Brushless DC Motor A/D Flash MCU

HT66FM5440

Revision: V1.10 Date: November 20, 2019

www.holtek.com

Table of Contents

Features **7**
CPU Features 7
Peripheral Features 7
General Description **8**
Block Diagram **9**
Pin Assignment **9**
Pin Description **10**
Absolute Maximum Ratings **13**
D.C. Characteristics **13**
A.C. Characteristics **14**
LVR/LVD Electrical Characteristics **14**
A/D Converter Electrical Characteristics **15**
D/A Converter Electrical Characteristics **16**
Operational Amplifier 0 Electrical Characteristics **16**
Operational Amplifier 1 & 2 Electrical Characteristics **17**
Unit Gain Buffer Electrical Characteristics **17**
Comparators Electrical Characteristics **18**
Power-on Reset Characteristics **18**
System Architecture **19**
Clocking and Pipelining 19
Program Counter 20
Stack 20
Arithmetic and Logic Unit – ALU 21
Flash Program Memory **22**
Structure 22
Special Vectors 22
Look-up Table 22
Table Program Example 23
In Circuit Programming – ICP 24
On-Chip Debug Support – OCDS 25
Data Memory **25**
Structure 25
Data Memory Addressing 26
General Purpose Data Memory 26
Special Purpose Data Memory 26
Special Function Register Description **28**
Indirect Addressing Registers – IAR0, IAR1, IAR2 28
Memory Pointers – MP0, MP1L, MP1H, MP2L, MP2H 28

Accumulator – ACC 29
Program Counter Low Register – PCL 30
Look-up Table Registers – TBLP, TBHP, TBLH 30
Status Register – STATUS 30

Oscillators 32
Oscillator Overview 32
System Clock Configurations 32
Internal High Speed RC Oscillator – HIRC 33
Internal 32kHz Oscillator – LIRC 33

Operating Modes and System Clocks 33
System Clocks 33
System Operation Modes 34
Control Registers 35
Operating Mode Switching 37
Standby Current Considerations 38
Wake-up 39

Watchdog Timer 40
Watchdog Timer Clock Source 40
Watchdog Timer Control Register 40
Watchdog Timer Operation 41

Reset and Initialisation 42
Reset Functions 42
Reset Initial Conditions 44

Input/Output Ports 51
Pull-high Resistors 52
Port A Wake-up 52
I/O Port Control Registers 53
Pin-shared Functions 53
I/O Pin Structures 60
Programming Considerations 60

Timer Modules – TM 61
Introduction 61
TM Operation 61
TM Clock Source 61
TM Interrupts 62
TM External Pins 62
TM Input/Output Pin Selection 62
Programming Considerations 63

Periodic Type TM – PTM 64
Periodic TM Operation 65
Periodic Type TM Register Description 66
Periodic Type TM Operating Modes 72

Capture Timer Module – CAPTM 84
Capture Timer Overview 84
Capture Timer Register Description 84
Capture Timer Operation 87
Noise Filter 89
Analog to Digital Converter 91
A/D Converter Overview 91
A/D Converter Register Description 92
A/D Converter Operation 101
Summary of A/D Conversion Steps 103
Considerations for simutaniously using both trigger methods 105
Programming Considerations 106
A/D Conversion Function 106
A/D Conversion Programming Examples 107
Comparators 109
Comparator Operation 109
Comparator Register 110
Over Current Detection 111
Over Current Detect Functional Description 111
Over Current Detect Register 112
Phase Current Detection 113
Phase Current Detect Functional Description 113
Phase Current Detect Register 113
BLDC Motor Control Circuit 114
Functional Description 114
PWM Counter Control Circuit 115
Mask Function 119
Other Functions 124
Hall Sensor Decoder 126
Motor Protection Function 133
I²C Interface 140
I²C Interface Operation 140
I²C Registers 141
I²C Bus Communication 144
I²C Bus Start Signal 145
I²C Slave Address 145
I²C Bus Read/Write Signal 145
I²C Bus Slave Address Acknowledge Signal 146
I²C Bus Data and Acknowledge Signal 146
I²C Time-out Control 147
UART Interface 149
UART External Pins 150
UART Data Transfer Scheme 150

UART Status and Control Registers 150
Baud Rate Generator 156
UART Setup and Control 156
UART Transmitter 158
UART Receiver 159
Managing Receiver Errors 160
UART Interrupt Structure 161
UART Power Down and Wake-up 162

Low Voltage Detector – LVD 163
LVD Register 163
LVD Operation 164

Multiplication Division Unit – MDU 165
MDU Registers 166
MDU Operation 167

Interrupts 169
Interrupt Registers 169
Interrupt Priority Configuration 178
Interrupt Preempt Function 180
Interrupt Operation 181
Hall Sensor Interrupts 184
External Interrupt 1 184
Noise Filter Input Interrupt 185
Comparator Interrupt 185
CAPTM Interrupts 185
Multi-function Interrupts 186
PWM Module Interrupts 186
A/D Converter Interrupts 187
TM Interrupts 187
UART Interrupt 188
I^2C Interrupt 188
Time Base Interrupt 188
LVD Interrupt 189
Interrupt Wake-up Function 190
Programming Considerations 190

Application Circuits 191
Introduction 191
Functional Description 191
Hardware Block Diagram 192
Hardware Circuit 193

Instruction Set 194
Introduction 194
Instruction Timing 194
Moving and Transferring Data 194
Arithmetic Operations 194

Logical and Rotate Operation 195
Branches and Control Transfer 195
Bit Operations 195
Table Read Operations 195
Other Operations 195
Instruction Set Summary 196
Table Conventions 196
Extended Instruction Set 198
Instruction Definition 200
Extended Instruction Definition 209
Package Information 216
28-pin SSOP (150mil) Outline Dimensions 217

Features

CPU Features

- Operating voltage
  - f_{SYS}=16MHz: 4.5V~5.5V
- 0.0625μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 16MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: NORMAL, IDLE and SLEEP
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 4K×16
- RAM Data Memory: 384×8
- Watchdog Timer function
- 26 bidirectional I/O lines
- Interrupt priority programmable with two interrupt preempt funcions
- Five pin-shared external interrupts – H1, H2, H3, NFIN and INT1
- Multiple Timer Modules for time measurement, input capture, compare match output or PWM output or single pulse output function
- Single 16-bit CAPTM for motor protection
- 3-channel 10-bit PWM with complementary outputs for BLDC application
- 6 external channels 10/12-bit resolution A/D converter
  - Normal A/D conversion
  - Delay auto-triggered A/D conversion, up to 4 channels available for each conversion
- I^2C Interface
- Single Fully-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- Four comparator functions
- Over current detection – Operational Amplifier 0, Comparator 0 and 8-bit D/A converter
- Phase current detection – Operational Amplifier 1 and 2
- Single Time-Base function for generation of fixed time interrupt signals
- Integrated Multiplication/Division Units
  - One 8-bit Multiplication/Division Unit – MDU0
  - One 16-bit Multiplication/Division Unit – MDU1
- Low voltage reset function
- Low voltage detect function
- Flash program memory can be re-programmed up to 10,000 times
- Flash program memory data retention > 10 years
- Package type: 28-pin SSOP

General Description

The BLDC Motor Flash MCU HT66FM5440 provides PWM configuration flexibility and complete protection mechanism which are necessary for brushless DC motor applications. The device uses the HT8-1T architecture to achieve one oscillation cycle for one instruction. With the integrated 16MHz oscillator, executing a single instruction only needs 0.0625μs.

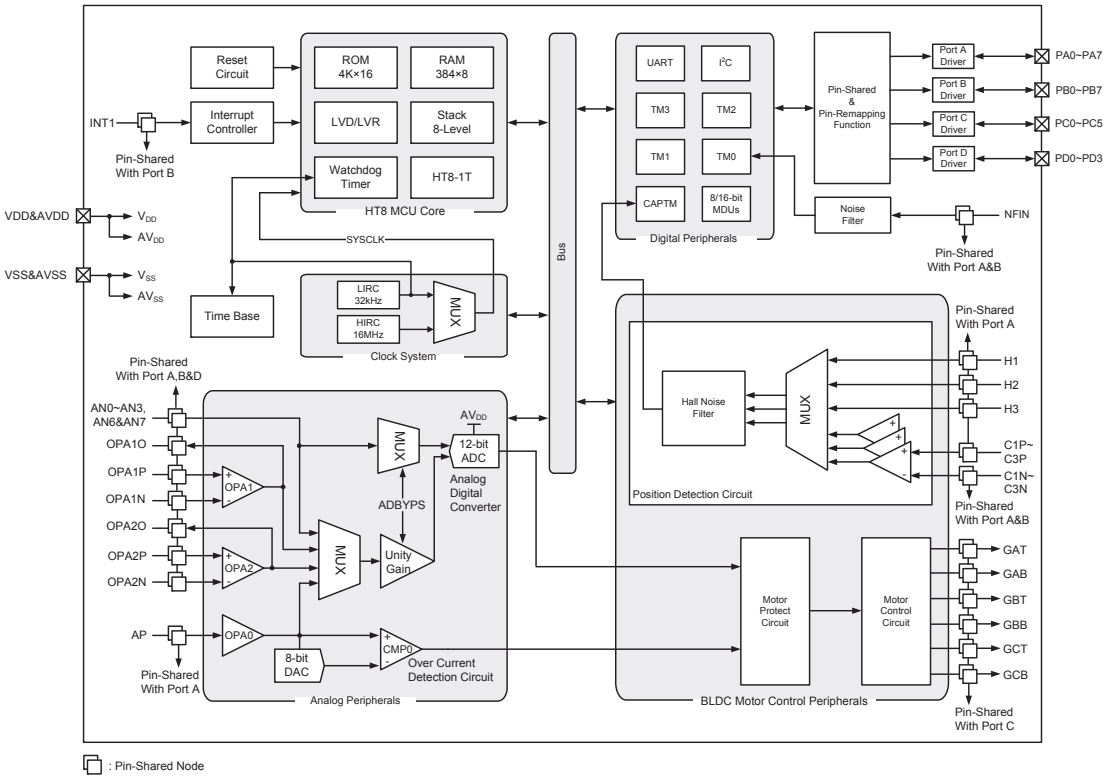
Compared with its predecessor, the HT66FM5440 adds the interrupt priority configuration function and two interrupt preempt functions, which can be flexibly configured according to different interrupt priority requirements for different systems. Two unsigned Multiplication and Division Units (MDU), one 8-bit type and one 16-bit type, are provided for complicated operational requirements. The 16-bit MDU additionally integrates a 32-bit/16-bit unsigned divider.

The 10-bit PWM generator dedicated for motor control provides different duty configurations for the four channels and up to three PWM complimentary output pairs. The motor current and voltage often require real-time detection and accurate trigger timing, to achieve this purpose an A/D auto-scan function is provided. Users can flexibly select to use the 10-bit PWM period or duty signal to trigger the A/D conversion. If using the interrupt to check for the end of the A/D conversion, the main program or subroutine program running will not be affected during the conversion process.

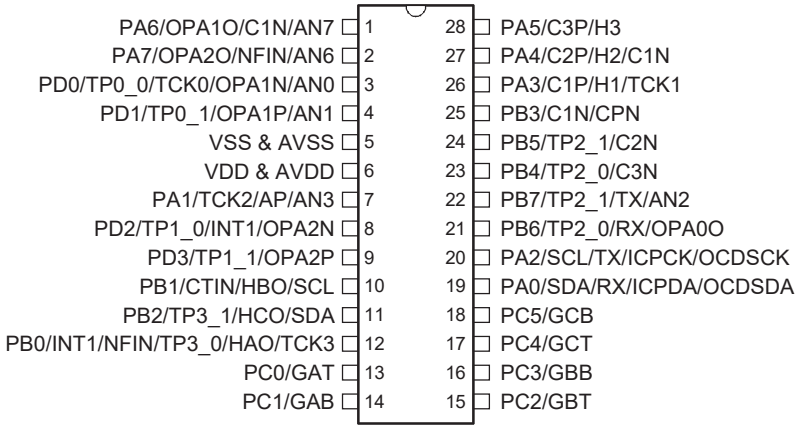
With the A/D auto-scan continuous sampling function provided, the A/D converter also supports 10-bit or 12-bit resolution, which is adjustable according to the actual resolution and sampling time requirement. In addition, each channel has a unity-gain buffer which is used to avoid voltage interference when switching between A/D channels. An integrated OCP circuit composed of OPA0 and CMP0, and two OPAs, OPA1 and OPA2, can be flexibly used for motor current detection. The OCP function can effectively provide motor instant large current protection and turn off the internal motor drive signal to achieve instantaneous system protection. After the OPA0~OPA2 output signals are sampled and converted by the A/D converter, the converted data will be compared with the preset boundary values. If the converted data exceeds the preset range, an interrupt will be generated for the user to do the corresponding processing. With regard to motor stalling protection, a 16-bit Capture Timer is provided to monitor the motor speed loop thus achieving stalling protection.

With the aforementioned advantages, the HT66FM5440 is especially suitable for brushless motor square-wave Hall solutions, square-wave sensorless solutions and sine-wave Hall driving solutions.

Block Diagram



Pin Assignment



HT66FM5440/HT66VM5440
28 SSOP-A

Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by corresponding software control bits.
2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the HT66VM5440 device which is the OCDS EV chip for the HT66FM5440 device.
3. The "VDD&AVDD" means that the VDD and AVDD are internally bonded while the "VSS&AVSS" means that the VSS and AVSS are internally bonded.

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/SDA/RX/ ICPDA/OCDSDA	PA0	PAPU PAWU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SDA	PAPS0 PRM	ST	NMOS	I²C data/address line
	RX	PAPS0 PRM	ST	—	External UART RX serial data input pin
	ICPDA	—	ST	CMOS	ICP data/address pin
	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only.
PA1/TCK2/AP/AN3	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	TCK2	PAPS0	ST	—	PTM2 clock input
	AP	PAPS0	AN	—	Operational Amplifier 0 positive input
	AN3	PAPS0	AN	—	A/D Converter external input 3
PA2/SCL/TX/ ICPCK/OCDSCK	PA2	PAPU PAWU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SCL	PAPS0 PRM	ST	—	I²C clock line
	TX	PAPS0 PRM	—	CMOS	External UART TX serial data output pin
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/C1P/H1/TCK1	PA3	PAPU PAWU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	C1P	PAPS0	AN	—	Comparator 1 positive intput
	H1	PAPS0	ST	—	Hall sensor input
	TCK1	PAPS0	ST	—	PTM1 clock input
PA4/C2P/H2/C1N	PA4	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	C2P	PAPS1	AN	—	Comparator 2 positive intput
	H2	PAPS1	ST	—	Hall sensor input
	C1N	PAPS1 PRM	AN	—	Comparator 1 negative input
PA5/C3P/H3	PA5	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	C3P	PAPS1	AN	—	Comparator 3 positive intput
	H3	PAPS1	ST	—	Hall sensor input
PA6/OPA1O/ C1N/AN7	PA6	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OPA1O	PAPS1	—	AN	Operational Amplifier 1 output
	C1N	PAPS1 PRM	AN	—	Comparator 1 negative input
	AN7	PAPS1	AN	—	A/D Converter external input 7

Pin Name	Function	OPT	I/T	O/T	Description
PA7/OPA2O/ NFIN/AN6	PA7	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OPA2O	PAPS1	—	AN	Operational Amplifier 2 output
	NFIN	PAPS1 PRM	ST	—	Noise Filter External input
	AN6	PAPS1	AN	—	A/D Converter external input 6
PB0/INT1/NFIN/ TP3_0/HAO/TCK3	PB0	PBPU PBPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT1	PBPS0 PRM INTEG	ST	—	External interrupt 1 input
	NFIN	PBPS0 PRM	ST	—	Noise Filter External input
	TP3_0	PBPS0	ST	CMOS	PTM3 input/output
	HAO	PBPS0	—	CMOS	Test pin for SA
	TCK3	PBPS0	ST	—	PTM3 clock input
PB1/CTIN/HBO/ SCL	PB1	PBPU PBPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTIN	PBPS0	ST	—	CAPTM capture input
	HBO	PBPS0	—	CMOS	Test pin for SB
	SCL	PBPS0 PRM	ST	—	I^2C clock line
PB2/TP3_1/HCO/ SDA	PB2	PBPU PBPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TP3_1	PBPS0	—	CMOS	PTM3 output
	HCO	PBPS0	—	CMOS	Test pin for SC
	SDA	PBPS0 PRM	ST	NMOS	I^2C data/address line
PB3/C1N/CPN	PB3	PBPU PBPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	C1N	PBPS0 PRM	AN	—	Comparator 1 negative input
	CPN	PBPS0	—	AN	Comparator 1&2&3 negative input when C1N&C2N&C3N are shorted
PB4/TP2_0/C3N	PB4	PBPU PBPS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	TP2_0	PBPS1	ST	CMOS	PTM2 input/ouput
	C3N	PBPS1	AN	—	Comparator 3 negative input
PB5/TP2_1/C2N	PB5	PBPU PBPS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	TP2_1	PBPS1	—	CMOS	PTM2 ouput
	C2N	PBPS1	AN	—	Comparator 2 negative input
PB6/TP2_0/RX/ OPA0O	PB6	PBPU PBPS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	TP2_0	PBPS1	ST	CMOS	PTM2 input/ouput
	RX	PBPS1 PRM	ST	—	External UART RX serial data input pin
	OPA0O	PBPS1	—	AN	Operational Amplifier 0 output

Pin Name	Function	OPT	I/T	O/T	Description
PB7/TP2_1/TX/AN2	PB7	PBPU PBPS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	TP2_1	PBPS1	—	CMOS	PTM2 ouput
	TX	PBPS1 PRM	—	CMOS	External UART TX serial data output pin
	AN2	PBPS1	AN	—	A/D Converter external input 2
PC0/GAT	PC0	PCPU PCPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	GAT	PCPS0	—	CMOS	Pulse width modulation complementary output
PC1/GAB	PC1	PCPU PCPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	GAB	PCPS0	—	CMOS	Pulse width modulation complementary output
PC2/GBT	PC2	PCPU PCPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	GBT	PCPS0	—	CMOS	Pulse width modulation complementary output
PC3/GBB	PC3	PCPU PCPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	GBB	PCPS0	—	CMOS	Pulse width modulation complementary output
PC4/GCT	PC4	PCPU PCPS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	GCT	PCPS1	—	CMOS	Pulse width modulation complementary output
PC5/GCB	PC5	PCPU PCPS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	GCB	PCPS1	—	CMOS	Pulse width modulation complementary output
PD0/TP0_0/TCK0/OPA1N/AN0	PD0	PDPU PDPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TP0_0	PDPS0	ST	CMOS	PTM0 input/ouput
	TCK0	PDPS0	ST	—	PTM0 clock input
	OPA1N	PDPS0	AN	—	Operational Amplifier 1 negative input
	AN0	PDPS0	AN	—	A/D Converter external input 0
PD1/TP0_1/OPA1P/AN1	PD1	PDPU PDPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TP0_1	PDPS0	—	CMOS	PTM0 ouput
	OPA1P	PDPS0	AN	—	Operational Amplifier 1 positive input
	AN1	PDPS0	AN	—	A/D Converter external input 1
PD2/TP1_0/INT1/OPA2N	PD2	PDPU PDPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TP1_0	PDPS0	ST	CMOS	PTM1 input/ouput
	INT1	PDPS0 PRM INTEG	ST	—	External interrupt 1 input
	OPA2N	PDPS0	AN	—	Operational Amplifier 2 negative input
PD3/TP1_1/OPA2P	PD3	PDPU PDPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TP1_1	PDPS0	—	CMOS	PTM1 ouput
	OPA2P	PDPS0	AN	—	Operational Amplifier 2 positive input
VDD&AVDD	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply

