Memory has a capacity of 4K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer registers.



Program Memory Structure

Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 0000H 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 registers, TBLP and TBHP. These registers define the total address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the Program Memory using the “TABRD [m]” or “TABRDL [m]” instruction respectively. When the memory [m] is located in sector 0. If the memory [m] is located in other sectors except sector 0, the data can be retrieved from the program memory using the corresponding extended table read instruction “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. Any unused bits in this transferred higher order byte will be read as “0”.

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

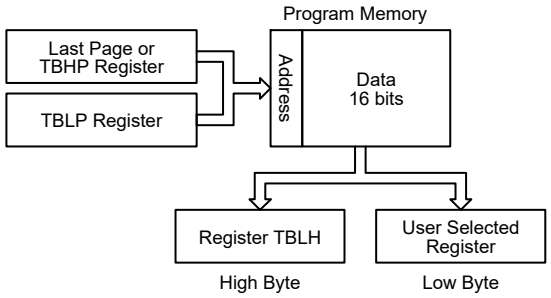


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "0F00H" which refers to the start address of the last page within the 4K words Program Memory of the device. The table pointer low byte register is set here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by the TBLP and TBHP registers if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" 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           ; sets initial address of last page
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
```

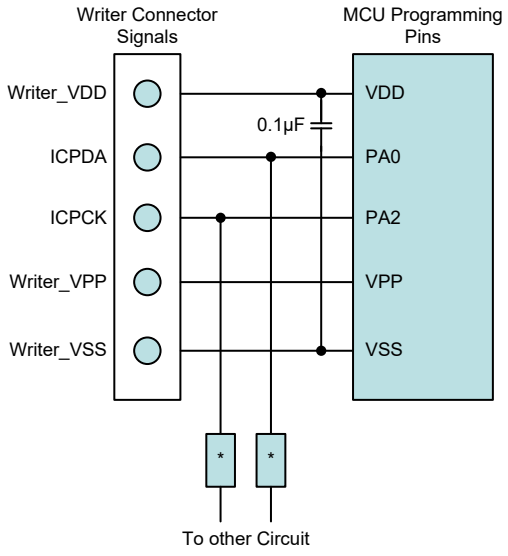
In Circuit Programming – ICP

The OTP type Program Memory is provided for users to program their application code one time into the device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 5-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with an un-programmed microcontroller, and then programming the program at a later stage.

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VPP	VPP	Programming OTP ROM power supply (8.5V)
VDD	VDD	Power Supply. A 0.1μF capacitor is required to be connected between VDD and VSS for programming.
VSS	VSS	Ground

The Program Memory can be programmed serially in-circuit using this 5-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Three additional lines are required for the power supply. The technical details regarding the in-circuit programming of the device is beyond the scope of this document and will be supplied in supplementary literature.

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



Note: 1. A 0.1μF capacitor is required to be connected between VDD and VSS for ICP programming, and as close to these pins as possible.

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

On-Chip Debug Support – OCDS

There is an EV chip named BD66EVM2541G which can emulate the BD66RM2541G device. The EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function and the package type. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Program 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	MCU Chip Pins	Pin Description
OCDSDA	OCDSDA	On-Chip Debug Support Data/Address input/output
OCDSCK	OCDSCK	On-Chip Debug Support Clock input
VDD	VDD	Power Supply
VSS	VSS	Ground

OTP ROM Parameter Program – ORPP

This device contains an ORPP function. The provision of the ORPP function offers users the convenience of OTP Memory programming features. Note that the Write operation only writes data to the last page of OTP Program Memory, and the data can only be written once and cannot be erased.

Before the write operation is implemented, the VPP pin must be connected to an 8.5V power and after the write operation is completed, the high voltage power should be removed from the VPP pin. If the VPP function is pin-shared with an I/O port, the correspoonding I/O port cannot be set as an output when it is used as the VPP function.

ORPP Registers

Three registers control the overall operation of the internal ORPP function. These are data registers ODL and ODH, and a control register OCR, which are all located in Sector 1. The ODL and ODH registers can be addressed directly using the extended instructions or can be read from or written to indirectly using the MP1H/ MP1L or MP2H/MP2L Memory Pointer pairs and Indirect Addressing Register, IAR1 or IAR2. However, the OCR regiser can only be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pair and Indirect Addressing Register, IAR1 or IAR2. Because the OCR control register is located at address 43H in Sector 1, the Memory Pointer low byte register, MP1L or MP2L, must first be set to the value 43H and the Memory Pointer high byte register, MP1H or MP2H, set to the value, 01H, before any operations on the OCR register are executed.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OCR	—	—	—	—	WREN	WR	—	—
ODL	D7	D6	D5	D4	D3	D2	D1	D0
ODH	D15	D14	D13	D12	D11	D10	D9	D8

ORPP Register List

• **ODL 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**: ORPP program memory data bit 7~bit 0

• **ODH 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**: ORPP program memory data bit 15~bit 8

• **OCR Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3 **WREN**: ORPP Write Enable
0: Disable
1: Enable
This is the ORPP Write Enable Bit which must be set high before write operations are carried out. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Clearing this bit to zero will inhibit ORPP write operations.

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

Bit 1~0 Unimplemented, read as "0"

Note: 1. The WREN and WR cannot be set high at the same time in one instruction.
2. Note that the CPU will be stopped when a write operation is successfully activated.
3. Ensure that the f_{SUB} clock is stable before executing the write operation.
4. Ensure that the write operation is totally complete before executing other operations.

ORPP Writing Data to the OTP Program Memory

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

ORPP Reading Data from the OTP Program Memory

For ORPP read operation the desired address should first be placed in the TBLP register. Then the data can be retrieved from the program memory using the "TABRDL [m]" instruction. 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.

Programming Considerations

Care must be taken that data is not inadvertently written to the OTP Program Memory. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Memory Pointer high byte register, MP1H or MP2H, could be normally cleared to zero as this would inhibit access to Sector 1 where the ORPP control register exists. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process.

When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write cycle is executed and then set high again after a write activation procedure has completed. Note that the device should not enter the IDLE or SLEEP mode until the ORPP write operation is totally complete. Otherwise, the ORPP write operation will fail.

Programming Examples

ORPP Reading Data from the OTP Program Memory

```
Tempreg1 db?         ; temporary register
MOV  A, 03H
MOV  TBLP, A         ; set read address 03H
TABRDL Tempreg1      ; transfers value in table (last page) referenced by table
                     ; pointer, data at program memory address "0F03H" transferred
                     ; to tempreg1 and TBLH
```

ORPP Writing Data to the OTP Program Memory

```
MOV  A, ORPP_ADRES    ; user defined address
MOV  TBLP, A
LMOV A, ORPP_DATA_L   ; user defined data
LMOV ODL, A
LMOV A, ORPP_DATA_H
LMOV ODH, A
MOV  A, 43H           ; set memory pointer low byte MP1L
MOV  MP1L, A          ; MP1L points to OCR register
MOV  A, 01H           ; set memory pointer high byte MP1H
MOV  MP1H, A
CLR  IAR1
CLR  EMI
SET  IAR1.3           ; set WREN bit, enable write operation
SET  IAR1.2           ; start Write Cycle - set WR bit - executed immediately
                      ; after setting WREN bit
SET  EMI
BACK:
SZ   IAR1.2           ; check for write cycle end
JMP  BACK
CLR  MP1H
```

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.

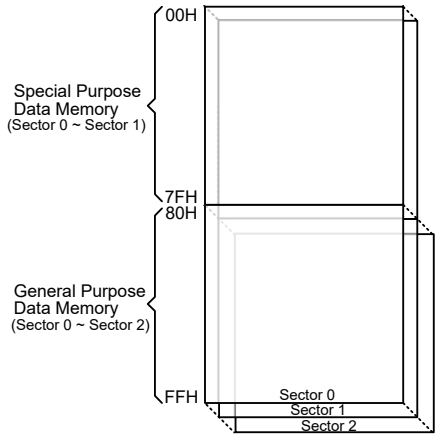
Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide RAM. Each of the Data Memory Sector is categorized into two types, the special Purpose Data Memory and the General Purpose Data Memory. 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.

Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value if using the indirect addressing method.

Special Purpose Data Memory	General Purpose Data Memory	
Located Sectors	Capacity	Sector: Address
0, 1	384×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH

Data Memory Summary



Data Memory Structure

Data Memory Addressing

For the device that supports the extended instructions, there is no Bank Pointer for Data Memory addressing. For Data Memory the desired Sector is pointed by the MP1H or MP2H register and the certain Data Memory address in the selected sector is specified by the MP1L or MP2L register when using indirect addressing access.

Direct Addressing can be used in all sectors using the extended instructions which can address all available data memory space. For the accessed data memory which is located in any data memory sectors 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 10 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 programming 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	RSTFC	
10H	HIRCC	
11H	LVDC	
12H	LVRC	
13H	SCC	
14H	PA	PAS0
15H	PAC	PAS1
16H	PAPU	
17H	PAWU	
18H		PCS0
19H		PCS1
1AH		
1BH	PC	
1CH	PCC	
1DH	PCPU	
1EH		
1FH		
20H		
21H		
22H	PSCR	SADC0
23H	WDTC	SADC1
24H	TBC	
25H	INTEG	ADCR2
26H	INTC0	ADDL
27H	INTC1	
28H	INTC2	CMPC0
29H	INTC3	CMPC1
2AH	MFI0	OPOMS
2BH	MFI1	OPCM
2CH	MFI2	OPACAL
2DH	MFI3	SADOL
2EH	MFI4	SADOH
2FH	MFI5	ADLVDL
30H	MFI6	ADLVDH
31H		ADHVDL
32H	PTMC0	ADHVDH
33H	PTMC1	
34H	PTMC2	
35H	PTMDL	
36H	PTMDH	
37H	PTMAL	
38H	PTMAH	
39H	PTMBL	
3AH	PTMBH	
3BH	PTMRPL	
3CH	PTMRPH	
3DH		
3EH	CTM0C0	
3FH	CTM0C1	

	Sector 0	Sector 1
40H	CTM0DL	
41H	CTM0DH	
42H	CTM0AL	
43H	CTM0AH	OCR
44H	CTM0RP	ODL
45H		ODH
46H		
47H	CTM1C0	
48H	CTM1C1	
49H	CTM1DL	
4AH	CTM1DH	
4BH	CTM1AL	
4CH	CTM1AH	
4DH	CTM1RP	
4EH		
4FH		
50H	CTM2C0	IFS
51H	CTM2C1	
52H	CTM2DL	PWMC
53H	CTM2DH	PWMCS
54H	CTM2AL	DUTR0L
55H	CTM2AH	DUTR0H
56H	CTM2RP	DUTR1L
57H		DUTR1H
58H		
59H	MDUWR0	
5AH	MDUWR1	PRDRL
5BH	MDUWR2	PRDRH
5CH	MDUWR3	PWMRL
5DH	MDUWR4	PWMRH
5EH	MDUWR5	PWMME
5FH	MDUWCTRL	PWMMD
60H		MCF
61H		MCD
62H		DTS
63H		PLC
64H		HINTEG
65H		HDCR
66H		HDCD
67H	USR	HDCT0
68H	UCR1	HDCT1
69H	UCR2	HDCT2
6AH	UCR3	HDCT3
6BH	BRDH	
6CH	BRDL	
6DH	UFCR	
6EH	TXR_RXR	
6FH	RxCNT	
70H		
71H	INHLVC	
72H	TLVRC	
73H		HCHK_NUM
74H	NF_VIH	HNF_MSEL
75H	NF_VIL	MPTC1
76H		MPTC2
77H		OCPS
78H		CAPTC0
79H		CAPTC1
7AH		CAPTMDL
7BH		CAPTMDH
7CH		CAPTMAL
7DH		CAPTMAH
7EH		CAPTMCL
7FH		CAPTMCH

: Unused, read as 00H

: Reserved, cannot be changed

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 mp1h, a
    mov a, offset adres1    ; Accumulator loaded with first RAM address
    mov mp1l, a             ; setup memory pointer with first RAM address
loop:
    clr IAR1                ; clear the data at address defined by MP1L
    inc mp1l                ; 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 Byte 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 operations are selected through a combination of configuration options and the relevant control registers.

Oscillator Overview

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

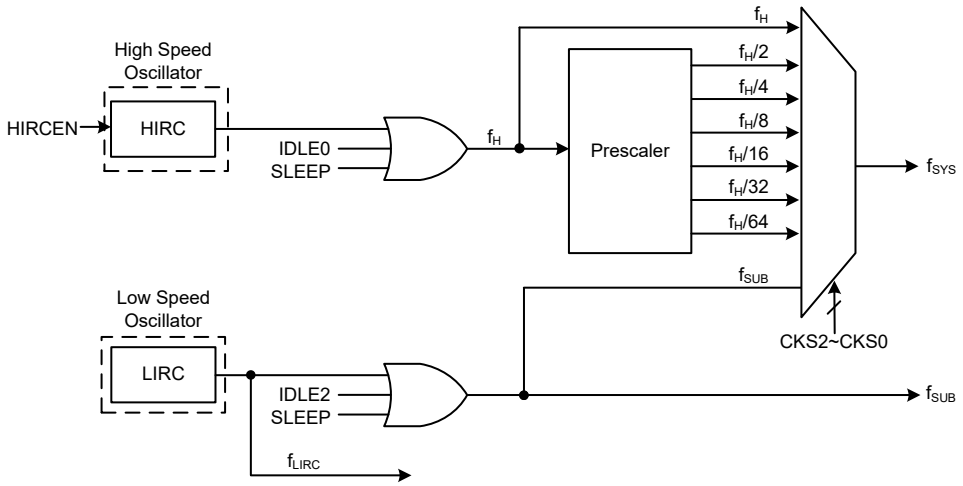
Type	Name	Frequency
Internal High Speed RC	HIRC	16/20MHz
Internal Low Speed RC	LIRC	32kHz

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 16/20MHz 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 oscillator is implemented using the CKS2~CKS0 bits in the SCC register and the system clock can be dynamically selected.

The frequency of the slow speed or high speed system clock is determined using the CKS2~CKS0 bits in the SCC register.



System Clock Configurations

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal high speed RC oscillator has two fixed frequencies of 16MHz and 20MHz, which is selected using a configuration option. The HIRC1~HIRC0 bits in the HIRCC register must also be setup to match the selected configuration option frequency. Setting up these bits is necessary to ensure that the HIRC frequency accuracy specified in the A.C. Characteristics is achieved. Devices 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 a fully integrated low frequency RC oscillator with a typical frequency of 32kHz, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

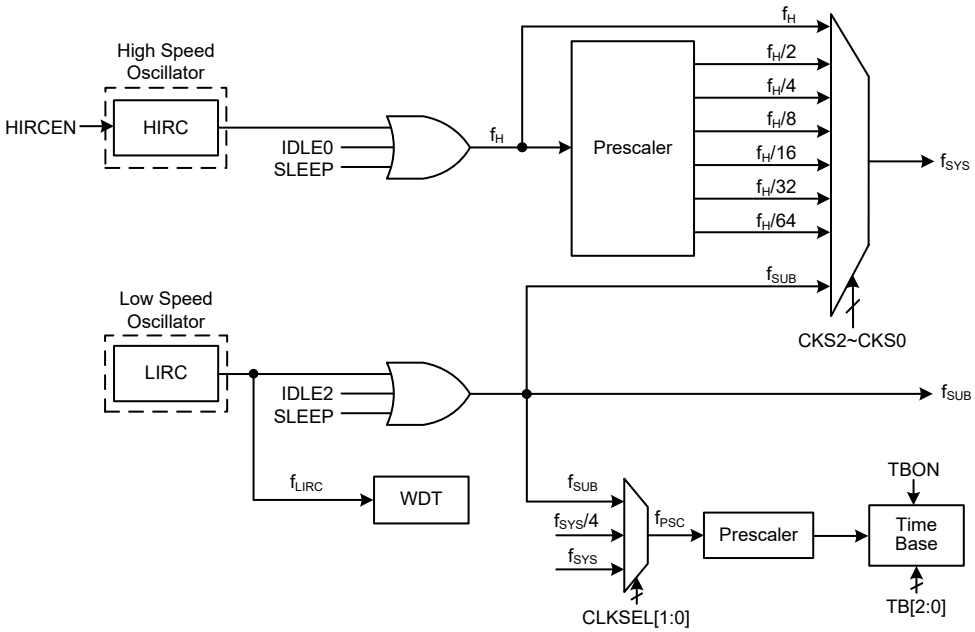
Operating Modes and System Clocks

Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice versa, lower speed clocks reduce current consumption. As Holtek has provided the device with both high and low speed clock sources and the means to switch between them dynamically, the user can optimise the operation of their microcontroller to achieve the best performance/power ratio.

System Clocks

The device has many different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock options using register programming, a clock system can be configured to obtain maximum application performance.

The main system clock can come from either a high speed frequency, f_H, or a low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.



Device Clock Configurations

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

System Operation Modes

There are six different modes of operation for the microcontrollers, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the FAST Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Modes are used when the microcontroller CPU is switched off to conserve power.

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

"x": Don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.

2. The f_{LIRC} clock can be on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

FAST Mode

This is one of the main operating modes where the microcontrollers have all of their 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 which will come from the high speed oscillator, HIRC. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontrollers at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}, which is derived from the LIRC oscillator.

SLEEP Mode

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

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN

System Operating Mode Control Register List

• **SCC Register**

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

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

Bit 4~2 Unimplemented, read as "0"

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

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

Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.
Clock switching delay time=$4\times t_{SYS}+[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$, where $t_{Curr.}$ indicates the current clock period, $t_{Tar.}$ indicates the target clock period and t_{SYS} indicates the current system clock period.

• **HIRCC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection
00: Reserved
01: 16MHz
10: 20MHz
11: Reserved
When the HIRC oscillator is enabled or the HIRC frequency selection is changed by the application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1.
It is recommended that the HIRC frequency selected by these two bits should be the same with the frequency determined by the configuration option to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

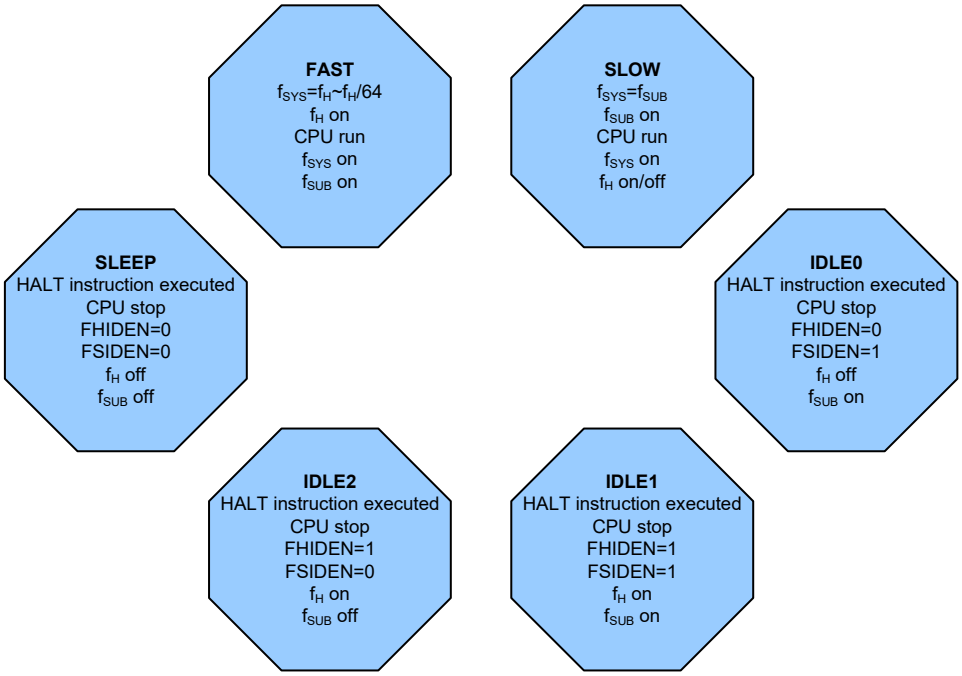
Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set high to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set high after the HIRC oscillator is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

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

Operating Mode Switching

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

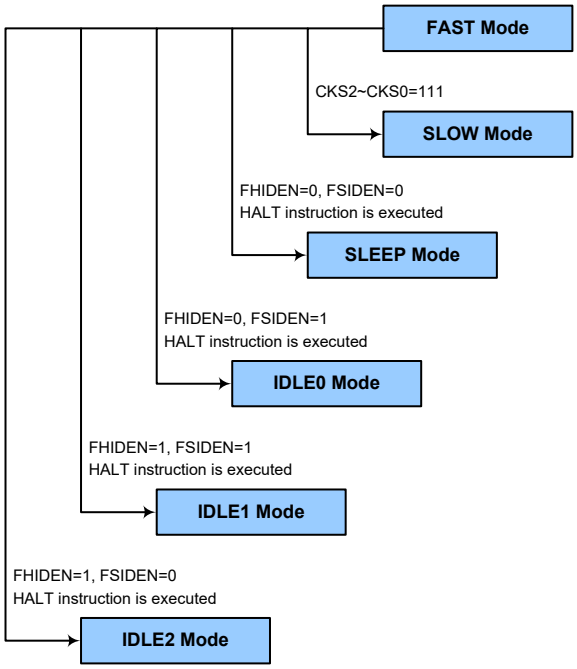
In simple terms, mode switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



FAST Mode to SLOW Mode Switching

When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by setting the CKS2~CKS0 bits to "111" in the SCC register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

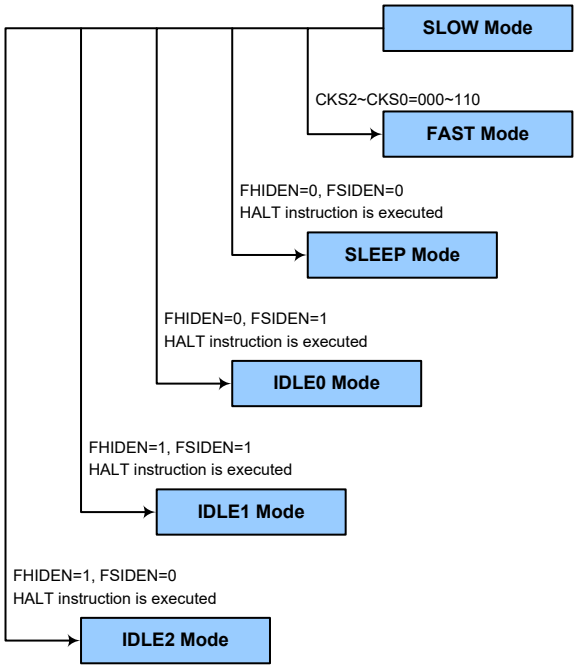
The SLOW Mode system clock is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

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

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



Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "0". In this mode all the clocks and functions will be switched off except the WDT function. When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE0 Mode

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

- The f_H clock will be stopped and the application program will stop at the "HALT" instruction, but the f_{SUB} clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE1 Mode

There is only one way for the device to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The f_H and f_{SUB} clocks will be on but the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE2 Mode

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

- The f_H clock will be on but the f_{SUB} clock will be off and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These must either be set as outputs or if set as inputs must have pull-high resistors connected. In addition, the I/O pin-shared with VPP must not be set to output high, as this could result in increased current consumption.

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

In the IDLE1 and IDLE2 Mode the high speed oscillator is on, if the peripheral function clock source is derived from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

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

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

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

When the device executes the "HALT" instruction, it will enter the IDLE or SLEEP mode and the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction. If the system is woken up by a WDT overflow, a Watchdog Timer reset will be initiated and the TO flag will be set to 1. The TO flag is set if a WDT time-out occurs and causes a wake-up that only resets the Program Counter and Stack Pointer, other flags remain in their original status.

Each pin on Port A can be set using the PAWU register to permit a negative transition on the pin to wake up the system. When a pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction. If the system is woken up by an interrupt, then two

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{LIRC}. 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 time-out periods, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period, Watchdog Timer enable/disable and software reset MCU.

• **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
10101: Disable
01010: Enable
Others: Reset MCU
When these bits are changed by the environmental noise or software setting to reset the microcontroller, the reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set high.

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

• **RSTFC Register**

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

"x": Unknown

Bit 7~3 Unimplemented, read as "0"

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

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

Bit 0 **WRF**: WDT control register software reset flag
0: Not occurred
1: Occurred
This bit is set high by the WDT control register software reset and cleared by the application program. Note that this bit can only be cleared to zero 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/disable control and MCU reset. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B while the WDT function will be enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after a delay time, t_{SRESET}. After power-on these bits will have a value of 01010B.

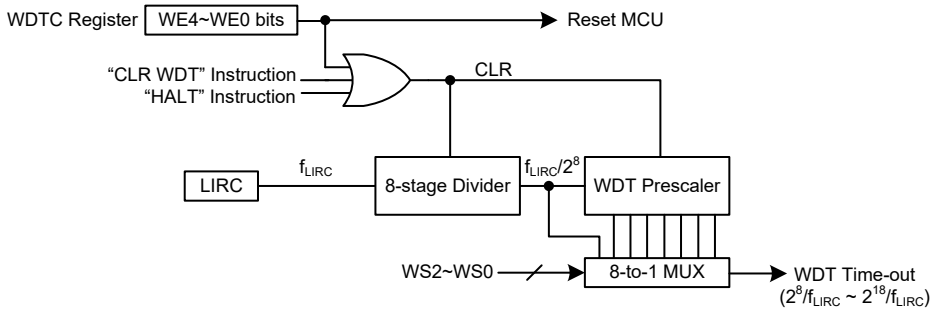
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other values	Reset MCU

Watchdog Timer Function Control

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

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

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



Watchdog Timer

Reset and Initialisation

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

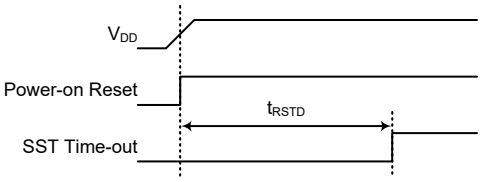
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

Reset Functions

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

Power-on Reset

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



Power-on Reset Timing Chart

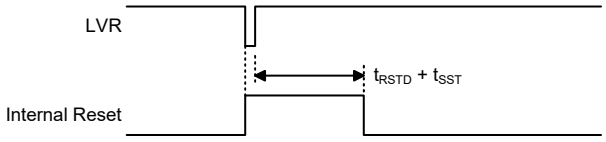
Low Voltage Reset – LVR

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

The LVR function is always enabled in FAST/SLOW mode with a specific LVR voltage, V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing

the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVR/LVD Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual t_{LVR} value can be selected by the TLVR1~TLVR0 bits in the TLVRC register.

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 the environmental noise or software setting, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01100110B. Note that the LVR function will be automatically disabled when the device enters the IDLE/SLEEP mode.



Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select control
01100110: 3.15V
01010101: 3.15V
00110011: 3.15V
10011001: 3.15V
10101010: 3.15V
11110000: 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, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps more than a t_{LVR} time. The actual t_{LVR} value can be selected by the TLVR1~TLVR0 bits in the TLVRC register. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than defined six register values above, will 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.

- **TLVRC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TLVR1	TLVR0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	1

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **TLVR1~TLVR0**: Minimum low voltage width to reset (t_{LVR}) selection
00: (7~8)×t_{LIRC}
01: (31~32)×t_{LIRC}
10: (63~64)×t_{LIRC}
11: (127~128)×t_{LIRC}

• **RSTFC Register**

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

"x": Unknown

Bit 7~3 Unimplemented, read as "0"

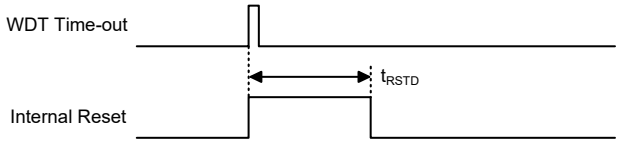
Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set high when a specific Low Voltage Reset situation occurs. This bit can only be cleared to 0 by the application program.

Bit 1 **LRF**: LVR control register software reset flag
0: Not occurred
1: Occurred
This bit is set high if the LVRC register contains any non-defined register setting 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
Refer to the "Watchdog Timer Control Register" section.

Watchdog Time-out Reset during Normal Operation

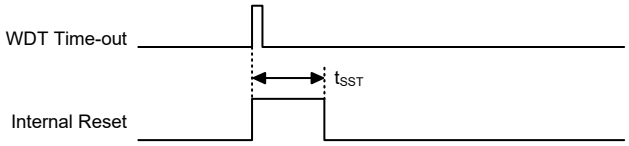
The Watchdog time-out flag TO will be set to "1" when the Watchdog time-out Reset during normal operation.



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 and PDF flags will be set to "1". Refer to the System Start Up Time Characteristics for t_{SST} details.



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

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

"u" stands for unchanged

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

Item	Condition after RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Base	Cleared after reset, WDT begins counting
Timer 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.

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	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	---- xxxx	---- uuuu	---- uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu
HIRCC	---- 0001	---- 0001	---- uuuu
LVDC	0000 -000	0000 -000	uuuu –uuu
LVRC	0110 0110	0110 0110	uuuu uuuu
SCC	010- --00	010- --00	uuu- --uu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PC	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	uuuu uuuu

Register	Power-on Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PCPU	0000 0000	0000 0000	uuuu uuuu
PSCR	---- --00	---- --00	---- --uu
WDTC	0101 0011	0101 0011	uuuu uuuu
TBC	0--- -000	0--- -000	u--- -uuu
INTEG	---- --00	---- --00	---- -- uu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	00-0 00-0	00-0 00-0	uu-u uu-u
MFI0	---0 ---0	---0 ---0	---u ---u
MFI1	0000 0000	0000 0000	uuuu uuuu
MFI2	0-00 0-00	0-00 0-00	u-uu u-uu
MFI3	--00 --00	--00 --00	--uu --uu
MFI4	--00 --00	--00 --00	--uu --uu
MFI5	--00 --00	--00 --00	--uu --uu
MFI6	--00 --00	--00 --00	--uu --uu
PTMC0	0000 0---	0000 0---	uuuu u---
PTMC1	0000 0000	0000 0000	uuuu uuuu
PTMC2	---- -000	---- -000	---- -uuu
PTMDL	0000 0000	0000 0000	uuuu uuuu
PTMDH	0000 0000	0000 0000	uuuu uuuu
PTMAL	0000 0000	0000 0000	uuuu uuuu
PTMAH	0000 0000	0000 0000	uuuu uuuu
PTMBL	0000 0000	0000 0000	uuuu uuuu
PTMBH	0000 0000	0000 0000	uuuu uuuu
PTMRPL	0000 0000	0000 0000	uuuu uuuu
PTMRPH	0000 0000	0000 0000	uuuu uuuu
CTM0C0	0000 0---	0000 0---	uuuu u---
CTM0C1	0000 0000	0000 0000	uuuu uuuu
CTM0DL	0000 0000	0000 0000	uuuu uuuu
CTM0DH	0000 0000	0000 0000	uuuu uuuu
CTM0AL	0000 0000	0000 0000	uuuu uuuu
CTM0AH	0000 0000	0000 0000	uuuu uuuu
CTM0RP	0000 0000	0000 0000	uuuu uuuu
CTM1C0	0000 0---	0000 0---	uuuu u---
CTM1C1	0000 0000	0000 0000	uuuu uuuu
CTM1DL	0000 0000	0000 0000	uuuu uuuu
CTM1DH	0000 0000	0000 0000	uuuu uuuu
CTM1AL	0000 0000	0000 0000	uuuu uuuu
CTM1AH	0000 0000	0000 0000	uuuu uuuu
CTM1RP	0000 0000	0000 0000	uuuu uuuu
CTM2C0	0000 0---	0000 0---	uuuu u---
CTM2C1	0000 0000	0000 0000	uuuu uuuu
CTM2DL	0000 0000	0000 0000	uuuu uuuu
CTM2DH	0000 0000	0000 0000	uuuu uuuu
CTM2AL	0000 0000	0000 0000	uuuu uuuu
CTM2AH	0000 0000	0000 0000	uuuu uuuu

Register	Power-on Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
CTM2RP	0000 0000	0000 0000	uuuu uuuu
MDUWR0	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR1	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR2	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR3	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR4	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR5	xxxx xxxx	0000 0000	uuuu uuuu
MDUWCTRL	00-- ----	00-- ----	uu-- ----
USR	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	uuuu uuuu
UCR3	---- -000	---- -000	---- -uuu
BRDH	0000 0000	0000 0000	uuuu uuuu
BRDL	0000 0000	0000 0000	uuuu uuuu
UFCR	--00 0000	--00 0000	--uu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
RxCNT	---- -000	---- -000	---- -uuu
INHLVC	---- 0000	---- 0000	---- uuuu
TLVRC	---- --01	---- --01	---- --uu
NF_VIH	00-0 0010	00-0 0010	uu-u uuuu
NF_VIL	00-- ----	00-- ----	uu-- ----
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PCS0	0000 0000	0000 0000	uuuu uuuu
PCS1	0000 00--	0000 00--	uuuu uu--
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	000- -000	000- -000	uuu- -uuu
ADCR2	0000 0-00	0000 0-00	uuu u-uu
ADDL	0000 0000	0000 0000	uuuu uuuu
CMPC0	---- 00-0	---- 00-0	---- uu-u
CMPC1	---- --00	---- --00	---- --uu
OPOMS	110- -000	110- -000	uuu- -uuu
OPCM	0000 0000	0000 0000	uuuu uuuu
OPACAL	00-1 0000	00-1 0000	uu-u uuuu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
ADLVDL	0000 ----	0000 ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)

Register	Power-on Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
ADLVDH	0000 0000	0000 0000	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
ADHVDL	1111 ----	1111 ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
ADHVDH	1111 1111	1111 1111	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
OCR	---- 00--	---- 00--	---- uu--
ODL	0000 0000	0000 0000	uuuu uuuu
ODH	0000 0000	0000 0000	uuuu uuuu
IFS	--00 0000	--00 0000	--uu uuuu
PWMC	0000 0000	0000 0000	uuuu uuuu
PWMCS	---- 0000	---- 0000	---- uuuu
DUTR0L	0000 0000	0000 0000	uuuu uuuu
DUTR0H	---- --00	---- --00	---- --uu
DUTR1L	0000 0000	0000 0000	uuuu uuuu
DUTR1H	---- --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	---- 0000	---- 0000	---- uuuu
PWMMD	---- 0000	---- 0000	---- uuuu
MCF	0--- 0100	0--- 0100	u--- uuuu
MCD	--00 0--x	--00 0--x	--uu u--u
DTS	0000 0000	0000 0000	uuuu uuuu
PLC	---- 0000	---- 0000	---- uuuu
HINTEG	-0-- --00	-0-- --00	-u-- --uu
HDCR	000- 0000	000- 0000	uuu- uuuu
HDCD	---- ---0	---- ---0	---- ---u
HDCT0	--00 00--	--00 00--	--uu uu--
HDCT1	--00 00--	--00 00--	--uu uu--
HDCT2	--00 00--	--00 00--	--uu uu--
HDCT3	--00 00--	--00 00--	--uu uu--
HCHK_NUM	---0 0000	---0 0000	---u uuuu
HNF_MSEL	---- 0000	---- 0000	---- uuuu
MPTC1	0000 0000	0000 0000	uuuu uuuu
MPTC2	-000 0000	-000 0000	-uuu uuuu
OCPS	---- 0000	---- 0000	---- 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

Register	Power-on Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
CAPTMAH	0000 0000	0000 0000	uuuu uuuu
CAPTMCL	xxxx xxxx	xxxx xxxx	uuuu uuuu
CAPTMCH	xxxx xxxx	xxxx xxxx	uuuu uuuu

Note: "u" stands for unchanged
"x" stands for unknown
"-" stands for unimplemented

Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
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
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
INHLVC	—	—	—	—	INHLVC3	D2	INHLVC1	INHLVC0

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 a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using registers, namely PAPU and PCPU, 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 or C. 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 register only when the pin is selected as a general purpose input and the MCU enters 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

PAWUn: Port A 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 PCC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

• **PxC Register**

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

PxCn: I/O port x pin 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 or C, However, the actual available bits for each I/O Port may be different.