Pin Name	Function	OPT	I/T	O/T	Description
VSS&AVSS	VSS	—	PWR	—	Digital negative power supply, ground
	AVSS	—	PWR	—	Analog 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;
NMOS: NMOS output; AN: Analog signal

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}−0.3V to V_{SS}+6.0V
Input Voltage ..... V_{SS}−0.3V to V_{DD}+0.3V
Storage Temperature ..... -50˚C to 125˚C
Operating Temperature ..... -40˚C to 85˚C
I_{OL} Total 80mA
I_{OH} 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

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (HIRC)	—	f_{SYS}=16MHz	4.5	—	5.5	V
I_{DD}	Operating Current (HIRC)	5V	$f_{SYS}=f_H$=16MHz, No load, ADC off, WDT enable, Motor_CTL off, IR_RX off	—	29	39	mA
		5V	$f_{SYS}=f_H$=16MHz/2, No load, ADC off, WDT enable, Motor_CTL off, IR_RX off	—	20	30	mA
I_{STB}	Standby Current	—	LIRC and LVR on, LVD off, WDT enable	—	17	26	μA
V_{IL}	Input Low Voltage for I/O Ports	—	—	0	—	$0.3V_{DD}$	V
V_{IH}	Input High Voltage for I/O Ports	—	—	$0.7V_{DD}$	—	V_{DD}	V
I_{OL}	Sink Current for I/O Ports	5V	$V_{OL}=0.1V_{DD}$	32	64	—	mA
I_{OH}	Source Current for I/O Ports	5V	$V_{OH}=0.9V_{DD}$	-8	-15	—	mA
R_{PH}	Pull-high Resistance for I/O Ports	5V	—	10	30	50	kΩ

A.C. Characteristics

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (HIRC)	4.5V~5.5V	$f_{SYS}=f_{HIRC}$=16MHz	—	16	—	MHz
f_{HIRC}	High Speed Internal RC Oscillator Frequency (HIRC)	5V	Ta=25°C	-1.0%	16	+1.0%	MHz
			Ta= -40°C~85°C	-2.0%	16	+2.0%	
		4.5V~5.5V	Ta=25°C	-2.5%	16	+2.5%	MHz
			Ta= -40°C~85°C	-3.0%	16	+3.0%	
f_{LIRC}	Low Speed Internal RC Oscillator Frequency (LIRC)	4.5V~5.5V	Ta=25°C	-5%	32	+5%	kHz
			Ta= -40°C~85°C	-10%	32	+10%	
t_{START}	LIRC Start Up Time	—	—	—	—	100	μs
f_{I2C}	I^2C Standard Mode (100kHz) f_{SYS} Frequency	—	No clock debounce	2	—	—	MHz
			2 system clock debounce	4	—	—	
			4 system clock debounce	8	—	—	
	I^2C Fast Mode (400kHz) f_{SYS} Frequency	—	No clock debounce	5	—	—	MHz
			2 system clock debounce	10	—	—	
			4 system clock debounce	20	—	—	
t_{TCK}	TM Capture Input Pin Minimum Pulse Width	—	—	0.2	—	—	μs
t_{TPI}	TM Capture Input Pin Minimum Pulse Width	—	—	—	4	6	ms
t_{INT}	External Interrupt Minimum Pulse Width	—	—	1	5	10	t_{SYS}
t_{SST}	System Start-up Timer Period (Wake-up from HALT Status)	—	$f_{SYS}=f_{HIRC}$	—	15~16	—	t_{SYS}
t_{RSTD}	System Reset Delay Time (Power-on Reset)	—	—	25	50	100	ms
	System Reset Delay Time (Any Reset except Power On Reset)	—	—	8.3	16.7	33.3	ms

Note: 1. $t_{SYS}=1/f_{SYS}$
2. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS and located as close to the device as possible.

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 3.8V	-5%	3.8	+5%	V
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 4.0V	-5%	4.0	+5%	V
$I_{LVRLVDBG}$	Operating Current	3V	LVD enable, LVR enable, VBGEN=0	—	13	16	μA
		5V		—	21	23	μA
		3V	LVD enable, LVR enable, VBGEN=1	—	14	17	μA
		5V		—	20	22	μA
I_{LVR}	Additional Current for LVR Enable	—	LVD disable, VBGEN=0	—	—	26	μA
I_{LVD}	Additional Current for LVD Enable	—	LVR disable, VBGEN=0	—	—	26	μA
t_{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	15	μs
		—	For LVR disable, VBGEN=0, LVD off → on	—	—	150	μs

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs

A/D Converter Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
AV_{DD}	A/D Converter Operating Voltage	—	—	4.5	5	5.5	V
I_{OP}	A/D Converter Operating Current	5V	—	—	0.7	—	mA
I_{STB}	A/D Converter Standby Current	—	Digital input not changed	—	—	1	μA
V_{REF}	A/D Converter Reference Voltage	—	—	2	AV_{DD}	AV_{DD} +0.1	V
t_{CONV}	12-bit Conversion Time	—	—	—	—	19	t_{ADCK}
	10-bit Conversion Time	—	—	—	—	17	t_{ADCK}
DNL	A/D Differential Non-linearity	4.5V 5.5V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=0.1μs, 10-bit	-3	—	+3	LSB
		4.5V 5.5V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=10μs, 10-bit				
		4.5V 5.5V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=0.16μs, 12-bit	-3	—	+3	LSB
		4.5V 5.5V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=10μs, 12-bit				
INL	A/D Integral Non-linearity	4.5V 5.5V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=0.1μs, 10-bit	-4	—	+4	LSB
		4.5V 5.5V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=10μs, 10-bit				
		4.5V 5.5V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=0.16μs, 12-bit	-4	—	+4	LSB
		4.5V 5.5V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=10μs, 12-bit				
t_{ADCK}	A/D Converter Clock Period	—	12-bit	0.16	—	10	μs
		—	10-bit	0.1	—	10	
t_{CKH}	A/D Converter Clock High Pulse	—	—	225	—	—	ns
t_{CKL}	A/D Converter Clock Low Pulse	—	—	225	—	—	ns
t_{ST}	A/D Converter Turn-on Setup Time	—	—	2	—	—	ns
	A/D Converter Start Bit Setup Time	—	—	2	—	—	ns
t_{STH}	A/D Converter Start Bit High Pulse	—	—	25	—	—	ns
t_{DEOC}	EOCB Bit Output Delay	—	AV_{DD}=5V	—	3	—	ns
t_{DOUT}	Conversion Data Output Delay	—	AV_{DD}=5V	—	3	—	ns
t_{ON}	A/D Converter Wake-up Time	—	—	2	—	—	μs
t_{OFF}	A/D Converter Sleep Time	—	—	—	—	5	ns

D/A Converter Electrical Characteristics

Ta=25°C

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

Operational Amplifier 0 Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	V_{LVR}	5	5.5	V
I_{PD}	Power Down Current	5V	—	—	—	0.1	μA
V_{OPOS}	Input Offset Voltage	5V	Without calibration, A0OF[4:0]=10000B	-15	—	+15	mV
		5V	With calibration	-2	—	+2	
V_{CM}	Common Mode Voltage Range	5V	—	V_{SS}	—	V_{DD}-1.4	V
V_{OR}	Maximum Output Voltage Range	5V	—	V_{SS} +0.2	—	V_{DD}-0.2	V
I_{OP}	Operating Current	5V	—	—	300	—	μA
PSRR	Power Supply Rejection Ratio	5V	—	90	—	96	dB
CMRR	Common Mode Rejection Ratio	5V	V_{CM}=0~(V_{DD}-1.4V)	—	106	—	dB
SR	Slew Rate+, Slew Rate-	5V	R_L=600Ω, C_L=100pF	1.8	2.5	—	V/μs
GBW	Gain Band Width	5V	R_L=600Ω, C_L=100pF	2.05	3.7	7.16	MHz
A_{OL}	Open Loop Gain	5V	R_L=600Ω, C_L=100pF	—	96	—	dB
PM	Phase Margin	5V	R_L=600Ω, C_L=100pF	—	90	—	—

Operational Amplifier 1 & 2 Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	V_{LVR}	5	5.5	V
I_{PD}	Power Down Current	5V	—	—	—	0.1	μA
V_{OPOS}	Input Offset Voltage	5V	Without calibration	-15	—	+15	mV
V_{CM}	Common Mode Voltage Range	5V	—	V_{SS} +0.2	—	V_{DD} -0.2	V
V_{OR}	Maximum Output Voltage Range	5V	—	V_{SS} +0.2	—	V_{DD} -0.2	V
I_{OP}	Operating Current	5V	—	—	800	—	μA
PSRR	Power Supply Rejection Ratio	5V	—	50	60	—	dB
CMRR	Common Mode Rejection Ratio	5V	V_{CM}=0~V_{DD}	50	60	—	dB
SR	Slew Rate+, Slew Rate-	5V	R_L=100kΩ, C_L=100pF	1.8	2.5	—	V/μs
GBW	Gain Band Width	5V	R_L=100kΩ, C_L=100pF	—	1	—	MHz
A_{OL}	Open Loop Gain	5V	R_L=100kΩ, C_L=100pF	60	80	—	dB
PM	Phase Margin	5V	R_L=100kΩ, C_L=100pF	50	60	—	—

Unit Gain Buffer Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	V_{LVR}	5	5.5	V
I_{PD}	Power Down Current	5V	—	—	—	0.1	μA
V_{OPS}	Input Offset Voltage	5V	Without calibration	-15	—	+15	mV
V_{CM}	Common Mode Voltage Range	5V	—	V_{SS} +0.2	—	V_{DD} -0.2	V
V_{OR}	Maximum Output Voltage Range	5V	—	V_{SS} +0.2	—	V_{DD}- 0.2	V
I_{OP}	Operating Current	5V	—	—	800	—	μA
PSRR	Power Supply Rejection Ratio	5V	—	50	60	—	dB
CMRR	Common Mode Rejection Ratio	5V	V_{CM}=0~V_{DD}	50	60	—	dB
SR	Slew Rate+, Slew Rate-	5V	R_L=100kΩ, C_L=100pF	1.8	2.5	—	V/μs
GBW	Gain Band Width	5V	R_L=100kΩ, C_L=100pF	—	1	—	MHz
A_{OL}	Open Loop Gain	5V	R_L=100kΩ, C_L=100pF	60	80	—	dB
PM	Phase Margin	5V	R_L=100kΩ, C_L=100pF	50	60	—	—

Comparators Electrical Characteristics

Ta=25°C

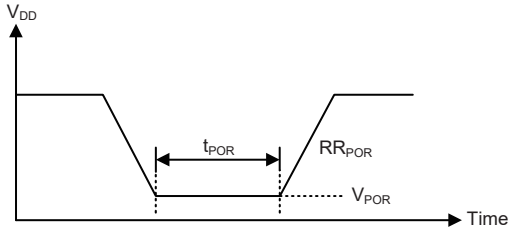
Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Comparator Operating Voltage	—	—	V_{LVR}	5.0	5.5	V
I_{CMP}	Comparator Operating Current	5V	—	—	300	450	μA
I_{OFF}	Comparator Power Down Current	5V	Comparator disabled	—	—	0.1	μA
V_{CMPOS}	Comparator Input Offset Voltage	5V	—	-10	—	+10	mV
V_{HYS}	Hysteresis Width	5V	Comparator 0	-50	100	+50	mV
		5V	Comparator 1, 2, 3	10	30	50	mV
V_{CM}	Input Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD} -1.4	V
A_{OL}	Comparator Open Loop Gain	—	—	100	120	—	dB
t_{PD}	Comparator Response Time	5V	V_{CM}=0~(V_{DD}-1.4V), With 10mV overdrive	—	—	1	μs
		5V	With 100mV overdrive (Note)	—	—	200	ns

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

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



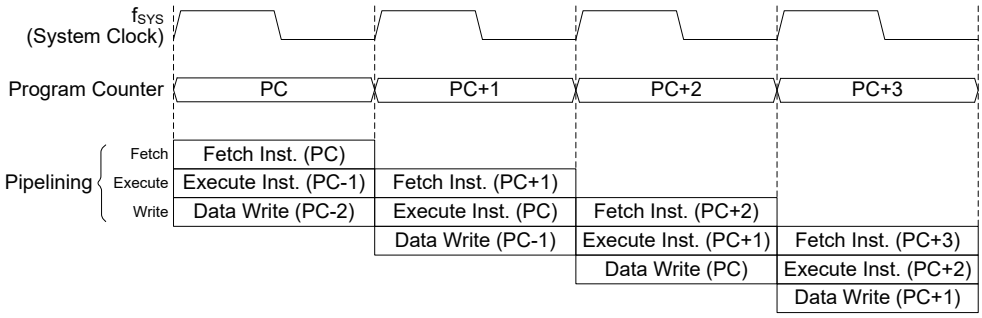
System Architecture

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

Clocking and Pipelining

The main system clock, derived from the HIRC oscillator is also used as the instruction clock. The Program Counter is incremented at the beginning of the the system clock during which time a new instruction is fetched. The following two system clocks carry out the instruction decoding/execution and the data write-back functions respectively. Although the instruction fetching, decoding/execution and data write-back operations take place in consecutive instruction cycles, the pipelining structure of the microcontroller ensures that instructions are effectively executed in one instruction cycle. The exception to this are instructions where the contents of the Program Counter are changed, such as subroutine calls or jumps, in which case the instruction will take one more instruction cycle to execute.

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



System Clocking and Pipelining

Program Counter

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

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

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

Program Counter

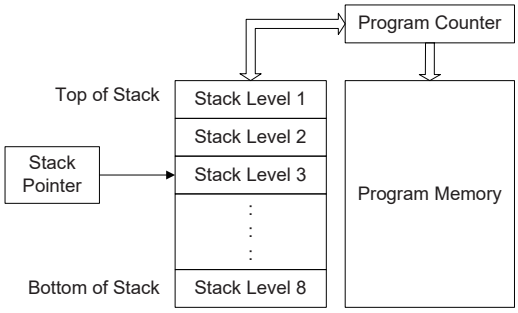
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into multiple 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:

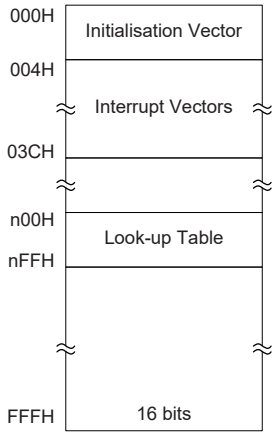
- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
  LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
  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

Flash Program Memory

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

Structure

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



Program Memory Structure

Special Vectors

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

Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the corresponding table read instruction such as "TABRD [m]" or "TABRDL [m]" respectively when the memory [m] is located in sector 0. If the memory [m] is located in other sectors, the data can be retrieved from the program memory using the corresponding extended table read instruction such as "LTABRD [m]" or "LTABRDL [m]" respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

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

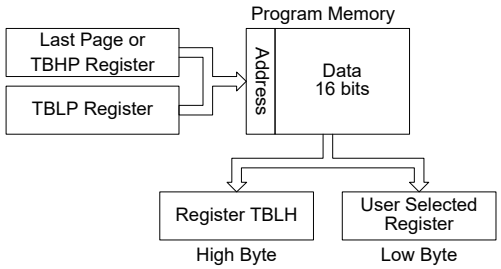


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "0F00H" which refers to the start address of the last page within the 4K words Program Memory of the device. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0F06H" or 6 locations after the start of the last page. 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          ; set initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

In Circuit Programming – ICP

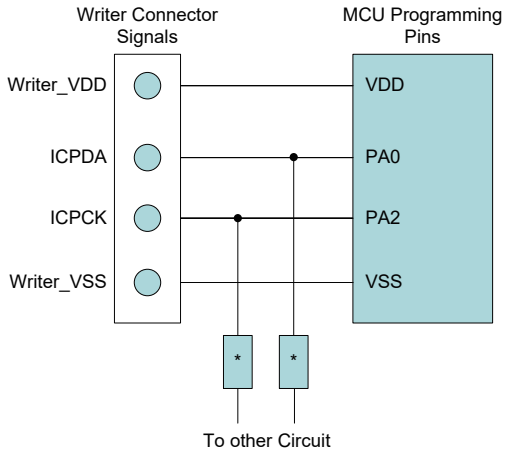
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

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

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

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



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

On-Chip Debug Support – OCDS

There is an EV chip named HT66VM5440 which is used to emulate the HT66FM5440 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. 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 Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

Holtek e-Link Pins	EV Chip Pins	Pin Description
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

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, known as the Special Function Data Memory. 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 known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

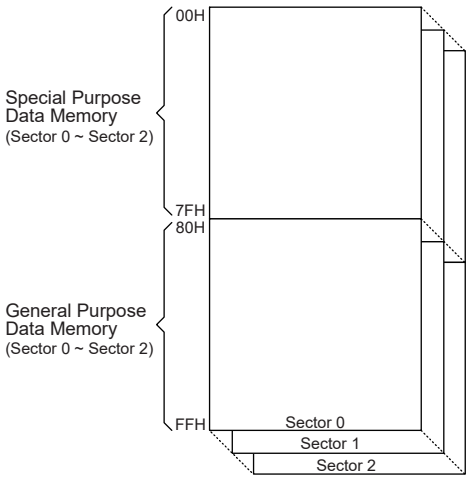
Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value if using indirect addressing method.

Structure

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

Special Purpose Data Memory	General Purpose Data Memory	
Located Sectors	Capacity	Sector: Address
0, 1, 2	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. 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 corresponding instruction 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 programing for both reading and writing operations. By using the bit operation instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

Special Purpose Data Memory

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

	Sector 0, 2	Sector 1
00H	IAR0	IAR0
01H	MP0	MP0
02H	IAR1	IAR1
03H	MP1L	MP1L
04H	MP1H	MP1H
05H	ACC	ACC
06H	PCL	PCL
07H	TBLP	TBLP
08H	TBLH	TBLH
09H	TBHP	TBHP
0AH	STATUS	STATUS
0BH		
0CH	IAR2	IAR2
0DH	MP2L	MP2L
0EH	MP2H	MP2H
0FH	SMOD	SMOD
10H	LVDC	LVDC
11H	LVRC	LVRC
12H	WDTC	WDTC
13H		TBC
14H	INTEG1	Pri_name0
15H	INTC0	Pri_name1
16H	INTC1	Pri_name2
17H	INTC2	Pri_name3
18H	INTC3	Pri_name4
19H	MFI0	Pri_name5
1AH	MFI1	Pri_name6
1BH	MFI2	Pri_name7
1CH	MFI3	MFI3
1DH	MFI4	MFI4
1EH	MFI5	MFI5
1FH	MFI6	MFI6
20H	MFI7	MFI7
21H	PAWU	PAWU
22H	PAPU	PTM0C2
23H	PA	PA
24H	PAC	PAC
25H	PBPU	PTM1C2
26H	PB	PB
27H	PBC	PBC
28H	PCPU	PTM2C2
29H	PC	PC
2AH	PCC	PCC
2BH	PDPU	PTM3C2
2CH	PD	PD
2DH	PDC	PDC
2EH	CAPTC0	PTM0BL
2FH	CAPTC1	PTM0BH
30H	CAPTMDL	PTM1BL
31H	CAPTMDH	PTM1BH
32H	CAPTMAL	PTM2BL
33H	CAPTMAH	PTM2BH
34H	CAPTMCL	PTM3BL
35H	CAPTMCH	PTM3BH
36H	ADRL	ISRL0
37H	ADRH	ISRH0
38H	ADCR0	ISRL1
39H	ADCR1	ISRH1
3AH	ADCR2	ISRL2
3BH	ADISG1	ISRH2
3CH	ADISG2	ISRL3
3DH	ADDL	ISRH3
3EH	ADLVDL	ADLVDL
3FH	ADLVDH	ADLVDH

	Sector 0, 2	Sector 1
40H	ADHVDL	
41H	ADHVDH	ADBYPS
42H	PBPS0	PBPS0
43H	PBPS1	PBPS1
44H	OCPS	ADCR3
45H	INTEG0	INTEG0
46H	CTRL	CTRL
47H		PTM1C0
48H		PTM1C1
49H	HDCR	PTM1DL
4AH	HDCD	PTM1DH
4BH	MPTC1	PTM1AL
4CH	MPTC2	PTM1AH
4DH	DUTR3L	PTM1RPL
4EH	DUTR3H	PTM1RPH
4FH	PTM0C0	PTM0C0
50H	PTM0C1	PTM0C1
51H	PTM0DL	PTM0DL
52H	PTM0DH	PTM0DH
53H	PTM0AL	PTM0AL
54H	PTM0AH	PTM0AH
55H	PTM0RPL	PWMCS
56H	PTM0RPH	OPA1CAL
57H	PWMC	OPA2CAL
58H	DUTR0L	HCHK_NUM
59H	DUTR0H	HNF_MSEL
5AH	DUTR1L	PTM2C0
5BH	DUTR1H	PTM2C1
5CH	DUTR2L	PTM2DL
5DH	DUTR2H	PTM2DH
5EH	PRDRL	PTM2AL
5FH	PRDRH	PTM2AH
60H	PWMRL	PTM2RPL
61H	PWMRH	PTM2RPH
62H	PWMME	HDCT0
63H	PWMMD	HDCT1
64H	MCF	HDCT2
65H	MCD	HDCT3
66H	DTS	HDCT4
67H	PLC	HDCT5
68H	IICC0	HDCT6
69H	IICC1	HDCT7
6AH	IICD	HDCT8
6BH	IICA	HDCT9
6CH	IICTOC	HDCT10
6DH	USR	HDCT11
6EH	UCR1	OPOMS
6FH	UCR2	OPCM
70H	BRG	OPA0CAL
71H	TXR_RXR	PTM3C0
72H	CMPC	PTM3C1
73H	MDU0R0	PTM3DL
74H	MDU0R1	PTM3DH
75H	MDU0CTRL	PTM3AL
76H	MDU1R0	PTM3AH
77H	MDU1R1	PTM3RPL
78H	MDU1R2	PTM3RPH
79H		
7AH	MDU1R3	PAPS0
7BH	MDU1R4	PAPS1
7CH	MDU1R5	PCPS0
7DH	MDU1CTRL	PCPS1
7EH	NF_VIH	PDPS0
7FH	NF_VIL	PRM

: 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 instruction 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 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	R	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 zero flag.

For other instructions, the CZ flag will not be affected.

Bit 5 **TO**: Watchdog Time-out flag
0: After power up or executing the "CLR WDT" or "HALT" instruction
1: A watchdog time-out occurred.

Bit 4 **PDF**: Power down flag
0: After power up or executing the "CLR WDT" instruction
1: By executing the "HALT" instruction

Bit 3 **OV**: Overflow flag
0: No overflow
1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa.

Bit 2 **Z**: Zero flag
0: The result of an arithmetic or logical operation is not zero
1: The result of an arithmetic or logical operation is zero

Bit 1 **AC**: Auxiliary flag
0: No auxiliary carry
1: An operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction

Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

The "C" flag is also affected by a rotate through carry instruction.

Oscillators

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

Oscillator Overview

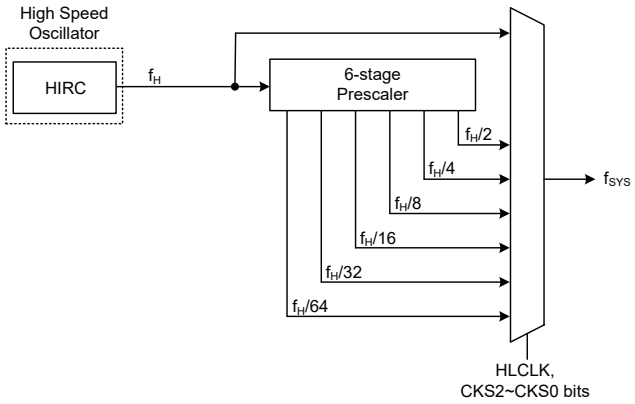
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Fully integrated internal oscillators, requiring no external components, are provided for fast system oscillator and other low frequency requirements. The 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 different system clocks, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

Type	Name	Frequency
Internal High Speed RC	HIRC	16MHz
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 clocks is sourced from the internal 16MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Note that the system clock only comes from HIRC. Selecting whether the f_H or $f_H/2$~$f_H/64$ clock is used as the system clock is implemented using the HLCLK bit and CKS2~CKS0 bits in the SMOD register and as the system clock can be dynamically selected. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.



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 RC oscillator has a fixed frequency of 16MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that if this internal system clock option is selected, as it requires no external pins for its operation.

Internal 32kHz Oscillator – LIRC

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

Operating Modes and System Clocks

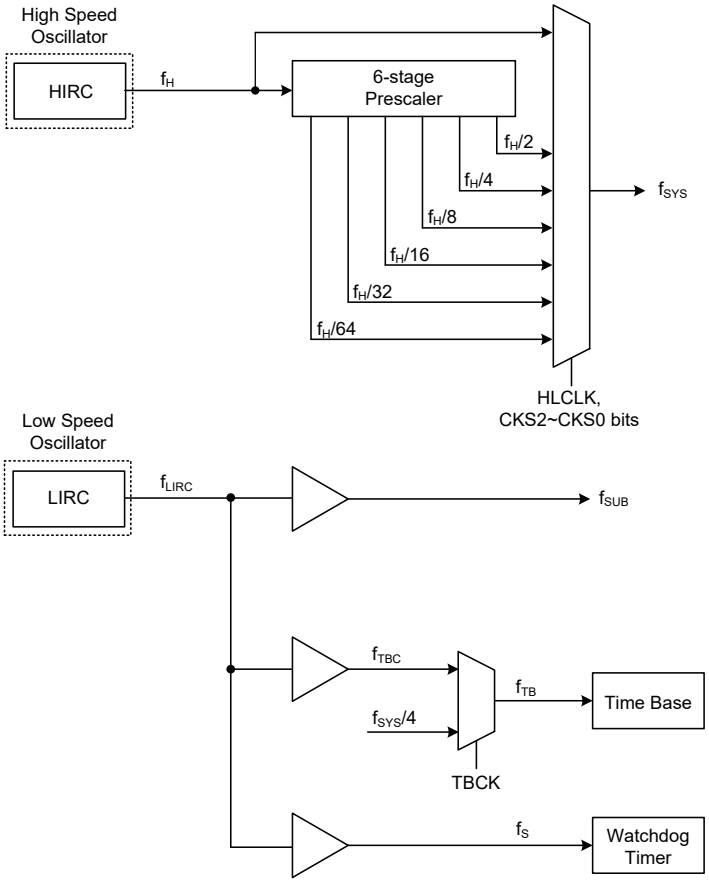
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their 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 comes from a high frequency, f_H, or its divided version, $f_H/2$~$f_H/64$, and is selected using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. The high speed system clock is sourced from the HIRC oscillator.

There are two additional internal clocks for the peripheral circuits, the substitute clock, f_{SUB}, and the Time Base clock, f_{TBC}. Each of these internal clocks is sourced from the LIRC oscillator.



Device Clock Configurations

System Operation Modes

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

Operating Mode	Description				
	CPU	f_{SYS}	f_{SUB}	f_S	f_{TBC}
NORMAL Mode	On	f_H~$f_H/64$	On	On	On
IDLE0 Mode	Off	Off	On	On	On
IDLE1 Mode	Off	On	On	On	On
SLEEP Mode	Off	Off	On	On	Off

NORMAL Mode

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

SLEEP Mode

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

IDLE0 Mode

The IDLE0 Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU but some peripheral functions will remain operational such as the Watchdog Timer, TMs and Time Base. In the IDLE0 Mode, the system oscillator will be stopped, the f_{SUB}, f_S and f_{TBC} clocks will be on.

IDLE1 Mode

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

Control Registers

The SMOD register and the FSYSON bit in the CTRL register are used to control the internal clocks within the device.

SMOD Register

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

Bit 7~5 **CKS2~CKS0**: System clock selection when HLCLK is "0"
000: Reserved
001: Reserved
010: $f_H/64$
011: $f_H/32$
100: $f_H/16$
101: $f_H/8$
110: $f_H/4$
111: $f_H/2$
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from f_H, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as "0"

Bit 3 **LTO**: Low speed oscillator ready flag
0: Not ready
1: Ready
This is the low speed system oscillator ready flag which indicates when the low speed oscillator is stable after power on reset or a wake-up has occurred.

Bit 2 **HTO**: High speed system oscillator ready flag
0: Not ready
1: Ready
This is the high speed system oscillator ready flag which indicates when the high speed system oscillator is stable. This flag is cleared to "0" by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable. Therefore this flag will always be read as "1" by the application program after device power-on.

Bit 1 **IDLEN**: IDLE Mode Control
0: Disable
1: Enable
This is the IDLE Mode Control bit and determines what happens when the HALT instruction is executed. If this bit is high, when a HALT instruction is executed the device will enter the IDLE Mode. In the IDLE1 Mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if FSYSON bit is high. If FSYSON bit is low, the CPU and the system clock will all stop in IDLE0 mode. If the bit is low the device will enter the SLEEP Mode when a HALT instruction is executed.

Bit 0 **HLCLK**: System Clock Selection
0: $f_H/2 \sim f_H/64$
1: f_H
This bit is used to select if the f_H clock or the $f_H/2 \sim f_H/64$ clock is used as the system clock. When the bit is high the f_H clock will be selected and if low the $f_H/2 \sim f_H/64$ clock will be selected.

CTRL Register

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

"x" unknown

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

Bit 6~3 Unimplemented, read as "0"

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

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

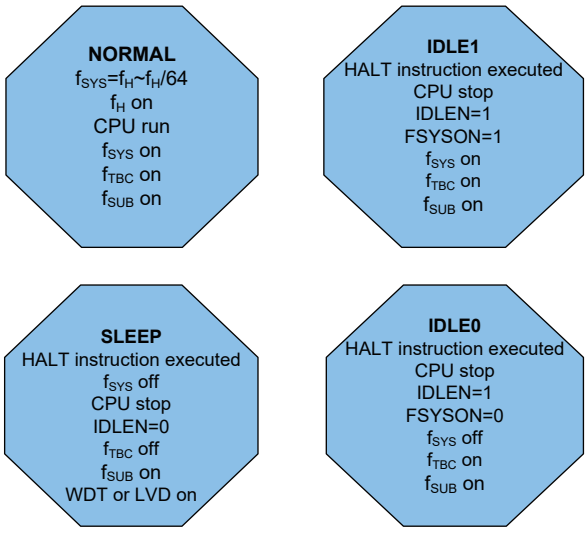
Bit 0 **WRF**: WDT Control register software reset flag
Describe elsewhere

Operating Mode Switching

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

In simple terms, Mode Switching from the NORMAL Mode to the SLEEP/IDLE Modes is executed via the HALT instruction. When an HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the condition of the IDLEN bit in the SMOD register and the FSYSON bit in the CTRL register.

When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~$f_H/64$. The accompanying flowchart shows what happens when the device moves between the various operating modes.



Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

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

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

Standby Current Considerations

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

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

In the IDLE1 Mode the sytem 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, it will take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

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

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

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

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

Watchdog Timer

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

Watchdog Timer Clock Source

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

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

Watchdog Timer Control Register

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

WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT function software control
01010 or 10101: Enable
Other values: Reset MCU
When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after 2~3 f_{LIRC} clock cycles and the WRF bit in the CTRL register will be set high.

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

CTRL Register

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

"x": unknown

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

Bit 6~3 Unimplemented, read as "0"

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

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

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

Watchdog Timer Operation

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

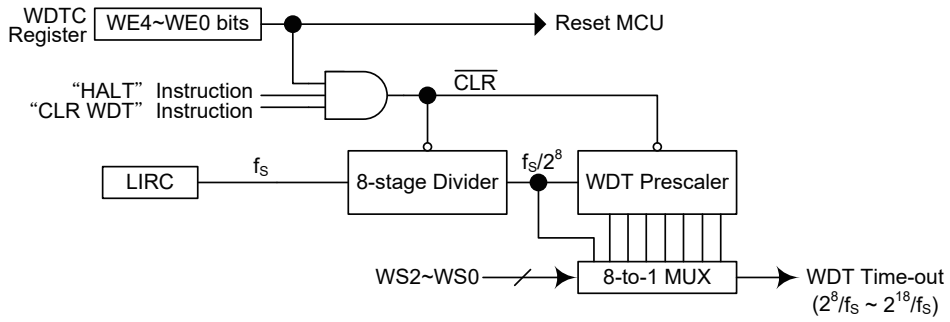
WE4~WE0 Bits	WDT Function
01010B or 10101B	Enable
Any other value	Reset MCU

Watchdog Timer Enable Control

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

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

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

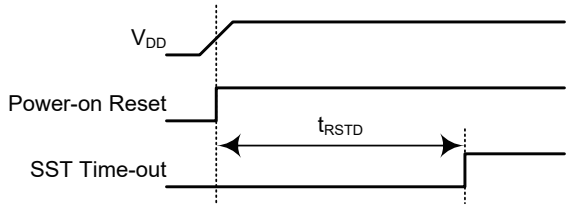
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

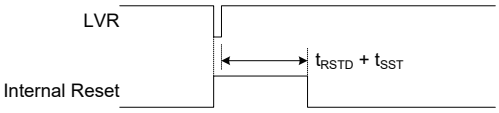


Power-on Reset Timing Chart

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 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 in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the CTRL register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVR/LVD 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 V_{LVR} value is fixed at 3.8V by setting the LVS bits in the LVRC register. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after 2~3 f_{LIRC} clock cycles. When this happens, the LRF bit in the CTRL register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the SLEEP/IDLE 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	0	1	0	1	0	1

Bit 7~0 **LVS7~LVS0**: LVR Voltage Select control
01010101: 3.8V
00110011: 3.8V
10011001: 3.8V
10101010: 3.8V
Any other value: Generates 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 2~3 f_{LIRC} clock cycles. In this situation the register contents will remain the same after such a reset occurs.

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

- **CTRL Register**

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

"x": unknown

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

Bit 6~3 Unimplemented, read as "0"

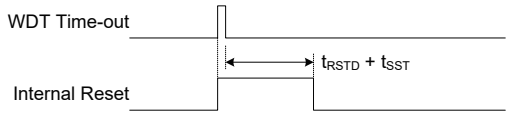
Bit 2 **LVRF**: LVR function reset flag
0: Not occur
1: Occurred
This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

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

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

Watchdog Time-out Reset during Normal Operation

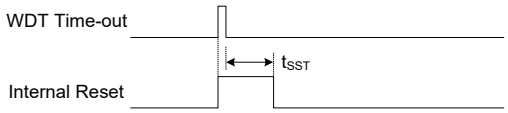
The Watchdog time-out Reset during normal operation is the same as a LVR reset except that the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



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

"u" stands for unchanged

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

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

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

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	xx1u uuuu	uu11 uuuu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
SMOD	000- 0011	000- 0011	uuu- uuuu
LVDC	--00 0000	--00 0000	--uu uuuu
LVRC	0101 0101	0101 0101	uuuu uuuu
WDTC	0101 0011	0101 0011	uuuu uuuu
INTEG1	---- 0000	---- 0000	---- 0000
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	0000 0000	0000 0000	uuuu uuuu
MFI0	-000 -000	-000 -000	-uuu —uuu
MFI1	0000 0000	0000 0000	uuuu uuuu
MFI2	--00 --00	--00 --00	--uu --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
MFI7	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PCPU	--00 0000	--00 0000	--uu uuuu
PC	--11 1111	--11 1111	--uu uuuu
PCC	--11 1111	--11 1111	--uu uuuu
PDPU	---- 0000	---- 0000	---- uuuu
PD	---- 1111	---- 1111	---- uuuu
PDC	---- 1111	---- 1111	---- 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

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
CAPTMDH	0000 0000	0000 0000	uuuu uuuu
CAPTMAL	0000 0000	0000 0000	uuuu uuuu
CAPTMAH	0000 0000	0000 0000	uuuu uuuu
CAPTMCL	xxxx xxxx	xxxx xxxx	uuuu uuuu
CAPTMCH	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=0)
			uu-- ---- (ADRFS=0, ADCRL_SEL=1)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=1)
ADRH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0, ADCRL_SEL=0)
			---- uuuu (ADRFS=1, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=0, ADCRL_SEL=1)
			---- --uu (ADRFS=1, ADCRL_SEL=1)
ADCR0	0110 0000	0110 0000	uuuu uuuu
ADCR1	0000 0000	0000 0000	uuuu uuuu
ADCR2	0000 0-00	0000 0-00	uuuu u-uu
ADISG1	0000 0000	0000 0000	uuuu uuuu
ADISG2	0000 0000	0000 0000	uuuu uuuu
ADDL	0000 0000	0000 0000	uuuu uuuu
ADLVDL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=0)
			uu-- ---- (ADRFS=0, ADCRL_SEL=1)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=1)
ADLVDH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0, ADCRL_SEL=0)
			---- uuuu (ADRFS=1, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=0, ADCRL_SEL=1)
			---- --uu (ADRFS=1, ADCRL_SEL=1)
ADHVDL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=0)
			uu-- ---- (ADRFS=0, ADCRL_SEL=1)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=1)

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
ADHVDH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0, ADCRL_SEL=0)
			---- uuuu (ADRFS=1, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=0, ADCRL_SEL=1)
			---- --uu (ADRFS=1, ADCRL_SEL=1)
PBPS0	0000 0000	0000 0000	uuuu uuuu
PBPS1	0000 0000	0000 0000	uuuu uuuu
OCPS	--00 0000	--00 0000	--uu uuuu
INTEG0	-000 0000	-000 0000	-uuu uuuu
CTRL	0--- -x00	0--- -x00	u--- -uuu
HDCR	0001 0000	0001 0000	uuuu uuuu
HDCD	---- -000	---- -000	---- -uuu
MPTC1	0000 00-0	0000 00-0	uuuu uu-u
MPTC2	---0 0000	---0 0000	---u uuuu
DUTR3L	0000 0000	0000 0000	uuuu uuuu
DUTR3H	---- --00	---- --00	---- --uu
PTM0C0	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	uuuu uuuu
PTM0DL	0000 0000	0000 0000	uuuu uuuu
PTM0DH	0000 0000	0000 0000	uuuu uuuu
PTM0AL	0000 0000	0000 0000	uuuu uuuu
PTM0AH	0000 0000	0000 0000	uuuu uuuu
PTM0RPL	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	0000 0000	0000 0000	uuuu uuuu
PWMC	0000 0000	0000 0000	uuuu uuuu
DUTR0L	0000 0000	0000 0000	uuuu uuuu
DUTR0H	---- --00	---- --00	---- --uu
DUTR1L	0000 0000	0000 0000	uuuu uuuu
DUTR1H	---- --00	---- --00	---- --uu
DUTR2L	0000 0000	0000 0000	uuuu uuuu
DUTR2H	---- --00	---- --00	---- --uu
PRDRL	0000 0000	0000 0000	uuuu uuuu
PRDRH	---- --00	---- --00	---- --uu
PWMRL	0000 0000	0000 0000	uuuu uuuu
PWMRH	---- --00	---- --00	---- --uu
PWMME	--00 0000	--00 0000	--uu uuuu
PWMMD	--00 0000	--00 0000	--uu uuuu
MCF	0--- 0100	0--- 0100	u--- uuuu
MCD	--00 0xxx	--00 0xxx	--uu uuuu
DTS	0000 0000	0000 0000	uuuu uuuu
PLC	--00 0000	--00 0000	--uu uuuu
IICC0	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	0000 000-	0000 000-	uuuu uuu-