For the internally connected lines, PC0/GAT~PC3/GBB and PC4, the relevant PxCn bit should be properly configured depending on the function that each line is actually used for internal connection. Set the line as an output or an input with pull-high if necessary.

I/O Port Input Voltage Level Selection

The I/O ports can be configured with different input voltage levels, which is selected by the corresponding pin input voltage level select bits. Users should refer to the Input/Output Characteristics section to obtain the exact value for different applications.

• **INHLVC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INHLVC3	D2	INHLVC1	INHLVC0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **INHLVC3**: PC7~PC5 input V_{IL} & V_{IH} selection
0: V_{IL} & V_{IH} selection = Level 0
1: V_{IL} & V_{IH} selection = Level 1

Bit 2 **D2**: Reserved, must be fixed at "0"

Bit 1 **INHLVC1**: PA7~PA4 input V_{IL} & V_{IH} selection
0: V_{IL} & V_{IH} selection = Level 0
1: V_{IL} & V_{IH} selection = Level 1

Bit 0 **INHLVC0**: PA3~PA0 input V_{IL} & V_{IH} selection
0: V_{IL} & V_{IH} selection = Level 0
1: V_{IL} & V_{IH} selection = Level 1

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 PxSn, and input function selection register, labeled as IFS, which can select the desired functions of the multi-function pin-shared pins.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as INT, PTCK, PTPI, 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 bits. 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
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	—	—
IFS	—	—	IFS5	IFS4	IFS3	IFS2	IFS1	IFS0

"—": unimplemented, read as "0"

Pin-shared Function Selection Register List

- **PAS0 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
01: C1N
10: AN2
11: PA3

Bit 5~4 **PAS05~PAS04**: PA2 pin-shared function selection
00: PA2/PTCK
01: RXFF/TXFF
10: GBB
11: AN3

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00: PA1
01: AP
10: AN0
11: PA1

Bit 1~0 **PAS01~PAS00**: PA0 pin-shared function selection
00: PA0
01: TXFF
10: GAB
11: AN4

• **PAS1 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/PTPI/INT/NFIN
01: RXFF/TXFF
10: AN1
11: PA7/PTPI/INT/NFIN

Bit 5~4 **PAS15~PAS14**: PA6 pin-shared function selection
00: PA6/VPP
01: HAO
10: PA6/VPP
11: PA6/VPP

Bit 3~2 **PAS13~PAS12**: PA5 pin-shared function selection
00: PA5/NFIN
01: OPAO
10: AN7
11: PA5/NFIN

Bit 1~0 **PAS11~PAS10**: PA4 pin-shared function selection
00: PA4/H1
01: C1P
10: PA4/H1
11: PA4/H1

• **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
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 **PCS07~PCS06**: PC3 pin-shared function selection
00: PC3
01: Reserved
10: GBB
11: Reserved

Bit 5~4 **PCS05~PCS04**: PC2 pin-shared function selection
00: PC2
01: Reserved
10: GBT
11: Reserved

Bit 3~2 **PCS03~PCS02**: PC1 pin-shared function selection
00: PC1
01: Reserved
10: GAB
11: Reserved

Bit 1~0 **PCS01~PCS00**: PC0 pin-shared function selection
00: PC0
01: GAT
10: Reserved
11: Reserved

• **PCS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	—	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	—
POR	0	0	0	0	0	0	—	—

Bit 7~6 **PCS17~PCS16**: PC7 pin-shared function selection
00: PC7/INT/CTCK2
01: PTPB
10: CTP0B
11: CTP1B

Bit 5~4 **PCS15~PCS14**: PC6 pin-shared function selection
00: PC6/CTIN/CTCK2
01: CTP1
10: TXFF
11: AN5

Bit 3~2 **PCS13~PCS12**: PC5 pin-shared function selection
00: PC5
01: PTP
10: RXFF/TXFF
11: AN6

Bit 1~0 Unimplemented, read as "0"

• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	IFS5	IFS4	IFS3	IFS2	IFS1	IFS0
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 5~4 **IFS5~IFS4**: RXFF/TXFF input source pin selection
00: PC5
01: PA7
10: PA2
11: PC5

Bit 3 **IFS3**: CTCK2 input source pin selection
0: PC6
1: PC7

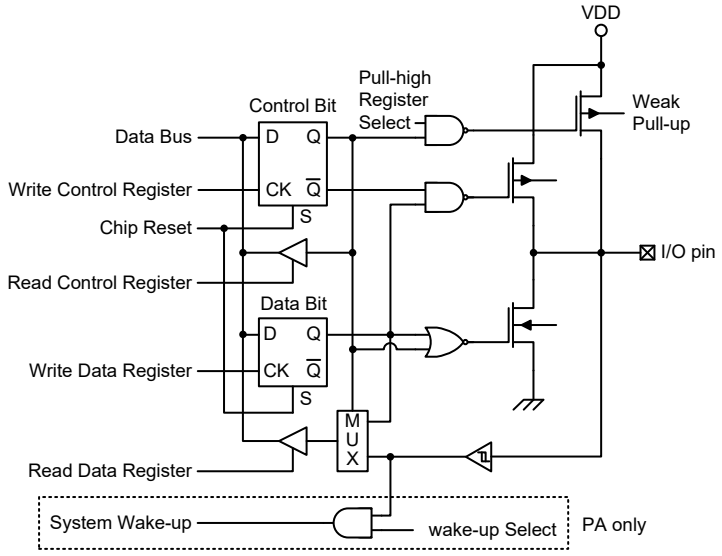
Bit 2 **IFS2**: INT input source pin selection
0: PC7
1: PA7

Bit 1 **IFS1**: H1 input source pin selection
0: PA4
1: Reserved

Bit 0 **IFS0**: NFIN input source pin selection
0: PA7
1: PA5

I/O Pin Structures

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



Logic Function Input/Output Structure

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.

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

Introduction

The device contains several TMs and each individual TM can be categorised as a certain type, namely Compact Type TM or Periodic Type TM. Although similar in nature, the different TM types vary in their feature complexity. The common features to all of the Compact and Periodic TMs will be described in this section and the detailed operation regarding each of the TM types will be described in separate sections. The main features of TMs are summarised in the accompanying table.

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

TM Function Summary

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 xTnCK2~xTnCK0 bits in the xTMn control registers, where “x” stands for C or P type TM and “n” stands for the specific TM serial number. For the PTM there is no serial number “n” in the relevant pins, registers and control bits since there is only one PTM in the device. The clock source can be a ratio of the system clock f_{SYS} or the internal high clock f_H, the f_{SUB} clock source or the external xTCKn pin. The xTCKn 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 have 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.

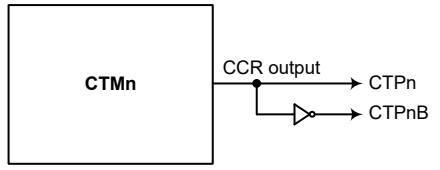
TM External Pins

The Compact type TM2 and Periodic type TM each have one input pin with the label xTCKn while the Periodic type TM has another input pin with the label PTPI. The xTMn input pin, xTCKn, is essentially a clock source for the xTM and is selected using the xTnCK2~xTnCK0 bits in the xTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The xTCKn input pin can be chosen to have either a rising or falling active edge. The PTCK pin is also used as the external trigger input pin in single pulse output mode for the PTM.

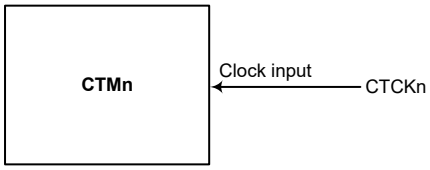
The other PTM input pin, PTPI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTIO1~PTIO0 bits in the PTMC1 register. There is another capture input, PTCK, for PTM capture input mode, which can be used as the external trigger input source except the PTPI pin.

The Compact and Periodic type TMs except CTM2 each have one or two output pins, xTPn and xTPnB. When the TM is in the Compare Match Output Mode, the xTPn pin can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The xTPnB pin outputs the inverted signal of the xTPn. The external output pins are also the pins where the TM generates the PWM output waveform.

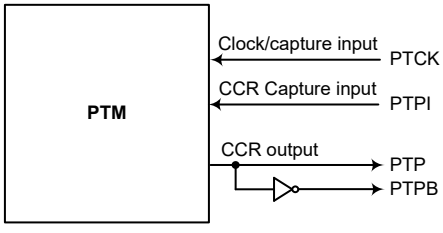
As the TM input and output pins are pin-shared with other functions, the TM input and output function must first be setup using relevant pin-shared function selection bits described in the Pin-shared Functions section. The details of the pin-shared function selection are described in the pin-shared function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



CTMn Function Pin Block Diagram (n=0~1)



CTMn Function Pin Block Diagram (n=2)

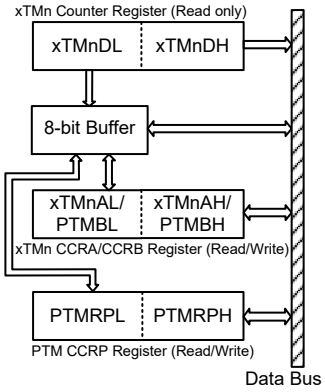


PTM Function Pin Block Diagram

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 xTMnAL, PTMBL and PTMRPL, using the following access procedures. Accessing the CCRA, CCRB or CCRP low byte registers without following these access procedures will result in unpredictable values.



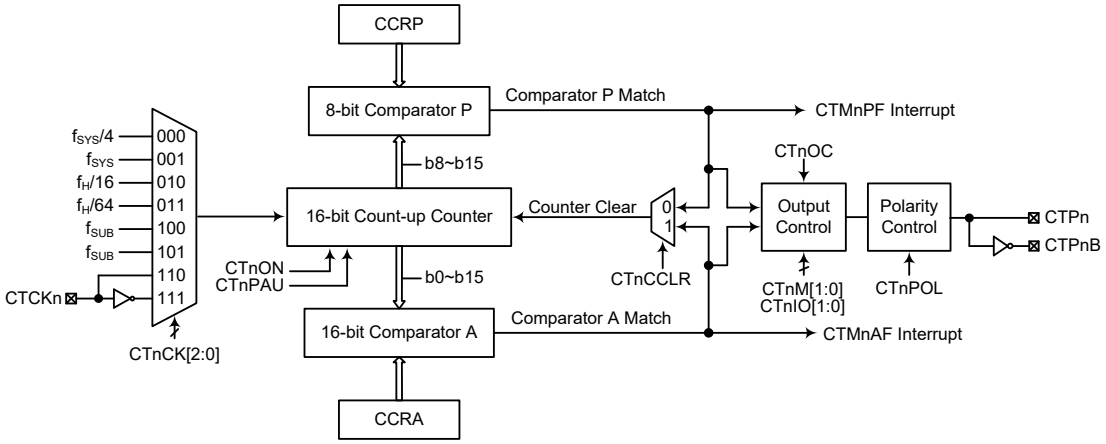
The following steps show the read and write procedures:

- Writing Data to CCRA, CCRB or CCRP
  - Step 1. Write data to Low Byte xTMnAL, PTMBL or PTMRPL
    - Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte xTMnAH, PTMBH or PTMRPH
    - 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 xTMnDH, xTMnAH, PTMBH or PTMRPH
    - 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 xTMnDL, xTMnAL, PTMBL or PTMRPL
    - This step reads data from the 8-bit buffer.

Compact Type TM – CTM

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

Name	TM Input Pin	TM Output Pin
CTM0	—	CTP0B
CTM1	—	CTP1, CTP1B
CTM2	CTCK2	—



Note: 1. The CTMn external pins are pin-shared with other functions, so before using the CTMn function, ensure that the relevant pin-shared function registers have been set properly to enable the CTMn pin function. The CTCKn pin, if used, must also be set as an input by setting the corresponding bits in the port control register.

2. The CTMn input and output pins, CTCK0, CTCK1, CTP0, CTP2 and CTP2B are not bounded to the external package.

16-bit Compact Type TM Block Diagram (n=0~2)

Compact Type TM Operation

At its core is a 16-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is 8-bit wide whose value is compared with the highest eight bits in the counter while the CCRA is 16-bit wide and therefore compares with all counter bits.

The only way of changing the value of the 16-bit counter using the application program, is to clear the counter by changing the CTnON 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 CTMn interrupt signal will also usually be generated. The Compact Type TMn can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control one output pin. All operating setup conditions are selected using relevant internal registers.

Compact Type TM Register Description

Overall operation of the Compact TM is controlled using a series of 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 CCRA value. The CTMnRP register is used to store the 8-bit CCRP 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
CTMnC0	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	—	—	—
CTMnC1	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
CTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnDH	D15	D14	D13	D12	D11	D10	D9	D8
CTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnAH	D15	D14	D13	D12	D11	D10	D9	D8
CTMnRP	D7	D6	D5	D4	D3	D2	D1	D0

16-bit Compact TMn Register List (n=0~2)

- **CTMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **CTnPAU**: CTMn 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 CTMn 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 **CTnCK2~CTnCK0**: CTMn counter clock selection
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: CTCKn rising edge clock
111: CTCKn falling edge clock

These three bits are used to select the clock source for the CTMn. The external pin clock source (only for CTM2) 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 **CTnON**: CTMn counter on/off control
0: Off
1: On

This bit controls the overall on/off function of the CTMn. Setting the bit high enables the counter to run while clearing the bit disables the CTMn. Clearing this bit to zero will stop the counter from counting and turn off the CTMn 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 CTMn is in the Compare Match Output Mode or the PWM Output Mode then the CTMn output pin will be reset to its initial condition, as specified by the CTnOC bit, when the CTnON bit changes from low to high.

Bit 2~0 Unimplemented, read as “0”

• **CTMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnNOC	CTnPOL	CTnDPX	CTnCCLR
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 **CTnM1~CTnM0**: CTMn operating mode selection
00: Compare Match Output Mode
01: Undefined
10: PWM Output Mode
11: Timer/Counter Mode

These bits set the required operating mode for the CTMn. To ensure reliable operation the CTMn should be switched off before any changes are made to the CTnM1 and CTnM0 bits. In the Timer/Counter Mode, the CTMn output pin state is undefined.

Note that CTM2 has no output pins and can only be used as a Timer/Counter by setting these bits to "00" or "11".

Bit 5~4 **CTnIO1~CTnIO0**: Select CTMn external pin function

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

PWM Output Mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Undefined

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a compare match occurs from the Comparator A. The CTMn output pin can be set 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 CTMn output pin should be configured using the CTnOC bit in the CTMnC1 register. Note that the output level requested by the CTnIO1 and CTnIO0 bits must be different from the initial value setup using the CTnOC bit otherwise no change will occur on the CTMn output pin when a compare match occurs. After the CTMn output pin changes state, it can be reset to its initial level by changing the level of the CTnON bit from low to high.

In the PWM Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn 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 change the values of the CTnIO1 and CTnIO0 bits only after the CTMn has been switched off. Unpredictable PWM outputs will occur if the CTnIO1 and CTnIO0 bits are changed when the CTMn is running.

Bit 3 **CTnOC**: CTMn CTPn output control

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

PWM Output Mode
0: Active low
1: Active high

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

Bit 2 **CTnPOL**: CTMn CTPn output polarity control
0: Non-invert
1: Invert

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

Bit 1 **CTnDPX**: CTMn PWM duty/period control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

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

Bit 0 **CTnCCLR**: CTMn counter clear condition selection
0: CTMn Comparator P match
1: CTMn Comparator A match

This bit is used to select the method which clears the counter. Remember that the Compact TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the CTnCCLR 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 CTnCCLR bit is not used in the PWM Output Mode.

• CTMnDL 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**: CTMn Counter Low Byte Register bit 7 ~ bit 0
CTMn 16-bit Counter bit 7 ~ bit 0

• CTMnDH 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**: CTMn Counter High Byte Register bit 7 ~ bit 0
CTMn 16-bit Counter bit 15 ~ bit 8

• CTMnAL 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**: CTMn CCRA Low Byte Register bit 7 ~ bit 0
CTMn 16-bit CCRA bit 7 ~ bit 0

• **CTMnAH 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**: CTMn CCRA High Byte Register bit 7 ~ bit 0
CTMn 16-bit CCRA bit 15 ~ bit 8

• **CTMnRP 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**: CTMn CCRP 8-bit register, compared with the CTMn Counter bit 15 ~ bit 8
Comparator P Match Period=
0: 65536 CTMn clocks
1~255: 256×(1~255) CTMn clocks

These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight bits. The result of this comparison can be selected to clear the internal counter if the CTnCCLR bit is set to zero. Setting the CTnCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

Compact Type TM Operating Modes

The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

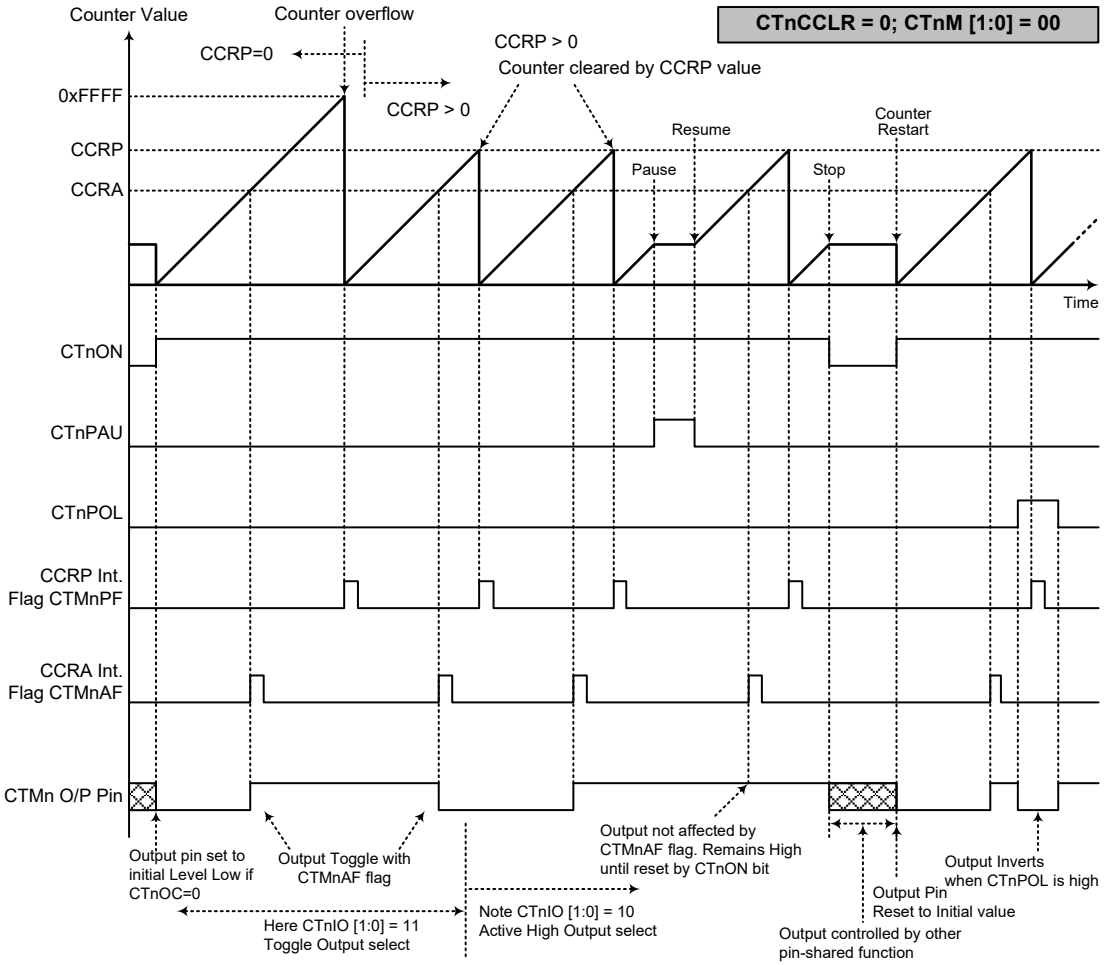
Compare Match Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 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 CTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match occurs from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMnAF and CTMnPF interrupt request flags for the Comparator A and Comparator P respectively, will both be generated.

If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is high no CTMnPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 16-bit, FFFF Hex, value, however here the CTMnAF interrupt request flag will not be generated.

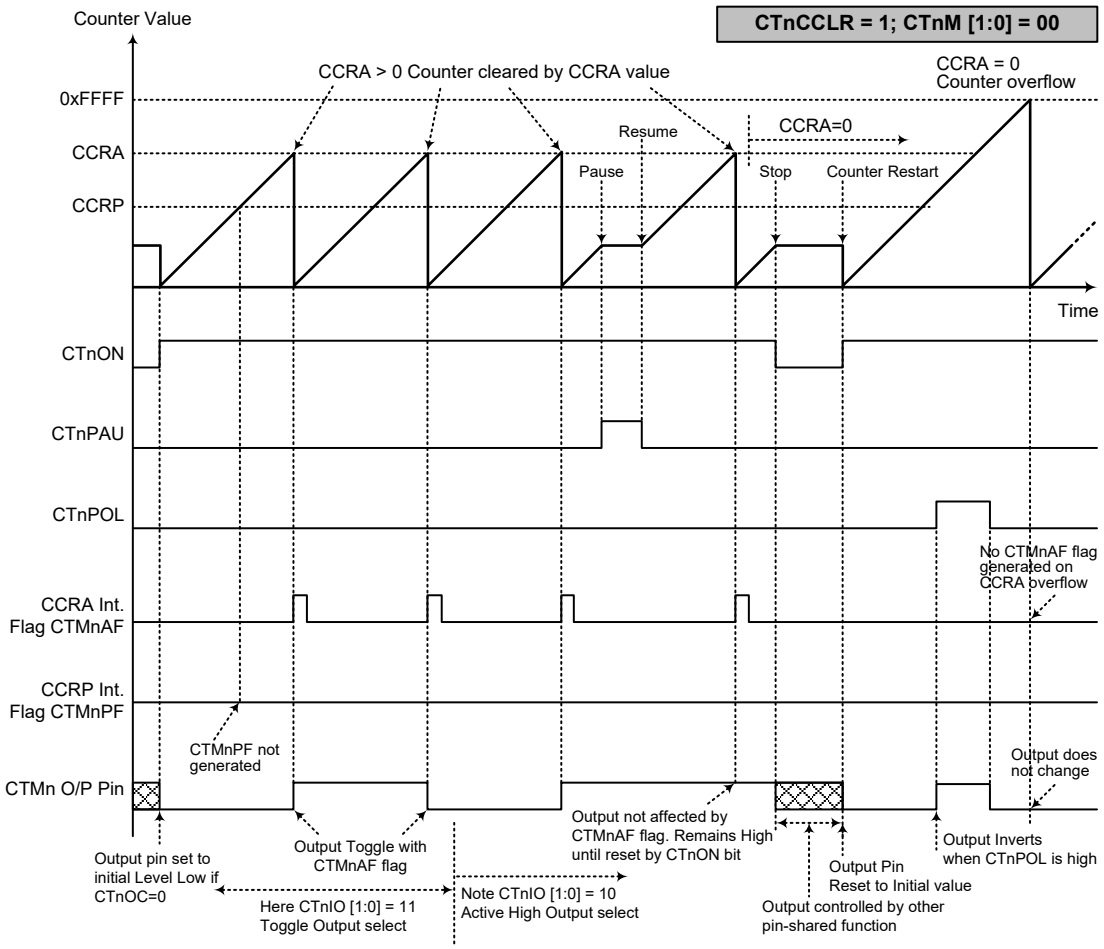
As the name of the mode suggests, after a comparison is made, the CTMn output pin will change state. The CTMn output pin condition however only changes state when a CTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on

the CTMn output pin. The way in which the CTMn output pin changes state are determined by the condition of the CTnIO1 and CTnIO0 bits in the CTMnC1 register. The CTMn output pin can be selected using the CTnIO1 and CTnIO0 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 CTMn output pin, which is setup after the CTnON bit changes from low to high, is setup using the CTnOC bit. Note that if the CTnIO1 and CTnIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – CTnCCLR=0 (n=0~2)

Note: 1. With CTnCCLR=0, a Comparator P match will clear the counter
2. The CTMn output pin is controlled only by the CTMnAF flag
3. The output pin is reset to its initial state by a CTnON bit rising edge
4. CTM2 has no output pins



Compare Match Output Mode – CTnCCLR=1 (n=0~2)

Note: 1. With CTnCCLR=1, a Comparator A match will clear the counter

2. The CTMn output pin is controlled only by the CTMnAF flag
3. The output pin is reset to its initial state by a CTnON bit rising edge
4. The CTMnPF flag is not generated when CTnCCLR=1
5. CTM2 has no output pins

Timer/Counter Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 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 CTMn 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 CTMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared functions.

PWM Output Mode (CTM0, CTM1)

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to "10" respectively. The PWM function within the CTMn 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 CTMn 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 CTnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTnDPX bit in the CTMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The CTnOC bit in the CTMnC1 register is used to select the required polarity of the PWM waveform while the two CTnIO1 and CTnIO0 bits are used to enable the PWM output or to force the CTMn output pin to a fixed high or low level. The CTnPOL bit is used to reverse the polarity of the PWM output waveform.

- **16-bit CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=0**

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

If f_{SYS}=20MHz, CTMn clock source is f_{SYS}/4, CCRP=2, CCRA=128,

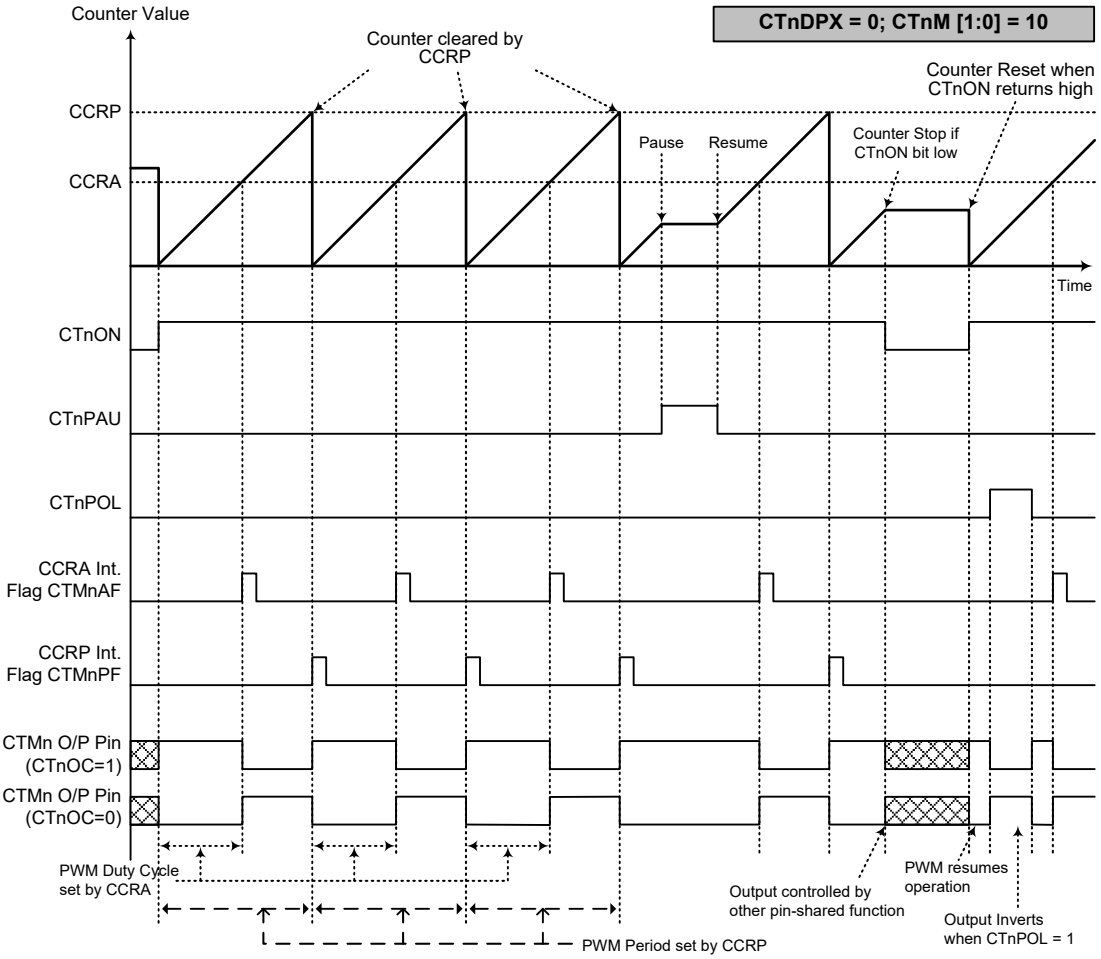
The CTMn PWM output frequency=(f_{SYS}/4)/512=f_{SYS}/2048=9.766kHz, duty=128/512=25%,

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%.