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IICTOC	0000 0000	0000 0000	uuuu uuuu
USR	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	uuuu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
CMPC	1111 0000	1111 0000	uuuu uuuu
MDU0R0	xxxx xxxx	xxxx xxxx	xxxx xxxx
MDU0R1	xxxx xxxx	xxxx xxxx	xxxx xxxx
MDU0CTRL	---- -000	---- -000	---- -uuu
MDU1R0	xxxx xxxx	xxxx xxxx	xxxx xxxx
MDU1R1	xxxx xxxx	xxxx xxxx	xxxx xxxx
MDU1R2	xxxx xxxx	xxxx xxxx	xxxx xxxx
MDU1R3	xxxx xxxx	xxxx xxxx	xxxx xxxx
MDU1R4	xxxx xxxx	xxxx xxxx	xxxx xxxx
MDU1R5	xxxx xxxx	xxxx xxxx	xxxx xxxx
MDU1CTRL	00-- ----	00-- ----	uu-- ----
NF_VIH	00-1 1001	00-1 1001	uu-u uuuu
NF_VIL	00-0 1010	00-0 1010	uu-u uuuu
TBC	0011 ----	0011 ----	uuuu ----
Pri_name0	0010 0001	0010 0001	uuuu uuuu
Pri_ name1	0100 0011	0100 0011	uuuu uuuu
Pri_ name2	0110 0101	0110 0101	uuuu uuuu
Pri_ name3	1000 0111	1000 0111	uuuu uuuu
Pri_ name4	10101001	10101001	uuuu uuuu
Pri_ name5	1100 1011	1100 1011	uuuu uuuu
Pri_ name6	1110 1101	1110 1101	uuuu uuuu
Pri_ name7	---- 1111	---- 1111	---- uuuu
PTM0C2	---- -000	---- -000	---- -uuu
PTM1C2	---- -000	---- -000	---- -uuu
PTM2C2	---- -000	---- -000	---- -uuu
PTM3C2	---- -000	---- -000	---- -uuu
PTM0BL	0000 0000	0000 0000	uuuu uuuu
PTM0BH	0000 0000	0000 0000	uuuu uuuu
PTM1BL	0000 0000	0000 0000	uuuu uuuu
PTM1BH	0000 0000	0000 0000	uuuu uuuu
PTM2BL	0000 0000	0000 0000	uuuu uuuu
PTM2BH	---- --00	---- --00	---- --uu
PTM3BL	0000 0000	0000 0000	uuuu uuuu
PTM3BH	---- --00	---- --00	---- --uu
ISRL0	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=0)
			uu-- ---- (ADRFS=0, ADCRL_SEL=1)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=1)

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
ISRH0	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0, ADCRL_SEL=0)
			---- uuuu (ADRFS=1, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=0, ADCRL_SEL=1)
			---- --uu (ADRFS=1, ADCRL_SEL=1)
ISRL1	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=0)
			uu-- ---- (ADRFS=0, ADCRL_SEL=1)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=1)
ISRH1	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0, ADCRL_SEL=0)
			---- uuuu (ADRFS=1, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=0, ADCRL_SEL=1)
			---- --uu (ADRFS=1, ADCRL_SEL=1)
ISRL2	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=0)
			uu-- ---- (ADRFS=0, ADCRL_SEL=1)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=1)
ISRH2	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0, ADCRL_SEL=0)
			---- uuuu (ADRFS=1, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=0, ADCRL_SEL=1)
			---- --uu (ADRFS=1, ADCRL_SEL=1)
ISRL3	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=0)
			uu-- ---- (ADRFS=0, ADCRL_SEL=1)
			uuuu uuuu (ADRFS=1, ADCRL_SEL=1)

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
ISRH3	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0, ADCRL_SEL=0)
			---- uuuu (ADRFS=1, ADCRL_SEL=0)
			uuuu uuuu (ADRFS=0, ADCRL_SEL=1)
			---- --uu (ADRFS=1, ADCRL_SEL=1)
ADBYPS	0-00 0000	0-00 0000	u-uu uuuu
ADCR3	---- -000	---- -000	---- -uuu
PTM1C0	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	uuuu uuuu
PTM1DL	0000 0000	0000 0000	uuuu uuuu
PTM1DH	0000 0000	0000 0000	uuuu uuuu
PTM1AL	0000 0000	0000 0000	uuuu uuuu
PTM1AH	0000 0000	0000 0000	uuuu uuuu
PTM1RPL	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	0000 0000	0000 0000	uuuu uuuu
PWMCS	---- -000	---- -000	---- -uuu
OPA1CAL	0--- ----	0--- ----	u--- ----
OPA2CAL	0--- ----	0--- ----	u--- ----
HCHK_NUM	---0 0000	---0 0000	---u uuuu
HNF_MSEL	---- 0000	---- 0000	---- uuuu
PTM2C0	0000 0---	0000 0---	uuuu u---
PTM2C1	0000 0000	0000 0000	uuuu uuuu
PTM2DL	0000 0000	0000 0000	uuuu uuuu
PTM2DH	---- --00	---- --00	---- --uu
PTM2AL	0000 0000	0000 0000	uuuu uuuu
PTM2AH	---- --00	---- --00	---- --uu
PTM2RPL	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	---- --00	---- --00	---- --uu
HDCT0	--00 0000	--00 0000	--uu uuuu
HDCT1	--00 0000	--00 0000	--uu uuuu
HDCT2	--00 0000	--00 0000	--uu uuuu
HDCT3	--00 0000	--00 0000	--uu uuuu
HDCT4	--00 0000	--00 0000	--uu uuuu
HDCT5	--00 0000	--00 0000	--uu uuuu
HDCT6	--00 0000	--00 0000	--uu uuuu
HDCT7	--00 0000	--00 0000	--uu uuuu
HDCT8	--00 0000	--00 0000	--uu uuuu
HDCT9	--00 0000	--00 0000	--uu uuuu
HDCT10	--00 0000	--00 0000	--uu uuuu
HDCT11	--00 0000	--00 0000	--uu uuuu
OPOMS	00-- -000	00-- -000	uu-- -uuu
OPCM	0000 0000	0000 0000	uuuu uuuu
OPA0CAL	-001 0000	-001 0000	-uuu uuuu
PTM3C0	0000 0---	0000 0---	uuuu u---
PTM3C1	0000 0000	0000 0000	uuuu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PTM3DL	0000 0000	0000 0000	uuuu uuuu
PTM3DH	---- --00	---- --00	---- --uu
PTM3AL	0000 0000	0000 0000	uuuu uuuu
PTM3AH	---- --00	---- --00	---- --uu
PTM3RPL	0000 0000	0000 0000	uuuu uuuu
PTM3RPH	---- --00	---- --00	---- --uu
PAPS0	0000 0000	0000 0000	uuuu uuuu
PAPS1	0000 0000	0000 0000	uuuu uuuu
PCPS0	0000 0000	0000 0000	uuuu uuuu
PCPS1	---- 0000	---- 0000	---- uuuu
PDPS0	0000 0000	0000 0000	uuuu uuuu
PRM	0000 0000	0000 0000	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~PD. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A, [m]", where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	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	PAW3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	—	PC5	PC4	PC3	PC2	PC1	PC0
PCC	—	—	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	—	—	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	—	—	—	—	PD3	PD2	PD1	PD0
PDC	—	—	—	—	PDC3	PDC2	PDC1	PDC0
PDPU	—	—	—	—	PDPU3	PDPU2	PDPU1	PDPU0

"—": Unimplemented, read as "0"

I/O Logic Function Registers List

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using registers, namely PAPU~PDPU, 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 an 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, B, C and D. However, the actual available bits for each I/O Port may be different.

Port A Wake-up

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

Note that the wake-up function can be controlled by the wake-up control registers only when the pin-shared functional pin is selected as general purpose input/output and the MCU enters the power down mode.

PAWU Register

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

Bit 7~0 **PAWU7~PAWU0**: PA7~PA0 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~PDC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

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 type selection

0: Output

1: Input

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

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

When the pin-shared input function is selected to be used, the copprsponding input and output functions selection should be properly managed. For example, if the I^2C SDA line is used, the corresponding output pin-shared function should be configured as the SDA function by configuring the PxPSn register and the SDA signal input should be properly selected using the PRM register. Note that the UART TX output pin function must also be properly configured using the PxPSn and PRM registers. If the external interrupt function is selected to be used, the relevant output pin-shared function should be selected as an I/O function and the interrupt input signal should be selected.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital

input pins, such as INTn, TCKn, etc, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bit fields. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be setup as an input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAPS0	PA3S1	PA3S0	PA2S1	PA2S0	PA1S1	PA1S0	PA0S1	PA0S0
PAPS1	PA7S1	PA7S0	PA6S1	PA6S0	PA5S1	PA5S0	PA4S1	PA4S0
PBPS0	PB3S1	PB3S0	PB2S1	PB2S0	PB1S1	PB1S0	PB0S1	PB0S0
PBPS1	PB7S1	PB7S0	PB6S1	PB6S0	PB5S1	PB5S0	PB4S1	PB4S0
PCPS0	PC3S1	PC3S0	PC2S1	PC2S0	PC1S1	PC1S0	PC0S1	PC0S0
PCPS1	—	—	—	—	PC5S1	PC5S0	PC4S1	PC4S0
PDPS0	PD3S1	PD3S0	PD2S1	PD2S0	PD1S1	PD1S0	PD0S1	PD0S0
PRM	NFINPS	INT1PS	RXPS	TXPS	C1NPS1	C1NPS0	SDAPS	SCLPS

Pin-shared Function Selection Registers List

- **PAPS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PA3S1	PA3S0	PA2S1	PA2S0	PA1S1	PA1S0	PA0S1	PA0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PA3S1~PA3S0**: PA3 Pin-Shared function selection
00: PA3/TCK1/H1
01: C1P
10: PA3/TCK1/H1
11: PA3/TCK1/H1

Bit 5~4 **PA2S1~PA2S0**: PA2 Pin-Shared function selection
00: PA2
01: SCL
10: TX
11: PA2
Note that to select this TX pin function, the corresponding output source selection bit in the PRM register should be set to 1.

Bit 3~2 **PA1S1~PA1S0**: PA1 Pin-Shared function selection
00: PA1/TCK2
01: AP/AN3
10: PA1/TCK2
11: PA1/TCK2

Bit 1~0 **PA0S1~PA1S0**: PA0 Pin-Shared function selection
00: PA0
01: SDA
10: RX
11: PA0

- **PAPS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PA7S1	PA7S0	PA6S1	PA6S0	PA5S1	PA5S0	PA4S1	PA4S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PA7S1~PA7S0**: PA7 Pin-Shared function selection
00: PA7/NFIN
01: AN6
10: PA7/NFIN
11: OPA2O

Bit 5~4 **PA6S1~PA6S0**: PA6 Pin-Shared function selection
00: PA6
01: C1N
10: AN7
11: OPA1O

Bit 3~2 **PA5S1~PA5S0**: PA5 Pin-Shared function selection
00: PA5/H3
01: PA5/H3
10: C3P
11: PA5/H3

Bit 1~0 **PA4S1~PA4S0**: PA4 Pin-Shared function selection
00: PA4/H2
01: PA4/H2
10: C2P
11: C1N

- **PBPS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PB3S1	PB3S0	PB2S1	PB2S0	PB1S1	PB1S0	PB0S1	PB0S0
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 **PB3S1~PB3S0**: PB3 Pin-Shared function selection
00: PB3
01: C1N
10: CPN
11: PB3
When these bits are set to 10B, the C1N, C2N and C3N will be shorted internally and the CPN pin function is selected. Refer to the Comparators chapter for more details about this option.

Bit 5~4 **PB2S1~PB2S0**: PB2 Pin-Shared function selection
00: PB2
01: HCO
10: TP3_1
11: SDA

Bit 3~2 **PB1S1~PB1S0**: PB1 Pin-Shared function selection
00: PB1/CTIN
01: HBO
10: PB1/CTIN
11: SCL

Bit 1~0 **PB0S1~PB0S0**: PB0 Pin-Shared function selection
00: PB0/INT1/TCK3/NFIN
01: HAO
10: PB0/INT1/TCK3/NFIN

11: TP3_0

- **PBPS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PB7S1	PB7S0	PB6S1	PB6S0	PB5S1	PB5S0	PB4S1	PB4S0
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 **PB7S1~PB7S0**: PB7 Pin-Shared function selection
00: PB7
01: TX
10: TP2_1
11: AN2
Note that to select this TX pin function, the corresponding output source selection bit in the PRM register should be cleared to 0.

Bit 5~4 **PB6S1~PB6S0**: PB6 Pin-Shared function selection
00: PB6
01: RX
10: TP2_0
11: OPA0O

Bit 3~2 **PB5S1~PB5S0**: PB5 Pin-Shared function selection
00: PB5
01: TP2_1
10: C2N
11: PB5

Bit 1~0 **PB4S1~PB4S0**: PB4 Pin-Shared function selection
00: PB4
01: TP2_0
10: C3N
11: PB4

- **PCPS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PC3S1	PC3S0	PC2S1	PC2S0	PC1S1	PC1S0	PC0S1	PC0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PC3S1~PC3S0**: PC3 Pin-Shared function selection
00: PC3
01: PC3
10: GBB
11: PC3

Bit 5~4 **PC2S1~PC2S0**: PC2 Pin-Shared function selection
00: PC2
01: PC2
10: GBT
11: PC2

Bit 3~2 **PC1S1~PC1S0**: PC1 Pin-Shared function selection
00: PC1
01: PC1
10: GAB
11: PC1

Bit 1~0 **PC0S1~PC0S0**: PC0 Pin-Shared function selection
00: PC0
01: PC0
10: GAT
11: PC0

- **PCPS1 Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **PC5S1~PC5S0**: PC5 Pin-Shared function selection
00: PC5
01: PC5
10: GCB
11: PC5

Bit 1~0 **PC4S1~PC4S0**: PC4 Pin-Shared function selection
00: PC4
01: PC4
10: GCT
11: PC4

- **PDPS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PD3S1	PD3S0	PD2S1	PD2S0	PD1S1	PD1S0	PD0S1	PD0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PD3S1~PD3S0**: PD3 Pin-Shared function selection
00: PD3
01: TP1_1
10: PD3
11: OPA2P

Bit 5~4 **PD2S1~PD2S0**: PD2 Pin-Shared function selection
00: PD2/INT1
01: TP1_0
10: PD2/INT1
11: OPA2N

Bit 3~2 **PD1S1~PD1S0**: PD1 Pin-Shared function selection
00: PD1
01: AN1
10: TP0_1
11: OPA1P

Bit 1~0 **PD0S1~PD0S0**: PD0 Pin-Shared function selection
00: PD0/TCK0
01: AN0
10: TP0_0
11: OPA1N

- **PRM Register**

Bit	7	6	5	4	3	2	1	0
Name	NFINPS	INT1PS	RXPS	TXPS	C1NPS1	C1NPS0	SDAPS	SCLPS
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 **NFINPS**: NFIN input source pin selection
0: PA7
1: PB0

Bit 6 **INT1PS**: INT1 input source pin selection
0: PB0
1: PD2

Bit 5 **RXPS**: RX input source pin selection
0: PB6
1: PA0

Bit 4 **TXPS**: TX output source pin selection
0: PB7
1: PA2

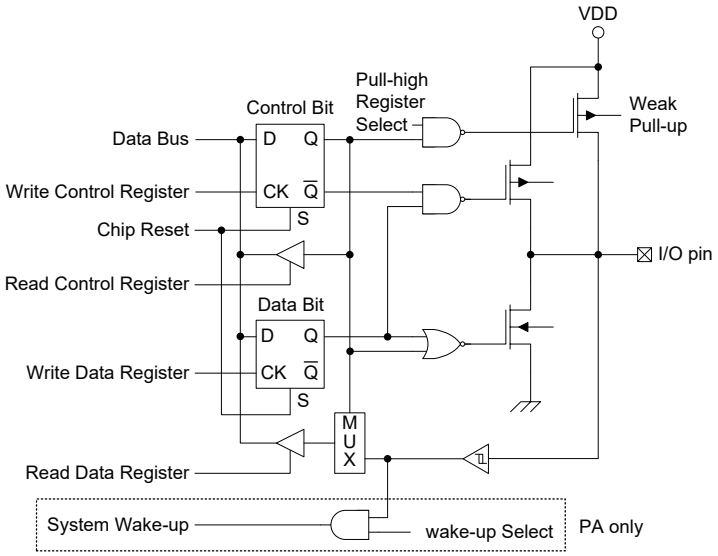
Bit 3~2 **C1NPS1~C1NPS0**: C1N input source pin selection
00: PB3
01: PA4
10: PA6
11: Reserved

Bit 1 **SDAPS**: SDA input source pin selection
0: PA0
1: PB2

Bit 0 **SCLPS**: SCL input source pin selection
0: PA2
1: PB1

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 port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

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

Timer Modules – TM

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

Introduction

The device contains four TMs with each TM having a reference name of PTMn. The PTM0 and PTM1 are 16-bit Periodic Type TMs while the PTM2 and PTM3 are 10-bit Periodic Type TMs. The common features to the 16-bit and 10-bit Periodic TMs will be described in this section and the detailed operation will be described in the Periodic Type TMs section. The main features of TMs are summarised in the accompanying table.

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

TM Function Summary

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

TM Name/Type Reference

TM Operation

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

TM Clock Source

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

TM Interrupts

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

TM External Pins

Each of the TMs has one TM input pin, with the label TCKn. The PTMn input pin, TCKn, is essentially a clock source for the PTMn and is selected using the PTnCK2~PTnCK0 bits in the PTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The TCKn input pin can be chosen to have either a rising or falling active edge. The TCKn pin is also used as the external trigger input pin in single pulse output mode.

The TMs each have two output pin with the label TPn_0 and TPn_1. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external TPn_0 and TPn_1 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 Function section.

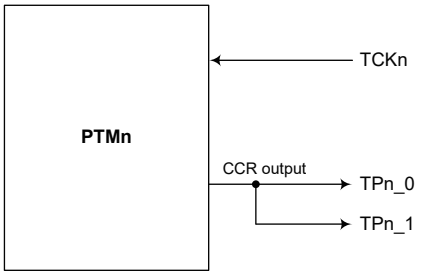
The TCKn and TPn_0 pins are also the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTnIO1~PTnIO0 bits in the PTMnC1 register. The capture input coming from TCKn or TPn_0 is determined by the PTnCAPTS bit in the PTMnC1 register.

PTM0		PTM1		PTM2		PTM3	
Input	Output	Input	Output	Input	Output	Input	Output
TCK0, TP0_0	TP0_0, TP0_1	TCK1, TP1_0	TP1_0, TP1_1	TCK2, TP2_0	TP2_0, TP2_1	TCK3, TP3_0	TP3_0, TP3_1

TM External Pins

TM Input/Output Pin Selection

Selecting to have a TM input/output or whether to retain its other shared function is implemented using the relevant pin-shared function selection registers, with the corresponding selection bits in each pin-shared function register corresponding to a TM input/output pin. Configuring the selection bits correctly will setup the corresponding pin as a TM input/output. The details of the pin-shared function selection are described in the pin-shared function section.

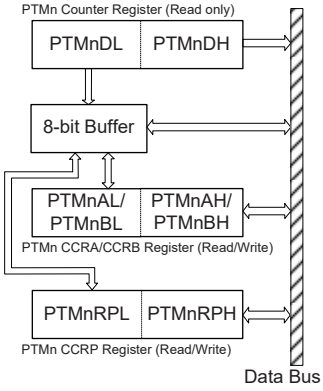


PTM Function Pin Block Diagram (n=0~3)

Programming Considerations

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

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



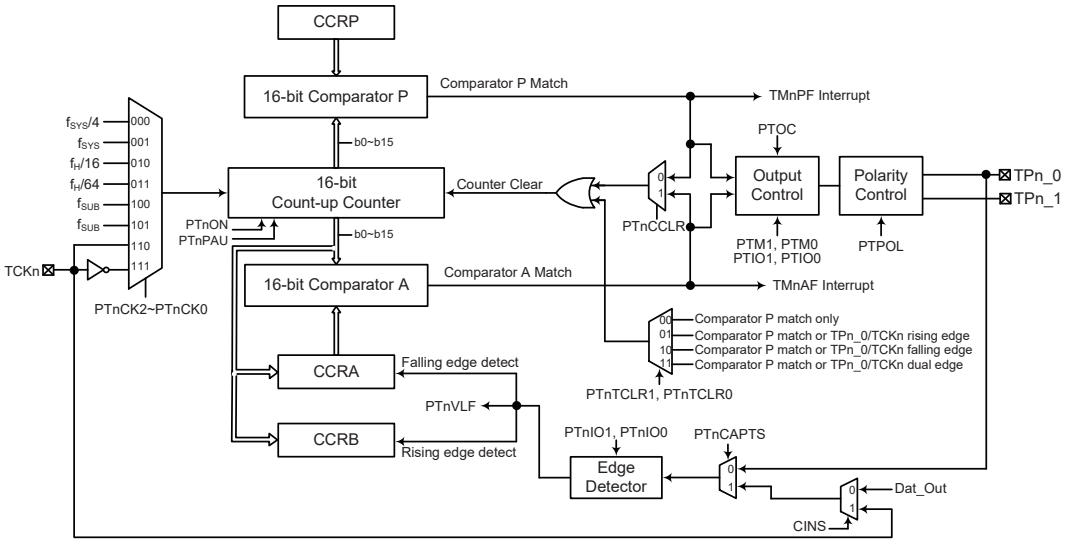
The following steps show the read and write procedures:

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

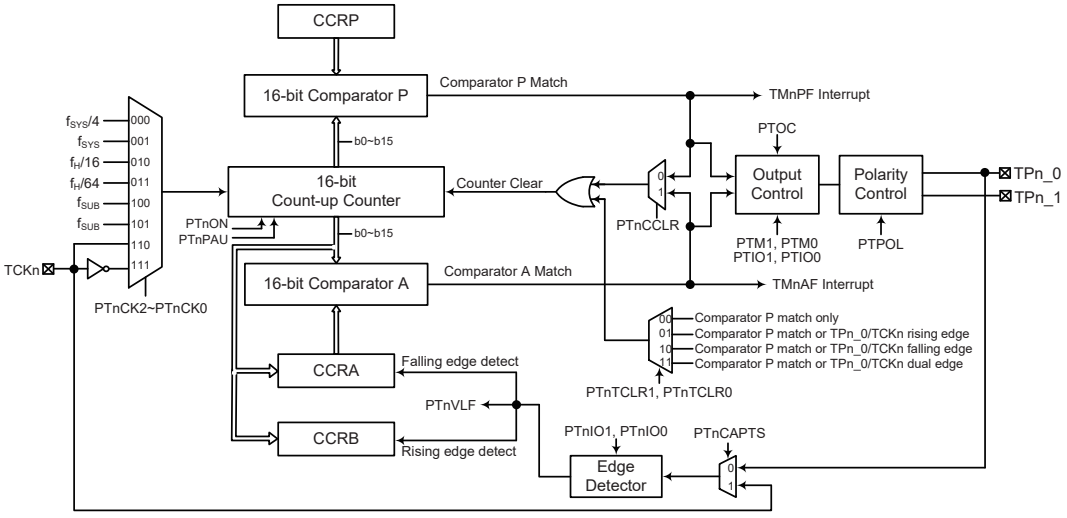
Periodic Type TM – PTM

The Periodic Type TM contains five operating modes, which are Compare Match Output, Timer/ Event Counter, Capture Input, Single Pulse Output and PWM Output modes. The Periodic TM can be controlled with one or two external input pins and can drive one or two external output pins. Note that the TM input and output pin functions should be selected by proper configuration which is described in the Pin-shared Functions section.

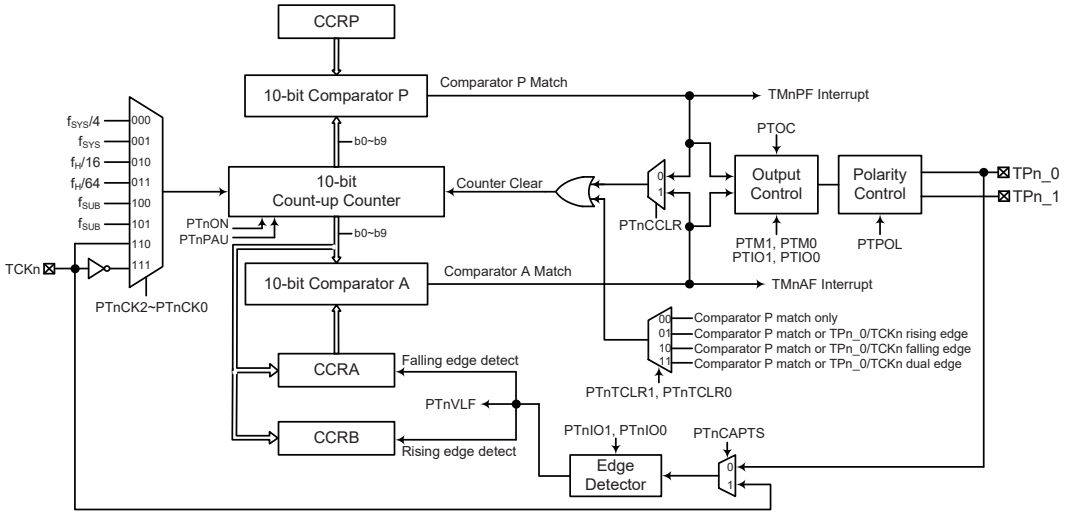
Name	TM No.	TM Input Pin	TM Output Pin
16-bit PTM	0, 1	TCK0, TP0_0; TCK1, TP1_0	TP0_0, TP0_1; TP1_0, TP1_1
10-bit PTM	2, 3	TCK2, TP2_0; TCK3, TP3_0	TP2_0, TP2_1; TP3_0, TP3_1



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



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



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

Periodic TM Operation

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

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

Periodic Type TM Register Description

Overall operation of the Periodic Type TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10-bit or 16-bit value, while three read/write register pairs exist to store the internal 10-bit or 16-bit CCRA value, CCRP value and CCRB 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
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnC2	—	—	—	—	—	PTnTCLR1	PTnTCLR0	PTnVLF
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnBL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnBH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	D15	D14	D13	D12	D11	D10	D9	D8

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnC2	—	—	—	—	—	PTnTCLR1	PTnTCLR0	PTnVLF
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnBL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnBH	—	—	—	—	—	—	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

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

PTMnC0 Register (n=0~3)

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

Bit 7 **PTnPAU**: PTMn Counter Pause Control
0: Run
1: Pause

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

Bit 6~4 **PTnCK2~PTnCK0**: Select PTMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: TCKn rising edge clock
111: TCKn falling edge clock

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

Bit 3 **PTnON**: PTMn Counter On/Off Control
0: Off
1: On

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

If the PTMn is in the Compare Match Output Mode, PWM output Mode or Single Pulse Output Mode then the PTMn output pin will be reset to its initial condition, as specified by the PTnOC bit, when the PTnON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

PTMnC1 Register (n=0~3)

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PTnM1~PTnM0**: Select PTMn Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

Bit 5~4 **PTnIO1~PTnIO0**: Select PTMn external pin function

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

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

Capture Input Mode

PTnTCLR[1:0]=00B:

00: Input capture at rising edge of TPn_0 or TCKn and the counter value will be latched into CCRA
01: Input capture at falling edge of TPn_0 or TCKn and the counter value will be latched into CCRA
10: Input capture at falling/rising edge of TPn_0 or TCKn and the counter value will be latched into CCRA
11: Input capture disabled

PTnTCLR[1:0]=01B or 10B or 11B:
00: Input capture at rising edge of TPn_0 or TCKn and the counter value will be latched into CCRB
01: Input capture at falling edge of TPn_0 or TCKn and the counter value will be latched into CCRA
10: Input capture at falling/rising edge of TPn_0 or TCKn and the counter value will be latched into CCRA at falling edge and into CCRB at rising edge
11: Input capture disabled

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn output pin changes state when a compare match occurs from the Comparator A. The PTMn output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the PTMn output pin should be setup using the PTnOC bit in the PTMnC1 register. Note that the output level requested by the PTnIO1 and PTnIO0 bits must be different from the initial value setup using the PTnOC bit otherwise no change will occur on the PTMn output pin when a compare match occurs. After the PTMn output pin changes state, it can be reset to its initial level by changing the level of the PTnON bit from low to high.

In the PWM Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the PTnIO1 and PTnIO0 bits only after the PTMn has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the PTMn is running.

In the Capture Input Mode, the capture input trigger source is selected by the PTnCAPTS bit in the PTMnC1 register for PTM1, PTM2 and PTM3. However, the capture input trigger source is also determined by the CINS bit in the NF_VIH register together with the PT0CAPTS bit in the PTM0C1 register for PTM0.

Bit 3 **PTnOC**: PTMn Output control bit

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

PWM Mode/Single Pulse Output Mode
0: Active low
1: Active high

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

Bit 2 **PTnPOL**: PTMn Output polarity Control
0: Non-invert
1: Invert
This bit controls the polarity of the output pins. When the bit is set high the PTMn output pin will be inverted and not inverted when the bit is zero. It has no effect if the PTMn is in the Timer/Counter Mode.

Bit 1 **PTnCAPTS**: PTMn Capture Trigger Source Selection
0: From TPn_0 pin
1: From TCKn pin
This bit is used to select the PTMn capture input trigger source. However for PTM0, the capture trigger source is also determined by the CINS bit in the NF_VIH register. When the PT0CAPTS bit is set to 1, the capture input trigger source can be TCK0 or noise filtered Dat_Out signal determined using the CINS bit.

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

PTMnC2 Register (n=0~3)

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

Bit 7~3 Unimplemented, read as "0"

Bit 2~1 **PTnTCLR1~PTnTCLR0**: Select PTMn Counter clear condition in capture input mode only
00: Comparator P match clear only
01: Comparator P match clear or TPn_0/TCKn rising edge clear
10: Comparator P match clear or TPn_0/TCKn falling edge clear
11: Comparator P match clear or TPn_0/TCKn dual edge clear

Bit 0 **PTnVLF**: PTMn Counter value latch trigger edge flag
0: Falling edge trigger the counter value latch
1: Rising edge trigger the counter value latch
When the PTnTCLR1~PTnTCLR0 bits equal to 00B, ignore this flag status.

PTMnDL Register (n=0~3)

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

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

PTMnDH Register (n=0~1)

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

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

PTMnDH Register (n=2~3)

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

Bit 7~2 Unimplemented, read as "0"
Bit 1~0 **D9~D8**: PTMn Counter High Byte Register bit 1 ~ bit 0
PTMn 10-bit Counter bit 9 ~ bit 8

PTMnAL Register (n=0~3)

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

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

PTMnAH Register (n=0~1)

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

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

PTMnAH Register (n=2~3)

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

Bit 7~2 Unimplemented, read as "0"
Bit 1~0 **D9~D8**: PTMn CCRA High Byte Register bit 1 ~ bit 0
PTMn 10-bit CCRA bit 9 ~ bit 8

PTMnBL Register (n=0~3)

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

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

PTMnBH Register (n=0~1)

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

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

PTMnBH Register (n=2~3)

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

Bit 7~2 Unimplemented, read as "0"

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

PTMnRPL Register (n=0~3)

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

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

PTMnRPH Register (n=0~1)

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

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

PTMnRPH Register (n=2~3)

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

Bit 7~2 Unimplemented, read as "0"

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

Periodic Type TM Operating Modes

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

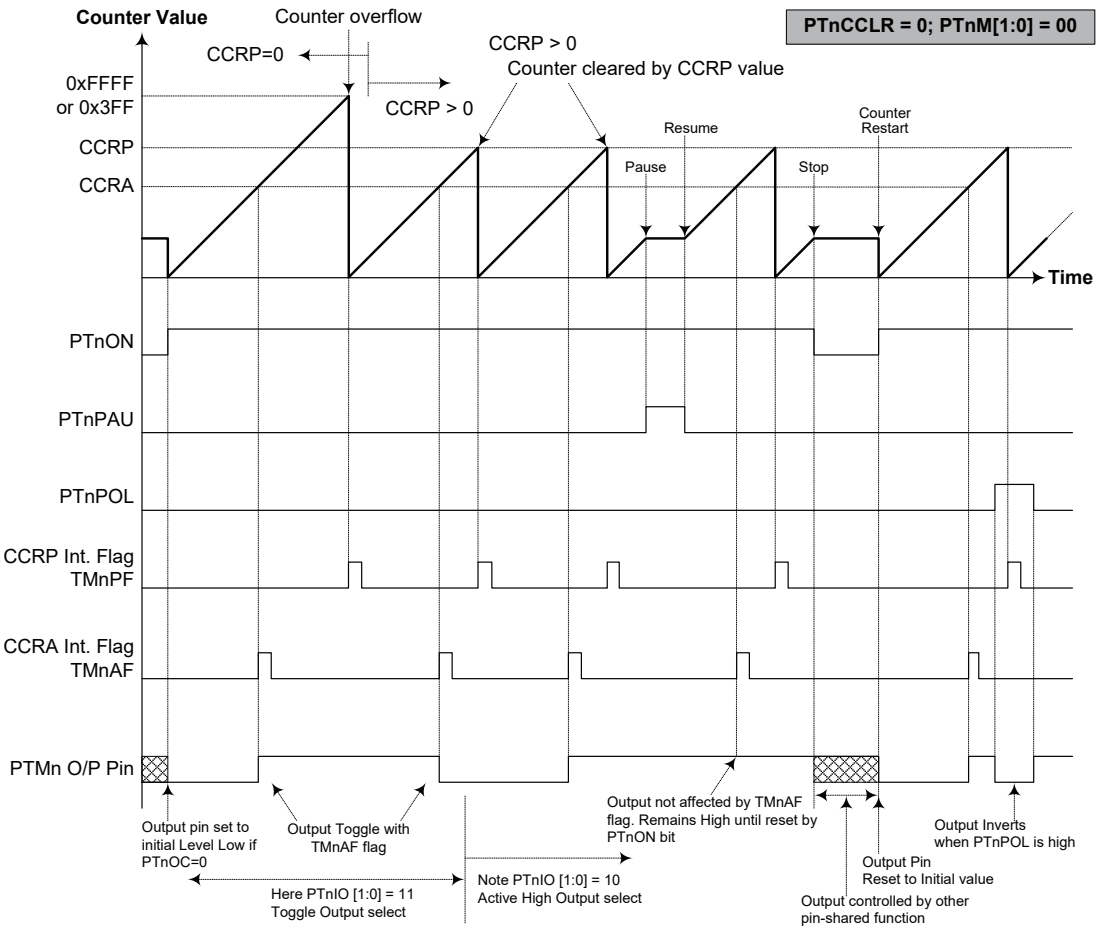
Compare Match Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the PTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both TMnAF and TMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

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

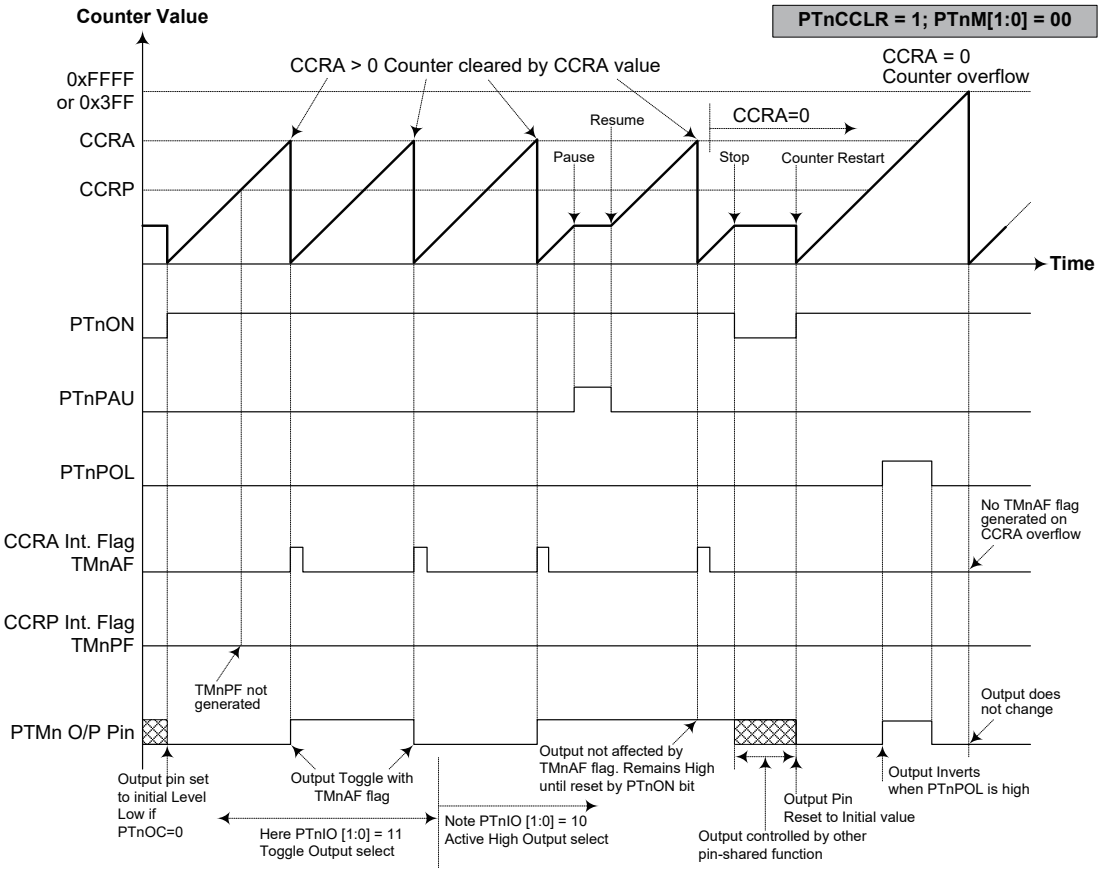
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, or 16-bit, FFFF Hex, value, however here the TMnAF interrupt request flag will not be generated.