- **16-bit CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=1**

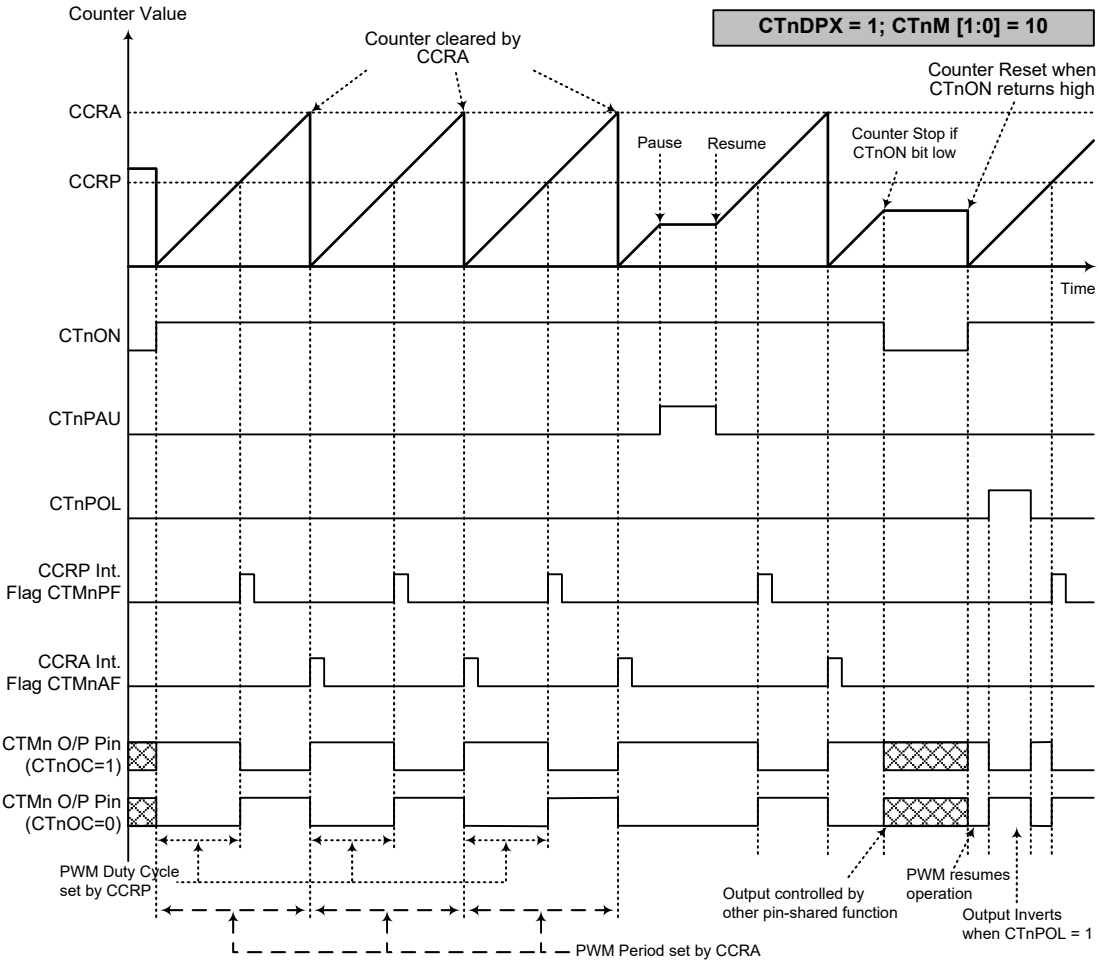
CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

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



PWM Output Mode – CTnDPX=0 (n=0,1)

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

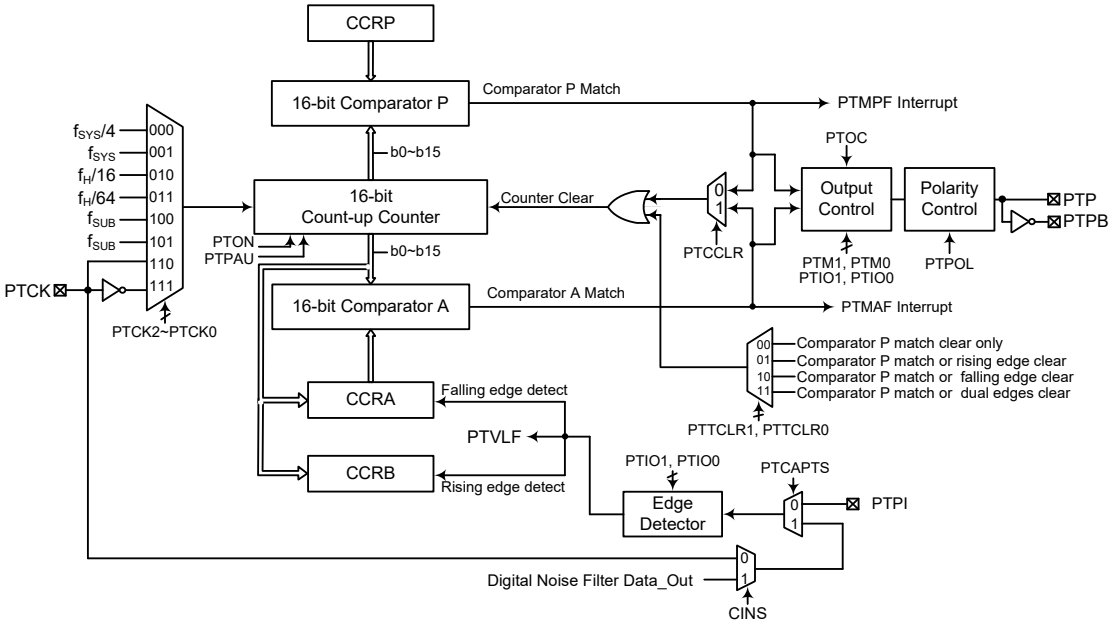


PWM Output Mode – CTnDPX=1 (n=0,1)

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

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 Type TM can be controlled with two external input pins and can drive two external output pin.



Note: The PTM external pin is pin-shared with other functions and can input or output on several pins, so before using the PTM functions, the pin-shared function registers must be set properly to enable the PTM pin function. The PTCK and PTPI pin, if used, must also be set as an input by setting the corresponding bits in the port control register.

16-bit Periodic Type TM Block Diagram

Periodic Type TM Operation

The size of Periodic TM is 16-bit wide and its core is a 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 16-bit wide whose value is respectively compared with all counter bits.

The only way of changing the value of the 16-bit counter using the application program is to clear the counter by changing the PTON 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 PTM 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 16-bit value, while three read/write register pairs exist to store the internal 16-bit CCRA value, CCRB value and CCRP value. The remaining three registers are control registers which setup the different operating and control modes.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMC2	—	—	—	—	—	PTTCLR1	PTTCLR0	PTVLF
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	D15	D14	D13	D12	D11	D10	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	D15	D14	D13	D12	D11	D10	D9	D8
PTMBL	D7	D6	D5	D4	D3	D2	D1	D0
PTMBH	D15	D14	D13	D12	D11	D10	D9	D8
PTMRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMRPH	D15	D14	D13	D12	D11	D10	D9	D8

16-Bit Periodic Type TM Register List

- **PTMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTPAU**: PTM 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 PTM 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 **PTCK2~PTCK0**: Select PTM counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: PTCK rising edge clock
111: PTCK falling edge clock
These three bits are used to select the clock source for the PTM. 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 **PTON**: PTM counter on/off control
0: Off
1: On
This bit controls the overall on/off function of the PTM. Setting the bit high enables the counter to run, clearing the bit disables the PTM. Clearing this bit to zero will stop the counter from counting and turn off the PTM 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 PTM is in the Compare Match Output Mode, PWM Output Mode or Single Pulse Output Mode then the PTM output pin will be reset to its initial condition, as specified by the PTOC bit, when the PTON bit changes from low to high.

Bit 2~0 Unimplemented, read as “0”

• **PTMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
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 **PTM1~PTM0**: Select PTM 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 PTM. To ensure reliable operation the PTM should be switched off before any changes are made to the PTM1 and PTM0 bits. In the Timer/Counter Mode, the PTM output pin state is undefined.

Bit 5~4 **PTIO1~PTIO0**: Select PTM external 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

PTTCLR[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

PTTCLR[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 PTM external pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the PTM is running.

In the Compare Match Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a compare match occurs from the Comparator A. The PTM 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 PTM output pin should be setup using the PTOC bit in the PTMC1 register. Note that the output level requested by the PTIO1 and PTIO0 bits must be different from the initial value setup using the PTOC bit otherwise no change will occur on the PTM output pin when a compare match occurs. After the PTM output pin changes state, it can be reset to its initial level by changing the level of the PTON bit from low to high.

In the PWM Output Mode, the PTIO1 and PTIO0 bits determine how the PTM 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 PTIO1 and PTIO0 bits after the PTM has been switched off. Unpredictable PWM outputs will occur if the PTIO1 and PTIO0 bits are changed when the PTM is running.

In the Capture Input Mode, the PTIO1 and PTIO0 bits are used to select the active trigger edge on the selected input signal.

Bit 3 **PTOC**: PTM PTP 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 PTM output pin. Its operation depends upon whether PTM 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 PTM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the PTM 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 PTM output pin when the PTON bit changes from low to high.

Bit 2 **PTPOL**: PTM output polarity control
0: Non-invert
1: Invert

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

Bit 1 **PTCAPTS**: PTM capture trigger source selection
0: From PTPI pin
1: From PTCK pin or Digital Noise Filter Data_Out

When the PTCAPTS bit is set high, the capture input trigger source can be PTCK pin or noise filtered Data_Out signal determined using the CINS bit in the NF_VIH register.

Bit 0 **PTCCLR**: Select PTM counter clear condition
0: PTM Comparator P match
1: PTM Comparator A match

This bit is used to select the method which clears the counter. Remember that the Periodic Type TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the PTCCLR 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 PTCCLR bit is not used in the PWM Output Mode, Single Pulse Output Mode or Capture Input Mode.

• PTMC2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PTTCLR1	PTTCLR0	PTVLF
R/W	—	—	—	—	—	R/W	R/W	R
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as “0”

Bit 2~1 **PTTCLR1~PTTCLR0**: Select PTM 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 **PTVLF**: PTM counter value latch trigger edge flag
0: Falling edge trigger the counter value latch
1: Rising edge trigger the counter value latch
When the PTTCLR1~PTTCLR0 bits are 00B, ignore this flag state.

• PTMBL 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**: PTM CCRB low byte register bit 7 ~ bit 0
PTM 16-bit CCRB bit 7 ~ bit 0

• PTMBH 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**: PTM CCRB high byte register bit 7 ~ bit 0
PTM 16-bit CCRB bit 15 ~ bit 8

• PTMDL 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**: PTM counter low byte register bit 7 ~ bit 0
PTM 16-bit Counter bit 7 ~ bit 0

• PTMDH 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**: PTM counter high byte register bit 7 ~ bit 0
PTM 16-bit Counter bit 15 ~ bit 8

• PTMAL 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**: PTM CCRA low byte register bit 7 ~ bit 0
PTM 16-bit CCRA bit 7 ~ bit 0

• **PTMAH 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**: PTM CCRA high byte register bit 7 ~ bit 0
PTM 16-bit CCRA bit 15 ~ bit 8

• **PTMRPL 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**: PTM CCRP low byte register bit 7 ~ bit 0
PTM 16-bit CCRP bit 7 ~ bit 0

• **PTMRPH 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**: PTM CCRP high byte register bit 7 ~ bit 0
PTM 16-bit CCRP bit 15 ~ 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 PTM1 and PTM0 bits in the PTMC1 register.

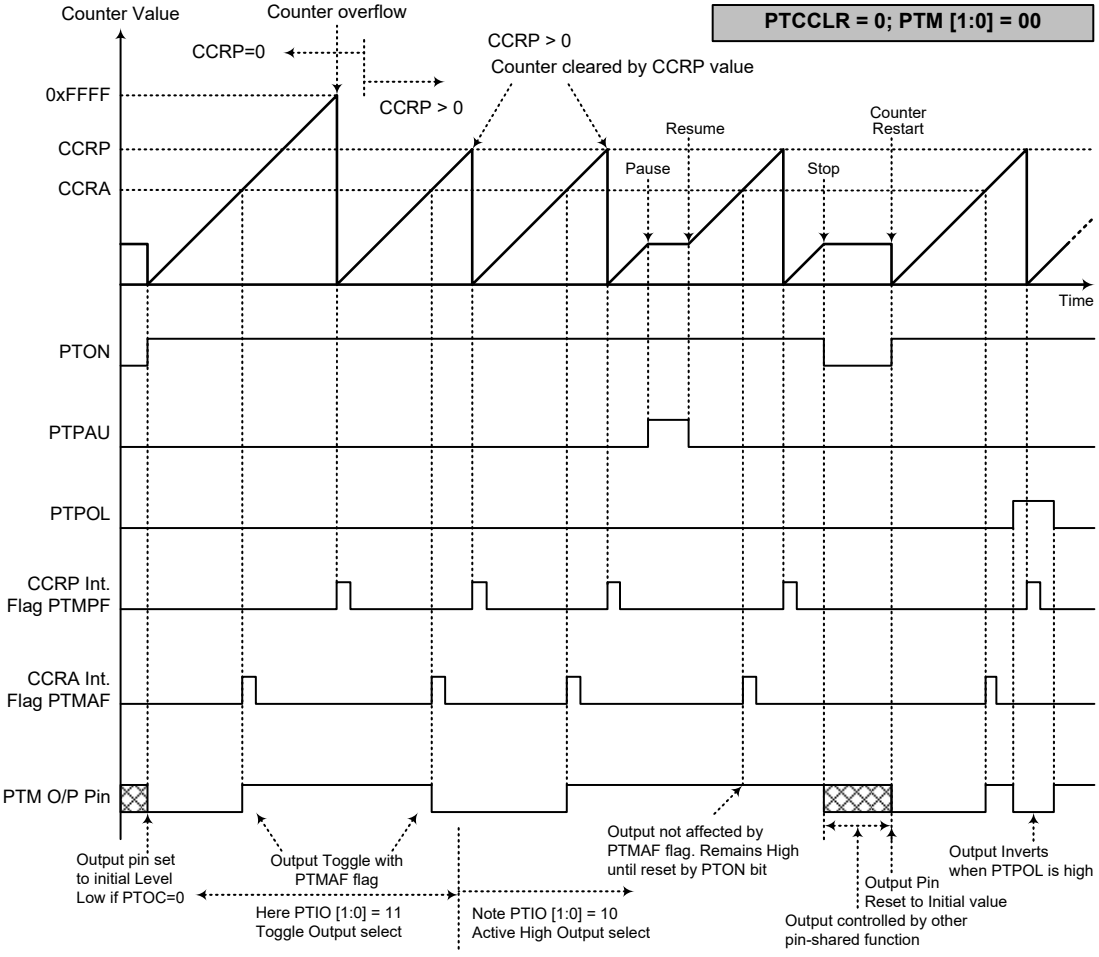
Compare Match Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 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 PTCCLR 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 PTMAF and PTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the PTCCLR bit in the PTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTCCLR is high no PTMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be cleared to zero.

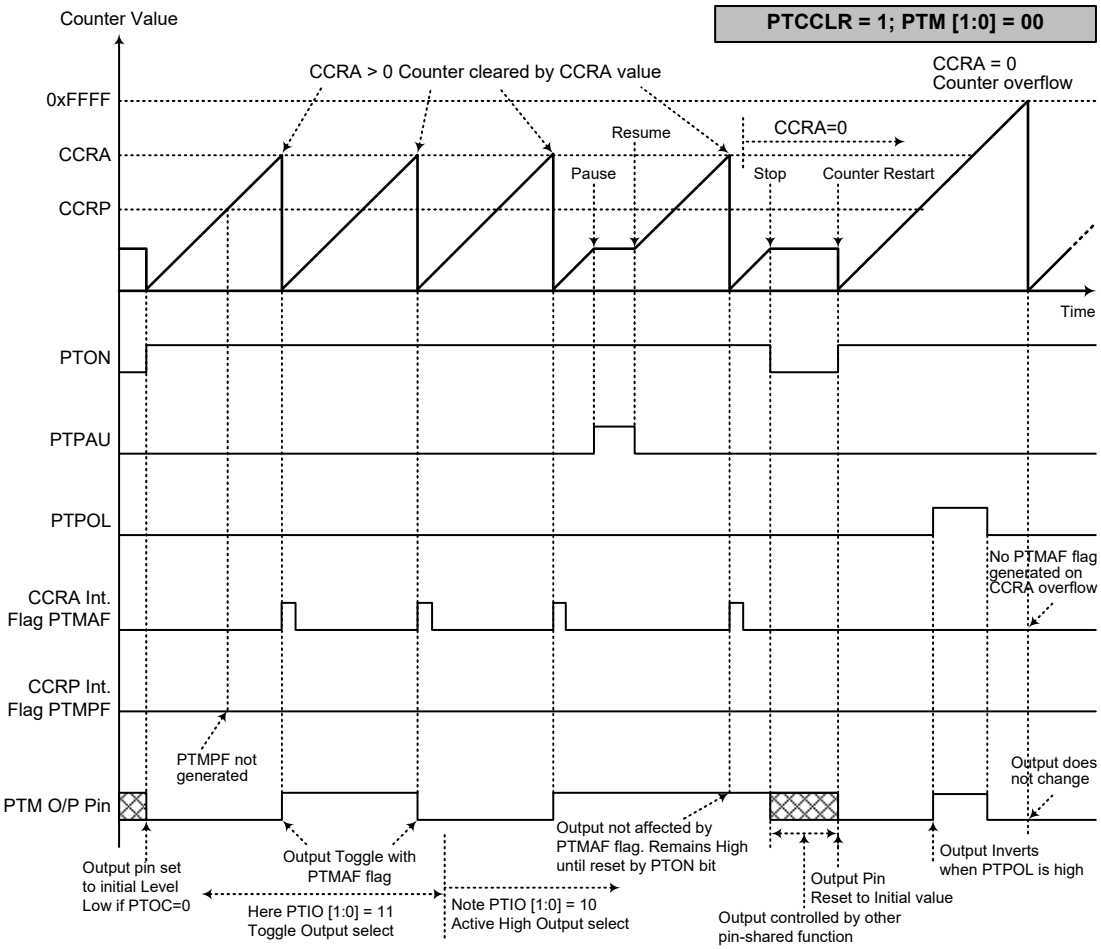
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 16-bit, FFFFH Hex, value, however here the PTMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is matched, the PTM output pin will change state. The PTM output pin condition however only changes state when a PTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the PTM output pin. The way in which the PTM output pin changes state are determined by the condition of the PTIO1 and PTIO0 bits in the PTMC1 register. The PTM output pin can be selected using the PTIO1 and PTIO0 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 PTM output pin, which is setup after the PTON bit changes from low to high, is setup using the PTOC bit. Note that if the PTIO1 and PTIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – PTCCLR=0

Note: 1. With PTCCLR=0 a Comparator P match will clear the counter
2. The PTM output pin controlled only by the PTMAF flag
3. The output pin reset to its initial state by a PTON bit rising edge



Compare Match Output Mode – PTCCLR=1

Note: 1. With PTCCLR=1 a Comparator A match clear the counter
2. The PTM output pin controlled only by the PTMAF flag
3. The output pin reset to its initial state by a PTON bit rising edge
4. A PTMPF flag is not generated when PTCCLR=1

Timer/Counter Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 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 PTM 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 PTM1 and PTM0 in the PTMC1 register should be set to "10" respectively. The PWM function within the PTM 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 PTM 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 PTCCLR 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 PTOC bit in the PTMC1 register is used to select the required polarity of the PWM waveform while the two PTIO1 and PTIO0 bits are used to enable the PWM output or to force the PTM output pin to a fixed high or low level. The PTPOL bit is used to reverse the polarity of the PWM output waveform.

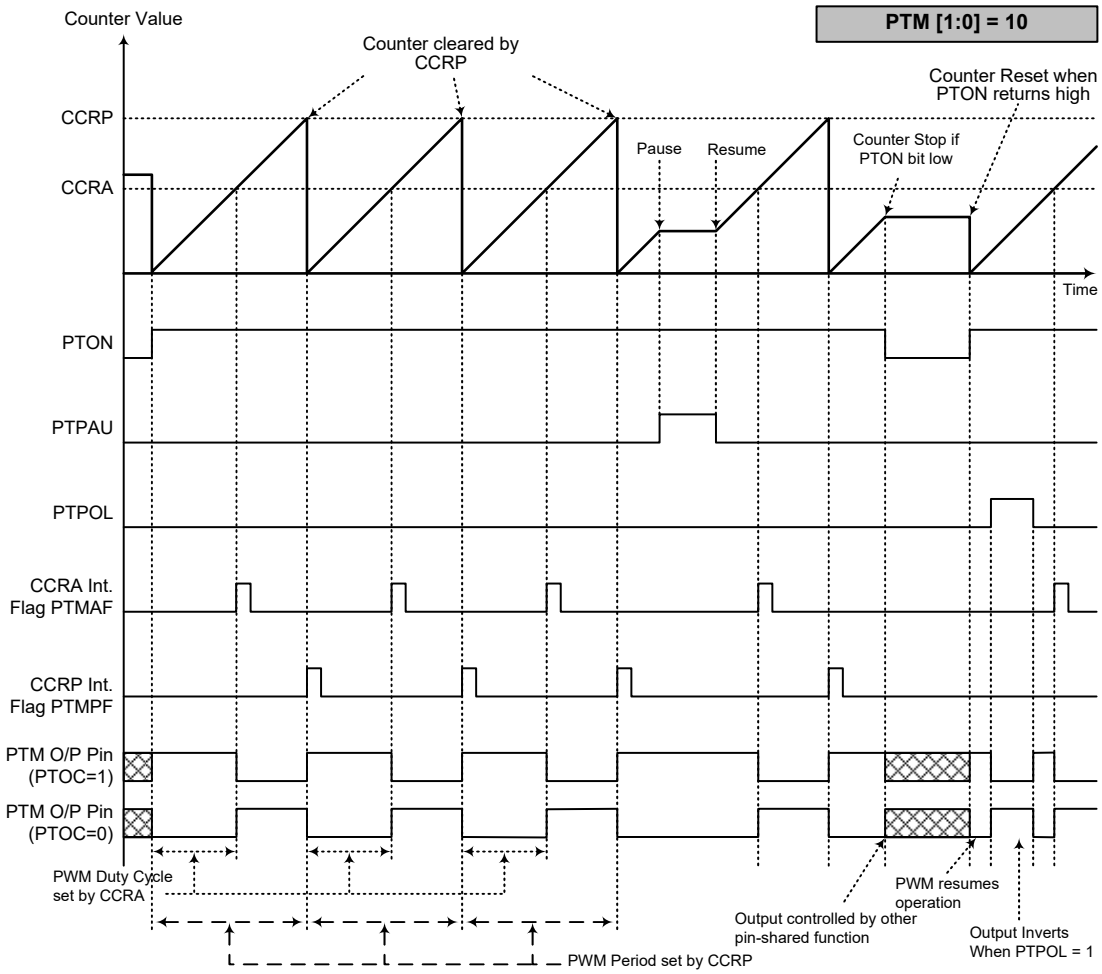
- **16-bit PTM, PWM Output Mode, Edge-aligned Mode**

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

If f_{SYS}=20MHz, PTM clock source select f_{SYS}/4, CCRP=512 and CCRA=128,

The PTM PWM output frequency = (f_{SYS}/4)/512=f_{SYS}/2048=9.766kHz, 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%.



PWM Output Mode

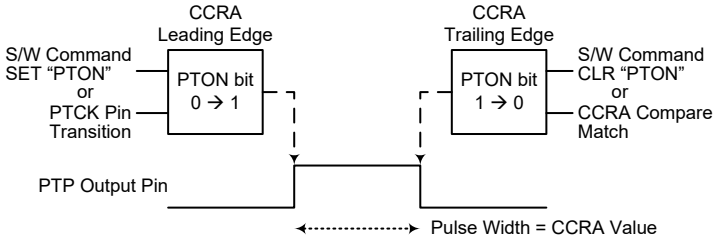
Note: 1. Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTIO[1:0]=00 or 01
4. The PTCCLR bit has no influence on PWM operation

Single Pulse Output Mode

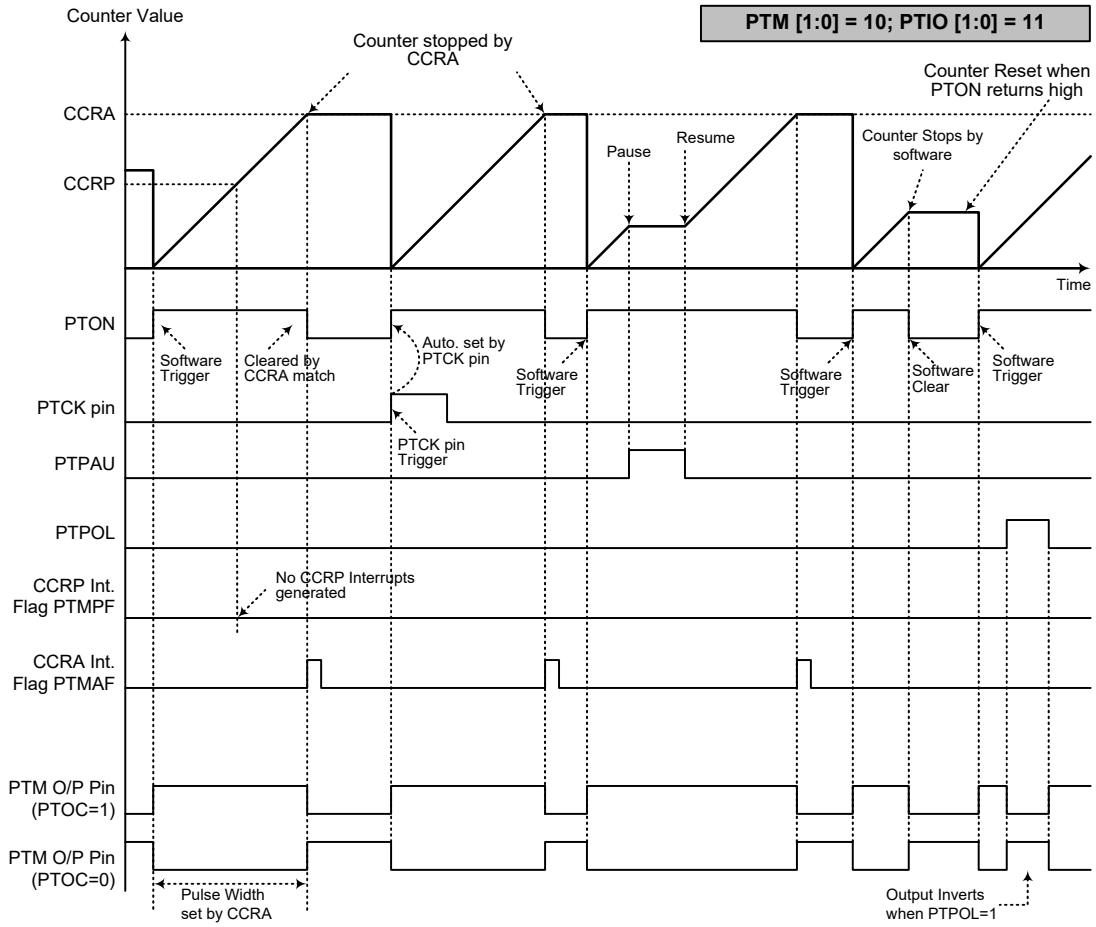
To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to "10" respectively and also the PTIO1 and PTIO0 bits should be set to "11" respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

The trigger for the pulse output leading edge is a low to high transition of the PTON bit, which can be implemented using the application program. However in the Single Output Pulse Mode, the PTON bit can also be made to automatically change from low to high using the external PTCK pin, which will in turn initiate the Single Pulse output. When the PTON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The PTON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the PTON 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 PTON 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 PTM interrupt. The counter can only be reset back to zero when the PTON bit changes from low to high when the counter restarts. In the Single Pulse Output Mode CCRP is not used. The PTCCLR bit is not used in this Mode.



Single Pulse Generation



Single Pulse Output Mode

Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse is triggered by the PTCK pin or by setting the PTON bit high
4. A PTCK pin active edge will automatically set the PTON bit high
5. In the Single Pulse Output Mode, PTIO[1:0] must be set to “11” and cannot be changed

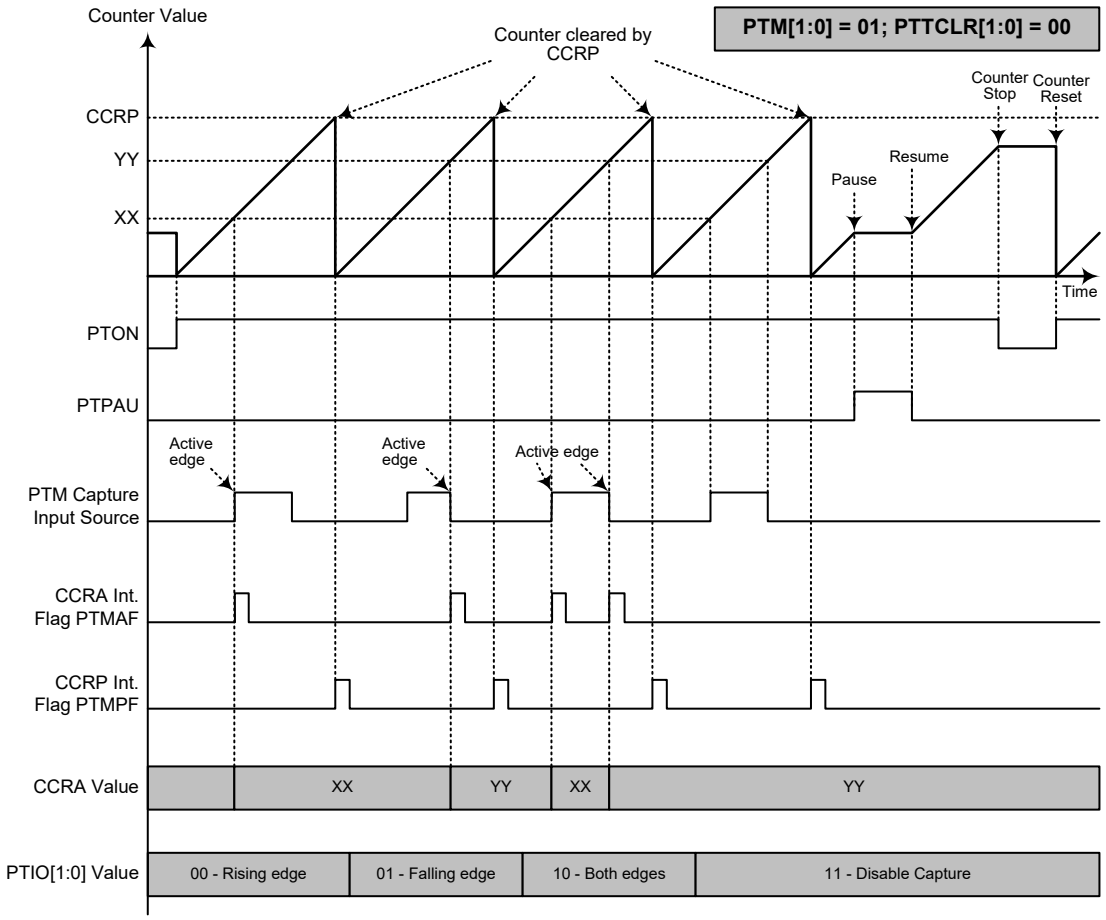
Capture Input Mode

To select this mode bits PTM1 and PTM0 in the PTMC1 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 PTPI or PTCK pin which is selected using the PTCAPTS bit in the PTMC1 register. The signal sourced from the Noise Filter Data_Out, which is a filtered signal of the NFIN pin input, can also 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 PTIO1 and PTIO0 bits in the PTMC1 register. The counter is started when the PTON bit changes from low to high which is initiated using the application program.

The PTIO1 and PTIO0 bits decide which active edge transition type to be latched and to generate an interrupt. The PTTCLR1 and PTTCLR0 bits decide the condition that the counter reset back to zero. The present counter value latched into CCRA or CCRB is decided by PTIO1~PTIO0 together with PTTCLR1~PTTCLR0 setting. The PTIO1~PTIO0 and PTTCLR1~PTTCLR0 bits are setup independently on each other.

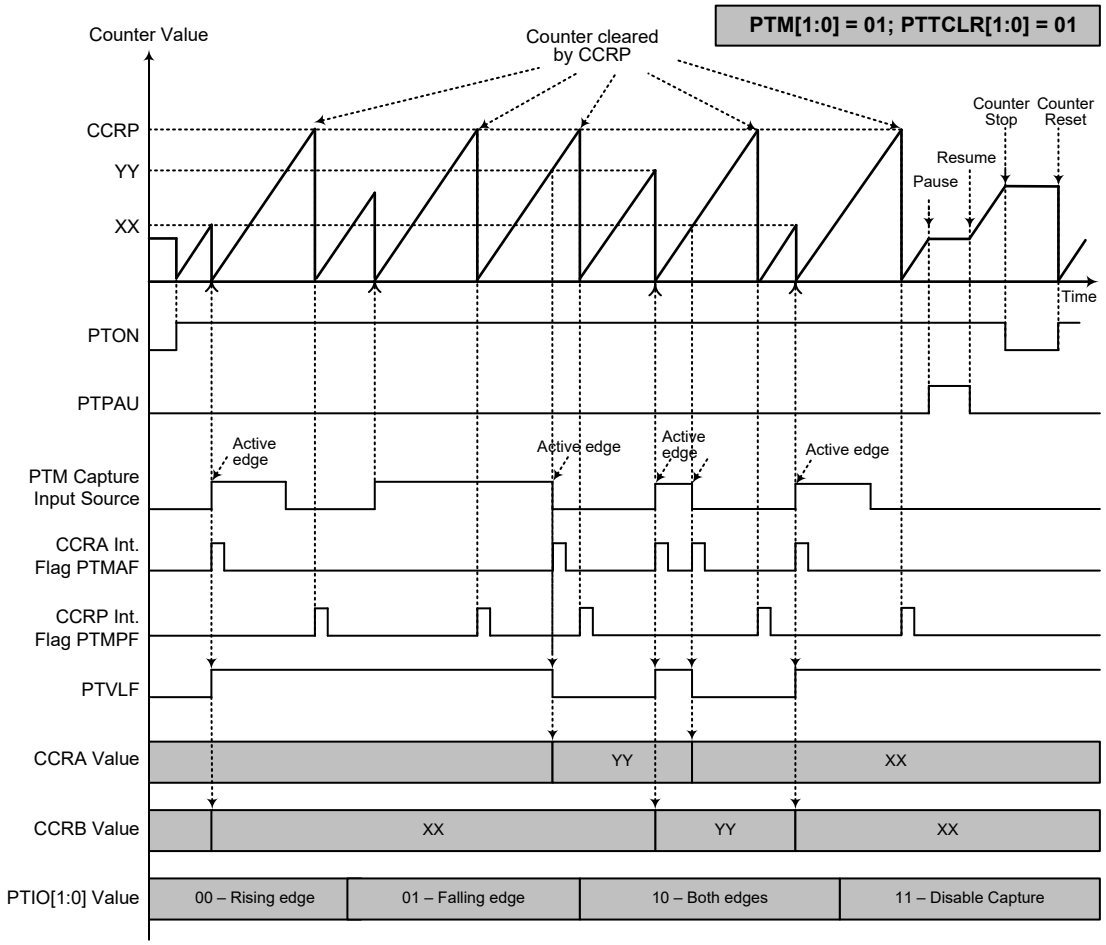
When the required edge transition appears on the PTPI or PTCK pin or Data_Out signal the present value in the counter will be latched into the CCRA or CCRB registers and a PTM interrupt generated. Irrespective of what events occur on the PTPI or PTCK pin or Data_Out signal, the counter will continue to free run until the PTON 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 PTM 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 PTIO1 and PTIO0 bits can select the active trigger edge on the PTPI or PTCK pin or Data_Out signal to be a rising edge, falling edge or both edge types. If the PTIO1 and PTIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPI or PTCK pin or Data_Out signal, however it must be noted that the counter will continue to run. The PTCCLR, PTOC and PTPOL bits are not used in this Mode.

There are some considerations that should be noted. If PTCK is used as the capture input source, then it cannot be selected as the PTM clock source. If the captured pulse width is less than 2 timer clock periods, it may be ignored by hardware. After the counter value is latched to CCRA or CCRB registers by an active capture edge, the PTMAF flag will be set high and the PTVLF flag status will be changed after 0.5 timer clock period. The delay time from the active capture edge received to the action of latching counter value to CCRA or CCRB registers is less than 1.5 timer clock periods.



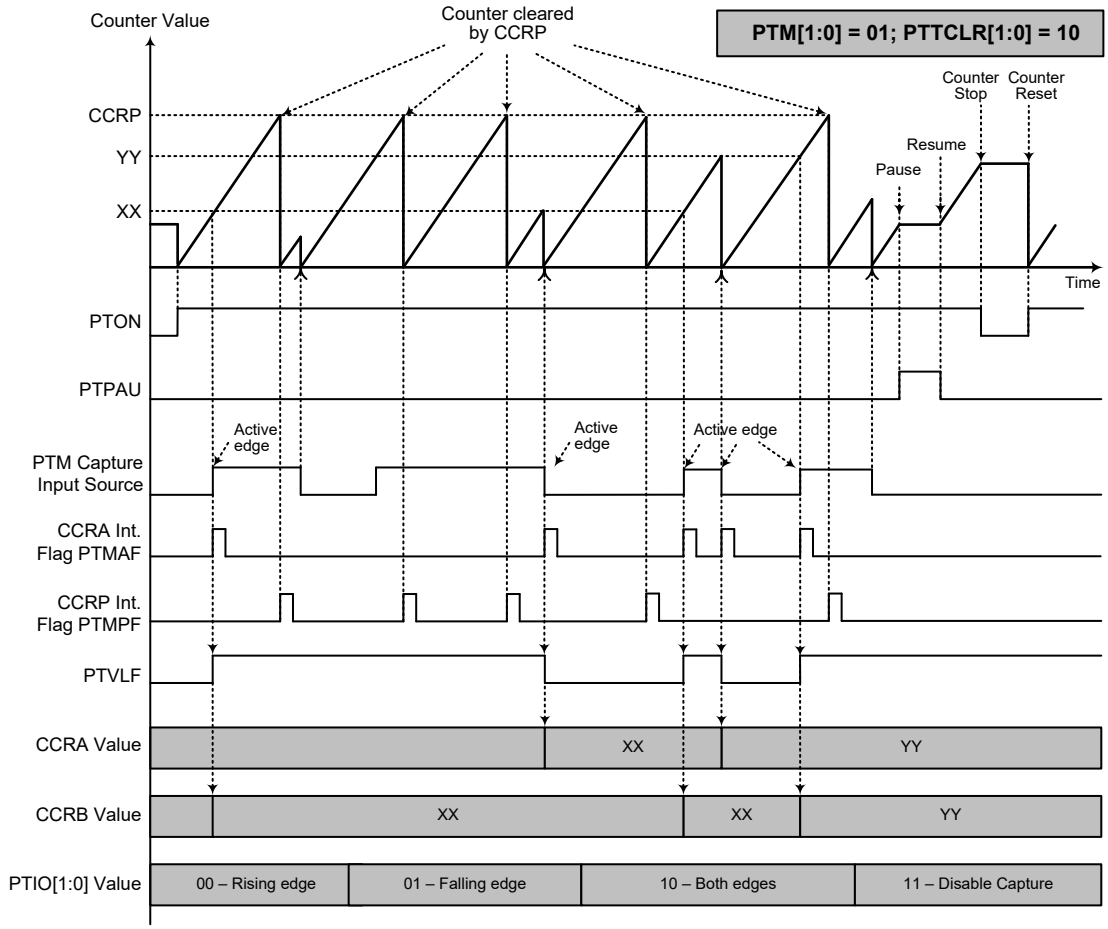
Capture Input Mode – PTTCLR[1:0]=00

Note: 1. PTM[1:0]=01, PTTCLR[1:0]=00 and active edge set by the PTIO[1:0] bits
2. A PTM Capture input pin active edge transfers the counter value to CCRA
3. Comparator P match will clear the counter
4. PTCCLR bit not used
5. No output function – PTOC and PTPOL 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 PTVLF bit status when PTTCLR[1:0]=00
8. The capture input mode cannot be used if the selected PTM counter clock is not available



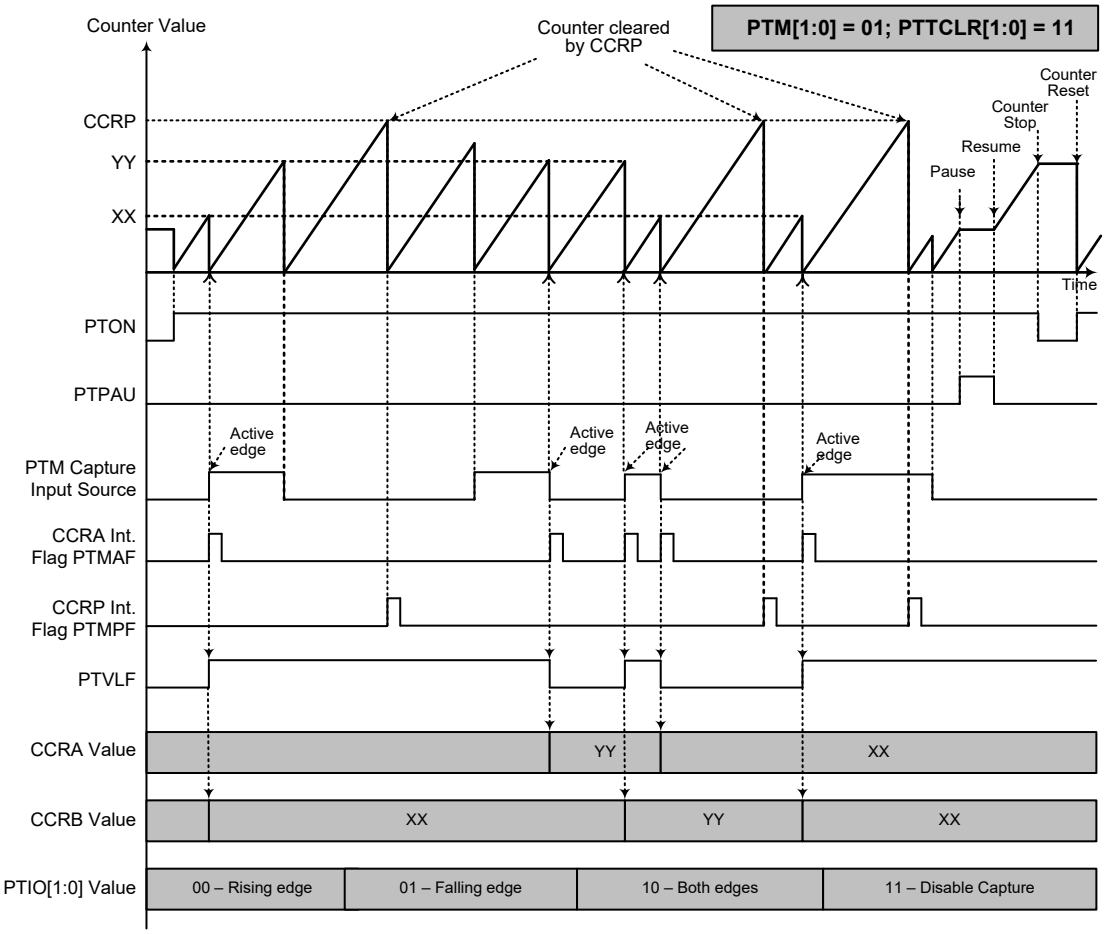
Capture Input Mode – PTTCLR[1:0]=01

Note: 1. PTM[1:0]=01, PTTCLR[1:0]=01 and active edge set by the PTIO[1:0] bits
2. A PTM Capture input pin active edge transfers the counter value to CCRA or CCRB
3. Comparator P match or PTM capture input rising edge will clear the counter
4. PTCCLR bit is not used
5. No output function – PTOC and PTPOL 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. The capture input mode cannot be used if the selected PTM counter clock is not available



Capture Input Mode – PTTCLR[1:0]=10

Note: 1. PTM[1:0]=01, PTTCLR[1:0]=10 and active edge set by the PTIO[1:0] bits
2. A PTM Capture input pin active edge transfers the counter value to CCRA or CCRB
3. Comparator P match or PTM capture input falling edge will clear the counter
4. PTCCLR bit is not used
5. No output function – PTOC and PTPOL 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. The capture input mode cannot be used if the selected PTM counter clock is not available



Capture Input Mode – PTTCLR[1:0]=11

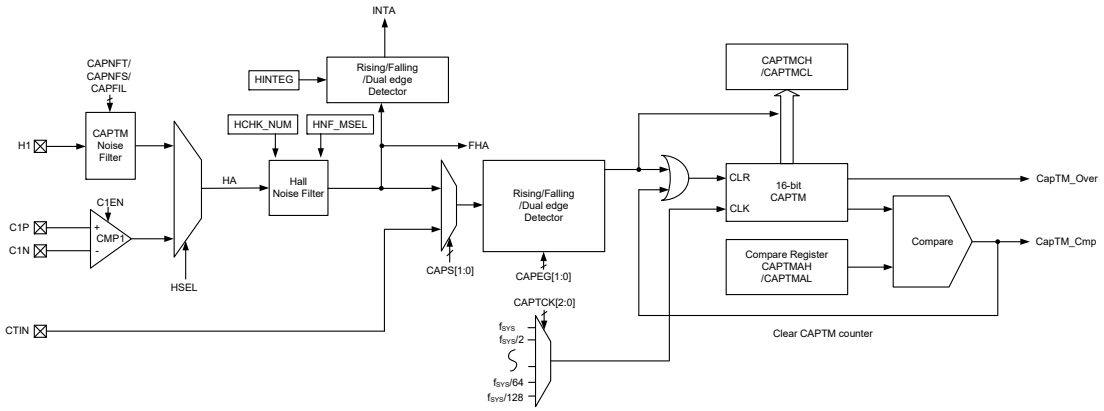
Note: 1. PTM[1:0]=01, PTTCLR[1:0]=11 and active edge set by the PTIO[1:0] bits
2. A PTM Capture input pin active edge transfers the counter value to CCRA or CCRB
3. Comparator P match or PTM capture input pin rising or falling edge will clear the counter
4. PTCCLR bit is not used
5. No output function, PTOC and PTPOL bits not used
6. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero
7. The capture input mode cannot be used if the selected PTM counter clock is not available

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 a divided version of the system clock. 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 FHA or 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
CAPTMDL	D7	D6	D5	D4	D3	D2	D1	D0
CAPTMDH	D15	D14	D13	D12	D11	D10	D9	D8
CAPTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CAPTMAH	D15	D14	D13	D12	D11	D10	D9	D8
CAPTMCL	D7	D6	D5	D4	D3	D2	D1	D0
CAPTMCH	D15	D14	D13	D12	D11	D10	D9	D8

Capture Timer Register List

• **CAPTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CAPTPAU	CAPTCK2	CAPTCK1	CAPTCK0	CAPTON	—	CAPS1	CAPS0
R/W	R/W	R/W	R/W	R/W	R/W	—	R/W	R/W
POR	0	0	0	0	0	—	0	0

Bit 7 **CAPTPAU**: CAPTM counter pause control
0: Run
1: Pause
The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the CAPTM will remain 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: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$
These three bits are used to select the clock source for the CAPTM.

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: Reserved
10: Reserved
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: Capture at rising edge
10: Capture at falling edge
11: Capture at dual edges

Bit 5 **CAPEN**: CAPTM capture input control
0: Disable
1: Enable

Bit 4 **CAPNFT**: CAPTM Noise Filter sampling times
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: f_{SYS}
1: $f_{SYS}/4$
The clock source for the Capture Timer Module Counter is provided by f_{SYS} or $f_{SYS}/4$.

Bit 2 **CAPFIL**: CAPTM capture input noise filter control
0: Disable
1: Enable
This bit is used to enable and disable the CAPTM capture input noise filter. If the CAPTON bit is equal to "1" and the CAPFIL bit is also equal to "1", the CAPTM capture input noise filter will be enabled.

Bit 1 **CAPCLR**: CAPTM counter capture auto-reset control
0: Disable
1: Enable
If this bit is set high, when FHA or CTIN generates the required capture edge, the hardware will automatically transfer the value in the CAPTMDL and CAPTMDH registers to the capture register pair CAPTMCL and CAPTMCH, after which the CAPTM counter will be cleared and restart to count automatically.

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. When CAPTMAH/CAPTMAL=0000H, it also can generate a compare match interrupt.

• 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

• **CAPTMCL Register**

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

"x": Unknown

Bit 7~0 **D7~D0**: CAPTM capture low byte register bit 7 ~ bit 0
CAPTM 16-bit capture register bit 7 ~ bit 0

• **CAPTMCH Register**

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

"x": Unknown

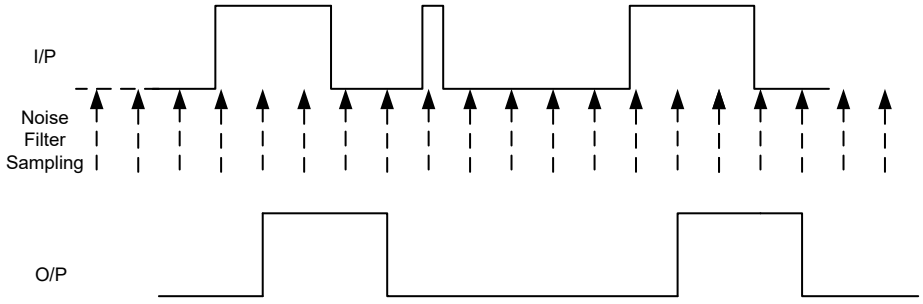
Bit 7~0 **D15~D8**: CAPTM capture high byte register bit 7 ~ bit 0
CAPTM 16-bit capture register bit 15 ~ bit 8

Capture Timer Operation

The Capture Timer is used to detect and measure input signal pulse widths and periods. It can operate in either Capture or Compare Mode. There are two timer capture inputs, FHA and CTIN. The capture input has an edge detection 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 function. Disabling the Capture Module when not used will reduce the device power consumption. Additionally the capture input control is enabled/disabled using the CAPEN control bit. The trigger edge options are setup using the CAPEG1 and CAPEG0 bits, to select either positive edge, negative edge or both edges.

The capture timer also includes an internal noise filter which is used to filter out unwanted glitches or pulses on the H1 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 number of sampling times is based on either f_{SYS} or $f_{SYS}/4$ determined using the CAPNFS bit.



Noise Filter Timing – CAPNFT=0 & CAPNFS=0

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 tranferred into the capture registers, CAPTMCL and CAPTMCH. If the count overflows, the CAPOF interrupt flag bit in the interrupt control register will be set. If the CAPCLR bit is set high, then the 16-bit counter will be automatically reset to zero 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.

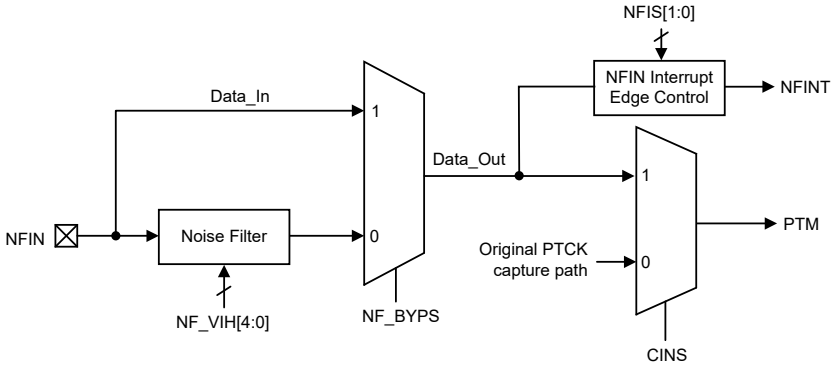
For motor applications, the rotor speed or a motor stall condition can be detected by setting the compare registers to compare the captured signal edge transition time. If a motor stall condition occurs, a compare interrupt will then be generated, after which the PWM motor drive circuit can be shut down by hardware or software to prevent a motor burn out situation.

Digital Noise Filter

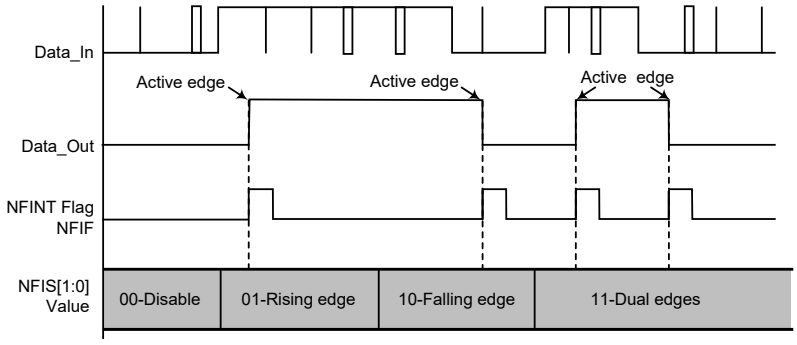
The external NFIN pin is connected to an internal filter to reduce the possibility of unwanted event counting events or inaccurate pulse width measurements due to adverse noise or spikes on the NFIN input signal and then is output to the 16-bit PTM capture circuit in order to ensure that the motor control circuit works normally.

The noise filter circuit is an input/output surge filtering analog circuit which can filter micro-second grade sharp-noise. The maximum antinoise pulse width can be calculated by the following equation:

$$NF\_VIH[4:0]\times(1/f_{SYS})\times4, \text{ where } NF\_VIH[4:0]>1$$



Digital Noise Filter Block Diagram



Digital Noise Filter Interrupt Trigger

Noise Filter Register Description

The operations of the digital noise filter are controlled by the NF_VIH and NF_VIL registers.

- **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	—	0	0	0	1	0

Bit 7 **NF_BYPS**: Bypass noise filter control
0: Noise Filter used
1: Noise Filter Bypass, Data_Out=Data_In

Bit 6 **CINS**: PTM capture input source selection (When PTCAPTS=1)
0: Noise Filter Data_Out not selected (remains original PTCK capture path)
1: Noise Filter Data_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	—	—	—	—	—	—
R/W	R/W	R/W	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

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

Bit 5~0 Unimplemented, read as "0"

Analog to Digital Converter

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

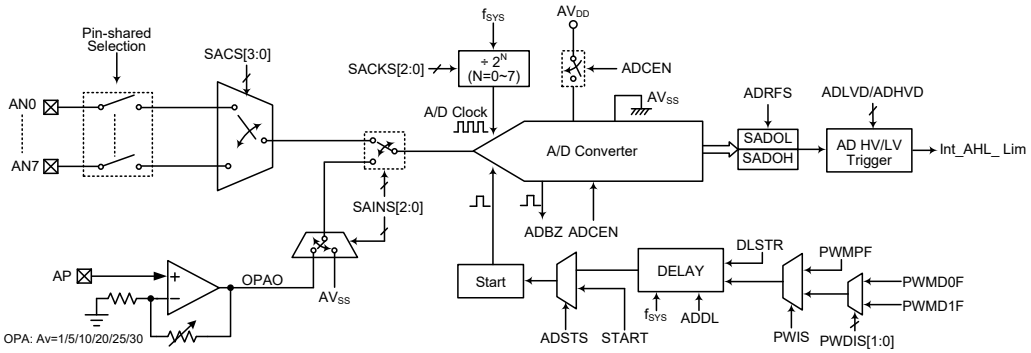
A/D Converter Overview

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 siganls directly into a 12-bit digital value. It also can convert the internal signals, such as OPA output of the over current detection, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 bits together with the SACS3~SACS0 bits. More detailed information about the A/D input signal is described in the "A/D Converter Input Signals" section.

Two sets 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.

External Input Channels	Internal Signals	A/D Input Select Bits
AN0~AN7	OPAO, AV_{SS}	SAINS2~SAINS0 SACS3~SACS0

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, SADOH and SADOL, 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
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
ADLVDL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
ADLVDL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADLVDH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADLVDH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
ADHVDL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
ADHVDL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADHVDH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADHVDH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	—	—	SACKS2	SACKS1	SACKS0
ADCR2	ADSTS	DLSTR	PWIS	ADCHVE	ADCLVE	—	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 SADOH, and a low byte register, known as SADOL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 12 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRFS bit in the SADC0 register as shown in the accompanying table. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero. The A/D data register contents will be unchanged if the A/D converter is disabled.

ADRFS	SADOH								SADOL							
	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 Boundary Registers

The device contains two sets of register pairs which are known as boundary registers to store fixed values for comparison with the A/D converted data value stored in SADOH and SADOL to trigger an A/D converter comparison result interrupt. These two sets of register pairs are a high boundary register pair, known as ADHVDL and ADHVDH, and a low boundary register pair known as ADLVDL and ADLVDH. The register contents will be unchanged when if the A/D converter is disabled.

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

A/D Low Boundary Data Registers

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

A/D High Boundary Data Registers

A/D Converter Control Registers – SADC0, SADC1, ADCR2, ADDL

To control the function and operation of the A/D converter, several control registers known as SADC0, SADC1 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. As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SAINS field in the SADC1 register and the SACS field in the SADC0 register are used to determine which analog signal derived from the external or internal signals will be connected to the 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.

• **SADC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **START**: 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 **ADBZ**: A/D converter busy flag
0: No A/D conversion is in progress
1: A/D conversion is in progress
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set high to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is completed.

Bit 5 **ADCEN**: A/D converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is cleared to zero, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOH and SADOL as well as the contents of the A/D boundary register pairs known as ADLVDH/ADLVDL and ADHVDH/ADHVDL will remain unchanged.

Bit 4 **ADRFS**: A/D conversion data format and boundary register data format selection
0: SADOH/ADHVDH/ADLVDH=D[11:4]; SADOL/ADHVDL/ADLVDL=D[3:0]
1: SADOH/ADHVDH/ADLVDH=D[11:8]; SADOL/ADHVDL/ADLVDL=D[7:0]
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers as well as the 12-bit data format of the low/high boundary registers. Details are provided in the A/D converter data register and boundary register sections.

Bit 3~0 **SACS3~SACS0**: A/D converter external analog input channel selection
0000: External AN0 input
0001: External AN1 input
0010: External AN2 input
0011: External AN3 input
0100: External AN4 input
0101: External AN5 input
0110: External AN6 input
0111: External AN7 input
1000~1111: Undefined, input floating

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	—	—	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7~5 **SAINS2~SAINS0**: A/D converter input signal selection
000: External source – External analog channel input, ANn
001: Internal source – Internal OPA output, OPAO
010: Internal source – Connected to ground, AV_{SS}
011: Internal source – Connected to ground, AV_{SS}
100~111: External source – External analog channel input, ANn

When the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off regardless of the SACS field value. It will prevent the external channel input from being connected together with the internal analog signal.

Bit 4~3 Unimplemented, read as "00"

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source selection
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

• ADCR2 Register

Bit	7	6	5	4	3	2	1	0
Name	ADSTS	DLSTR	PWIS	ADCHVE	ADCLVE	—	PWDIS1	PWDIS0
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 **ADSTS**: A/D conversion start trigger circuit selection
0: Select START bit trigger circuit
1: Select DELAY start

Bit 6 **DLSTR**: DELAY start function control
0: Disable
1: Enable
When this bit is cleared to zero, the ADDL register value must be cleared to zero. The ADDL register value must be set to a value except 00H if this bit is set high.

Bit 5 **PWIS**: PWM module interrupt source selection
0: Select PWM period match interrupt (PWMP_Int)
1: Select PWM duty match interrupt (PWMD0~1_Int)

Bit 4~3 **ADCHVE~ADCLVE**: A/D conversion comparison result interrupt trigger condition configuration (for SAIN[2:0]=001 only)
00: ADLVD[11:0] < SADO[11:0] < ADHVD[11:0]
01: SADO[11:0] ≤ ADLVD[11:0]
10: SADO[11:0] ≥ ADHVD[11:0]
11: SADO[11:0] ≤ ADLVD[11:0] or SADO[11:0] ≥ ADHVD[11:0]

Bit 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: Reserved
11: Reserved

• 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 (counts based on system clock)
Delay time=$(1/f_{SYS})\times D[7:0]$
Note that when PWMPF or PWMDnF changes from 0 to 1, Delay circuit starts counting.

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 START bit in the SADC0 register to start the A/D conversion. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. 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 PWM duty interrupt signal is selected, the actual PWM duty interrupt trigger source can be determined 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 ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in progress or not. This bit will be automatically cleared to zero by the microcontroller after an A/D conversion cycle has ended. This bit will be automatically set high by the microcontroller after an A/D conversion is successfully initiated. When the A/D conversion is complete, the ADBZ will be cleared to 0. In addition, the corresponding A/D interrupt request flag in the interrupt control register will be set high, and if the relevant 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 interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can poll the ADBZ bit in the SADC0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

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

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 SACKS2~SACKS0 bits in the SADC1 register. Although the A/D clock source is determined by the system clock f_{SYS} and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK}, is from 0.8μs to 12.8μs, care must be taken for system clock frequencies. For example, when the system clock operates at a frequency of 10MHz, the SACKS2~SACKS0 bits should not be set to 000, 001 or 010. Doing so will give A/D clock periods that are less or larger than the minimum or maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where special care must be taken, as the values may be less or larger than the specified A/D Clock Period range.

f_{SYS}	A/D Clock Period (t_{ADCK})							
	SACKS[2:0] =000 (f_{SYS})	SACKS[2:0] =001 (f_{SYS}/2)	SACKS[2:0] =010 (f_{SYS}/4)	SACKS[2:0] =011 (f_{SYS}/8)	SACKS[2:0] =100 (f_{SYS}/16)	SACKS[2:0] =101 (f_{SYS}/32)	SACKS[2:0] =110 (f_{SYS}/64)	SACKS[2:0] =111 (f_{SYS}/128)
4MHz	250ns*	500ns*	1μs	2μs	4μs	8μs	16μs*	32μs*
5MHz	200ns*	400ns*	800ns	1.6μs	3.2μs	6.4μs	12.8μs	25.6μs*
8MHz	125ns*	250ns*	500ns*	1μs	2μs	4μs	8μs	16μs*
10MHz	100ns*	200ns*	400ns*	800ns	1.6μs	3.2μs	6.4μs	12.8μs
16MHz	62.5ns*	125ns*	250ns*	500ns*	1μs	2μs	4μs	8μs
20MHz	50ns*	100ns*	200ns*	400ns*	800ns	1.6μs	3.2μs	6.4μs

A/D Clock Period Examples

Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry, a certain delay as indicated in the timing diagram must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs, if the ADCEN bit is high, then some power will still be consumed. In power conscious applications it is therefore recommended that the ADCEN is set low to reduce power consumption when the A/D converter function is not being used.

A/D Converter Input Signals

All the external A/D analog channel input pins are pin-shared with the I/O pins as well as other functions. The corresponding control bits for each A/D external input pin in the PxS0 and PxS1 registers determine whether the input pins are setup as A/D converter analog input channel or whether they have other functions. If the pin is setup to be as an A/D analog channel input, the original pin functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

There is an internal analog signals derived from the operational amplifier output signal OPAO which can be connected to the A/D converter as the analog input signal by configuring the SAINS2~SAINS0 bits. If the external channel input is selected to be converted, the SAINS2~SAINS0 bits should be set to “000” or “100~111” and the SACS3~SACS0 bits can determine which external channel is selected. If the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off regardless of the SACS field value. It will prevent the external channel input from being connected together with the internal analog signal.

SAINS[2:0]	SACS[3:0]	Input Signals	Description
000, 100~111	0000~0111	AN0~AN7	External channel analog input ANn
	1000~1111	—	Non-existed channel, input is floating
001	xxxx	OPAO	Internal OPA output signal
010~011	xxxx	AV_{SS}	Connected to the ground

“x”: Don’t care

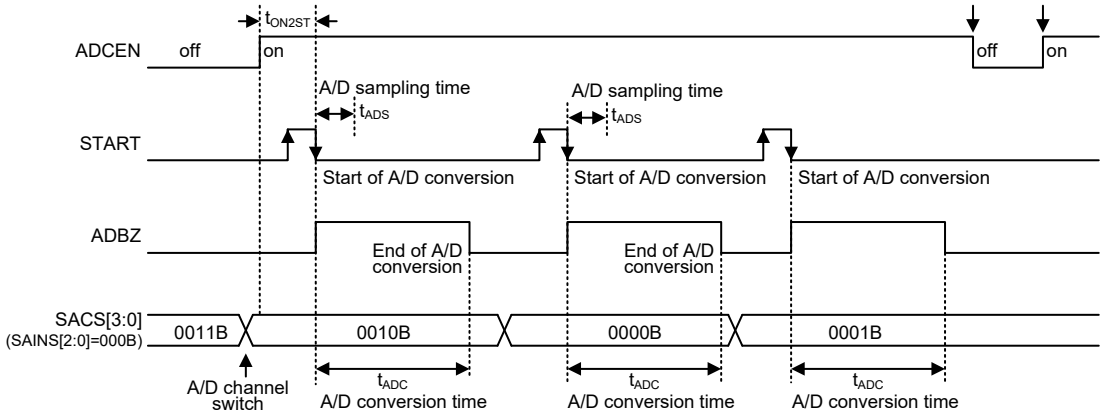
A/D Converter Input Signal Selection

Conversion Rate and Timing Diagram

A complete A/D conversion contains two parts, data sampling and data conversion. The data sampling which is defined as t_{ADS} takes 4 A/D clock periods and the data conversion takes 12 A/D clock periods. Therefore a total of 16 A/D clock periods for an analog signal A/D conversion which is defined as t_{ADC} are necessary.

Maximum single A/D conversion rate = 1/(A/D clock period×16)

The accompanying diagram shows graphically the various stages involved in an external channel input signal 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 16 t_{ADCK} where t_{ADCK} is equal to the A/D clock period.



A/D Conversion Timing – External Channel Input

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 properly programming the SACKS2~SACKS0 bits in the SADC1 register.

- Step 2

  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to one.

- Step 3

  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS and SACS bit fields.

  Selecting the external channel input to be converted, go to Step 4.

  Selecting the internal analog signal to be converted, go to Step 5.

- Step 4

  If the SAINS field is 000 or 100~111, the external channel input can be selected. The desired external channel input is selected by configuring the SACS field. When the A/D input signal comes from the external channel input, the corresponding pin should be configured as an A/D input function by selecting the relevant pin-shared function control bits. After this step, go to Step 6.

- Step 5

  If the SAINS field is set to 001, the internal OPA output signal will be selected. When the internal analog signal is selected to be converted, the external channel analog input will automatically be disconnected. Then select the A/D conversion comparison result interrupt trigger condition by configuring the ADCHVE~ADCLVE bits in the ADCR2 register. After this step, go to Step 6.

- Step 6

  Select which trigger circuit is to be used to start an A/D conversion by correctly programming the ADSTS bit in the ADCR2 register.

- Step 7

  Select the A/D converter output data format and A/D boundary data format by configuring the ADRFS bit.

- Step 8

  If A/D conversion interrupt is used, the relevant interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, the A/D conversion interrupt control bit, ADE or ALIME, and the relevant Multi-function interrupt control bit must all be set high in advance.

- Step 9

  If the Step 6 selects START trigger circuit, the analog to digital conversion process can be initiated by setting the START bit from low to high and then low again. Note that this bit should have been originally cleared to zero. If the Step 6 selects PWM interrupt trigger circuit, the DLSTR bit in the ADCR2 register and the ADDL register should be properly configured, refer to the relevant register description.

- Step 10

  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers. If the internal OPA output signal is converted, the data from the SADOH and SADOL registers will be compared with the contents in the boundary register pairs after the A/D conversion is completed. If the comparison result interrupt trigger condition occurs, an interrupt request will be generated.

Note: If the method of polling the ADBZ bit in the SADC0 register is used when checking for the end of the conversion process, and the method of polling the ALIMF bit in the MFI1 register is used when checking for the preset comparison result, the interrupt enable step above can be omitted.

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 ADCEN low in the SADC0 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

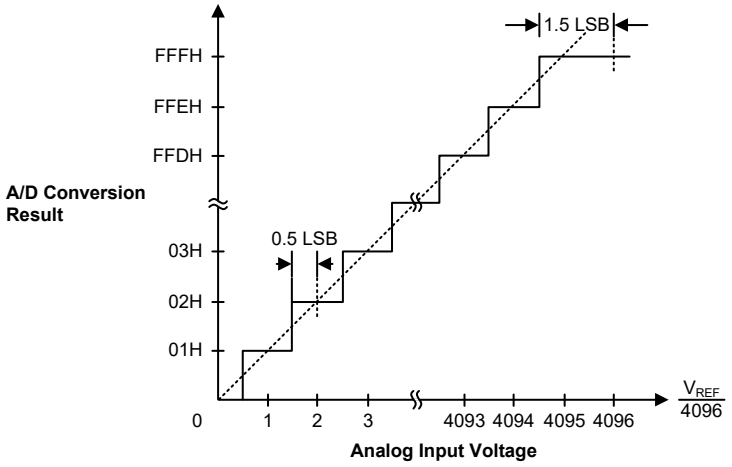
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the A/D converter reference voltage, V_{REF}, this gives a single bit analog input value of V_{REF} divided by 4096.

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

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

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

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_{REF} level. Note that here the V_{REF} is actually supplied by the A/D converter power supply.