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



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

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



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

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

Timer/Counter Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the PTMn output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the TM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively. The PWM function within the PTMn is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the PTMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

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

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The PTnOC bit in the PTMnC1 register is used to select the required polarity of the PWM waveform while the two PTnIO1 and PTnIO0 bits are used to enable the PWM output or to force the PTMn output pin to a fixed high or low level. The PTnPOL bit is used to reverse the polarity of the PWM output waveform.

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

CCRP	1~65535	0
Period	1~65535	1024
Duty	CCRA	

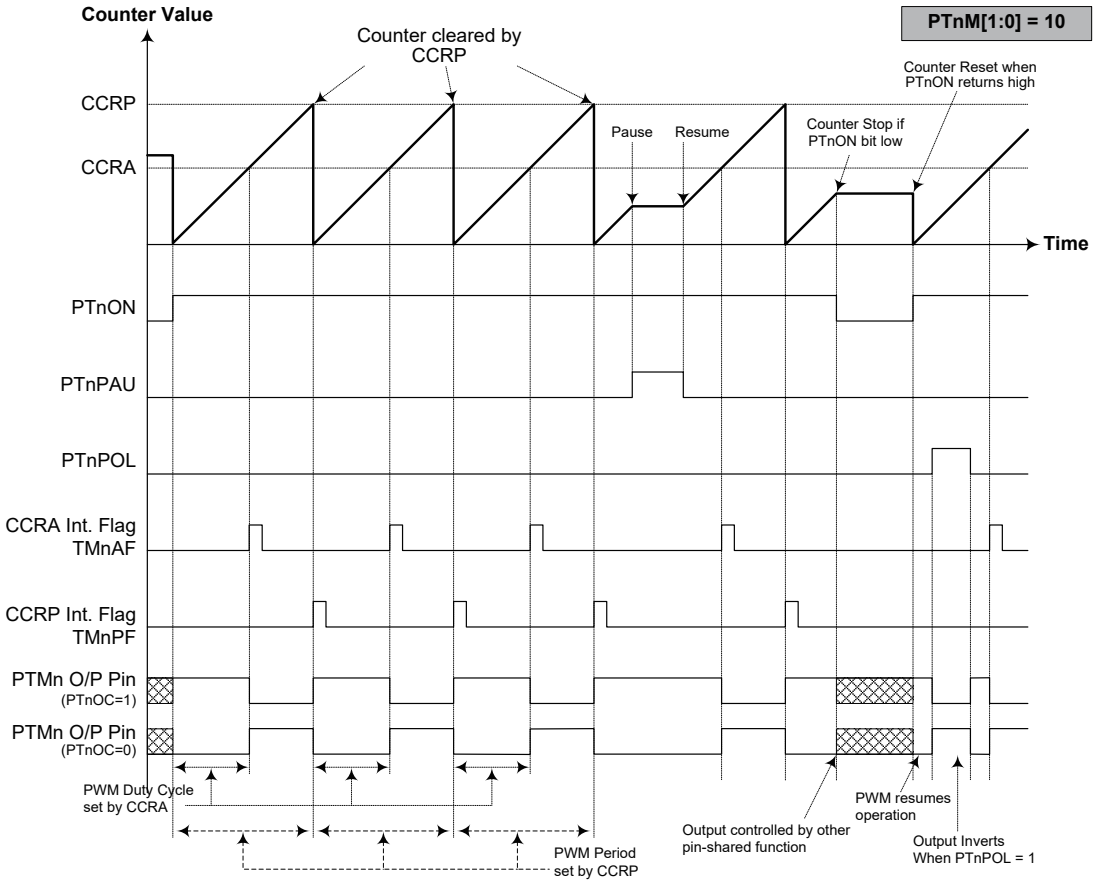
- **10-bit PTMn, PWM Output Mode, Edge-aligned Mode**

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

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

The PTMn PWM output frequency=($f_{SYS}/4$)/512=$f_{SYS}/2048$=1.9531kHz, 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 (n=0~3)

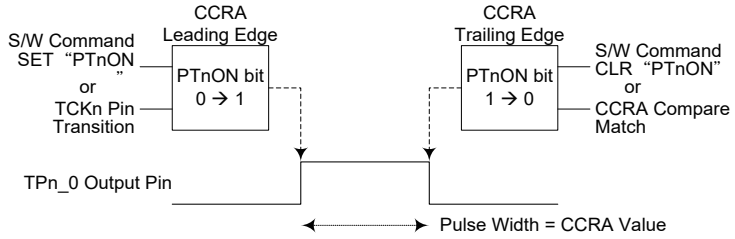
Note: 1. Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTnIO[1:0]=00 or 01
4. The PTnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

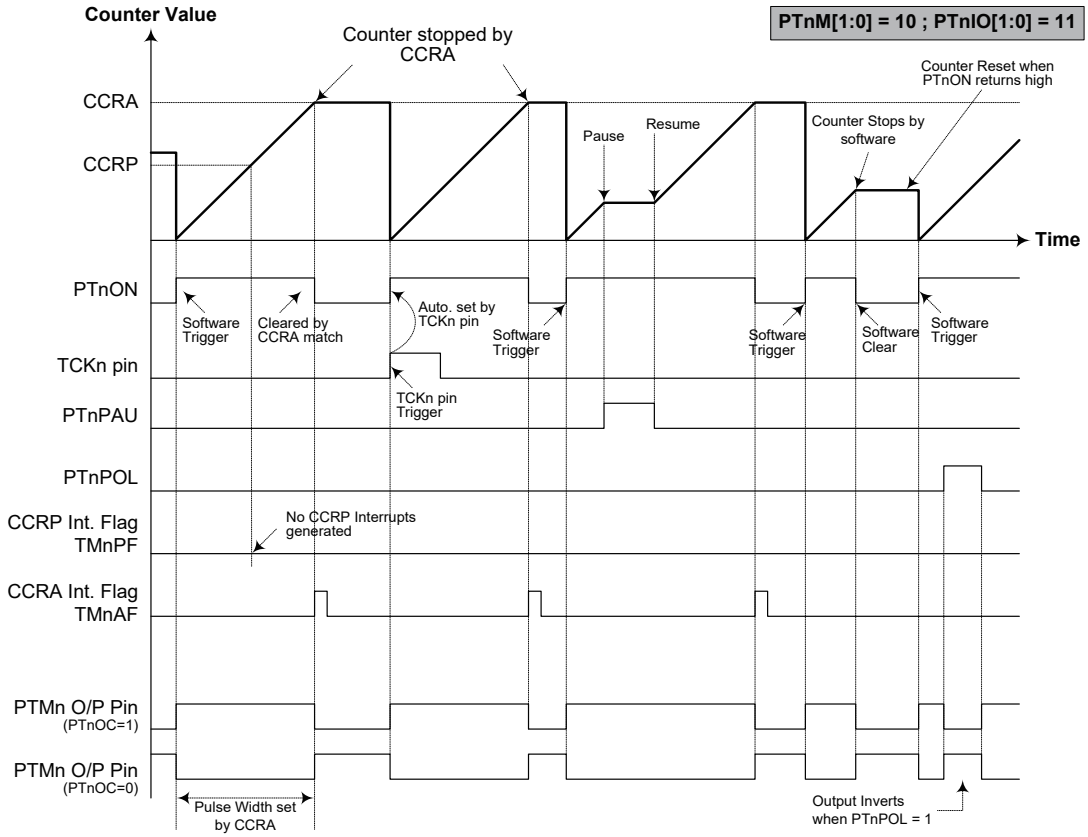
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

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

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



Single Pulse Generation (n=0~3)



Single Pulse Mode (n=0~3)

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

Capture Input Mode

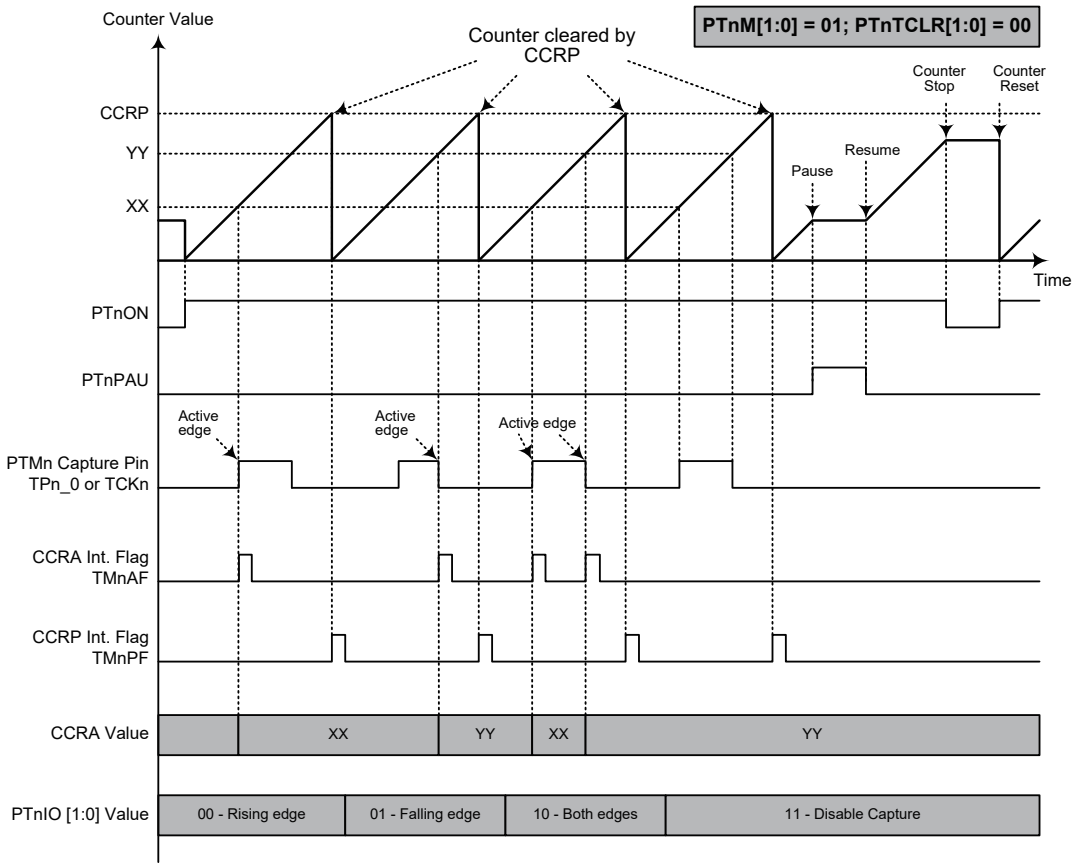
To select this mode bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the TPn_0 or TCKn pin which is selected using the PTnCAPTS bit in the PTMnC1 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The counter is started when the PTnON bit changes from low to high which is initiated using the application program.

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

When the required edge transition appears on the TPn_0 or TCKn pin the present value in the counter will be latched into the CCRA or CCRB registers and a PTMn interrupt generated. Irrespective of what events occur on the TPn_0 or TCKn pin, the counter will continue to free run until the PTnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTMn interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTnIO1 and PTnIO0 bits can select the active trigger edge on the TPn_0 or TCKn pin to be a rising edge, falling edge or both edge types. If the PTnIO1 and PTnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the TPn_0 or TCKn pin, however it must be noted that the counter will continue to run.

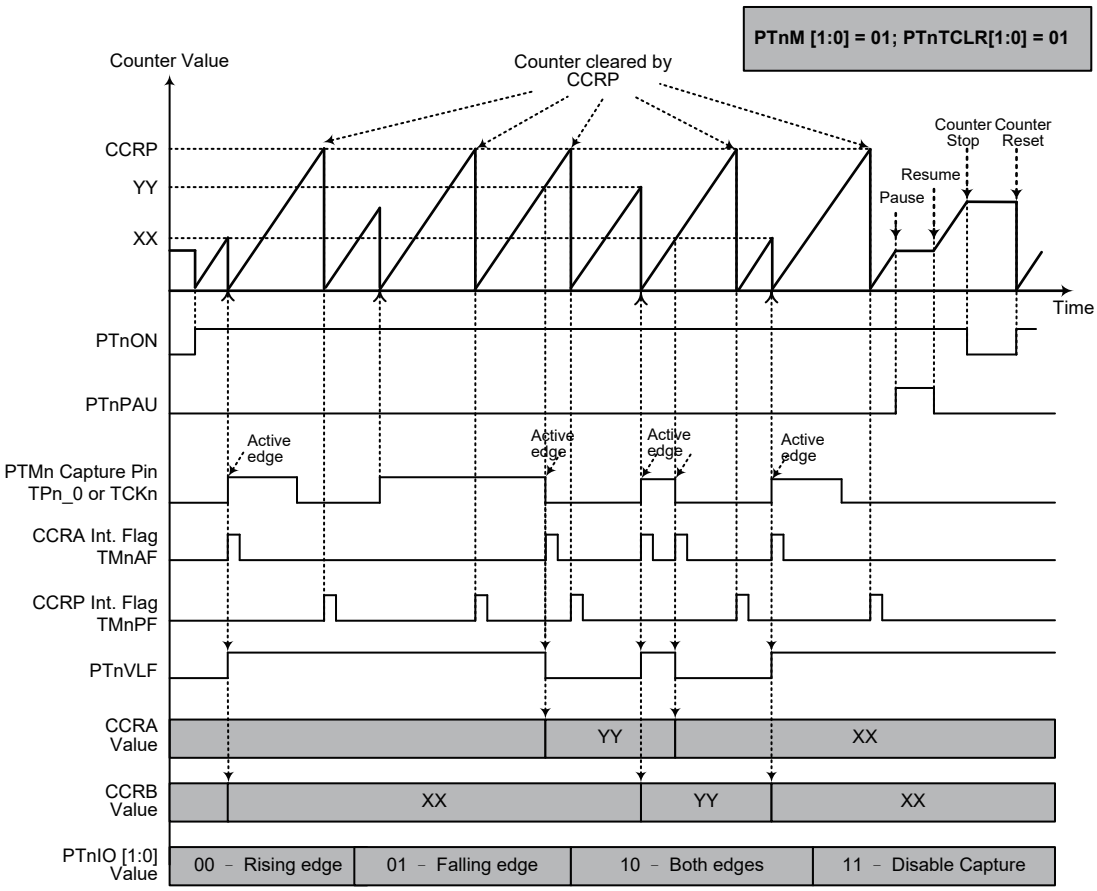
If the capture pulse width is less than two timer clock cycles, it may be ignored by hardware. The timer clock source must be equal to or less than 50MHz, otherwise the counter may fail to count.

As the TPn_0 or TCKn pin is pin shared with other functions, care must be taken if the PTMn is in the Capture Input Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The PTnCCLR, PTnOC and PTnPOL bits are not used in this Mode.



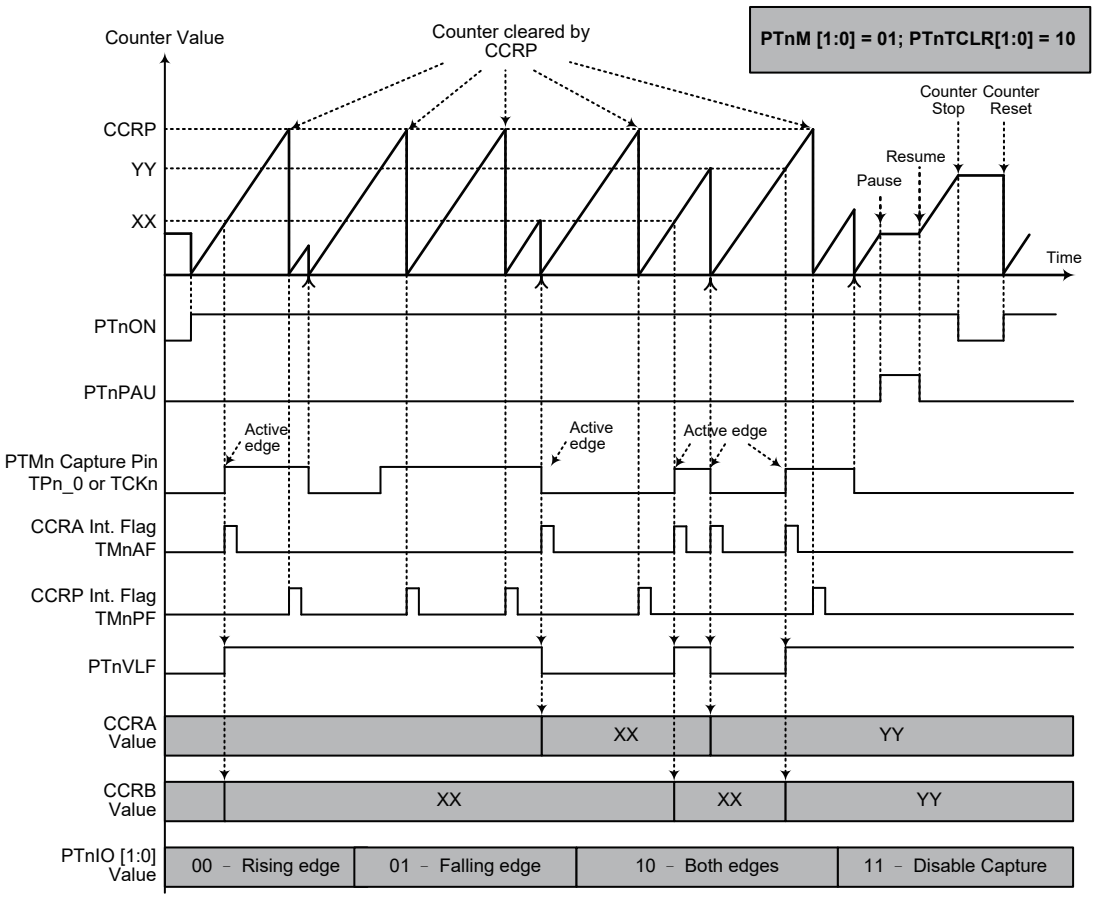
Capture Input Mode – PTnTCLR[1:0]=00 (n=0~3)