Ideal A/D Conversion Function

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 ADBZ bit in the SADC0 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.

The following examples all use direct addressing, and since most of the relevant registers are located in Sector 1, all programs are demonstrated using extended instructions except for some branch type instructions.

Example1: using an ADBZ polling method to detect the end of conversion

```
lclr ADE              ; disable ADC interrupt
lmov a,03H            ; select fSYS/8 as A/D clock and A/D input signal comes from
                      ; external channel
lmov SADC1,a
lmov a,08H            ; setup PAS0 to configure pin AN0
lmov PAS0,a
lmov a,20H            ; enable A/D converter and select AN0 as the A/D external
                      ; channel input
lmov SADC0,a
:
start_conversion:
lclr START            ; high pulse on START bit to initiate conversion
lset START            ; reset A/D
lclr START            ; start A/D
:
```

```
polling_EOC:
lsz  ADBZ            ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp  polling_EOC     ; continue polling
:
lmov a,SADOL         ; read low byte conversion result value
lmov SADOL_buffer,a  ; save result to user defined register
lmov a,SADOH         ; read high byte conversion result value
lmov SADOH_buffer,a  ; save result to user defined register
:
jmp start_conversion ; start next A/D conversion
```

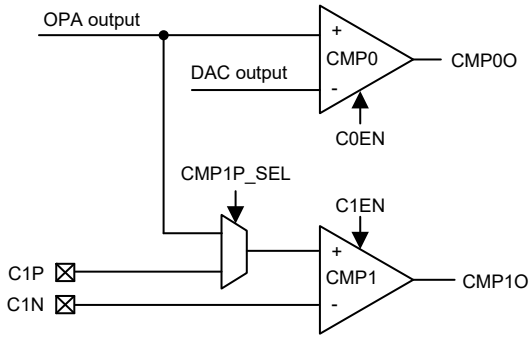
Example2: using the interrupt method to detect the end of conversion

```
lclr ADE             ; disable ADC interrupt
lmov a,03H           ; select fSYS/8 as A/D clock and A/D input signal comes from
                     ; external channel
lmov SADC1,a
lmov a,08H           ; setup PAS0 to configure pin AN0
lmov PAS0,a
lmov a,20H
lmov SADC0,a         ; enable A/D converter and select AN0 as the A/D external
                     ; channel input
:
Start_conversion:
lclr START           ; high pulse on START bit to initiate conversion
lset START           ; reset A/D
lclr START           ; start A/D
lclr ADF             ; clear ADC interrupt request flag
lset ADE             ; enable ADC interrupt
lset EMI             ; enable global interrupt
:
:
ADC_ISR:             ; ADC interrupt service routine
lmov acc_stack,a     ; save ACC to user defined memory
lmov a,STATUS
lmov status_stack,a  ; save STATUS to user defined memory
:
lmov a,SADOL         ; read low byte conversion result value
lmov SADOL_buffer,a  ; save result to user defined register
lmov a,SADOH         ; read high byte conversion result value
lmov SADOH_buffer,a  ; save result to user defined register
:
EXIT_INT_ISR:
lmov a,status_stack
lmov STATUS,a        ; restore STATUS from user defined memory
lmov a,acc_stack     ; restore ACC from user defined memory
reti
```

Comparators

Two independent analog comparators are contained within the device. These functions offer flexibility via their register controlled features such as power-down, 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.



Comparator Block Diagram

Comparator Operation

The device contains two 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, which 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 two comparators. The CnHYEN is the comparator n hysteresis control bit field and if set will apply a certain 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 IDLE/SLEEP mode. As the comparator 0 is used in the over current detector, there will be more description about it in the associated chapter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CMPC0	—	—	—	—	C1HYEN1	C1HYEN0	—	C0HYEN
CMPC1	—	—	—	—	—	—	C1EN	C0EN

Comparator Register List

• **CMPC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	C1HYEN1	C1HYEN0	—	C0HYEN
R/W	—	—	—	—	R/W	R/W	—	R/W
POR	—	—	—	—	0	0	—	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **C1HYEN1~C1HYEN0**: Comparator 1 hysteresis control
00: Off
01: Level 1
10: Level 2
11: Level 3

Bit 1 Unimplemented, read as "0"

Bit 0 **C0HYEN**: Comparator 0 hysteresis control
0: Off
1: On

• **CMPC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	C1EN	C0EN
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

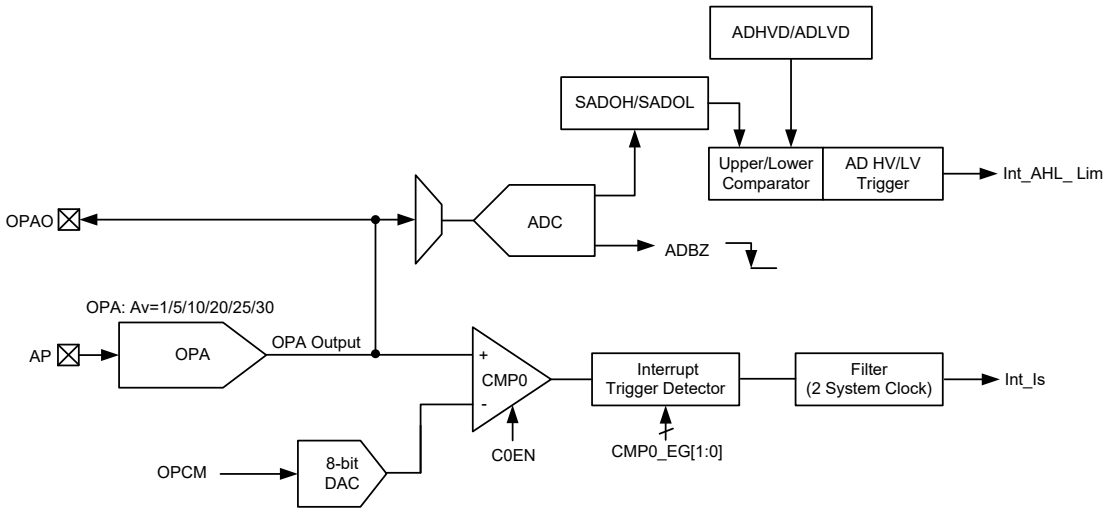
Bit 7~2 Unimplemented, read as "0"

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 includes a fully integrated over current detection circuit which is used for motor protection.



Over Current Detection Block Diagram

Over Current Detection 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 the 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 for current detection.

1. A/D converter comparison result interrupt – Int_AHL_Lim

2. Comparator 0 interrupt – Int_Is

Over Current Detection Registers

There are a series of registers to control the function and operation of the over current detection circuits. These 8-bit registers define functions such as comparator interrupt edge 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. The on/off control and hysteresis control of comparator 0 can be referred to the "Comparator Register" chapter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OPOMS	CMP0_EG1	CMP0_EG0	CMP1P_SEL	—	—	OPAVS2	OPAVS1	OPAVS0
OPCM	D7	D6	D5	D4	D3	D2	D1	D0
OPACAL	ARS	AOFM	—	AOF4	AOF3	AOF2	AOF1	AOF0

Over Current Detection Register List

- **OPOMS Register**

Bit	7	6	5	4	3	2	1	0
Name	CMP0_EG1	CMP0_EG0	CMP1P_SEL	—	—	OPAVS2	OPAVS1	OPAVS0
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	1	1	0	—	—	0	0	0

Bit 7~6 **CMP0_EG1~CMP0_EG0**: Interrupt trigger edge selection for Comparator 0
00: Disable – Comparator 0 and D/A converter both disabled
01: Rising edge
10: Falling edge
11: Dual edge
If these bits are set to "11", a comparator 0 interrupt can be normally generated when edges appear, but it cannot be used as the protection trigger signal.

Bit 5 **CMP1P_SEL**: CMP1 positive input source selection
0: C1P pin
1: OPA output

Bit 4~3 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: Av=25
101: Av=30
110: Av=1
111: Undefined
Note that when the OPA is used, the corresponding pin-shared control bit should be properly configured to enable the AP pin function.

• **OPCM Register**

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

Bit 7~0 **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 ensure that the D/A converter can output successfully.

DAC $V_{OUT} = (AV_{DD}/256) \times D[7:0]$

• **OPACAL Register**

Bit	7	6	5	4	3	2	1	0
Name	ARS	AOFM	—	AOF4	AOF3	AOF2	AOF1	AOF0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	1	0	0	0	0

Bit 7 **ARS**: Reference input selection for OPA offset calibration
0: Inverting input of OPA
1: Non-inverting input of OPA

Bit 6 **AOFM**: Normal or calibration mode selection
0: Normal Mode
1: Offset Calibration Mode

Bit 5 Unimplemented, read as "0"

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

BLDC Motor Control Circuit

This section describes how the device can be used to control Brushless DC Motors, also 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 and the low side, GAB/GBB, of the external Gate Driver Transistor Pairs under software control.

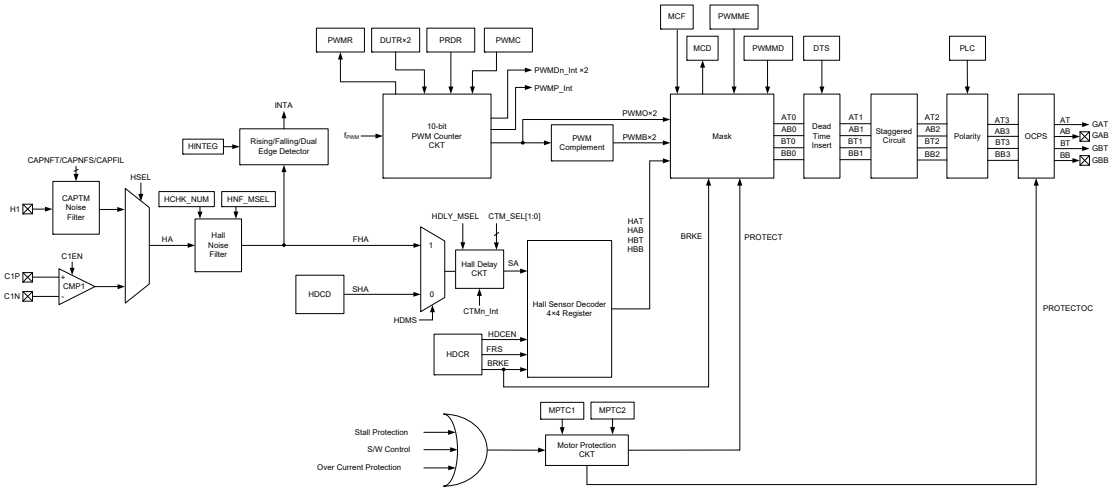
The Dead Time Insertion circuit is used to ensure the high and low sides of a Gate Driver Transistor will not be turned on at the same time to prevent a virtual power short circuit. The dead time should be configured in the range of 0.3μs~5μs by related register under software control.

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

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 to control the motor direction. Four registers, each using four bits, are used to control the direction of the motor. The motor forward, backward, brake and free running functions are controlled by the HDCD/HDCR registers. The HA or SHA signals can be selected as the Hall Sensor Decoder circuit inputs.

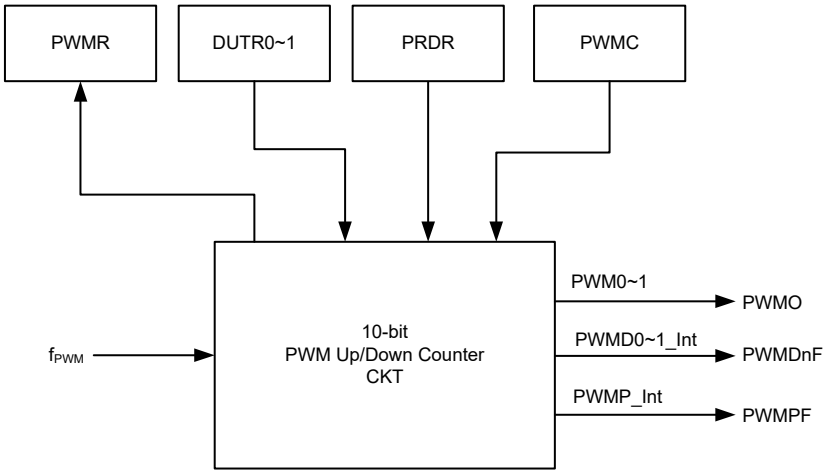


Note: The GAT, GAB, GBT, GBB are the PWM outputs of PWM0H, PWM0L, PWM1H, PWM1L respectively.

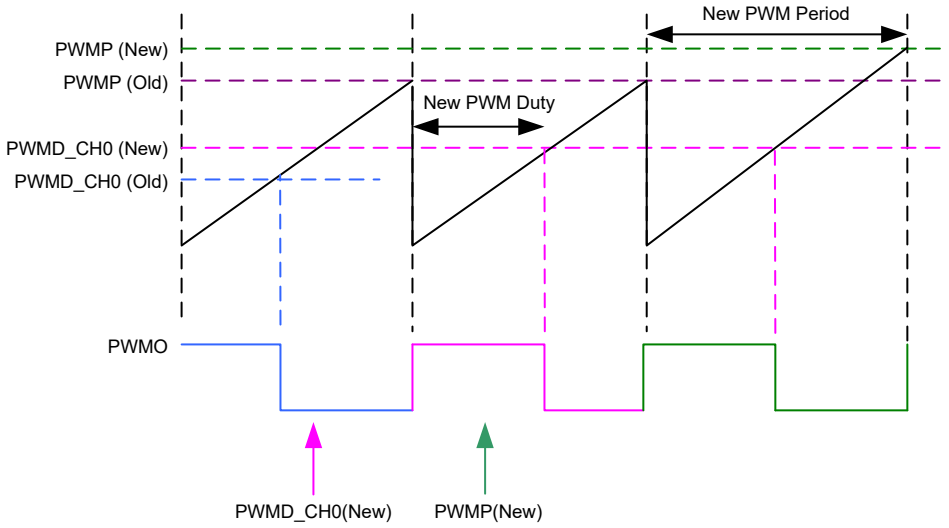
BLDC Motor Control Block Diagram

PWM Counter Control Circuit

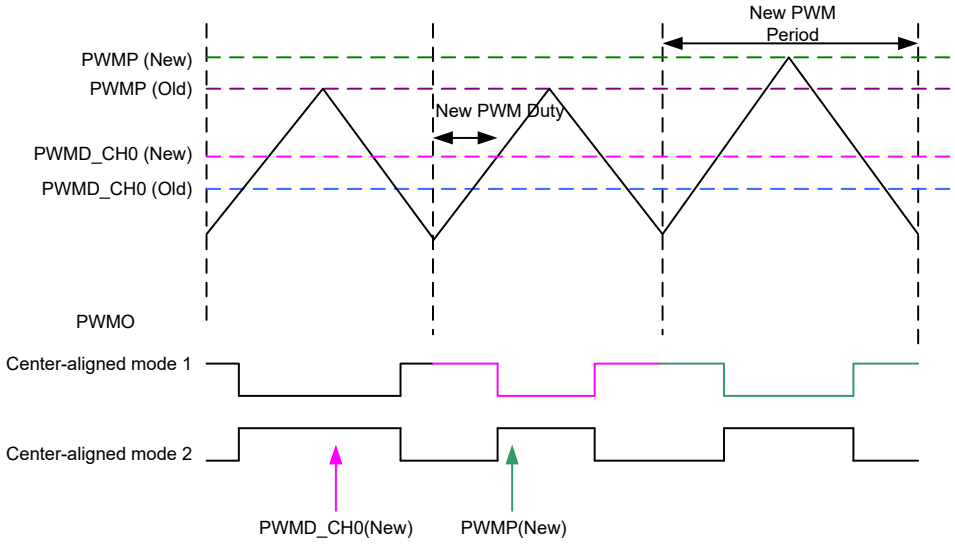
The BLDC Motor control circuit includes a 10-bit PWM generator. The duty cycle and frequency of the PWM signal can be adjusted by programming 10-bit data into the corresponding PWM duty and period control registers.



PWM Block Diagram



PWM Edge-Aligned Mode Timing Diagram



PWM Center-Aligned Mode Timing Diagram

PWM Duty Synchronous Update Modes

In high speed BLDC applications, using PWM interrupt to update the duty may result in asynchronous update for the two 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 two PWM duty values.

- DUTR0~DUTR1 PWM duty outputs are same

  Usually the two 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. In this way the PWM duty synchronous update is implemented with reduced instructions.

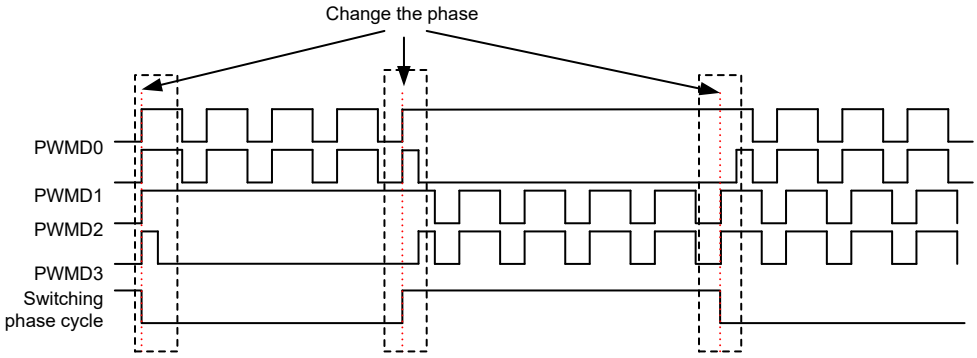
- DUTR0~DUTR1 PWM duty outputs are not the same

  If the two PWM duty outputs which require to be updated synchronously are not the same, set the PWMSU bit high to enable the PWM DUTR0~DUTR1 duty synchronous update function. When the PWM DUTR0~DUTR1 duty synchronous update request flag PWMSUF is set high, the hardware will synchronously update DUTR0 and DUTR1 values, after which the request flag will be automatically cleared to zero.

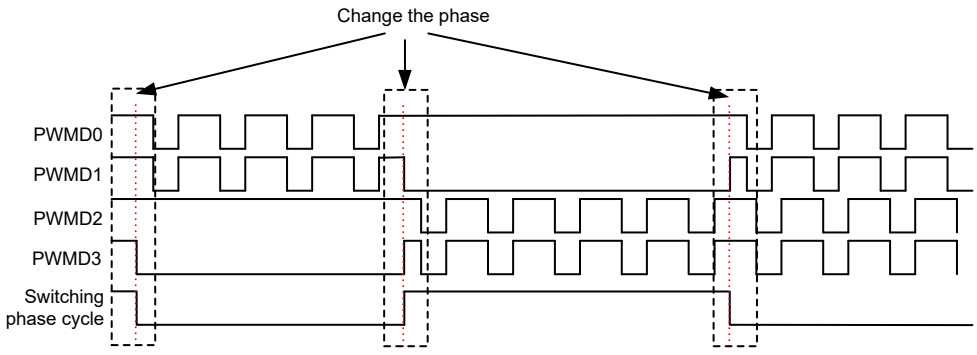
PWM Register Description

The overall PWM operation is controlled by a series of registers. The DUTRnL/DUTRnH register pairs are used for PWM duty control for adjustment of the motor output power, while the PRDRL/PRDRH register pair is 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 the10-bit PWM counter. The PWM counter clock source can be selected by PCKS[1:0] bits in the PWMC register. The PWMMS[1:0] bits in the PWMC register determines the PWM alignment type, which can be either edge or center type.

The PWMCS register is used to control PWM DUTR0~DUTR1 duty value synchronisation update function and determine if the PWM output state can be quickly updated 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 completed, and then output to the next phase, this circuit behavior is valid only for PMEn=0 (PWMME register) when in the Software Mask Mode, and is valid for HDCTn when in the Hardware Mask Mode. 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.



Phase Change Timing – PWMMEDO=0



Phase Change Timing – PWMMEDO=1

The DUTRn and PRDR registers are write-only registers, 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	PWMMS1	PWMMS0	PCKS1	PCKS0	PWMON	ITCMS1	ITCMS0	PWMLD
PWMCS	—	—	—	—	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
PRDRL	D7	D6	D5	D4	D3	D2	D1	D0
PRDRH	—	—	—	—	—	—	D9	D8
PWMRL	D7	D6	D5	D4	D3	D2	D1	D0
PWMRH	—	—	—	—	—	—	D9	D8

PWM Register List

• PWMC Register

Bit	7	6	5	4	3	2	1	0
Name	PWMMS1	PWMMS0	PCKS1	PCKS0	PWMON	ITCMS1	ITCMS0	PWMLD
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PWMMS1~PWMMS0**: 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 (f_{PWM})source selection
00: f_{SYS}
01: $f_{SYS}/2$
10: $f_{SYS}/4$
11: $f_{SYS}/8$

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 counting up condition
10: Center-aligned mode duty interrupt only in counting down condition
11: Center-aligned mode duty interrupt both in counting up and down conditions

Bit 0 **PWMLD**: PWM PRDR and DUTRn (n=0~1) register data update control
0: Do not reload the period value and duty 0~1 values in the PRDR and DUTRn (n=0~1) registers
1: Reload the period value and duty 0~1 values in the PRDR and DUTRn (n=0~1) registers after counter overflow/underflow

• **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 when PWM changes the phase
0: Disable
1: Enable

Bit 2 **PWMSUF**: PWM DUTR0~DUTR1 duty synchronous update request flag (when PWMSU=1)
0: No request
1: DUTR0~DUTR1 duty synchronous update request
Setting this bit high will request to update DUTR0~DUTR1 duty data synchronously. When synchronous update has finished, this bit will be automatically cleared to zero by hardware.

Bit 1 **PWMSU**: PWM DUTR0~DUTR1 duty synchronous update control (when PWMLD=1)
0: Disable
1: Enable

Bit 0 **PWMSV**: Synchronize DUTR0 content to DUTR1 control (when PWMLD=1)
0: Disable – DUTR0~DUTR1 should be configured separately
1: Enable – the PWM duty value written to DUTR0 is synchronised to DUTR1

• **DUTRnL Register (n=0~1)**

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=0~1)**

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

Edge-aligned Mode: PWM Period = (PRDR+1)/f_{PWM} (PRDR cannot be set to 000H)

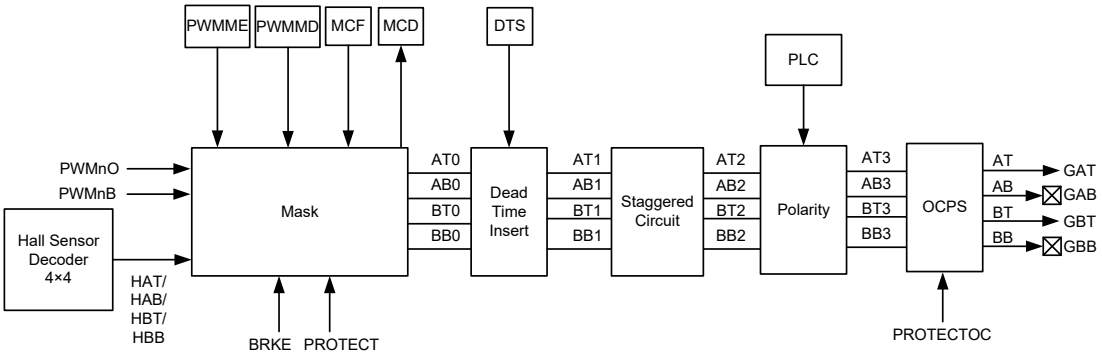
PWM Duty = DUTRn/f_{PWM}

Center-aligned Mode: PWM Period = (PRDR×2)/f_{PWM} (PRDR cannot be set to 000H)

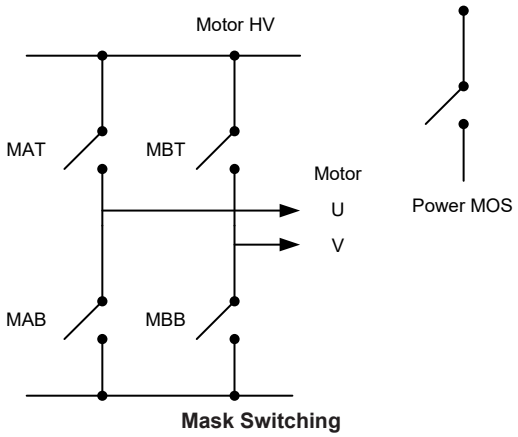
PWM Duty = (DUTRn×2-1)/f_{PWM}

MASK Function

The device includes a Mask function for Motor control to provide its control flexibility. The internal mask circuit supports three operating modes, which are known as the Normal Mode, Brake Mode and Motor Protection Mode. The Normal Mode has two sub-modes, namely Hardware Mask Mode and Software Mask Mode.



Mask Function Block Diagram



Mask Switching

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 will output PWM.
- When PWMS=1, Transistor pair high side GAT/GBT will output PWM.
- When MPWE=0, the PWM output is disabled and AT0/BT0/AB0/BB0 are all on.
- When MPWE=1, the PWM output is enabled and AT0/BT0/AB0/BB0 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 in the Software Mask Mode where the PWM output is determined by the PWMME and PWMMD registers.

Bit 2 **MPWE**: PWM output control
0: PWM output disable (AT0/BT0/AB0/BB0 cannot output PWM)
1: PWM output enable

Bit 1 **FMOS**: Mask output control when PROTECT is triggered
0: AT0/BT0=0, AB0/BB0=0
1: AT0/BT0=0, AB0/BB0=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	—	—	GAT	GAB	GBT	—	—	FHA
R/W	—	—	R	R	R	—	—	R
POR	—	—	0	0	0	—	—	x

"x": Unknown

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **GAT/GAB/GBT**: Gate-driver output

Bit 2~1 Unimplemented, read as "0"

Bit 0 **FHA**: HA filtered output via Hall Noise Filter
This signal is derived from the HA signal and filtered by the Hall Noise Filter.

Hardware Mask Mode

Complementary control, MPWMS=0

	HAT	HAB	AT0	AB0		HAT	HAB	AT0	AB0
PWMS=0	0	0	0	0	**PWMS=1**	0	0	0	0
	0	1	PWM0B	PWM0O		0	1	0	1
	1	0	1	0		1	0	PWM0O	PWM0B
	1	1	0	0		1	1	0	0

	HBT	HBB	BT0	BB0		HBT	HBB	BT0	BB0
PWMS=0	0	0	0	0	**PWMS=1**	0	0	0	0
	0	1	PWM1B	PWM1O		0	1	0	1
	1	0	1	0		1	0	PWM1O	PWM1B
	1	1	0	0		1	1	0	0

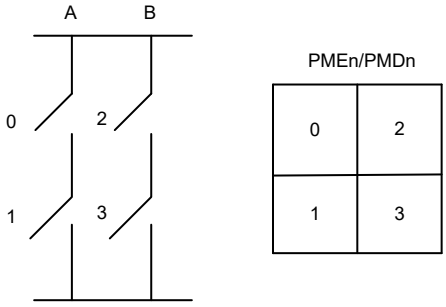
Non-complementary control, MPWMS=1

	HAT	HAB	AT0	AB0		HAT	HAB	AT0	AB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	0	PWM0O	PWMS=1	0	1	0	1
	1	0	1	0		1	0	PWM0O	0
	1	1	0	0		1	1	0	0

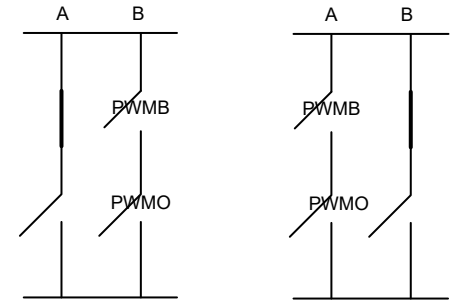
	HBT	HBB	BT0	BB0		HBT	HBB	BT0	BB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	0	PWM1O	PWMS=1	0	1	0	1
	1	0	1	0		1	0	PWM1O	0
	1	1	0	0		1	1	0	0

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 Gate-Driver Circuit is on or off.



1-Phase Inverter Symbol and Control Bits

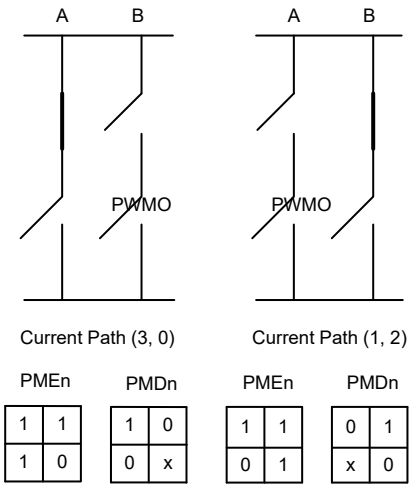


Current Path (3, 0) Current Path (1, 2)

PMEn		PMDn	
1	0	1	x
1	0	0	x

PMEn		PMDn	
0	1	x	1
0	1	x	0

Mask Complement Mode Examples



Mask Independent Mode Examples

Note: 1. If the mask mode is enabled, when PWMxH and PWMxL are masked at the same time, the two line data of each pair, PMD0 and PMD1, PMD2 and PMD3, cannot be both set to "1", otherwise the output will be forced to 0.

2. If PWM and complementary PWM are both enabled, one of the complementary channels outputs PWM and the other one cannot be masked to "1" but outputs "0" automatically by hardware.

3. If the GxT and GxB (x=A or B) 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	—	—	—	—	PME3	PME2	PME1	PME0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **PMEn**: PWM mask enable bit (n=0~3)
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	—	—	—	—	PMD3	PMD2	PMD1	PMD0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **PMDn**: PWM mask data bit (n=0~3)
0: Output logic low
1: Output logic high
When the corresponding mask enable bit PMEn=1, this Mask data bit can control the PWMn output.

Brake Mode

The brake mode has the highest priority. When this mode is activated, all the low sides of the external gate driver transistors are all on and the high sides are all 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	AB0	BB0
	0	0	1	1

Motor Protection Mode

When the PROTECT bit is set high, the motor protection mode 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	AT0	BT0	AB0	BB0
FMOS=0	0	0	0	0
FMOS=1	0	0	1	1

Other Functions

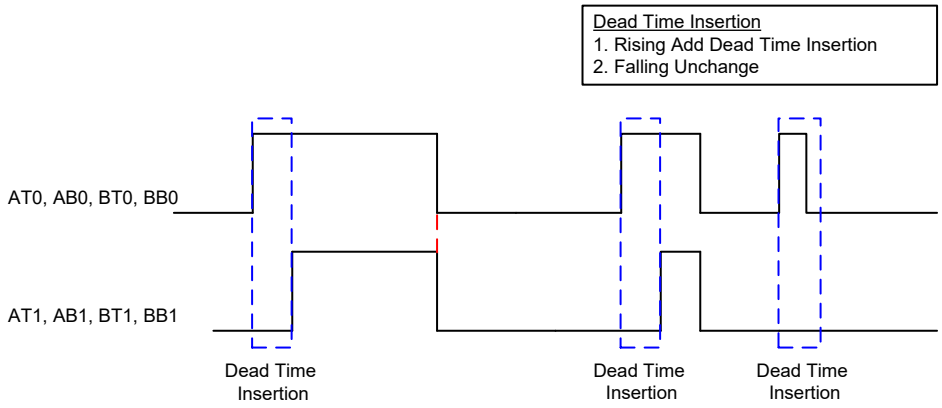
Several other functions exist for additional motor control drive signal flexibility. These are the Dead Time Insertion Function, Staggered Function and Polarity Control Function.

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 0.3μs~5μs and is selected by the application program.

- When the AT0/AB0/BT0/BB0 outputs at a rising edge, a Dead Time is inserted.
- When the AT0/AB0/BT0/BB0 outputs at a falling edge, their outputs remain unchanged.

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



Dead Time Insertion Timing

• **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: $f_{DT}=f_{SYS}$
01: $f_{DT}=f_{SYS}/2$
10: $f_{DT}=f_{SYS}/4$
11: $f_{DT}=f_{SYS}/8$

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 the external transistor pair high and low sides are both on resulting from a software error occurrence 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: In BLDC motor control circuits, external gate-driver transistor pairs are designed as N-type transistor pairs by default. This means a “1” value will switch the transistor on and a “0” value will switch it off. However, for P-type transistor pairs, a “0” value will switch the transistor on and a “1” value will switch it off.

Polarity Control Function

This function allows setup of the external gate-driver transistor output polarity status. A single register, PLC, is used for the overall control. Note that the default output status of the GAT/GAB/GBT/GBB pins are all high impedance.

• **PLC Register**

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

Bit 7~4 Unimplemented, read as “0”

Bit 3 **PBBC**: GBB output polarity control
0: Non-invert
1: Invert

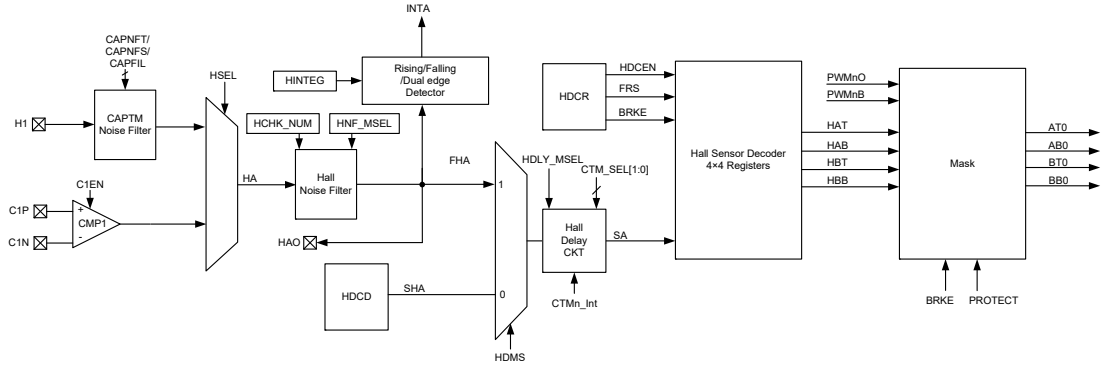
Bit 2 **PBTC**: GBT output polarity control
0: Non-invert
1: Invert

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

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.

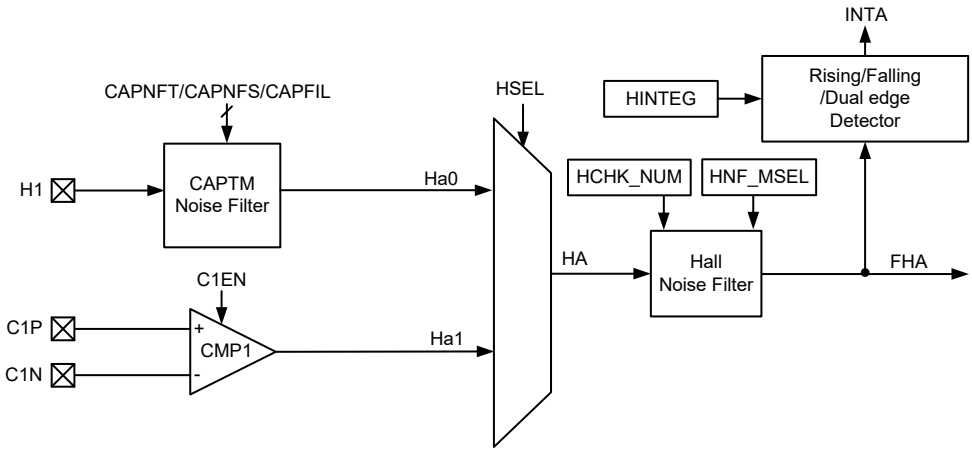


Hall Sensor Decoder Block Diagram

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

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 input (HA), 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 chapter. The comparator relevant control is described in the Comparators chapter.

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 for the overall enable/disable control of the noise filter.

HNF_EN bit	Status
0	Noise filter off – HA 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 set the number of Hall Sensor input comparisons.

HCK_N[4:0]×Sampling space=Anti-noise ability=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_SEL[1:0] 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. A low to high transition of the HDLY_MSEL bit is required to allow the hardware to initialize the Hall Delay circuit.

- Step 4

  Enable the Hall Decoder using the HDCEN bit.

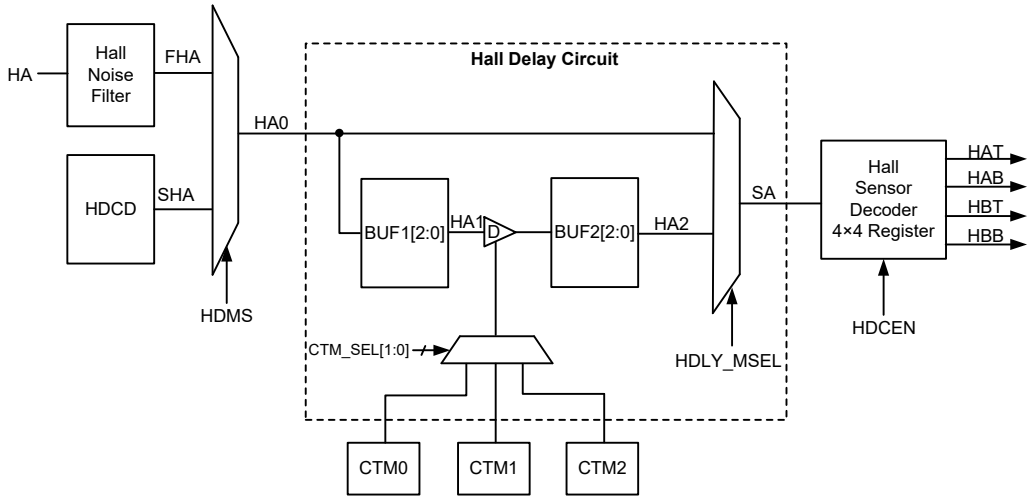
The following points should be noted regarding the HDLY_MSEL bit.

1. When this bit is low, BUF1[2:0] and BUF2[2:0] will be cleared to zero.
2. When this bit is low, CTM0, CTM1 and CTM2 remain their original TM functions.
3. When this bit is high, the CTMn 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 valid except for that the CTnON bit which will be controlled automatically by the hardware. And the selected CTMn should be properly configured according to the requirement.

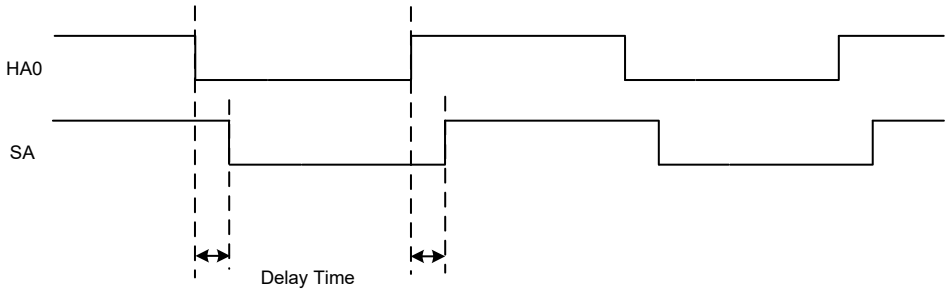
With regard to the selected CTMn functions the following notes should be considered before the Delay function is enabled.

1. Remain CTnON=0 and CTnPAU=0.
2. The CTMn should be set to operate in the Compare Match Output Mode.
3. Set CTnCCLR=1, therefore the CTMn counter is cleared with a comparator A match condition.
4. Setup the Delay time by properly setting the CTMn CCRA and selecting the counter clock.

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



Hall Delay Function Block Diagram



Delay Function Timing

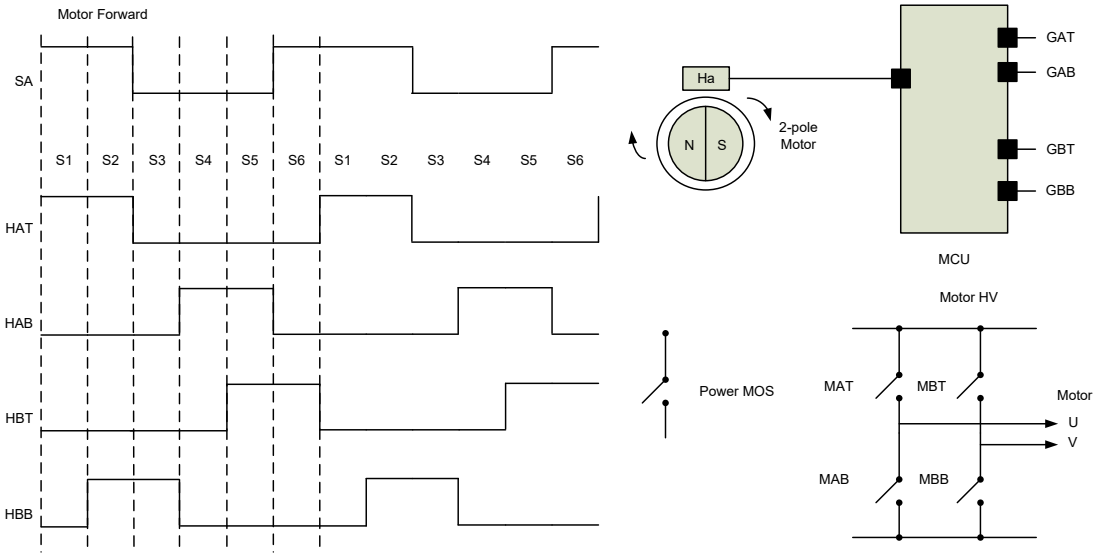
Motor Control Drive Signals

The direction of the BLDC motor is controlled using the HDCR and HDCD registers as well as a series of 4-bit HDCT registers, HDCT0~HDCT3. 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. The value of the 4-bit registers HDCT0~HDCT1 are used for the Motor Forward table, and The value of the 4-bit registers HDCT2~HDCT3 are used for the Motor Backward table.

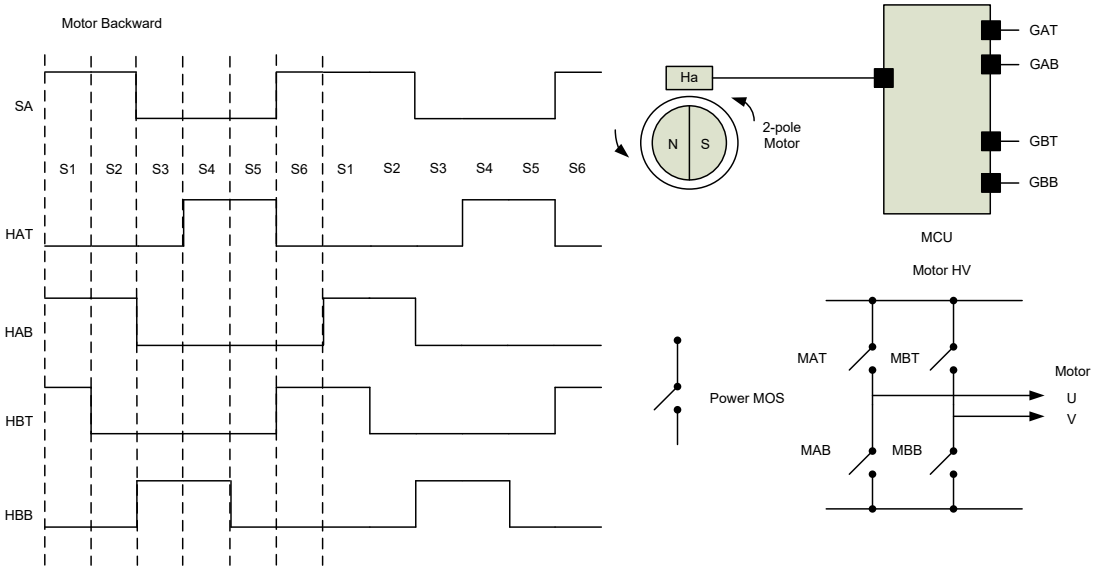
The accompanying tables show the truth tables for each of the registers.

Forward (HDCEN=1, FRS=0, BRKE=0)	**SA**	**Bit5**	**Bit4**	**Bit3**	**Bit2**
		HAT	**HAB**	**HBT**	**HBB**
	1	HDCT0[5:2]			
	0	HDCT1[5:2]			
Backward (BRKE=0, HDCEN=1, FRS=1)	**SA**	**Bit5**	**Bit4**	**Bit3**	**Bit2**
		HAT	**HAB**	**HBT**	**HBB**
	1	HDCT2[5:2]			
	0	HDCT3[5:2]			
Brake (BRKE=1, HDCEN=1, FRS=x)	**SA**	**Bit5**	**Bit4**	**Bit3**	**Bit2**
		HAT	**HAB**	**HBT**	**HBB**
	V	0	1	0	1
Hall Decoder Disabled (HDCEN=0)	**SA**	**Bit5**	**Bit4**	**Bit3**	**Bit2**
		HAT	**HAB**	**HBT**	**HBB**
	V	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.



Motor Drive Signal Timing Diagram – Forward Direction



Motor Drive Signal Timing Diagram – Backward Direction

Hall Sensor Decoder Register Description

The HDCR register is the Hall Sensor Decoder control register, HDCD is the Hall Sensor Decoder input data register, and HDCT0~HDCT3 are the Hall Sensor Decoder tables. The HCHK_NUM register defines the number of Hall Noise Filter detections and HNF_MSEL is the Hall Noise Filter Mode selection register. The HINTEG register is Hall Noise Filter input source selection and interrupt edge control register.

• **HINTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	—	HSEL	—	—	—	—	INTAS1	INTAS0
R/W	—	R/W	—	—	—	—	R/W	R/W
POR	—	0	—	—	—	—	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **HSEL**: HA source selection
0: H1
1: CMP1 output

Bit 5~2 Unimplemented, read as "0"

Bit 1~0 **INTAS1~INTAS0**: FHA trigger edge selection for INTA interrupt
00: Disable
01: Rising edge
10: Falling edge
11: Dual edge

• **HDCR Register**

Bit	7	6	5	4	3	2	1	0
Name	CTM_SEL1	CTM_SEL0	HDLY_MSEL	—	HDMS	BRKE	FRS	HDCEN
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 7~6 **CTM_SEL1~CTM_SEL0**: Select TM used for Hall Delay Circuit
00: CTM0 CCRA match
01: CTM1 CCRA match
10: CTM2 CCRA match
11: Unused

Bit 5 **HDLY_MSEL**: Hall Delay Circuit selection
0: Original path (Bypass Delay Circuit)
1: Hall Delay Circuit

Bit 4 Unimplemented, read as "0"

Bit 3 **HDMS**: Hall Sensor decoding mode selection
0: Software Mode (SHA in the HDCD register)
1: Hall Sensor Mode (FHA via noise filter)

Bit 2 **BRKE**: Motor brake control
0: AT/BT/AB/BB=V
1: AT/BT=0, AB/BB=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	—	—	—	—	—	—	—	SHA
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **SHA**: Software Hall A

• **HDCTn Register (n=0~3)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	HATDn	HABDn	HBTDn	HBBDn	—	—
R/W	—	—	R/W	R/W	R/W	R/W	—	—
POR	—	—	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 1~0 Unimplemented, read as "0"

• **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 detections

• **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

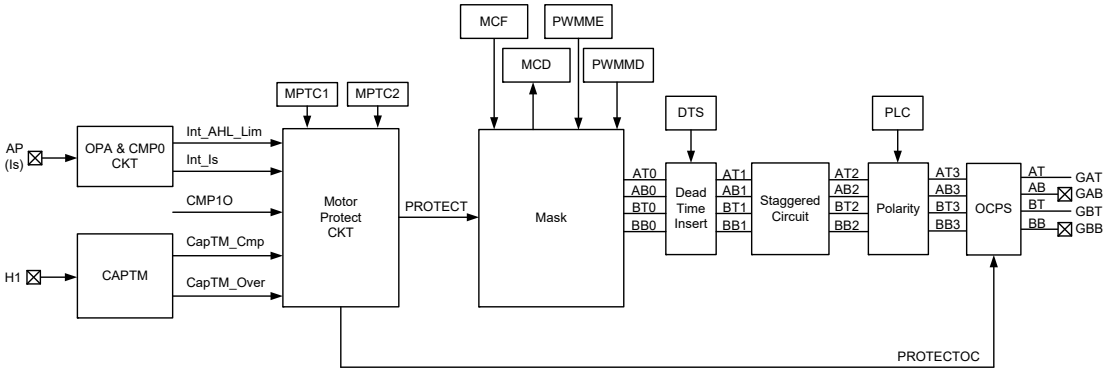
Bit 7~4 Unimplemented, read as "0"

Bit 3 **HNF_EN**: Hall Noise Filter control
0: Disable (bypass)
1: Enable

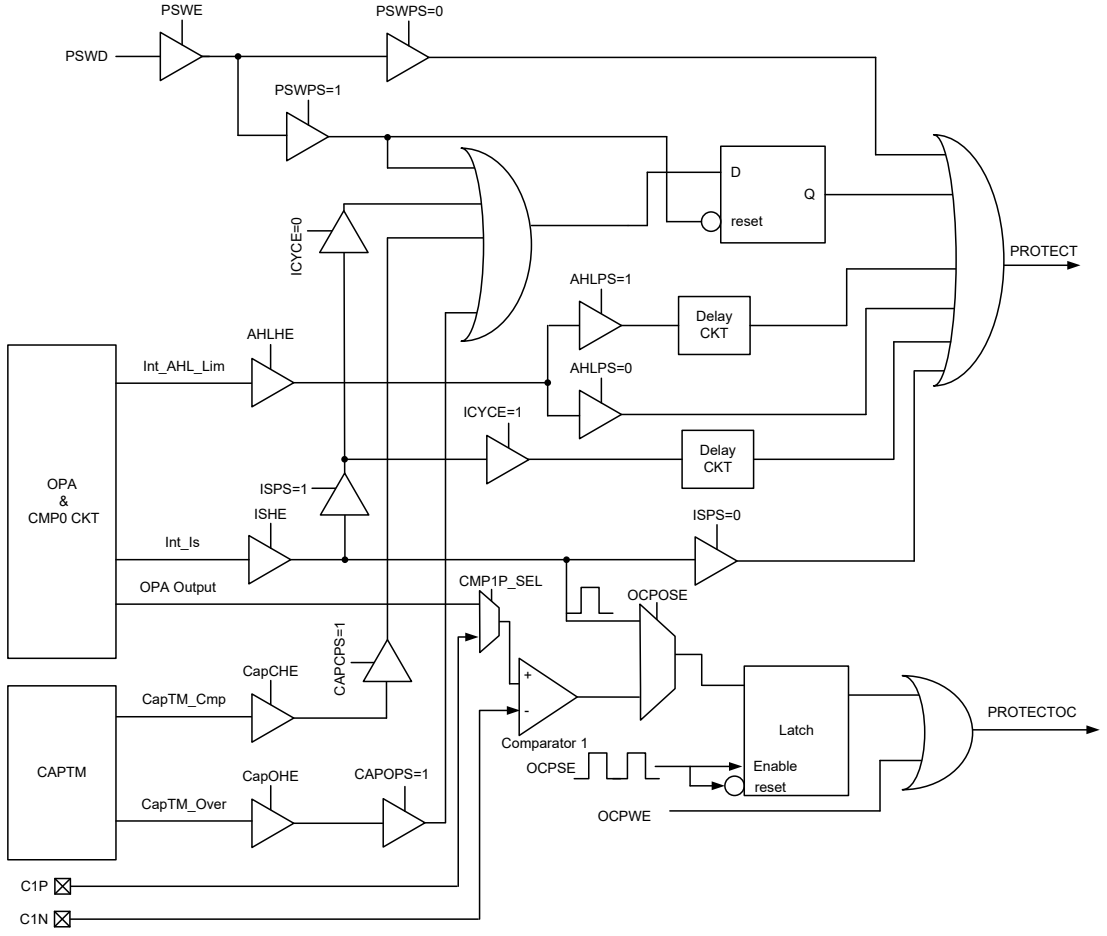
Bit 2~0 **HFR_SEL2~HFR_SEL0**: Hall Noise Filter clock source selection
000: $f_{SYS}/2$
001: $f_{SYS}/4$
010: $f_{SYS}/8$
011: $f_{SYS}/16$
100: $f_{SYS}/32$
101: $f_{SYS}/64$
110: $f_{SYS}/128$
111: Unused

Motor Protection Function

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



Motor Protection Function Block Diagram



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-driver transistor pair can be forced into two different status, which is determined by the FMOS bit in the MCF register. The first is the brake status where the high sides are all off and the low sides are all on, and the second is the free running status where both high and low sides are off.

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 cut off the driver signal output quickly, providing a more immediate current protection machenism. If an over current condition has been detected, the preset OCPS protection signals will be output immediately 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 trigger) or OCPWE (software trigger) bit in the MPTC1 register. Whatever the PROTECTOC signal triggerred source is, it can only be released by software.

Current Protection Machenism

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 occurs, 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 over current protection machenism must use the Fault Mode.

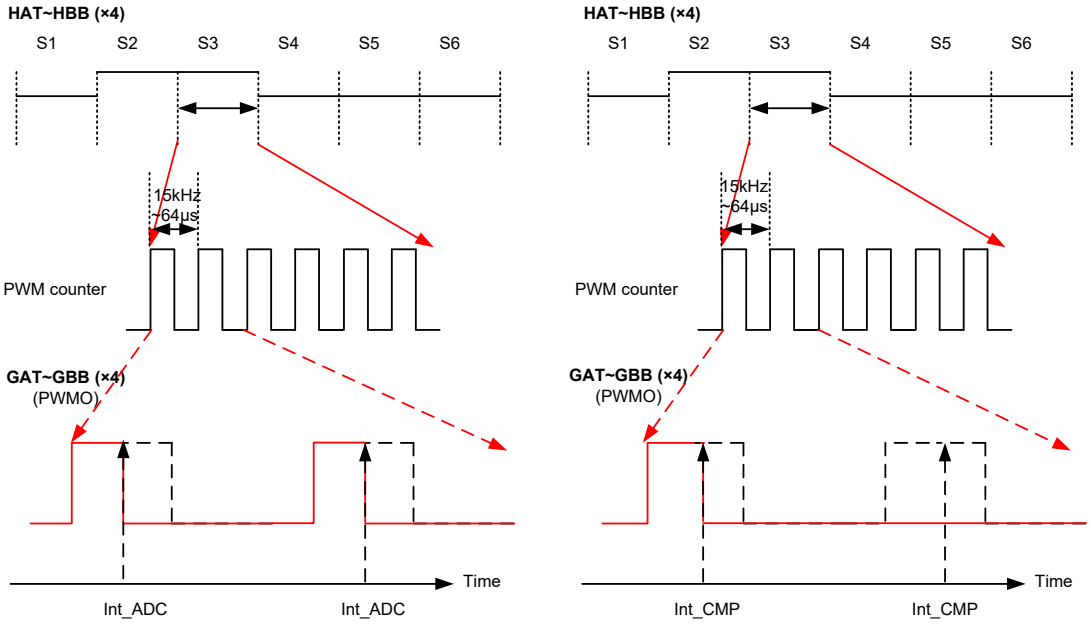
The comparator 0 output is filtered by a 2 system clock filter circuit to generate the Int_Is signal to trigger protection.

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

AHLPS=0 → The protection ciruit will allow the PWM output to immediately restart once the Int_AHL_Lim interrupt protection is 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 is 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. Then select the Fault mode by clearing the ISPS bit to zero.



MOS Current Limiting Protection: (AHLHE=1; AHLPS=1)
Start the next cycle of PWM output automatically by hardware

MOS Over Current Protection: (ISHE=1; ISPS=0)
PWM output must be restarted by software

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 zero to reset the PROTECTOC signal, after which the normal Polarity drive signals will be recovered.

Motor Stall Detection Function

For 1-phase BLDC applications with Hall Sensors, the integrated 16-bit CAPTM can be used to monitor the H1 input for rotor speed detection. The software will setup the CAPTMAH and CAPTMAL registers to monitor the Hall sensor input H1 for rotor speed control. If an abnormal situation occurs, a CapTM_Cmp or CapTM_Over interrupt will be generated, which is described in the CAPTM section.

CapTM_Cmp Stall Detection Mechanism: CapCHE=0 disables the hardware trigger mode, and CapCHE=1 enables the hadware trigger mode. The stall detection mechanism must use the Pause Mode which is selected by setting the CAPCPS bit high.

CapTM_Over Stall Detection Mechanism: CapOHE=0 disables the hardware trigger mode, and CapOHE=1 enables the hardware trigger Mode. Then select the Pause Mode by setting the CAPOPS bit high.

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
In the Pause Mode, the PSWD bit can be used to trigger protection and release protection. In the Fault Mode, the PSWD bit can only be used to release protection, in which case the PSWE bit should be set to "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 high to low then to high.

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

- **MPTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	OCPOSE	ICYCE	PSWPS	AHLPS	ISPS	CAPCPS	CAPOPS
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 **OCPOSE**: PROTECTOC hardware trigger source selection
0: Int_Is
1: CMP1 output rising edge

Bit 5 **ICYCE**: Over current hardware cycle-by-cycle protection
0: Disable
1: Enable

The protection circuit will switch off the PWM output once the Int_Is has an over current condition, and recover the PWM output on the next PWM period after the over current protection is removed. However the PWM output will be switch off again immediately if there is still an over current condition.

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 used
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 used
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	—	—	—	—	OCPBB	OCPBT	OCPAB	OCPAT
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as “0”

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

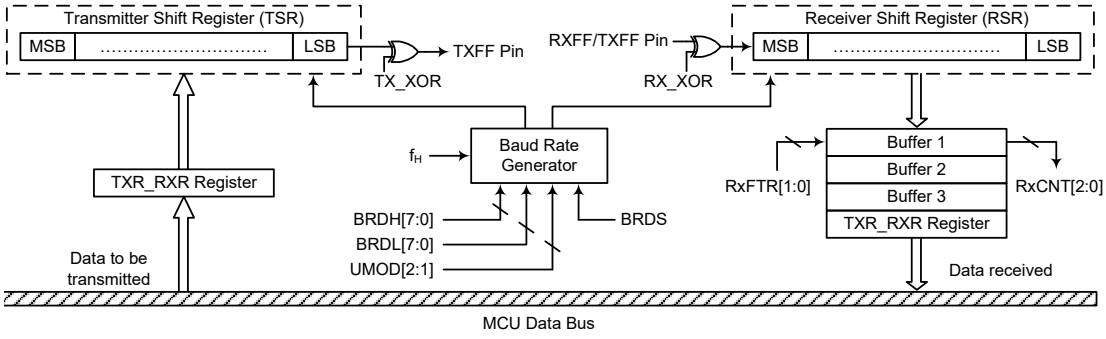
Bit 0 **OCPAT**: GAT over current protection output selection
0: Output 0
1: Output 1

UART Interface

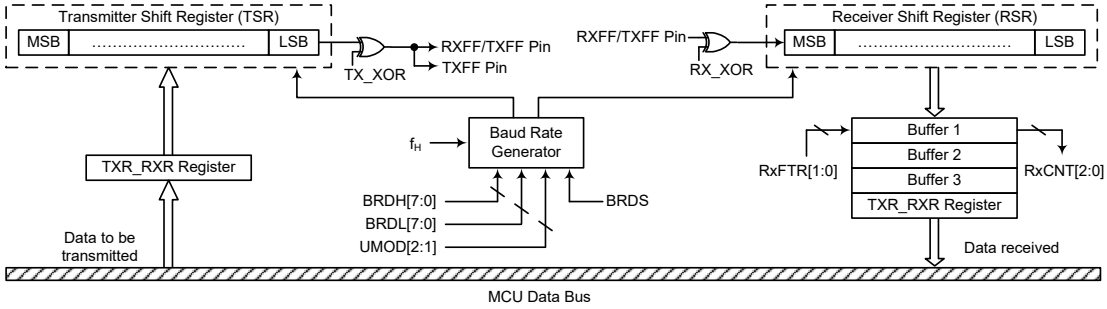
The device contains an integrated full-duplex or half-duplex asynchronous serial communication UART interface that enables communication with external devices that contain a serial interfaces. The UART function has many features and can transmit and receive data serially by transferring a frame of data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

Each integrated UART function contains the following features:

- Full-duplex or half-duplex (single wire mode) asynchronous communication
- 8 or 9 bits character length
- Even, odd, mark, space or no parity options
- One or two stop bits configurable for receiver
- Two stop bits for transmitter
- Baud rate generator with 16-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- Separately enabled transmitter and receiver inverter functions
- 4-byte Deep FIFO Receive Data Buffer
- 1-byte Deep FIFO Transmit Data Buffer
- RXFF/TXFF pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
  - Transmitter Empty
  - Transmitter Idle
  - Receiver reaching FIFO trigger level
  - Receiver Overrun
  - Address Mode Detect



UART Data Transfer Block Diagram – SWM=0



UART Data Transfer Block Diagram – SWM=1

UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as TXFF and RXFF/TXFF, which are pin-shared with I/O or other pin functions. The TXFF and RXFF/TXFF pin function should first be selected by the corresponding pin-shared function selection register before the UART function is used. Along with the UARTEN bit, the TXEN and RXEN bits, if set, will configure these pins to transmitter output and receiver input conditions. At this time the internal pull-high resistor related to the transmitter output pin will be disabled, while the internal pull-high resistor related to the receiver input pin is controlled by the corresponding I/O pull-high function control bit. When the TXFF or RXFF/TXFF pin function is disabled by clearing the UARTEN, TXEN or RXEN bit, the TXFF or RXFF/TXFF pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the TXFF or RXFF/TXFF pin or not is determined by the corresponding I/O pull-high function control bit.

UART Single Wire Mode

The UART function also supports a Single Wire Mode communication which is selected using the SWM bit in the UCR3 register. When the SWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single RXFF/TXFF pin can be used to transmit and receive data depending upon the corresponding control bits. When the RXEN bit is set high, the RXFF/TXFF pin is used as a receiver pin. When the RXEN bit is cleared to zero and the TXEN bit is set high, the RXFF/TXFF pin will act as a transmitter pin.

It is recommended not to set both the RXEN and TXEN bits high in the single wire mode. If both the RXEN and TXEN bits are set high, the RXEN bit will have the priority and the UART will act as a receiver.

It is important to note that the functional description in this UART chapter, which is described from the full-duplex communication standpoint, also applies to the half-duplex (single wire mode) communication except the pin usage. In the single wire mode, the TXFF pin mentioned in this chapter should be replaced by the RXFF/TXFF pin to understand the whole UART single wire mode function.

In the single wire mode, the data can also be transmitted on the TXFF pin in a transmission operation with proper software configurations. Therefore, the data will be output on the RXFF/TXFF and TXFF pins.

UART Data Transfer Scheme

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

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

It should be noted that the actual register for data transmission and reception only exists as a single shared register, TXR_RXR, in the Data Memory.

UART Status and Control Registers

There are nine control registers associated with the UART function. The UCR3 register is used to enable/disable the UART Single Wire Mode and control the transmitter/receiver inverter function. The USR, UCR1, UCR2, UFCR and RxCNT registers control the overall function of the UART, while the BRDH and BRDL registers control the baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXR data register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
UCR3	—	—	—	—	—	TX_XOR	RX_XOR	SWM
TXR_RXR	D7	D6	D5	D4	D3	D2	D1	D0
BRDH	D7	D6	D5	D4	D3	D2	D1	D0
BRDL	D7	D6	D5	D4	D3	D2	D1	D0
UFCR	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
RxCNT	—	—	—	—	—	D2	D1	D0

UART Register List

• USR Register

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

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

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

Bit 6 **NF**: Noise flag
0: No noise is detected
1: Noise is detected

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

Bit 5 **FERR**: Framing error flag
0: No framing error is detected
1: Framing error is detected

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

Bit 4 **OERR**: Overrun error flag
0: No overrun error is detected
1: Overrun error is detected

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

Bit 3 **RIDLE**: Receiver status
0: Data reception is in progress (Data being received)
1: No data reception is in progress (Receiver is idle)

The RIDLE flag is the receiver status flag. When this read only flag is "0", it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is "1", it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is "1" indicating that the UART receiver is idle and the RXFF/TXFF pin stays in logic high condition.

Bit 2 **RXIF**: Receive TXR_RXR data register status
0: TXR_RXR data register is empty
1: TXR_RXR data register has available data and reach Receiver FIFO trigger level is reached

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

Bit 1 **TIDLE**: Transmission idle
0: Data transmission is in progress (Data being transmitted)
1: No data transmission is in progress (Transmitter is idle)

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

Bit 0 **TXIF**: Transmit TXR_RXR data register status
0: Character is not transferred to the transmit shift register
1: Character has transferred to the transmit shift register (TXR_RXR data register is empty)

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

- **UCR1 Register**

The UCR1 register together with the UCR2 and UCR3 register are the three UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length and single wire mode communication etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

"x": Unknown

Bit 7 **UARTEN**: UART function enable control
0: Disable UART – TXFF and RXFF/TXFF pins are in a floating state
1: Enable UART – TXFF and RXFF/TXFF pins function as UART pins

The UARTEN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RXFF/TXFF pin as well as the TXFF pin will be set in a floating state. When the bit is equal to "1", the UART will be enabled and the TXFF and RXFF/TXFF pins will function as defined by the the SWM mode selection bit together with TXEN and RXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits as well as the RxCNT register will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6 **BNO**: Number of data transfer bits selection
0: 8-bit data transfer
1: 9-bit data transfer

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

Note that the 9th bit of data if BNO=1, or the 8th bit of data if BNO=0, which is used as the parity bit, does not transfer to RX8 or TXR_RXR.7 respectively when the parity function is enabled.

Bit 5 **PREN**: Parity function enable control
0: Parity function is disabled
1: Parity function is enabled

This is the parity enable bit. When this bit is equal to "1", the parity function will be enabled. If the bit is equal to "0", then the parity function will be disabled.

Bit 4~3 **PRT1~PRT0**: Parity type selection bits
00: Even parity for parity generator
01: Odd parity for parity generator
10: Mark parity for parity generator
11: Space parity for parity generator
These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.

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

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

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

• UCR2 Register

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

Bit	7	6	5	4	3	2	1	0
Name	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TXEN**: UART transmitter enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled
The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to "0", the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TXFF pin will be set in a floating state.
If the TXEN bit is equal to "1" and the UARTEN bit are also equal to "1", the transmitter will be enabled and the TXFF pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TXFF pin will be set in a floating state.

Bit 6 **RXEN**: UART Receiver enabled control
0: UART receiver is disabled
1: UART receiver is enabled
The bit named RXEN is the Receiver Enable Bit. When this bit is equal to "0", the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RXFF/TXFF pin will be set in a floating state. If the RXEN bit is equal to "1" and the UARTEN bit is also equal to "1",

the receiver will be enabled and the RXFF/TXFF pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RXFF/TXFF pin will be set in a floating state.

Bit 5 **STOPS**: Number of stop bits selection for receiver
0: One stop bit format is used
1: Two stop bits format is used

This bit determines if one or two stop bits are to be used for receiver. When this bit is equal to "1", two stop bits are used. If this bit is equal to "0", then only one stop bit is used. Two stop bits are used for transmitter.