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



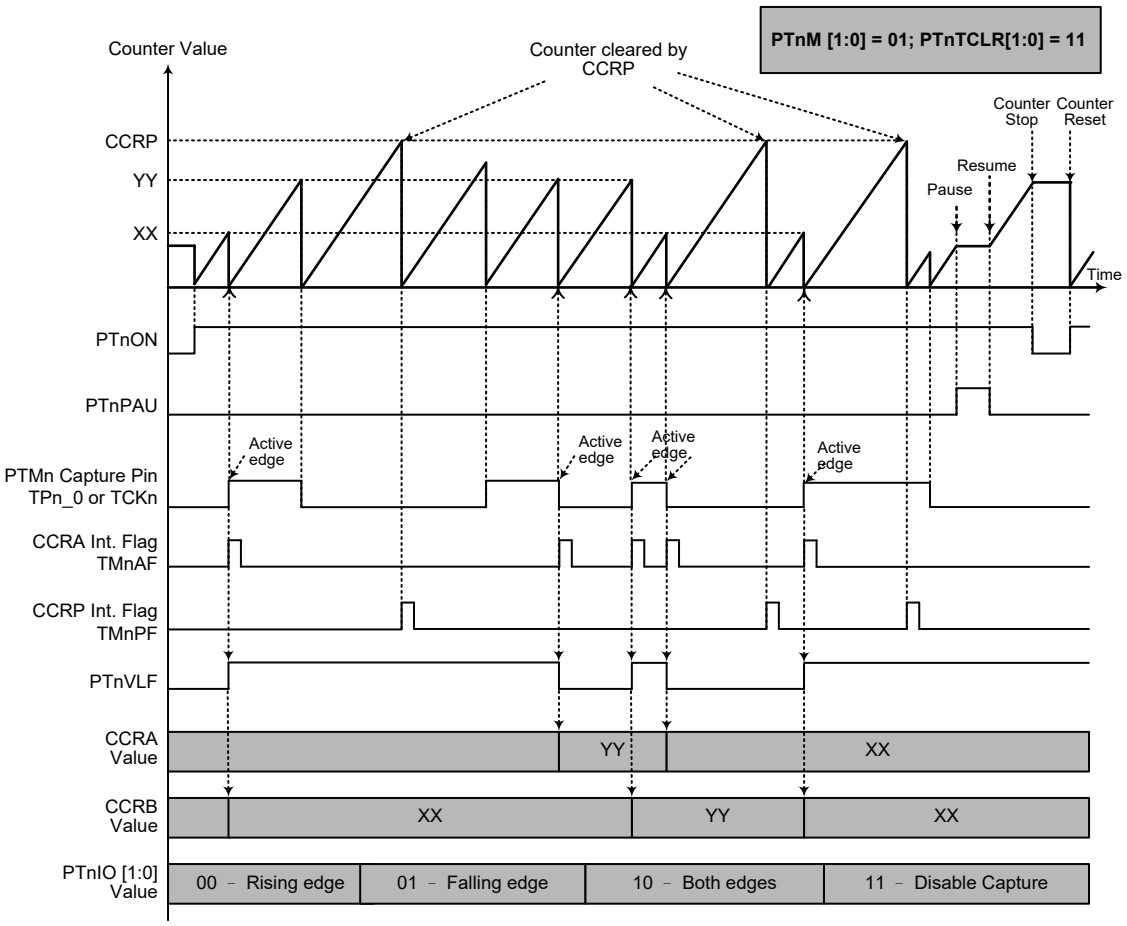
Capture Input Mode — PTnTCLR[1:0]=01 (n=0~3)

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



Capture Input Mode – PTnTCLR[1:0]=10 (n=0~3)

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



Capture Input Mode – PTnTCLR[1:0]=11 (n=0~3)

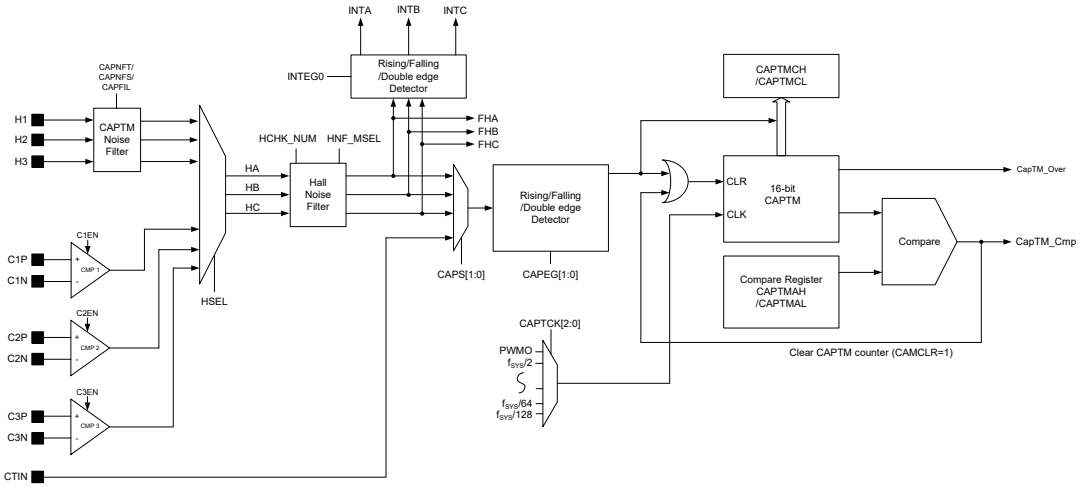
Note: 1. PTnM[1:0]=01, PTnTCLR[1:0]=11 and active edge set by the PTnIO[1:0] bits
2. A PTMn Capture input pin active edge transfers the counter value to CCRA or CCRB
3. Comparator P match or PTMn capture input pin rising or falling edge will clear the counter
4. PTnCCLR bit is not used
5. No output function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

Capture Timer Module – CAPTM

The Capture Timer Module is a timing unit specifically used for Motor Control purposes. The CAPTM is controlled by a program selectable clock source.

Capture Timer Overview

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



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

Capture Timer Block Diagram

Capture Timer Register Description

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

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 Registers List

CAPTC0 Register

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

Bit 7 **CAPTPAU**: CAPTM Counter Pause Control
0: Run
1: Pause

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

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

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

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

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

Bit 2 Unimplemented, read as "0"

Bit 1~0 **CAPS1~CAPS0**: CAPTM capture source selection
00: FHA
01: FHB
10: FHC
11: CTIN

CAPTC1 Register

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

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

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

Bit 4 **CAPNFT**: CAPTM Noise Filter sample times definition
0: Twice
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 clock is decided by the CAPNFS bit.

Bit 3 **CAPNFS**: CAPTM Noise Filter clock source selection
0: t_{SYS}
1: $4 \times t_{SYS}$
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

Bit 1 **CAPCLR**: CAPTM Counter capture auto-reset control
0: Disable
1: Enable
If this bit is set high, when FHA/FHB/FHC/CTIN generates the required capture edge, the hardware will automatically transfer the value in the CAPTMDL and CAPTMDH register to the capture register pair CAPTMCL and CAPTMCH, and then reset the CAPTM counter.

Bit 0 **CAMCLR**: CAPTM Counter compare match auto-reset control
0: Disable
1: Enable
If this bit is set high, when a compare match condition has occurred, the hardware will automatically reset the CAPTM counter.

CAPTMDL Register

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

Bit 7~0 **D7~D0**: CAPTM Counter Low Byte Register bit 7 ~ bit 0
CAPTM 16-bit Counter bit 7 ~ bit 0

CAPTMDH Register

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

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

CAPTMAL Register

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

Bit 7~0 **D7~D0**: CAPTM Compare Low Byte Register bit 7 ~ bit 0
CAPTM 16-bit Comare 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

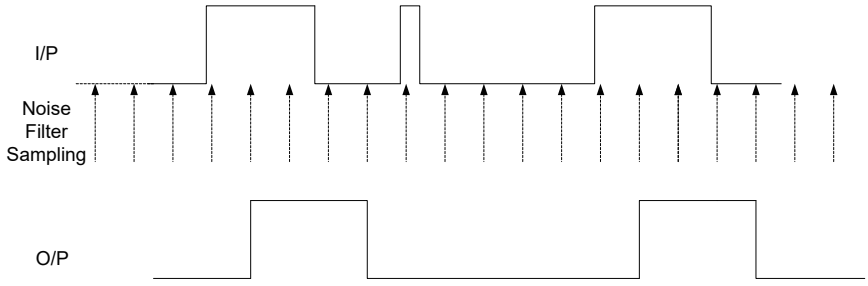
Bit 7~0 **D15~D8**: CAPTM Capture High Byte Register bit 7 ~ bit 0
CAPTM 16-bit Capture Register bit 15 ~ bit 8

Capture Timer Operation

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

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

The timer also includes a capture timer noise filter which is used to filter out unwanted glitches or pulses on the H1, H2 and H3 input pins. This function is enabled using the CAPFIL bit. If the noise filter is enabled, the capture input signals must be sampled either 2 or 4 times, in order to recognise an edge as a valid capture event. The sampling 2 or 4 time units are based on either t_{SYS} or $4 \times t_{SYS}$ determined using the CAPNFS bit.



Noise Filter with CAPNFT=0 and 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 captured into the capture registers, CAPTMCL and CAPTMCH. When this occurs, the relevant interrupt priority flag bit in the interrupt control register will be set. If this interrupt is enabled by setting the relevant interrupt priority enable bit high, an interrupt will be generated. If the CAPCLR bit is set high, then the 16-bit counter will be automatically reset after a capture event occurs.

Compare Mode Operation

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

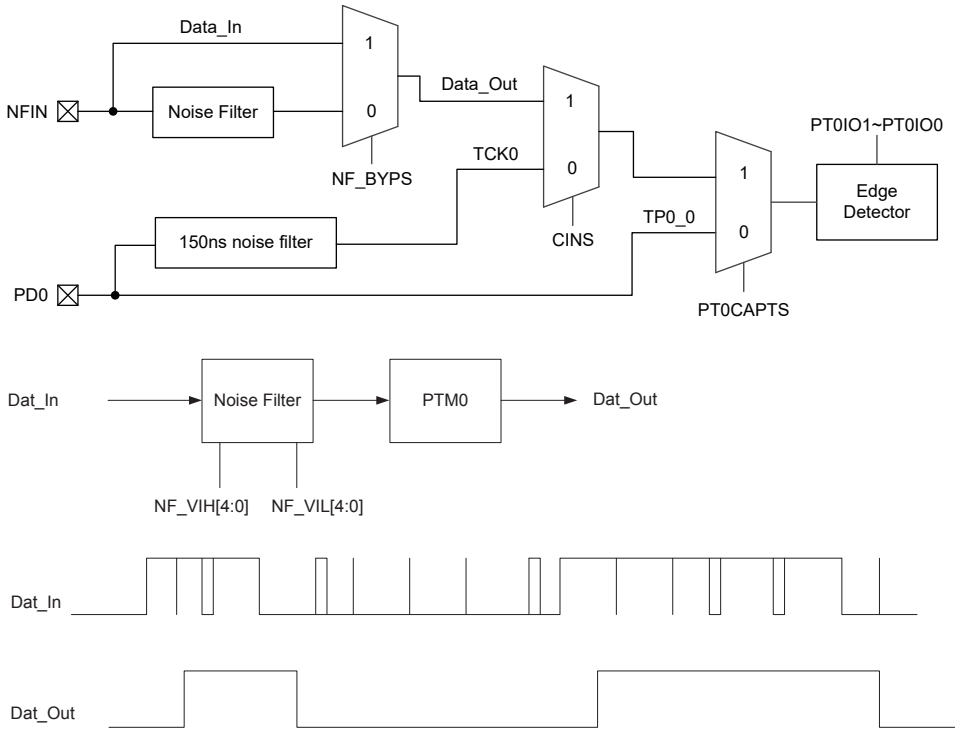
Noise Filter

The external NFIN pin is connected to an internal filter to reduce the possibility of unwanted event counting events or inaccurate pulse width measurements due to adverse noise or spikes on the NFIN input signal, and then outputs to the 16-bit PTM0 capture circuit in order to ensure that the motor control circuit works normally.

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

Antinoise pulse width maximum:

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



NF_VIH Register

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

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

Bit 6 **CINS**: PTM0 capture source selection
0: Noise Filter Dat_Out not selected (remains original PTM0 capture path with 150ns filter)
1: Noise Filter Dat_Out selected

Bit 5 Unimplemented, read as "0"

Bit 4~0 **D4~D0**: NF_VIH register bit 4 ~ bit 0

NF_VIL Register

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

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

Bit 5 Unimplemented, read as "0"

Bit 4~0 **D4~D0**: NF_VIL register bit 4 ~ bit 0

Analog to Digital Converter

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

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, or internal signals, and convert these signals directly into a 12-bit or 10-bit digital value.

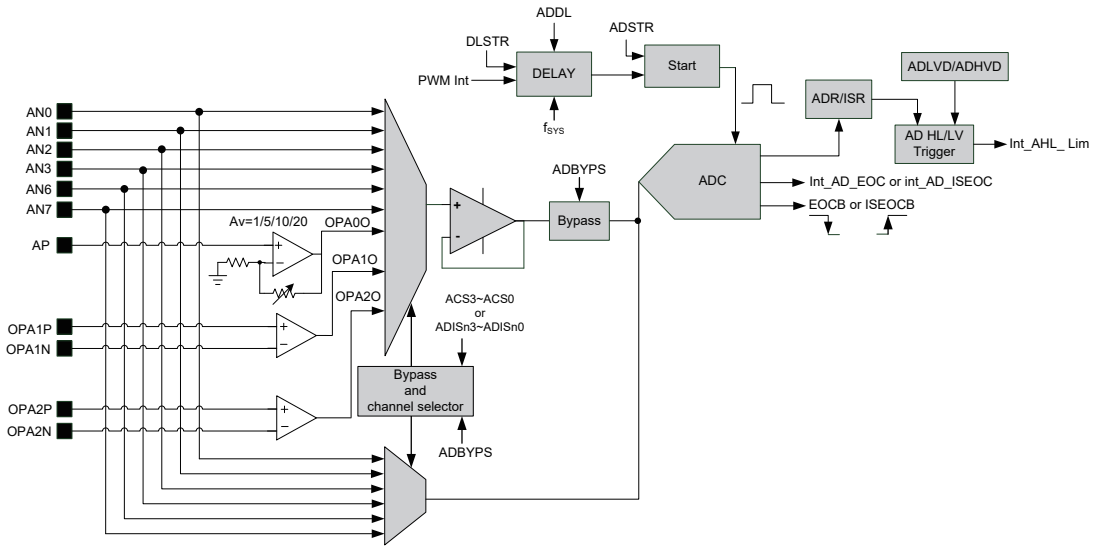
The six external channels can be individually configured to be isolated using the unity-gain buffer when connected to the internal A/D converter. This function can avoid transient voltages caused by channel switching from affecting the channel voltage to be measured. However, in this case the input voltage is limited in the range of V_{SS}+0.1V to V_{DD}-0.1V. To convert a full voltage range of 0~V_{DD}, the external channel can be configured to bypass the unity-gain buffer using the BYPSANn bit in the ADBYPS register.

The three internal channels, which are OPA0, OPA1 and OPA2 outputs, OPA0O~OPA2O, can not bypass the unity-gain buffer when connected to the A/D converter. This means that any one of these channels must go through the unity-gain buffer by enabling the UGB_ON bit in the ADBYPS register, otherwise the conversion result may be incorrect.

There are two trigger mechanisms to start an A/D conversion, one is using the ADSTR bit, another is using the DLSTR bit. For the ADSTR triggered A/D conversion, the input channel is selected using the ACS3~ACS0 bits. For the DLSTR triggered A/D conversion, the input channel is selected using the ADISn3~ADISn0 bits. More detailed information about the A/D input signal is described in the "A/D Converter Control Registers" section.

External Input Channels	Internal Input Signals	A/D Channel Select Bits
6: AN0~AN3, AN6~AN7	3: OPA0O~OPA2O	ACS3~ACS0, ADISn3~ADISn0 (n=0~3)

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



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using a series of registers. A read only register pair, ADRH and ADRL, exists to store the ADSTR triggered A/D converter data 12-bit or 10-bit value. Four read only register pairs, ISRHn and ISRLn, are used to store the DLSTR triggered A/D converter data 12-bit or 10-bit value. Two register pairs, ADLVDH/ADLVDL and ADHVDH/ADHVDL, are used to store A/D converter low and high boundary values reapectively. The ADDL register is used to determine the delay time between the DLSTR trigger action and the actual start of the A/D conversion. The remaining registers are control registers which setup the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
ADRL (ADRFS=0, ADCRL_SEL=0)	D3	D2	D1	D0	—	—	—	—
ADRL (ADRFS=1, ADCRL_SEL=0)	D7	D6	D5	D4	D3	D2	D1	D0
ADRL (ADRFS=0, ADCRL_SEL=1)	D1	D0	—	—	—	—	—	—
ADRL (ADRFS=1, ADCRL_SEL=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADRH (ADRFS=0, ADCRL_SEL=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADRH (ADRFS=1, ADCRL_SEL=0)	—	—	—	—	D11	D10	D9	D8
ADRH (ADRFS=0, ADCRL_SEL=1)	D9	D8	D7	D6	D5	D4	D3	D2
ADRH (ADRFS=1, ADCRL_SEL=1)	—	—	—	—	—	—	D9	D8
ADCR0	ADSTR	EOCB	ADOFF	ADRFS	ACS3	ACS2	ACS1	ACS0
ADCR1	ADRE	DLSTR	PWIS	ADCHVE	ADCLVE	ADCK2	ADCK1	ADCK0
ADCR2	IOEOCB	ADCRL_SEL	ADCH_SEL1	ADCH_SEL0	ISEOCB	—	PWMDIS1	PWMDIS0
ADCR3	—	—	—	—	—	OPA2LE	OPA1LE	OPA0LE
ADISG1	ADIS13	ADIS12	ADIS11	ADIS10	ADIS03	ADIS02	ADIS01	ADIS00
ADISG2	ADIS33	ADIS32	ADIS31	ADIS30	ADIS23	ADIS22	ADIS21	ADIS20
ADDL	D7	D6	D5	D4	D3	D2	D1	D0

Register Name	Bit							
	7	6	5	4	3	2	1	0
ADBYPS	UGB_ON	—	BYPSAN7	BYPSAN6	BYPSAN3	BYPSAN2	BYPSAN1	BYPSAN0
ADLVDL (ADRFS=0, ADCRL_SEL=0)	D3	D2	D1	D0	—	—	—	—
ADLVDL (ADRFS=1, ADCRL_SEL=0)	D7	D6	D5	D4	D3	D2	D1	D0
ADLVDL (ADRFS=0, ADCRL_SEL=1)	D1	D0	—	—	—	—	—	—
ADLVDL (ADRFS=1, ADCRL_SEL=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADLVDH (ADRFS=0, ADCRL_SEL=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADLVDH (ADRFS=1, ADCRL_SEL=0)	—	—	—	—	D11	D10	D9	D8
ADLVDH (ADRFS=0, ADCRL_SEL=1)	D9	D8	D7	D6	D5	D4	D3	D2
ADLVDH (ADRFS=1, ADCRL_SEL=1)	—	—	—	—	—	—	D9	D8
ADHVDL (ADRFS=0, ADCRL_SEL=0)	D3	D2	D1	D0	—	—	—	—
ADHVDL (ADRFS=1, ADCRL_SEL=0)	D7	D6	D5	D4	D3	D2	D1	D0
ADHVDL (ADRFS=0, ADCRL_SEL=1)	D1	D0	—	—	—	—	—	—
ADHVDL (ADRFS=1, ADCRL_SEL=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADHVDH (ADRFS=0, ADCRL_SEL=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADHVDH (ADRFS=1, ADCRL_SEL=0)	—	—	—	—	D11	D10	D9	D8
ADHVDH (ADRFS=0, ADCRL_SEL=1)	D9	D8	D7	D6	D5	D4	D3	D2
ADHVDH (ADRFS=1, ADCRL_SEL=1)	—	—	—	—	—	—	D9	D8
ISRLn (ADRFS=0, ADCRL_SEL=0)	D3	D2	D1	D0	—	—	—	—
ISRLn (ADRFS=1, ADCRL_SEL=0)	D7	D6	D5	D4	D3	D2	D1	D0
ISRLn (ADRFS=0, ADCRL_SEL=1)	D1	D0	—	—	—	—	—	—
ISRLn (ADRFS=1, ADCRL_SEL=1)	D7	D6	D5	D4	D3	D2	D1	D0
ISRHn (ADRFS=0, ADCRL_SEL=0)	D11	D10	D9	D8	D7	D6	D5	D4
ISRHn (ADRFS=1, ADCRL_SEL=0)	—	—	—	—	D11	D10	D9	D8
ISRHn (ADRFS=0, ADCRL_SEL=1)	D9	D8	D7	D6	D5	D4	D3	D2
ISRHn (ADRFS=1, ADCRL_SEL=1)	—	—	—	—	—	—	D9	D8

Note: n=0~3

A/D Converter Registers List

A/D Converter Data Registers – ADRL, ADRH, ISRLn, ISRHn

As this device contains an internal up to 12-bit A/D converter, it requires two data registers to store each converted value. These are a high byte register, known as ADRH, and a low byte register, known as ADRL, to store the converted value which is triggered by the ADSTR bit. The device also contains four register pairs to strore the converted value which is triggered by the DLSTR bit. Each register pair is composed of a high byte register, known as ISRHn, and a low byte register, known as ISRLn. 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 or 10 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 ADCR0 register together with the ADCRL_SEL bit in the ADCR2 register as shown in the accompanying tables. D0~D11 or D0~D9 are the A/D conversion result data bits. Any unused bits will be read as zero.