Bit 4 **ADDEN**: Address detect function enable control
0: Address detect function is disabled
1: Address detect function is enabled

The bit named ADDEN is the address detect function enable control bit. When this bit is equal to "1", the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXR_RXR.7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the received word is "0" with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **WAKE**: RXFF/TXFF pin wake-up UART function enable control
0: RXFF/TXFF pin wake-up UART function is disabled
1: RXFF/TXFF pin wake-up UART function is enabled

This bit is used to control the wake-up UART function when a falling edge on the RXFF/TXFF pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no RXFF/TXFF pin wake-up UART function if the UART clock (f_H) exists. If the WAKE bit is set high as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the RXFF/TXFF pin occurs. When this request happens and the corresponding interrupt is enabled, an RXFF/TXFF pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the RXFF/TXFF pin when the WAKE bit is cleared to 0.

Bit 2 **RIE**: Receiver interrupt enable control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled

This bit enables or disables the receiver interrupt. If this bit is equal to "1" and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.

Bit 1 **TIIE**: Transmitter Idle interrupt enable control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled

This bit enables or disables the transmitter idle interrupt. If this bit is equal to "1" and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

Bit 0 **TEIE**: Transmitter Empty interrupt enable control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled

This bit enables or disables the transmitter empty interrupt. If this bit is equal to "1" and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

• UCR3 Register

The UCR3 register is used to enable the UART Single Wire Mode communication and control the transmitter/receiver inverter. As the name suggests in the single wire mode the UART communication can be implemented in one single line, RXFF/TXFF, together with the control of the RXEN and TXEN bits in the UCR2 register.

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	TX_XOR	RX_XOR	SWM
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **TX_XOR**: Transmitter inverter control
0: Non-invert
1: Invert

Bit 1 **RX_XOR**: Receiver inverter control
0: Non-invert
1: Invert

Bit 0 **SWM**: Single Wire Mode enable control
0: Disable, the RXFF/TXFF pin is used as UART receiver function only
1: Enable, the RXFF/TXFF pin can be used as UART receiver or transmitter function controlled by the RXEN and TXEN bits

Note that when the Single Wire Mode is enabled, if both the RXEN and TXEN bits are high, the RXFF/TXFF pin will just be used as UART receiver input.

• TXR_RXR Register

The TXR_RXR register is the data register which is used to store the data to be transmitted on the TXFF pin or being received from the RXFF/TXFF pin.

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	x	x	x	x	x	x	x	x

"x": Unknown

Bit 7~0 **D7~D0**: UART transmit/receive data bit 7 ~ bit 0

• BRDH 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**: Baud rate divider high byte

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

Baud Rate=f_H/(BRD+UMOD/8)

BRD=16~65535 or 8~65535 depending on BRDS

Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.
2. The BRDL must be written first and then BRDH, otherwise errors may occur.
3. The BRDH register should not be modified during data transmission process.

• **BRDL 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**: Baud rate divider low byte

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

Baud Rate=f_H/(BRD+UMOD/8)

BRD=16~65535 or 8~65535 depending on BRDS

Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.
2. The BRDL must be written first and then BRDH, otherwise errors may occur.
3. The BRDL register should not be modified during data transmission process.

• **UFCR Register**

The UFCR register is the FIFO control register which is used for UART modulation control, BRD range selection and trigger level selection for RXIF and interrupt.

Bit	7	6	5	4	3	2	1	0
Name	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
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 5~3 **UMOD2~UMOD0**: UART Modulation Control bits

The modulation control bits are used to correct the baud rate of the received or transmitted UART signal. These bits determine if the extra UART clock cycle should be added in a UART bit time. The UMOD2~UMOD0 will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle.

Bit 2 **BRDS**: BRD range selection

0: BRD range is from 16 to 65535
1: BRD range is from 8 to 65535

The BRDS is used to control the sampling point in a UART bit time. If the BRDS is cleared to zero, the sampling point will be BRD/2, BRD/2+1×f_H, and BRD/2+2×f_H in a UART bit time. If the BRDS is set high, the sampling point will be BRD/2-1×f_H, BRD/2, and BRD/2+2×f_H in a UART bit time.

Note that the BRDS bit should not be modified during data transmission process.

Bit 1~0 **RxFTR1~RxFTR0**: Receiver FIFO trigger level (bytes)

00: 4 bytes in Receiver FIFO
01: 1 or more bytes in Receiver FIFO
10: 2 or more bytes in Receiver FIFO
11: 3 or more bytes in Receiver FIFO

For the receiver these bits define the number of received data bytes in the Receiver FIFO that will trigger the RXIF bit being set high, an interrupt will also be generated if the RIE bit is enabled. To prevent OERR from being set high, the receiver FIFO trigger level can be set to 2 bytes, avoiding an overrun state that cannot be processed by the program in time when more than 4 data bytes are received. After the reset the receiver FIFO is empty.

• **RxCNT Register**

The RxCNT register is the counter used to indicate the number of received data bytes in the Receiver FIFO which have not been read by the MCU. This register is read only.

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

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **D2~D0**: Receiver FIFO counter

The RxCNT register is the counter used to indicate the number of receiver data bytes in Receiver FIFO which is not read by MCU. When Receiver FIFO receives one byte data, the RxCNT will increase by one; when the MCU reads one byte data from Receiver FIFO, the RxCNT will decrease by one. If there are 4 bytes of data in the Receiver FIFO, the 5th data will be saved in the shift register. If there is 6th data, the 6th data will be saved in the shift register. But the RxCNT remains the value of 4. The RxCNT will be cleared when reset occurs or UARTEN=1. This register is read only.

Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRDH/BRDL register and the second is the UART modulation control bits, UMOD2~UMOD0. To prevent accumulated error of the receiver baud rate frequency, it is recommended to use two stop bits for resynchronization after each byte is received. If a baud rate BR is required with UART clock f_H.

f_H/BR=Integer Part+Fractional Part

The integer part is loaded into BRD (BRDH/BRDL). The fractional part is multiplied by 8 and rounded, then loaded into UMOD bit field as following:

BRD=TRUNC (f_H/BR)

UMOD=ROUND [MOD (f_H/BR)×8]

Therefore, the actual baud rate is as following:

Baud rate=$f_H/[BRD+(UMOD/8)]$

Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, determine the BRDH/BRDL register value, the actual baud rate and the error value for a desired baud rate of 230400.

From the above formula, the BRD=TRUNC (f_H/BR)=TRUNC(17.36111)=17

The UMOD=ROUND[MOD(f_H/BR)×8]=ROUND(0.36111×8)=ROUND(2.88888)=3

The actual Baud Rate=$f_H/[BRD+(UMOD/8)]$=230215.83

Therefore the error is equal to (230215.83-230400)/230400=-0.08%

Modulation Control Example

To get the best-fitting bit sequence for UART modulation control bits UMOD2~UMOD0, the following algorithm can be used: Firstly, the fractional part of the theoretical division factor is multiplied by 8. Then the product will be rounded and UMOD2~UMOD0 bits will be filled with the rounded value. The UMOD2~UMOD0 will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle. The following is an example using the fraction 0.36111 previously calculated: UMOD[2:0]=ROUND(0.36111×8)=011b.

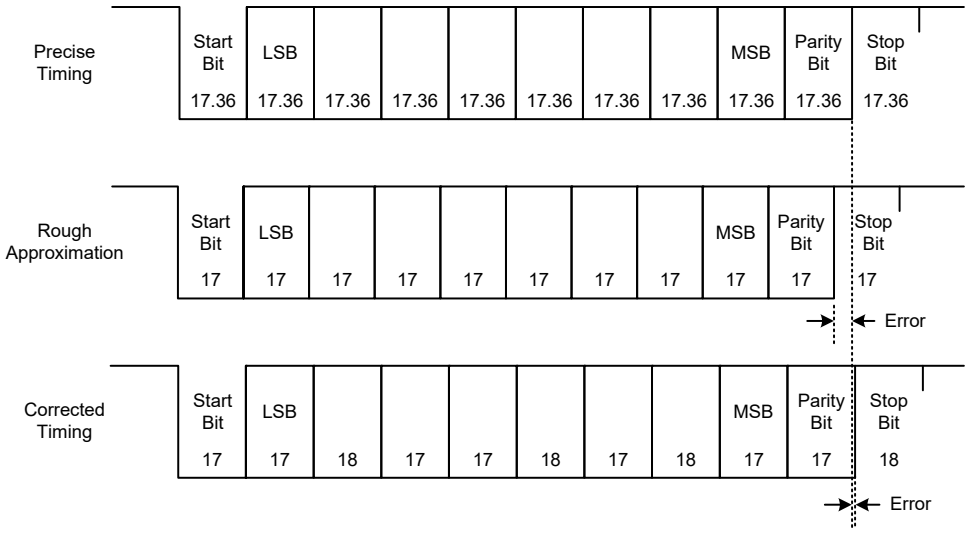
Fraction Addition	Carry to Bit 3	UART Bit Time Sequence	Extra UART Clock Cycle
0000b+0011b=0011b	No	Start bit	No
0011b+0011b=0110b	No	D0	No
0110b+0011b=1001b	Yes	D1	Yes
1001b+0011b=1100b	No	D2	No
1100b+0011b=1111b	No	D3	No
1111b+0011b=0010b	Yes	D4	Yes
0010b+0011b=0101b	No	D5	No
0101b+0011b=1000b	Yes	D6	Yes
1000b+0011b=1011b	No	D7	No
1011b+0011b=1110b	No	Parity bit	No
1110b+0011b=0001b	Yes	Stop bit	Yes

Baud Rate Correction Example

The following figure presents an example using a baud rate of 230400 generated with UART clock f_H. The data format for the following figure is: eight data bits, parity enabled, no address bit, two stop bits.

The following figure shows three different frames:

- The upper frame is the correct one, with a bit-length of 17.36 f_H cycles (4000000/230400=17.36).
- The middle frame uses a rough estimate, with 17 f_H cycles for the bit length.
- The lower frame shows a corrected frame using the best fit for the UART modulation control bits UMOD2~UMOD0.



UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd, mark, space or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits along with the parity are setup by programming the BNO, PRT1~PRT0 and PREN bits. The transmitter always uses two stop bits while the receiver uses one or two stop bits which is determined by the STOPS bit. The baud rate used to transmit and receive data is setup using the internal 16-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and

receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

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

Clearing the UARTEN bit will disable the TXFF and RXFF/TXFF pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF as well as register RxCNT being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

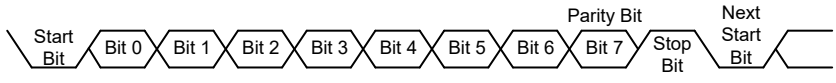
Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1 and UCR2 registers. The BNO bit controls the number of data bits which can be set to either 8 or 9, the PRT1~PRT0 bits control the choice of odd, even, mark or space parity, the PREN bit controls the parity on/off function and the STOPS bit decides whether one or two stop bits are to be used for the receiver, while the transmitter always uses two stop bits. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only configurable for the receiver. The transmitter uses two stop bits.

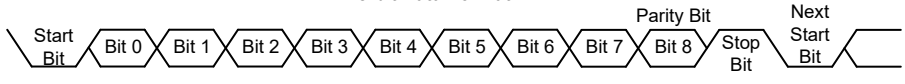
Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1 or 2
1	7	0	1	1 or 2
1	7	1	0	1 or 2
Example of 9-bit Data Formats				
1	9	0	0	1 or 2
1	8	0	1	1 or 2
1	8	1	0	1 or 2

Transmitter Receiver Data Format

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



8-bit Data Format



9-bit Data Format

UART Transmitter

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

Transmitting Data

When the UART is transmitting data, the data is shifted on the TXFF pin from the shift register, with the least significant bit first. In the transmit mode, the TXR_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8 bit in the UCR1 register.

The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT1~ PRT0 and PREN bits to define the required word length and parity type. Two stop bits are used for the transmitter.
- Setup the BRDH, BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.
- Set the TXEN bit to ensure that the TXFF pin is used as a UART transmitter pin.

Access the USR register and write the data that is to be transmitted into the TXR_RXR register. Note that this step will clear the TXIF bit.

This sequence of events can now be repeated to send additional data.

It should be noted that when TXIF=0, data will be inhibited from being written to the TXR_RXR register. Clearing the TXIF flag is always achieved using the following software sequence:

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

The read-only TXIF flag is set by the UART hardware and if set indicates that the TXR_RXR register is empty and that other data can now be written into the TXR_RXR register without overwriting the previous data. If the TEIE bit is set then the TXIF flag will generate an interrupt.

During a data transmission, a write instruction to the TXR_RXR register will place the data into the TXR_RXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set.

To clear the TIDLE bit the following software sequence is used:

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

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

Transmitting Break

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

UART Receiver

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

Receiving Data

When the UART receiver is receiving data, the data is serially shifted in on the external RXFF/TXFF input pin, LSB first. In the read mode, the TXR_RXR register forms a buffer between the internal bus and the receiver shift register. The TXR_RXR register is a four-byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR_RXR before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERR will be subsequently indicated. For continuous multi-byte data transmission, it is strongly recommended that the receiver uses two stop bits to avoid a receiving error caused by the accumulated error of the receiver baud rate frequency.

The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNO, PRT1~PRT0, PREN and STOPS bits to define the word length and parity type and number of stop bits.
- Setup the BRDH, BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.
- Set the RXEN bit to ensure that the RXFF/TXFF input pin is used as a UART receiver pin.

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

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

- The RXIF bit in the USR register will be set when the TXR_RXR register has data available. The number of the available data bytes can be checked by polling the RxCNT register content.
- When the contents of the shift register have been transferred to the TXR_RXR register and Receiver FIFO trigger level is reached, if the RIE bit is set, then an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

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

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

Receiving Break

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

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

Idle Status

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

Receiver Interrupt

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

When a subroutine will be called with an execution time longer than the time for UART to receive five data bytes, if the UART received data could not be read in time during the subroutine execution, clear the RXEN bit to zero in advance to suspend data reception. If the UART interrupt could not be served in time to process the overrun error during the subroutine execution, ensure that both EMI and RXEN bits are disabled during this period, and then enable EMI and RXEN again after the subroutine execution has been completed to continue the UART data reception.

Managing Receiver Errors

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

Overrun Error – OERR

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

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

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

When the OERR flag is set to "1", it is necessary to read five data bytes from the four-byte deep receiver FIFO and the shift register immediately to avoid unexpected errors, such as the UART is unable to receive data. If such an error occurs, clear the RXEN bit to "0" then set it to "1" again to continue data reception.

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

Noise Error – NF

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

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

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

Framing Error – FERR

The read only framing error flag, FERR, in the USR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, both stop bits must be high; otherwise the FERR flag will be set. The FERR flag and the received data will be recorded in the USR and TXR_RXR registers respectively, and the flag is cleared in any reset.

Parity Error – PERR

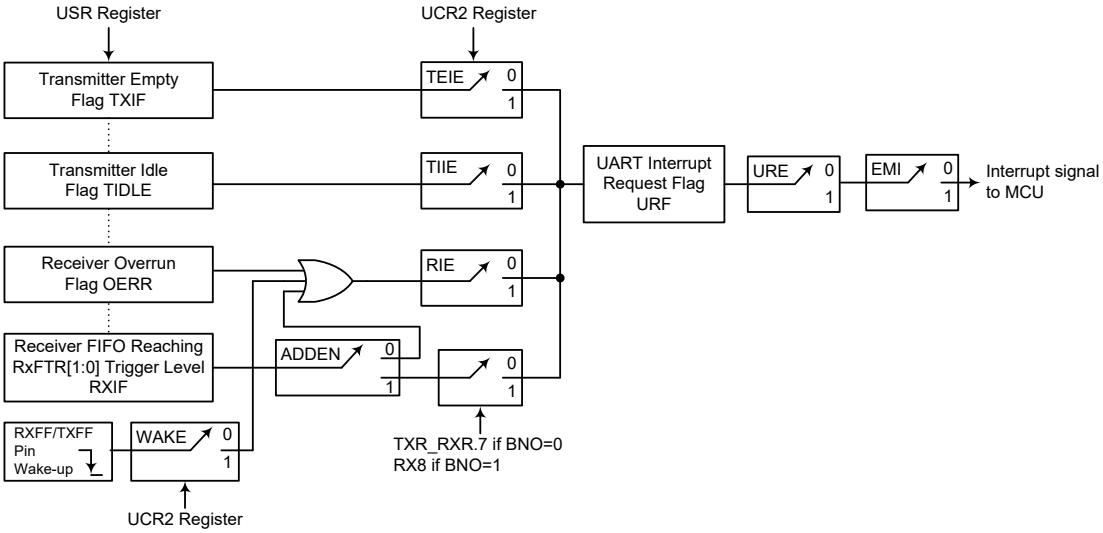
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd, even, mark or space, is selected. The read only PERR flag and the received data will be recorded in the USR and TXR_RXR registers respectively. It is cleared on any reset, it should be noted that the flags, FERR and PERR, in the USR register should first be read by the application program before reading the data word.

UART Interrupt Structure

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

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RXFF/TXFF pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the UART clock (f_H) source is switched off and the WAKE and RIE bits in the UCR2 register are set when a falling edge on the RXFF/TXFF pin occurs.

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



UART Interrupt Structure

Address Detect Mode

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the URE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to zero.

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

ADDEN Bit Function

UART Power Down and Wake-up

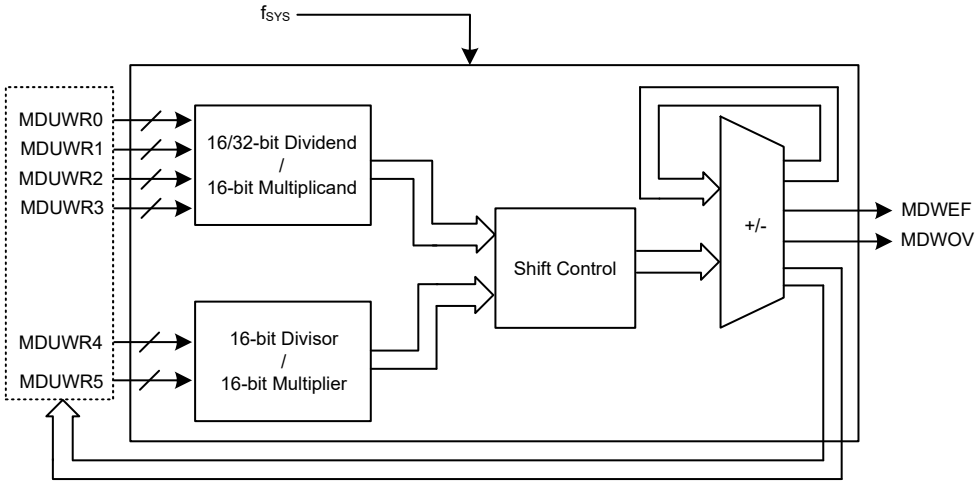
When the UART clock (f_H) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock (f_H) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the IDLE or SLEEP mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the USR, UCR1, UCR2, UCR3, UFCR, RxCNT, TXR_RXR, as well as the BRDH and BRDL registers will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RXFF/TXFF pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the UART clock (f_H) is off, then a falling edge on the RXFF/TXFF pin will trigger an RXFF/TXFF pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RXFF/TXFF pin will be ignored.

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

16-bit Multiplication Division Unit – MDU

The device has a 16-bit Multiplication Division Unit, MDU, which integrates a 16-bit unsigned multiplier and a 32-bit/16-bit unsigned divider. The MDU, in replacing the software multiplication and division operations, can therefore save large amounts of computing time as well as the Program and Data Memory space. It also reduces the overall microcontroller loading and results in the overall system performance improvements.



16-Bit MDU Block Diagram

MDU Registers

The multiplication and division operations are implemented in a specific way, a specific write access sequence of a series of MDU data registers. The status register, MDUWCTRL, provides the indications for the MDU operation. The data register each is used to store the data regarded as the different operand corresponding to different MDU operations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
MDUWR0	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR1	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR2	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR3	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR4	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR5	D7	D6	D5	D4	D3	D2	D1	D0
MDUWCTRL	MDWEF	MDWOV	—	—	—	—	—	—

MDU Register List

- **MDUWRn Register (n=0~5)**

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	x	x	x	x	x	x	x	x

"x": Unknown

Bit 7~0 **D7~D0**: 16-bit MDU data register n

• **MDUWCTRL Register**

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

Bit 7 **MDWEF**: 16-bit MDU error flag
0: Normal
1: Abnormal
This bit will be set high if the data register MDUWRn is written or read as the MDU operation is executing. This bit should be cleared to 0 by reading the MDUWCTRL register if it is equal to 1 and the MDU operation is completed.

Bit 6 **MDWOV**: 16-bit MDU overflow flag
0: No overflow occurs
1: Multiplication product > FFFFH or Divisor=0
When an operation is completed, this bit will be updated by hardware to a new value corresponding to the current operation situation.

Bit 5~0 Unimplemented, read as "0"

MDU Operation

For this MDU the multiplication or division operation is carried out in a specific way and is determined by the write access sequence of the six MDU data registers, MDUWR0~MDUWR5. The low byte data, regardless of the dividend, multiplicand, divisor or multiplier, must first be written into the corresponding MDU data register followed by the high byte data. All MDU operations will be executed after the MDUWR5 register is write-accessed together with the correct specific write access sequence of the MDUWRn. Note that it is not necessary to consecutively write data into the MDU data registers but must be in a correct write access sequence. Therefore, a non-write MDUWRn instruction or an interrupt, etc., can be inserted into the correct write access sequence without destroying the write operation. The relationship between the write access sequence and the MDU operation is shown in the following.

- 32-bit/16-bit division operation: Write data sequentially into the six MDU data registers from MDUWR0 to MDUWR5.
- 16-bit/16-bit division operation: Write data sequentially into the specific four MDU data registers in a sequence of MDUWR0, MDUWR1, MDUWR4 and MDUWR5 with no write access to MDUWR2 and MDUWR3.
- 16-bit ×16-bit multiplication operation: Write data sequentially into the specific four MDU data register in a sequence of MDUWR0, MDUWR4, MDUWR1 and MDUWR5 with no write access to MDUWR2 and MDUWR3.

After the specific write access sequence is determined, the MDU will start to perform the corresponding operation. The calculation time necessary for these MDU operations are different. During the calculation time any read/write access to the six MDU data registers is forbidden. After the completion of each operation, it is necessary to check the operation status in the MDUWCTRL register to make sure that whether the operation is correct or not. Then the operation result can be read out from the corresponding MDU data registers in a specific read access sequence if the operation is correctly finished. The necessary calculation time for different MDU operations is listed in the following.

- 32-bit/16-bit division operation: $17\times t_{SYS}$
- 16-bit/16-bit division operation: $9\times t_{SYS}$
- 16-bit×16-bit multiplication operation: $11\times t_{SYS}$

The operation results will be stored in the corresponding MDU data registers and should be read out from the MDU data registers in a specific read access sequence after the operation is completed. Noe that it is not necessary to consecutively read data out from the MDU data registers but must be in a correct read access sequence. Therefore, a non-read MDUWRn instruction or an interrupt, etc., can be inserted into the correct read access sequence without destroying the read operation. The relationship between the operation result read access sequence and the MDU operation is shown in the following.

- 32-bit/16-bit division operation: Read the quotient from MDUWR0 to MDUWR3 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit/16-bit division operation: Read the quotient from MDUWR0 and MDUWR1 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit×16-bit multiplication operation: Read the product sequentially from MDUWR0 to MDUWR3.

The overall important points for the MDU read/write access sequence and calculation time are summarized in the following table. Note that the device should not enter the IDLE or SLEEP mode until the MDU operation is totally completed, otherwise the MDU operation will fail.

Operations / Items	32-bit / 16-bit Division	16-bit / 16-bit Division	16-bit × 16-bit Multiplication
Write Sequence First write ↓ ↓ ↓ ↓ Last write	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Dividend Byte 2 written to MDUWR2 Dividend Byte 3 written to MDUWR3 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Multiplicand Byte 0 written to MDUWR0 Multiplier Byte 0 written to MDUWR4 Multiplicand Byte 1 written to MDUWR1 Multiplier Byte 1 written to MDUWR5
Calculation Time	17×t_{SYS}	9×t_{SYS}	11×t_{SYS}
Read Sequence First read ↓ ↓ ↓ ↓ Last read	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Quotient Byte 2 read from MDUWR2 Quotient Byte 3 read from MDUWR3 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Product Byte 0 written to MDUWR0 Product Byte 1 written to MDUWR1 Product Byte 2 written to MDUWR2 Product Byte 3 written to MDUWR3

MDU Operations Summary

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 one of three fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} voltage is above the preset low voltage value. The TLVD1~TLVD0 bits are used to select the minimum low voltage width to interrupt, t_{LVD}. 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	TLVD1	TLVD0	LVDO	LVDEN	—	VLVD2	VLVD1	VLVD0
R/W	R/W	R/W	R	R/W	—	R/W	R/W	R/W
POR	0	0	0	0	—	0	0	0

Bit 7~6 **TLVD1~TLVD0**: Minimum low voltage width to interrupt (t_{LVD}) selection
00: (1~2)×t_{LIRC}
01: (3~4)×t_{LIRC}
10: (7~8)×t_{LIRC}
11: (1~2)×t_{LIRC}

Bit 5 **LVDO**: LVD output flag
0: No Low Voltage Detected
1: Low Voltage Detected

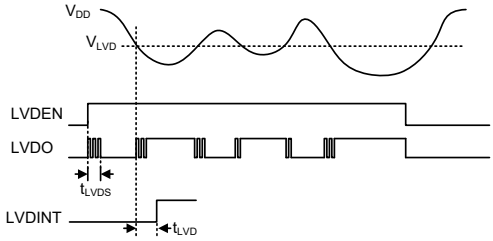
Bit 4 **LVDEN**: Low voltage detector enable control
0: Disable
1: Enable

Bit 3 Unimplemented, read as "0"

Bit 2~0 **VLVD2~VLVD0**: LVD voltage selection
000: Reserved
001: Reserved
010: Reserved
011: Reserved
100: Reserved
101: 3.3V
110: 3.6V
111: 4.0V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range ofbetween 3.3V~4.0V. 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 mode, the low voltage detector will be disabled even if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



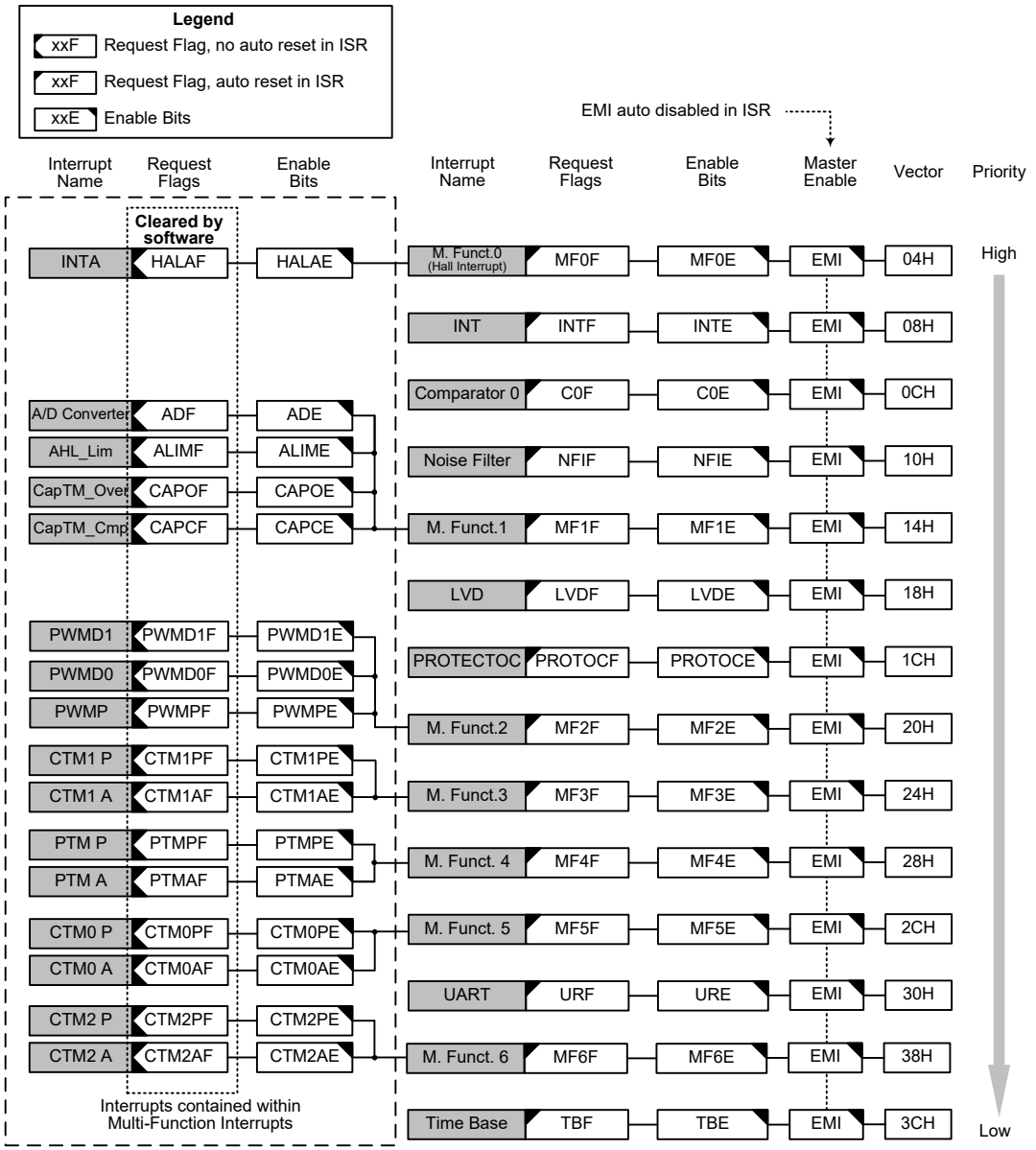
LVD Operation

The Low Voltage Detector also has its own interrupt, 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, i.e., V_{DD} falls below the preset LVD voltage. In this case, the LVDF interrupt request flag will be set, causing an interrupt to be generated. This will cause the device to wake up from the IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVDF flag should be first set high before the device enters the 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, NFIN and INT pins, while the internal interrupts are generated by various internal functions including the TMs, the Comparator 0, 16-bit CAPTM, Time Base, PWM, LVD, UART, A/D converter and PROTECTOC.

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.



Interrupt Structure

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 HINTEG, INTEG and NF_VIL 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 interrupt	HALAE	HALAF	Hall noise filter output
External interrupt	INTE	INTF	—
Comparator 0	C0E	C0F	Int_Is
Digital Noise Filter input	NFIE	NFIF	—
Low Voltage Detector	LVDE	LVDF	—
Protectoc	PROTOCE	PROTOCF	—
UART Interface	URE	URF	—
Time Base	TBE	TBF	—
Multi-function Interrupt	MFnE	MFnF	n=0~6
A/D Converter	ADE	ADF	—
	ALIME	ALIMF	Int_AHL_Lim
16-bit Capture Timer Module	CAPOE	CAPOF	—
	CAPCE	CAPCF	—
PWM function	PWMDnE	PWMDnF	n=0~1
	PWMPE	PWMPF	—
Timer Modules	CTMnPE	CTMnPF	n=0~2
	CTMnAE	CTMnAF	
	PTMPE	PTMPF	—
	PTMAE	PTMAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
HINTEG	—	HSEL	—	—	—	—	INTAS1	INTAS0
INTEG	—	—	—	—	—	—	INTS1	INTS0
NF_VIL	NFIS1	NFIS0	—	—	—	—	—	—
INTC0	—	C0F	INTF	MF0F	C0E	INTE	MF0E	EMI
INTC1	PROTOCF	LVDF	MF1F	NFIF	PROTOCE	LVDE	MF1E	NFIE
INTC2	MF5F	MF4F	MF3F	MF2F	MF5E	MF4E	MF3E	MF2E
INTC3	TBF	MF6F	—	URF	TBE	MF6E	—	URE
MFI0	—	—	—	HALAF	—	—	—	HALAE
MFI1	CAPCF	CAPOF	ALIMF	ADF	CAPCE	CAPOE	ALIME	ADE
MFI2	PWMPF	—	PWMD1F	PWMD0F	PWMPE	—	PWMD1E	PWMD0E
MFI3	—	—	CTM1AF	CTM1PF	—	—	CTM1AE	CTM1PE

Register Name	Bit							
	7	6	5	4	3	2	1	0
MFI4	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE
MFI5	—	—	CTM0AF	CTM0PF	—	—	CTM0AE	CTM0PE
MFI6	—		CTM2AF	CTM2PF	—		CTM2AE	CTM2PE

Interrupt Register List

• HINTEG Register

Bit	7	6	5	4	3	2	1	0
Name	—	HSEL	—	—	—	—	INTAS1	INTAS0
R/W	—	R/W	—	—	—	—	R/W	R/W
POR	—	0	—	—	—	—	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **HSEL**: HA source selection
Described elsewhere.

Bit 5~2 Unimplemented, read as "0"

Bit 1~0 **INTAS1~INTAS0**: FHA trigger edge selection for INTA interrupt
00: Disable
01: Rising edge
10: Falling edge
11: Dual edge

• INTEG Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **INTS1~INTS0**: INT interrupt trigger edge selection
00: Disable
01: Rising edge
10: Falling edge
11: Dual edge

• NF_VIL Register

Bit	7	6	5	4	3	2	1	0
Name	NFIS1	NFIS0	—	—	—	—	—	—
R/W	R/W	R/W	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

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

Bit 5~0 Unimplemented, read as "0"

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	C0F	INTF	MF0F	C0E	INTE	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 **INTF**: External interrupt request flag
0: No request
1: Interrupt request

Bit 4 **MF0F**: 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 **INTE**: External interrupt interrupt control
0: Disable
1: Enable

Bit 1 **MF0E**: 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	PROTOCF	LVDF	MF1F	NFIF	PROTOCE	LVDE	MF1E	NFIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PROTOCF**: PROTECTOC interrupt request flag
0: No request
1: Interrupt request

Bit 6 **LVDF**: LVD interrupt request flag
0: No request
1: Interrupt request

Bit 5 **MF1F**: Multi-function interrupt 1 request flag
0: No request
1: Interrupt request

Bit 4 **NFIF**: Digital Noise Filter NFIN input interrupt request flag
0: No request
1: Interrupt request

Bit 3 **PROTOCE**: PROTECTOC interrupt control
0: Disable
1: Enable

Bit 2 **LVDE**: LVD interrupt control
0: Disable
1: Enable

Bit 1 **MF1E**: Multi-function interrupt 1 control
0: Disable
1: Enable

Bit 0 **NFIE**: Digital 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	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **MF5F**: Multi-function interrupt 5 request flag
0: No request
1: Interrupt request

Bit 6 **MF4F**: Multi-function interrupt 4 request flag
0: No request
1: Interrupt request

Bit 5 **MF3F**: Multi-function interrupt 3 request flag
0: No request
1: Interrupt request

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

Bit 3 **MF5E**: Multi-function interrupt 5 control
0: Disable
1: Enable

Bit 2 **MF4E**: Multi-function interrupt 4 control
0: Disable
1: Enable

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

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

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	TBF	MF6F	—	URF	TBE	MF6E	—	URE
R/W	R/W	R/W	—	R/W	R/W	R/W	—	R/W
POR	0	0	—	0	0	0	—	0

Bit 7 **TBF**: Time Base interrupt request flag
0: No request
1: Interrupt request

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

Bit 5 Unimplemented, read as “0”

Bit 4 **URF**: UART interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TBE**: Time Base interrupt control
0: Disable
1: Enable

Bit 2 **MF6E**: Multi-function interrupt 6 control
0: Disable
1: Enable

Bit 1 Unimplemented, read as "0"

Bit 0 **URE**: UART interrupt control
0: Disable
1: Enable

• **MFI0 Register**

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

Bit 7~5 Unimplemented, read as "0"

Bit 4 **HALAF**: Hall sensor A interrupt (INTA) request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~1 Unimplemented, read as "0"

Bit 0 **HALAE**: Hall sensor A interrupt (INTA) control
0: Disable
1: Enable

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	CAPCF	CAPOF	ALIMF	ADF	CAPCE	CAPOE	ALIME	ADE
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
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 6 **CAPOF**: CAPTM capture overflow interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 5 **ALIMF**: A/D conversion comparison result interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **ADF**: A/D conversion end interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

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 comparison result interrupt control
0: Disable
1: Enable

Bit 0 **ADE**: A/D conversion end interrupt control
0: Disable
1: Enable

• **MFI2 Register**

Bit	7	6	5	4	3	2	1	0
Name	PWMPF	—	PWMD1F	PWMD0F	PWMPE	—	PWMD1E	PWMD0E
R/W	R/W	—	R/W	R/W	R/W	—	R/W	R/W
POR	0	—	0	0	0	—	0	0

Bit 7 **PWMPF**: PWM period match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 6 Unimplemented, read as "0"

Bit 5 **PWMD1F**: PWM1 duty match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **PWMD0F**: PWM0 duty match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3 **PWMPE**: PWM period match interrupt control
0: Disable
1: Enable

Bit 2 Unimplemented, read as "0"

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	—	—	CTM1AF	CTM1PF	—	—	CTM1AE	CTM1PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **CTM1AF**: CTM1 comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **CTM1PF**: CTM1 comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **CTM1AE**: CTM1 comparator A match interrupt control
0: Disable
1: Enable
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 0 **CTM1PE**: CTM1 comparator P match interrupt control
0: Disable
1: Enable

• **MFI4 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PTMAF**: PTM comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **PTMPF**: PTM comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTMAE**: PTM comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTMPE**: PTM comparator P match interrupt control
0: Disable
1: Enable

• **MFI5 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CTM0AF	CTM0PF	—	—	CTM0AE	CTM0PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **CTM0AF**: CTM0 comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **CTM0PF**: CTM0 comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **CTM0AE**: CTM0 comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **CTM0PE**: CTM0 comparator P match interrupt control
0: Disable
1: Enable

• **MFI6 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CTM2AF	CTM2PF	—	—	CTM2AE	CTM2PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **CTM2AF**: CTM2 comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **CTM2PF**: CTM2 comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **CTM2AE**: CTM2 comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **CTM2PE**: CTM2 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 an "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.

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 interrupt structure 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 Interrupt

The Hall sensor interrupt is contained in Multi-function interrupt 0. It is controlled by the signal transistion on the Hall sensor noise filter output signal, FHA. A Hall sensor interrupt request will take place when the Hall sensor interrupt request flag, HALAF, is set, which will occur when a transition appears on the Hall sensor noise filter output. The correct interrupt edge type must be selected using the HINTEG register.

To allow the program to branch to the corresponding interrupt vector address, the global interrupt enable bit, EMI, the Multi-function interrupt enable bit, MF0E, and the Hall Sensor interrupt enable bit, HALAE, must first be set. If the Hall noise filter input is selected to come from the external input pin, H1, which is pin-shared with the I/O pin, they should be configured as the external Hall sensor pin by the corresponding pin-shared function selection bits before the Hall sensor 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 Hall sensor noise filter output signal, 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 Multi-function interrupt request flag, MF0F, will be automatically reset, but the Hall sensor interrupt request flag, HALAF, must be manually cleared to zero by the application program.

The HINTEG 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 HINTEG register also can be used to disable the Hall sensor input interrupt function.

External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, is set, which will occur when a transition appears on the external interrupt pin. The correct interrupt edge type must be selected using the INTEG register.

To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the external interrupt enable bit, INTE, must first be set. As the external interrupt pin is pin-shared with I/O pin, it should be configured as external interrupt pin by the corresponding pin-shared function selection bits 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, INTF, 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 INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

Comparator 0 Interrupt

The comparator 0 interrupt (Int_Is) 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 a transition appears on the comparator 0 output. The correct interrupt edge type must be selected using the CMP0_EG1~CMP0_EG0 bits in the OPOMS register To allow the program to branch to its 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 output generates a preset edge 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.

Digital Noise Filter Interrupt

The digital noise filter interrupt is controlled by signal transitions on the NFIN pin. A digital noise filter interrupt request will take place when the digital noise filter input interrupt request flag, NFIF, is set, which will occur when a transition appears on the NFIN pin. The correct interrupt edge type must be selected using the NFIS1 and NFIS0 bits in the NF_VIL register.

To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the NFIN input interrupt enable bit, NFIE, must first be set. As the digital noise filter interrupt pin NFIN is pin-shared with I/O pin, it should be configured as the digital noise filter input pin by the corresponding pin-shared function selection bits before the digital noise filter 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 NFIN pin, a subroutine call to the digital noise filter interrupt vector will take place. When the interrupt is serviced, the digital noise filter 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 digital noise filter interrupt. A choice of either rising or falling or both edge types can be chosen to trigger a digital noise filter interrupt. Note that the NF_VIL register can also be used to disable the digital noise filter interrupt function.

LVD Interrupt

An LVD Interrupt request will take place when the LVD Interrupt request flag, LVDF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and Low Voltage Interrupt enable bit, LVDE, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the LVD Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the interrupt request flag, LVDF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

PROTECTOC Interrrupt

The PROTECTOC interrupt is controlled by the motor protection circuit. A PROTECTOC interrupt request will take place when the PROTECTOC interrupt request flag, PROTOCF, is set, a situation that will occur when a PROTECTOC signal is generated. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and PROTECTOC interrupt enable bit, PROTOCE, must first be set. When the interrupt is enabled, the stack is not full and a PROTECTOC signal is generated, a subroutine call to the PROTECTOC interrupt vector, will take place. When the interrupt is serviced, the PROTECTOC interrupt request flag, PROTOCF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

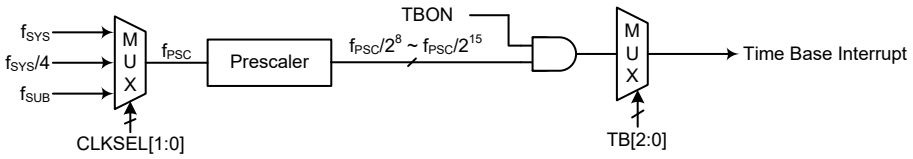
UART Interrupt

Several individual UART conditions can generate a UART Interrupt. When any one of these conditions exists, low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RXFF/TXFF pin wake-up. To allow the program to branch to its interrupt vector addresses, the global interrupt enable bit, EMI, and UART Interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of these conditions are created, a subroutine call to the UART Interrupt vector will take place. When the interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART section.

Time Base Interrupt

The function of the Time Base Interrupt is to provide regular time signals in the form of an internal interrupt. It is controlled by the overflow signals from its timer functions. When this happen its interrupt request flag, TBF will be set. To allow the program to branch to its interrupt vector addresses, 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. The clock source, f_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock sourcethat generates f_{PSC}, which in turn controls the Time Base interrupt period, is selected using the CLKSEL1~CLKSEL0 bits in the PSCR register.



Time Base Interrupt

• PSCR Register

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

Bit 7~2 Unimplemented, read as "0"

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

• TBC Register

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

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

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB2~TB0**: Time Base time-out period selection
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

Multi-function Interrupts

Within the device there are several 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 interrupt, PWM period and duty match interrupts, TM interrupts, A/D converter interrupts and CAPTM interrupts.

A Multi-function interrupt request will take place when the Multi-function interrupt request flag, MFnF, is set, which will occur when any of its included functions generate an interrupt request flag. To allow the program to branch to the corresponding interrupt vector address, the global interrupt enable bit, EMI, the Multi-function interrupt enable bit, MFnE, and the enable bits of the interrupts which are included in the Multi-function interrupt, and 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 the Multi-function interrupt occurs, a subroutine call to the corresponding Multi-function interrupt vector 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.

A/D Converter Interrupts

The A/D Converter has two interrupts which both are contained in Multi-function interrupt 1. One of the A/D Converter interrupts is controlled by the end of an A/D conversion process. The other is the A/D converter comparison result interrupt (Int_AHL_Lim) which is controlled by the comparison of the converted data registers (SADOH/SADOL) with the boundary register pairs (ADHVDH/ADHVDL and ADLVDH/ADLVDL). The comparison result interrupt trigger condition is set by ADCHVE and ADCLVE bits in the ADCR2 register. An A/D converter interrupt request will take place when the A/D converter interrupt request flag, ADF or ALIMF, is set, which occurs when the A/D conversion process finishes or a comparison result interrupt trigger condition occurs.

To allow the program to branch to the corresponding interrupt vector address, the global interrupt enable bit, EMI, the Multi-function interrupt enable bit, MF1E, and the A/D converter interrupt enable bit, ADE or ALIME, must first be set. When the interrupt is enabled, the stack is not full and an A/D conversion comparison result interrupt trigger condition occurs or an end of A/D conversion process occurs, a subroutine call to the corresponding Multi-function interrupt vector will take place. When the A/D converter interrpt is serviced, the Multi-function interrupt request flag, MF1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. However the ADF or ALIMF flag will not be automatically cleared to zero, they must be manually cleared by the application program.

Capture Timer Module Interrupts

The CAPTM has two interrupts, known as CapTM_Over and CapTM_Cmp, which are contained within the Multi-function Interrupt 1. 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 the corresponding interrupt vector address, the global interrupt enable bit, EMI, the Multi-function interrupt enable bit, MF1E, and the CAPTM interrupt enable bit, CAPOE or CAPCE, must first be set. When the interrupt is enabled, the stack is not full and the CAPTM capture overflows or compare matches, a subroutine call to the corresponding Multi-function interrupt vector will take

place. When the CAPTM interrupt is serviced, the Multi-function interrupt request flag, MF1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. As the CAPOF or CAPCF flag will not be automatically cleared to zero, they must be manually cleared by the application program.

PWM Module Interrupts

The PWM module has three interrupts, one PWM Period match interrupt known as PWMP interrupt and two PWM Duty match interrupts known as PWMDn(n=0~1) 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, is set, which occurs when the PWM Period or Duty n matches. To allow the program to branch to the corresponding interrupt vector address, the global intgerrupt enable bit, EMI, the PWM Period or Duty match interrupt enable bit, PWMPE or PWMDnE, and the Multi-function interrupt enable bit, MF2E, 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 the corresponding Multi-function interrupt vector will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the Multi-function interrupt request flag, MF2F, will also be automatically reset, but the interrupt request flag, PWMPF or PWMDnF, must be manually cleared by the application program.

TM Interrupts

The Compact and 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 3~6. For all of the TM types there are two interrupt request flags, xTMnPF and xTMnAF, and two enable control bits, xTMnPE and xTMnAE. 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 their respective interrupt vector addresses, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant multi-function interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the respectivel Multi-function interrupt vector 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, MFnF, 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.

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 an 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.

Voltage Regulator

The device includes a fully integrated 5V LDO regulator to provide power source for the device internal circuitry and external circuits, with a output current over 50mA. The LDO will act as a fully turned on switch when the power supply V_{CC} is less than 5V, in which condition its output voltage is almost equal to the power supply if there is no load.

Gate-driver Circuit

The device includes a 2-channel gate-driver, which can be used for external high-side and low-side N-channel MOSFET driving. It includes a 12V LDO, 2-channel high-side and low-side gate-driver circuits. The gate-driver also has five protection functions, which are Power Supply Input Under Voltage Lock-Out, 5V LDO Output Under Voltage Lock-Out, 12V LDO Output Under Voltage Lock-Out, Bootstrap Output Under Voltage Lock-Out and Over Temperature Protection, to avoid abnormal output situations.

The input signals of the INHx, $\overline{\text{INLx}}$ and EN are input to the control logic which will determine the high-side and low-side gate-driver outputs. The INHx and EN each have an internal pull-down resistor and the $\overline{\text{INLx}}$ has an internal pull-up resistor. Additionally, there is a fixed dead time insertion when switching between the high-side and low-side gate driving to avoid short-circuit between V_{CC} and ground.

The gate-driver output voltage will vary with the power supply when V_{CC} is less than 13V. When V_{CC} is greater than 13V, the gate-driver output will be clamped to 12V, providing a 0.7A peak source current and a 1A peak sink current. Either high-side and low-side gate has an internal hold-off resistor in order to avoid misconduction of external power MOSFET due to interference when the power is off.

The gate-driver also has integrated bootstrap diodes for bootstrap circuit implementation, allowing reduced system component requirements.

12V Voltage Regulator

The integrated 12V LDO, which supplies power for the low-side gate-drivers, cannot be used as power supply for other circuits.

Bootstrap Circuit Operation

The gate-driver uses 2 sets of bootstrap circuits as floating power supplies to power the high-side gate-driver circuits.

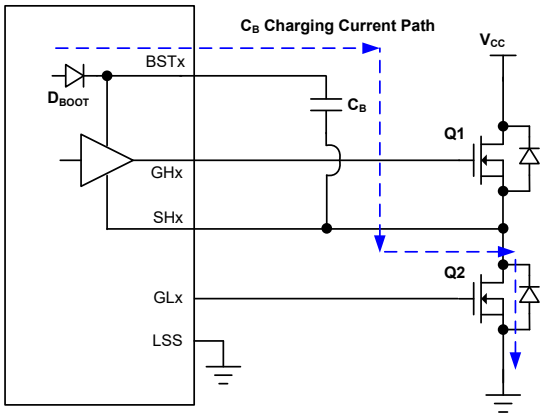
Each set of bootstrap circuit is composed of an external bootstrap capacitor, C_B, and an internal bootstrap diode, D_{BOOT}. The charging current path of the bootstrap capacitor in common applications is shown by the blue dotted line in the figure below. The bootstrap capacitor is charged after the low-side power MOSFET is turned on. After the gate-driver is enabled, an input command of INHx=$\overline{\text{INLx}}$='L' should be arranged before switching to the high-side power MOSFET for the first time, so that the low-side power MOSFET will be turned on for a period of time to charge the bootstrap capacitor. As shown in the Bootstrap Capacitor Charging Time figure, the high-side gate-driver output could not be controlled by inputs until the bootstrap capacitor has been charged exceeding the bootstrap under voltage lock-out threshold, V_{BST_UVLO+}. It is recommended to charge the bootstrap capacitor to the steady-state voltage of V1 before proceeding. The equation for estimating the charging time t_{BST} of the bootstrap capacitor is as follows:

$$t_{BST}\,(\text{ms}) > 0.3 + 1.1 \times C_B\,(\mu\text{F}) \div 2.2$$

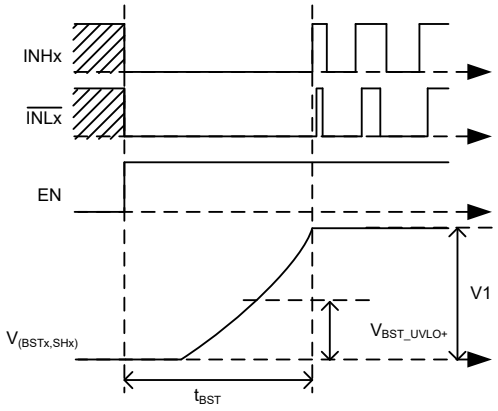
Where C_B is the bootstrap capacitance. The larger the capacitance, the longer it will take to charge. For example, the charging time t_{BST} should be at least 1.5ms for a capacitance of 2.2μF. After the charging is completed, the bootstrap voltage will reach the steady-state voltage V1, as shown in the figure. When the power supply V_{CC} is less than or equal to 13V, V1 will change along with V_{CC}. Then V1 will be clamped to a fixed value of 12V once V_{CC} is larger than 13V. V1 is calculated as follows:

V1=12V when V_{CC}>13V

V1=V_{CC}-1.5V when $V_{CC}\leq$13V

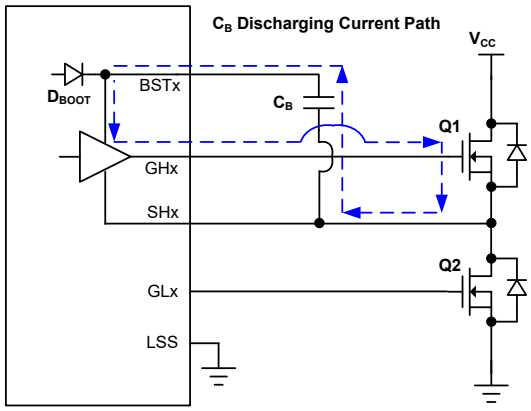


Bootstrap Capacitor (C_B) Charging Current Path



Bootstrap Capacitor Charging Time (t_{BST})

The charge stored in the bootstrap capacitor, C_B, is discharged during the high-side gate-driver output and the internal bootstrap diode, D_{BOOT}, is used to avoid current backflow, as shown in the following figure. When discharging, pay attention to whether the bootstrap capacitance value is sufficient. If the bootstrap capacitance value is too small, it will affect the high-side gate driving capability. Generally speaking, the bootstrap capacitance value is recommended to be more than 50 times the input power capacitance value of the high-side power MOSFET, and is recommended to be at least 2.2μF.



Bootstrap Capacitor (C_B) Discharging Current Path

Gate-driver Control Logic

As a gate-driver for driving high-side and low-side N-channel MOSFETs, the control signals EN, INHx, $\overline{INLx}$ are internally controlled by PC4, GxT and GxB on PC0~PC3. The dead time width is determined by the control signals but has a minimum value equal to the fixed dead time designed in the gate-driver.

Both high-side and low-side gate-driver outputs are controlled by the EN, INHx and $\overline{INLx}$ input signals. For example, the on/off true table of the external N-channel power MOSFETs is shown as follows.

EN	INHx	$\overline{INLx}$	GHx-to-SHx	GLx-to-LSS	External High-Side Power MOSFET	External Low-Side Power MOSFET
0	X	X	L	L	OFF	OFF
1	0	0	L	H	OFF	ON
1	0	1	L	L	OFF	OFF
1	1	0	L	L	OFF	OFF
1	1	1	H	L	ON	OFF

X: No care; L: Low; H: High

Operation Truth Table

Protection Function Operation

When the gate-driver operates in an abnormal situation, such as a power supply input under voltage lock-out, bootstrap output under voltage lock-out, 12V LDO output under voltage lock-out, 5V LDO output under voltage lock-out or over temperature condition has occurred, it will activate the corresponding protection mechanism to turn off the affected N-channel power MOSFET. The protection mechanisms are summarized below.

Protection	Protection Entry Condition	Protection Reaction				Release Condition
		V_{12P}	GHx-to-SHx	GLx-to-LSS	Bootstrap Function	
VCC_UVLO	$V_{CC}<V_{CC\_UVLO-}$	0V	L	L	Disable	$V_{CC}\geq V_{CC\_UVLO+}$
VBST_UVLO	$V_{(BSTx,SHx)}<V_{BST\_UVLO-}$	—	L	—	Keep Active	$V_{(BSTx,SHx)}\geq V_{BST\_UVLO+}$
V12P_UVLO	$V_{12P}<V_{12P\_UVLO-}$	—	—	L	Disable	$V_{12P}\geq V_{12P\_UVLO+}$
VREG_UVLO	$V_{REG}<V_{REG\_UVLO-}$	—	L	L	Disable	$V_{REG}\geq V_{REG\_UVLO+}$
OTP	$T_j>T_{SHD}$	—	L	L	Disable	$T_j\leq T_{REC}$

Protection Function Conditions

Power Supply Input Under Voltage Lock-Out – VCC_UVLO

This integrated protection function is to avoid unstable gate-driver output when the power supply voltage falls to a certain low level. During the power-on period, both high-side and low-side power MOSFETs are turned off before the power supply voltage reaching the threshold V_{CC_UVLO+}. When the power supply voltage is greater than V_{CC_UVLO+}, the gate-driver outputs are determined by the input signals. If the power supply voltage falls below the under voltage lock-out threshold V_{CC_UVLO-}, both high and low-side power MOSFETs will remain off.

Bootstrap Output Under Voltage Lock-Out – VBST_UVLO

This integrated protection function is to avoid that when the bootstrap capacitor is insufficiently charged, the output voltage of the high-side gate-driver will be insufficient making the high-side power MOSFET fully turned on. When the bootstrap output voltage is larger than the threshold V_{BST_UVLO+}, the high-side gate-driver output is determined by the input signals. If the bootstrap output voltage falls below the under voltage lock-out threshold V_{BST_UVLO-}, the high-side power MOSFET will remain off.

12V LDO Output Under Voltage Lock-Out – V12P_UVLO

When the internal 12V LDO output voltage, V_{12P}, is too low, the integrated 12V LDO output under voltage lock-out function will be activated to avoid that the output voltage of the low-side gate-driver is insufficient making the low-side power MOSFET fully turned on. After V_{12P} exceeds the threshold V_{12P_UVLO+}, the low-side gate-driver output is determined by the input signals. If V_{12P} is less than the under voltage lock-out threshold V_{12P_UVLO-}, the low-side power MOSFET will remain off.

5V LDO Output Under Voltage Lock-Out – VREG_UVLO

When the internal 5V LDO output voltage, V_{REG}, is too low, the integrated 5V LDO output under voltage lock-out function will be activated to avoid unstable input signals. After V_{REG} exceeds the threshold V_{REG_UVLO+}, the gate-driver output is determined by the input signals. If V_{REG} is less than the under voltage lock-out threshold V_{REG_UVLO-}, both high and low-side power MOSFETs will remain off.

Over Temperature Protection – OTP

If the internal junction temperature of the gate-driver exceeds the limit threshold T_{SHD}, the high-side and low-side power MOSFETs will be turned off until the junction temperature drops below the recovery temperature level, T_{REC}, at which the gate-driver output is determined by the

input signals. An over temperature protection being triggered means the overall power dissipation P_D of the gate-driver has exceeded the maximum allowable power dissipation, $P_{D(MAX)}$. For related content and calculations, refer to the "Thermal Consideration" chapter.

Thermal Consideration

The maximum power dissipation depends upon the thermal resistance of the IC package, PCB layout, rate of surrounding airflow and difference between the junction and ambient temperature. The maximum power dissipation can be calculated by the following formula:

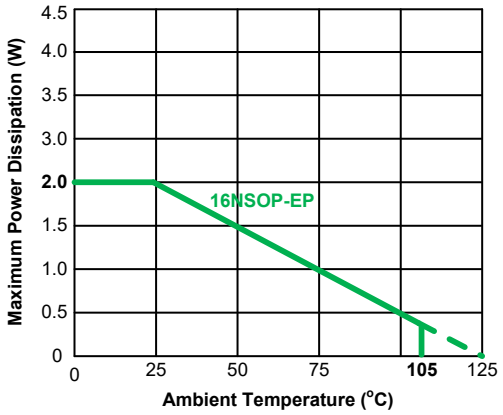
$$P_{D(MAX)} = (T_{J(MAX)} – T_A)/\theta_{JA} \quad (W)$$

where $T_{J(MAX)}$ is the maximum junction temperature, T_A is the ambient temperature and θ_{JA} is the junction-to-ambient thermal resistance of MCU package.

For maximum operating rating conditions, the maximum junction temperature is 125˚C. However, it's recommended that the maximum junction temperature does not exceed 125˚C during normal operation to maintain high reliability. The de-rating curve of the maximum power dissipation is show below:

$$P_{D(MAX)} = (125˚C – 25˚C)/(50˚C/W) = 2W \quad (24QFN)$$

For a fixed $T_{J(MAX)}$ of 125˚C, the maximum power dissipation depends upon the operating ambient temperature and the package's thermal resistance, θ_{JA}. The de-rating curve below shows the effect of rising ambient temperature on the maximum recommended power dissipation.



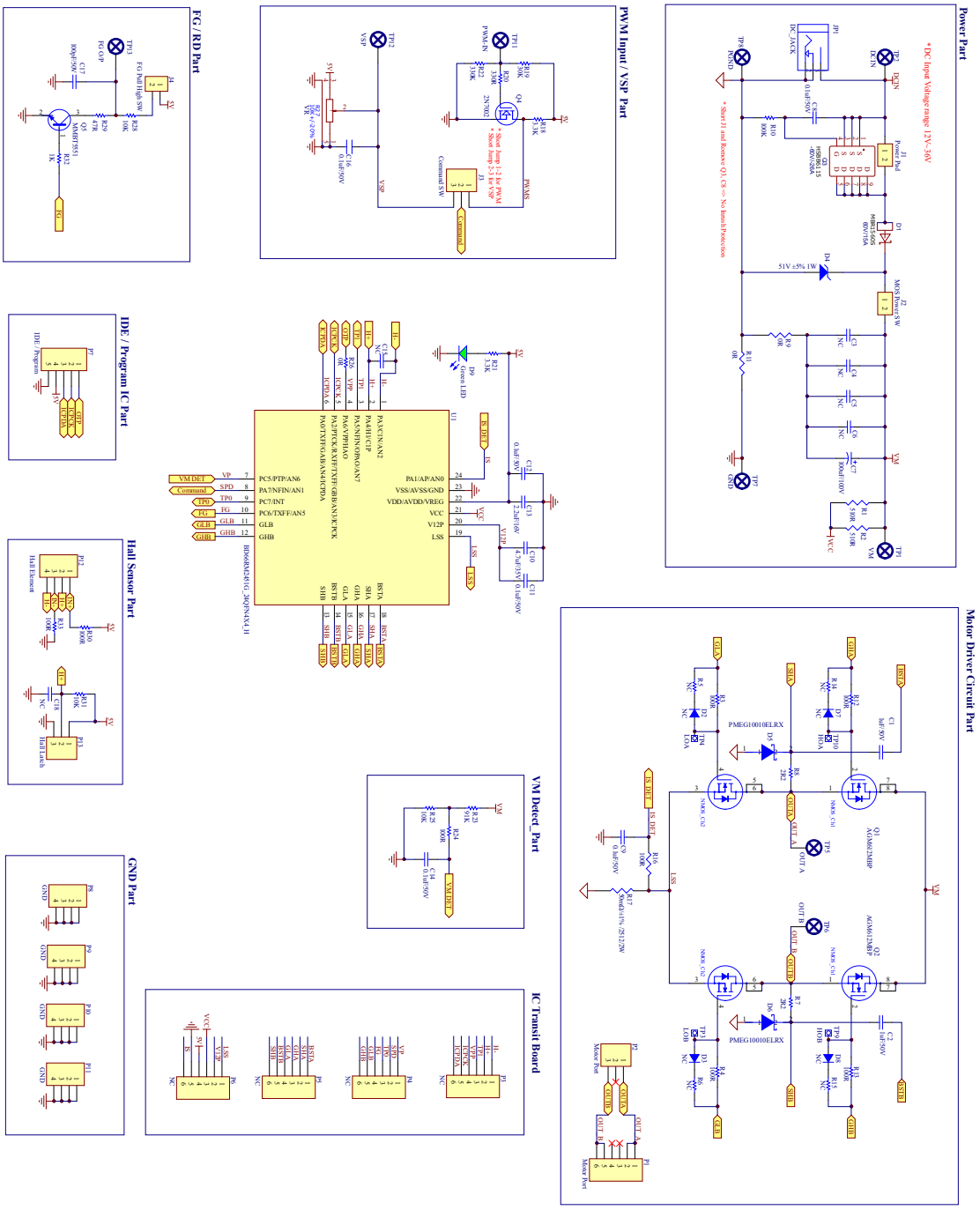
Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. The option must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Option	
1	HIRC frequency selection – f_{HIRC}: 16MHz or 20MHz

Note: When the HIRC has been configured at a frequency shown in this table, the HIRC1 and HIRC0 bits should also be set to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of 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	1Note	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	1Note	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	1Note	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	1Note	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C

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

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

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

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 sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

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	2Note	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	2Note	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	2Note	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	2Note	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2Note	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	2Note	Z
LORM A,[m]	Logical OR ACC to Data Memory	2Note	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2Note	Z
LCPL [m]	Complement Data Memory	2Note	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	2Note	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2Note	Z
Rotate			
LRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2Note	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2Note	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2Note	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2Note	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2Note	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2Note	None
LSET [m].i	Set bit of Data Memory	2Note	None

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

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← [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 ← [m]

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC "OR" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

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

Affected flag(s) None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. 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]≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C, 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] ← 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 ← 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~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

Description The low byte of the program code (specific page) addressed by the table pointer (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 (specific page) to TBLH and Data Memory Description Increment table pointer low byte, TBLP, first and then the program code (specific page) 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 ← 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] ← 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 ← 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] ← 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 ← ACC "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] ← ACC "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] ← 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 ← 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] ← [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 ← [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] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

LDEC [m] Decrement Data Memory

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

Operation [m] ← [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 ← [m] − 1

Affected flag(s) Z

LINC [m] Increment Data Memory

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

Operation [m] ← [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 ← [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 ← [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] ← 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 ← ACC "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] ← ACC "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) ← [m].i; (i=0~6)
[m].0 ← [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) ← [m].i; (i=0~6)
ACC.0 ← [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) ← [m].i; (i=0~6)
[m].0 ← C
C ← [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 theAccumulator 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

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 ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

LRRA [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

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 ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [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 ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [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 ← ACC − [m] − 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] ← ACC − [m] − 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 three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

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 three cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

LSET [m] Set Data Memory

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

Operation [m] ← 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 ← 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 three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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 three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written to the specified Data Memory again. 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 three 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 The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three 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 three 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 three 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 (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) 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

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 (specific page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code (specific page) 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

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

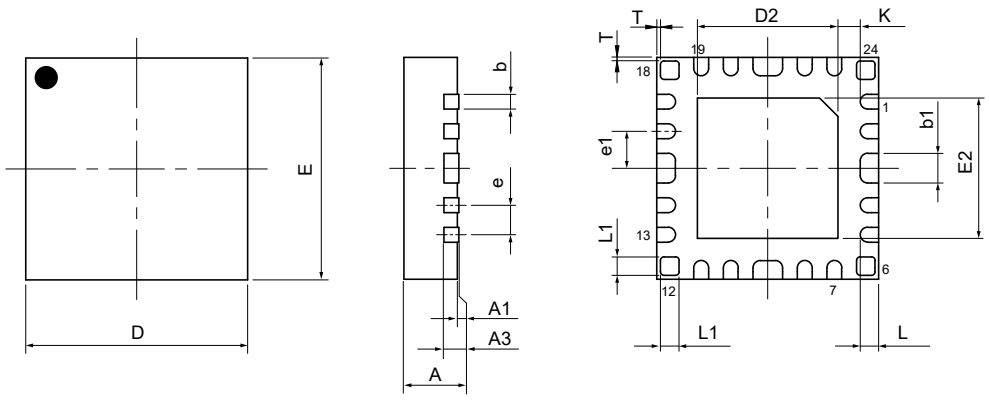
Package Information

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

SAW Type 24-pin QFN (4mm×4mm×0.75mm, lead: 0.325mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3		0.008 REF	
b	0.007	0.010	0.012
D		0.157 BSC	
E		0.157 BSC	
e		0.020 BSC	
D2	0.100	—	0.110
E2	0.100	—	0.110
L	0.011	0.013	0.015

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3		0.203 REF	
b	0.18	0.25	0.30
D		4.00 BSC	
E		4.00 BSC	
e		0.50 BSC	
D2	2.55	—	2.80
E2	2.55	—	2.80
L	0.28	0.33	0.38