- **ADCRL_SEL=0 → 12-bit Format**

ADRFS	ADRH (Read only)								ADRL (Read only)							
	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

ADRFS	ISRHn (Read only)								ISRLn (Read only)							
	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

Note: n=0~3

- **ADCRL_SEL=1 → 10-bit Format**

ADRFS	ADRH (Read only)								ADRL (Read only)							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0
1	0	0	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

ADRFS	ISRHn (Read only)								ISRLn (Read only)							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0
1	0	0	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Note: n=0~3

A/D Converter Data Registers

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

The device contains what are known as boundary registers to strore fixed values for comparison with the A/D converted data value stored in ADRH/ADRL or ISRHn/ISRLn. There are two pairs of boundary registers, a high boundary pair, known as ADHVDL and ADHVDH, and a low boundary pair known as ADLVDL and ADLVDH. Only the converted data of OPA0O, OPA1O or OPA2O, which are stored in ADRH/ADRL or ISRHn/ISRLn, can be compared with the low and high boundary values.

As only 12 bits or 10 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 ADCR0 register together with the ADCRL_SEL bit in the ADCR2 register as shown in the accompanying tables.

- **ADCRL_SEL=0 → 12-bit Format**

ADRFS	ADLVDH (R/W)								ADLVDL (R/W)							
	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

ADRFS	ADHVDH (R/W)								ADHVDL (R/W)							
	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

- **ADCRL_SEL=1 → 10-bit Format**

ADRFS	ADLVDH (R/W)								ADLVDL (R/W)							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0
1	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	D9	D8

ADRFS	ADHVDH (R/W)								ADHVDL (R/W)							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0
1	0	0	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D Boundary Data Registers

A/D Converter Control Registers – ADCRn (n=0~3), ADISG1, ADISG2, ADDL, ADBYPS

To control the function and operation of the A/D converter, several control registers known as 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 or internal analog signal inputs must be routed to the converter. The ACS3~ACS0 bits in the ADCR0 register are used to determine which external or internal channel input is selected to be converted in the case of ADSTR triggered A/D conversion. In the case of auto-scan triggered A/D conversion, the ADCH_SEL bit field is used to determine the number of channels to be scanned and the ADISn3~ADISn0 bits are used to select which external or internal channel input is selected to be converted. The ADDL register is used to store the A/D conversion start delay time, which is only available for the auto-scan triggered A/D conversion and is only required before scanning the first selected channel. The ADBYPS register is used to control the unity-gain buffer and setup ANn bypass unity-gain buffer function.

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

- **ADCR0 Register**

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

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

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

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

Bit 4 **ADRFS**: A/D converter data format select
12-bit data format (ADCRL_SEL=0):
0: High Byte=D[11:4]; Low Byte=D[3:0]
1: High Byte=D[11:8]; Low Byte=D[7:0]
10-bit data format (ADCRL_SEL=1):
0: High Byte=D[9:2]; Low Byte=D[1:0]
1: High Byte=D[9:8]; Low Byte=D[7:0]
This bit controls the format of the 12-bit or 10-bit A/D converted data or boundary data in the corresponding high byte and low byte registers. Details are respectively provided in the A/D data registers and A/D boundary registers sections.

Bit 3~0 **ACS3~ACS0**: A/D converter analog channel input select (for ADSTR triggered A/D conversion)
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: OPA2 output
0101: OPA1 output
0110: OPA0 output
0111: AN6
1000: AN7
1001~1111: Undefined

- **ADCR1 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADRE	DLSTR	PWIS	ADCHVE	ADCLVE	ADCK2	ADCK1	ADCK0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **ADRE**: ADSTR triggered A/D conversion interrupted by A/D auto-scan flag
0: ADSTR triggered A/D conversion interrupted by A/D auto-scan has not occurred
1: ADSTR triggered A/D conversion interrupted by A/D auto-scan has occurred
As the A/D auto-scan triggered conversion has a higher priority than the ADSTR triggered A/D conversion, it can interrupt an ongoing ADSTR triggered A/D conversion. The ADRE bit can be used to monitor whether such condition has occurred. If yes, this bit will be set high by hardware to indicate that the current ADSTR triggered A/D conversion is incompleted and the result in the ADRH and ADRL register pair is invalid.

Bit 6 **DLSTR**: A/D auto-scan conversion control
0: Disable, A/D auto-scan conversion off
1: Enable, A/D auto-scan conversion on
Setting this bit high will enable an A/D auto-scan triggered A/D conversion which can interrupt the current ADSTR triggered A/D conversion since it has a higher priority than the latter.

Bit 5 **PWIS**: Select PWM source to trigger A/D auto-scan
0: Select PWM period to trigger auto-scan
1: Select PWM duty to trigger auto-scan
When this bit is set high to select PWM duty to trigger auto-scan conversion, the detailed PWM duty source is furtherly determined by the PWDIS1~PWDIS0 bits in the ADCR2 register.

Bit 4~3 **ADCHVE, ADCLVE**: A/D compare interrupt trigger source selection
00: Low boundary value < Converted data < High boundary value
01: Converted data ≤ Low boundary value
10: Converted data ≥ High boundary value
11: Converted data ≤ Low boundary value or Converted data ≥ High boundary value
The low boundary value is the content in the ADLVDH and ADLVDL register pair, the high boundary value is the content in the ADHVDH and ADHVDL register pair. The converted data indicates the converted result of OPA0O, OPA1O and OPA2O, which are stored in ADRH and ADRL register pair or ISRHn and ISRLn register pair. When the preset compare interrupt source condition occurs, an A/D converter boundary interrupt will be generated.

Bit 2~0 **ADCK2~ADCK0**: A/D conversion clock source select
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: Undefined
These three bits are used to select the clock source for the A/D converter.

- **ADCR2 Register**

Bit	7	6	5	4	3	2	1	0
Name	IOEOCB	ADCRL_SEL	ADCH_SEL1	ADCH_SEL0	ISEOCB	—	PWDIS1	PWDIS0
R/W	R	R/W	R/W	R/W	R/W	—	R/W	R/W
POR	0	0	0	0	0	—	0	0

Bit 7 **IOEOCB**: A/D auto-scan circuit status flag
0: A/D auto-scan circuit is idle
1: A/D auto-scan circuit is busy
This bit will be set high by hardware when the A/D auto-scan circuit is in a busy status, in which case ADSTR triggered A/D conversions are not allowed.

Bit 6 **ADCRL_SEL**: A/D resolution selection
0: 12-bit, one conversion needs $18\times t_{ADCK}$
1: 10-bit, one conversion needs $16\times t_{ADCK}$

Bit 5~4 **ADCH_SEL1~ADCH_SEL0**: Number of channels to be scanned selection
00: 1 channel
01: 2 channels
10: 3 channels
01: 4 channels
This setup is only available for A/D auto-scan triggered conversion. After setting these bits, the actual channel to be converted is configured using the ADISn3~ADISn0 bits.

Bit 3 **ISEOCB**: A/D auto-scan conversion finished flag
0: A/D auto-scan has not been triggered or A/D auto-scan has not finished
1: A/D auto-scan has been triggered and has finished
When the A/D auto-scan has been triggered and all the selected channels have been converted, this bit will be set high by hardware to indicate that the ISRLn/ISRHn registers are available to read. After the ISRLn/ISRHn registers has been read, this bit must be cleared to zero by application program.

Bit 2 Unimplemented, read as "0"

Bit 1~0 **PWDIS1~PWDIS0**: PWMn duty trigger interrupt selection when PWIS=1
00: PWM0 duty trigger interrupt
01: PWM1 duty trigger interrupt
10: PWM2 duty trigger interrupt
11: PWM3 duty trigger interrupt

- **ADCR3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	OPA2LE	OPA1LE	OPA0LE
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **OPA2LE**: OPA2 output compare with boundary values control
0: Disable
1: Enable

Bit 1 **OPA1LE**: OPA1 output compare with boundary values control
0: Disable
1: Enable

Bit 0 **OPA0LE**: OPA0 output compare with boundary values control
0: Disable
1: Enable

- **ADISG1 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADIS13	ADIS12	ADIS11	ADIS10	ADIS03	ADIS02	ADIS01	ADIS00
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 7~4 **ADIS13~ADIS10**: Auto-scan triggered A/D conversion second analog channel input selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: OPA2 output
0101: OPA1 output
0110: OPA0 output
0111: AN6
1000: AN7
1001~1111: Undefined

When using the auto-scan mechnism to trigger the A/D conversion and multiple channels are selected, the A/D converter will orderly switch to the channel selected by the ADIS1 bit field. After this conversion the result is stored in the ISRH1 and ISRL1 registers. If it is not the last channel, the A/D converter will switch to the ADIS2 bit field defined channel. While if it is the last channel the ISEOCB bit will be set high by hardware. Then the A/D converter will return to the channel selected by ACS bit field for use in ADSTR triggered conversions.

Bit 3~0 **ADIS03~ADIS00**: Auto-scan triggered A/D conversion first analog channel input selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: OPA2 output
0101: OPA1 output
0110: OPA0 output
0111: AN6
1000: AN7
1001~1111: Undefined

When using the auto-scan mechnism to trigger the A/D conversion, the A/D converter will firstly switch to the channel selected by the ADIS0 bit field. After this conversion the result is stored in the ISRH0 and ISRL0 registers. If it is not the last channel, the A/D converter will switch to the ADIS1 bit field defined channel. While if it is the last channel the ISEOCB bit will be set high by hardware. Then the A/D converter will return to the channel selected by ACS bit field for use in ADSTR triggered conversions.

- **ADISG2 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADIS33	ADIS32	ADIS31	ADIS30	ADIS23	ADIS22	ADIS21	ADIS20
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 7~4 **ADIS33~ADIS30**: Auto-scan triggered A/D conversion fourth analog channel input selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: OPA2 output
0101: OPA1 output
0110: OPA0 output
0111: AN6
1000: AN7
1001~1111: Undefined

When using the auto-scan mechnism to trigger the A/D conversion and multiple channels are selected, the A/D converter will orderly switch to the channel selected by the ADIS3 bit field. After this conversion the result is stored in the ISRH3 and ISRL3 registers and the ISEOCB bit will be set high by hardware. Then the A/D converter will return to the channel selected by ACS bit field for use in ADSTR triggered conversions.

Bit 3~0 **ADIS23~ADIS20**: Auto-scan triggered A/D conversion third analog channel input selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: OPA2 output
0101: OPA1 output
0110: OPA0 output
0111: AN6
1000: AN7
1001~1111: Undefined

When using the auto-scan mechnism to trigger the A/D conversion and multiple channels are selected, the A/D converter will orderly switch to the channel selected by the ADIS2 bit field. After this conversion the result is stored in the ISRH2 and ISRL2 registers. If it is not the last channel, the A/D converter will switch to the ADIS3 bit field defined channel. While if it is the last channel the ISEOCB bit will be set high by hardware. Then the A/D converter will return to the channel selected by ACS bit field for use in ADSTR triggered conversions.

- **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**: A/D delay time register bit 7 ~ bit 0 (count by f_{SYS})
A/D delay time value=$(1/f_{SYS}) \times$ ADDL[7:0]

- **ADBYPS Register**

Bit	7	6	5	4	3	2	1	0
Name	UGB_ON	—	BYPSAN7	BYPSAN6	BYPSAN3	BYPSAN2	BYPSAN1	BYPSAN0
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 **UGB_ON**: Unity-gain buffer on/off control
0: Disable
1: Enable

Bit 6 Unimplemented, read as "0"

Bit 5 **BYPSAN7**: AN7 bypass unity-gain buffer control
0: Disable, AN7 go through unity-gain buffer
1: Enable, AN7 bypass unity-gain buffer

Bit 4 **BYPSAN6**: AN6 bypass unity-gain buffer control
0: Disable, AN6 go through unity-gain buffer
1: Enable, AN6 bypass unity-gain buffer

Bit 3 **BYPSAN3**: AN3 bypass unity-gain buffer control
0: Disable, AN3 go through unity-gain buffer
1: Enable, AN3 bypass unity-gain buffer

Bit 2 **BYPSAN2**: AN2 bypass unity-gain buffer control
0: Disable, AN2 go through unity-gain buffer
1: Enable, AN2 bypass unity-gain buffer

Bit 1 **BYPSAN1**: AN1 bypass unity-gain buffer control
0: Disable, AN1 go through unity-gain buffer
1: Enable, AN1 bypass unity-gain buffer

Bit 0 **BYPSAN0**: AN0 bypass unity-gain buffer control
0: Disable, AN0 go through unity-gain buffer
1: Enable, AN0 bypass unity-gain buffer

A/D Converter Operation

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

The second method of initiating a conversion is to use the PWM interrupt signal to trigger the A/D auto-scan which is enabled by the DLSTR bit in the ADCR1 register. The PWM interrupt signal can be sourced from either the PWM period or the PWM duty interrupt signal, selected using the PWIS bit. If selects PWM duty interrupt signal, the actual PWM duty interrupt trigger source can be selected by the PWDIS1 and PWDIS0 bits in the ADCR2 register. The DLSTR bit can also activate a delay function which inserts a delay time between the incoming PWM interrupt signal and the actual start of the A/D conversion process, with the actual time being setup using the ADDL register. The actual delay time is calculated by the register content multiplied by the system clock period. The delay time is used to reduce the possibility of erroneous analog samples being taken during the time of large transient current swithing by the motor drive tansistors.

The EOCB bit in the ADCR0 register is used to indicate whether the ADSTR triggered analog to digital conversion process is completed. This bit will be automatically cleared to zero by the microcontroller after an A/D conversion cycle has ended. Similarly, the ISEOCB bit in the ADCR2 register is used to indicate whether the A/D auto-scan triggered analog to digital conversion process is completed. This bit will be automatically set high by the microcontroller after all the selected channels for the A/D auto-scan conversion have been converted. In addition, the corresponding A/D interrupt request flag AEOCF or ISAEOCF will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can poll the EOCB bit in the ADCR0 register to check whether it has been cleared or poll the ISOECB bit in the ADCR2 register to check whether is has been set high as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the ADCK2~ADCK0 bits in the ADCR1 register. Although the A/D clock source is determined by the system clock f_{SYS} and by bits ADCK2~ADCK0, there are some limitations on the maximum and minimum A/D clock source speeds that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK}, is from 0.16μs to 10μs for 12-bit format, and 0.1μs to 10μs for 10-bit format, care must be taken for system clock frequencies. Refer to the following table for examples, where values marked with an asterisk * show where, special care must be taken, as the values may exceed the specified A/D Clock Period range.

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

12-bit A/D Clock Period Examples

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

10-bit A/D Clock Period Examples (A/D Auto-scan Bypass Unity-gain Buffer)

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

Note: In 10-bit A/D conversion applications, if multiple channels are used and go through the unity-gain buffer, the minimum A/D clock period is 200ns, if the A/D clock period is configured to be smaller than 200ns, the A/D auto-scan conversion results will be incorrect.

10-bit A/D Clock Period Examples (A/D Auto-scan By Unity-gain Buffer)

Controlling the power on/off function of the A/D converter circuitry is implemented using the ADOFF bit in the ADCR0 register. This bit must be cleared to power on the A/D converter. When the ADOFF bit is cleared to power on the A/D converter internal circuitry, 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 ADOFF bit is low, then some power will still be consumed. In power conscious applications it is therefore recommended that the ADOFF is set high to reduce power consumption when the A/D converter function is not being used.

The boundary register pairs, ADHVDH/ADHVDL and ADLVDH/ADLVDL contain preset values which can be compared with the converted values of OPA0O, OPA1O and OPA2O that are stored in ADRH/ADRL registers or ISRHn/ISRLn registers. Various types of comparisons can be made as defined by the ADCLVE and ADCHVE bits and an interrupt will be generated to inform the system that either the lower or higher boundary has been exceeded. This function can be used to ensure that the motor current operates within safe working limits.

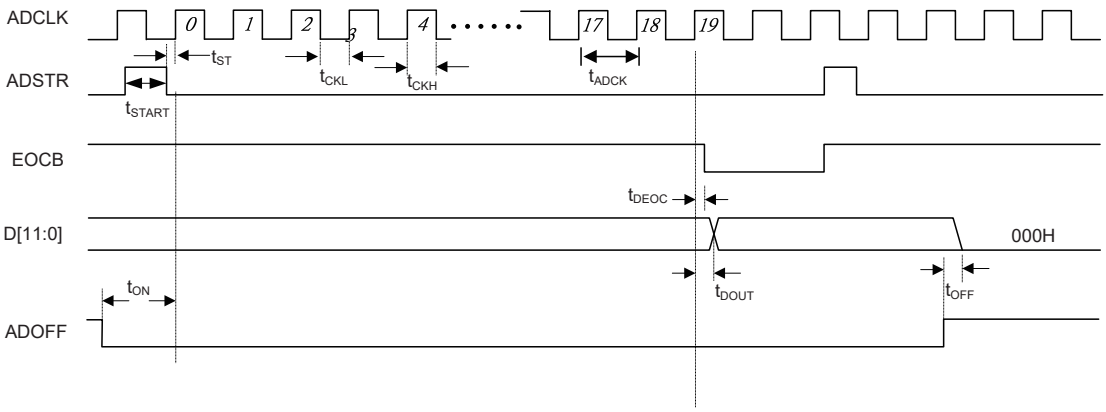
Summary of A/D Conversion Steps

ADSTR Triggered A/D Conversion Steps

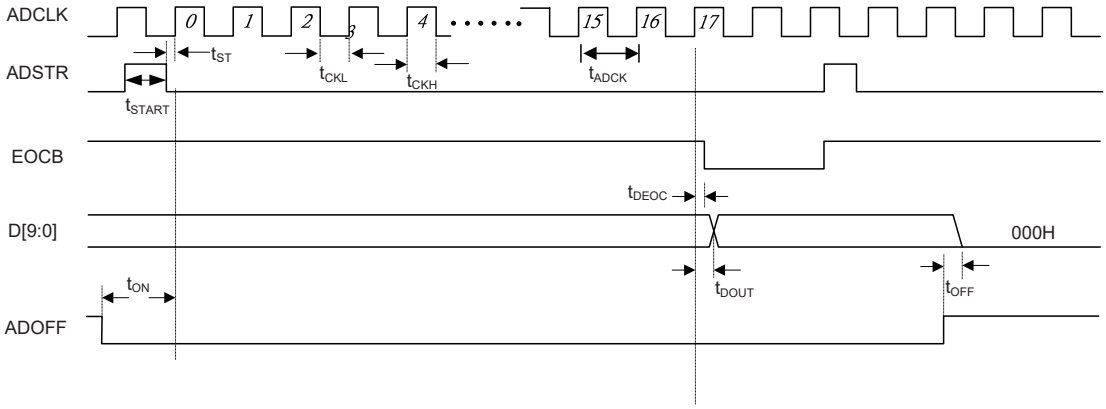
The following summarises the individual steps that should be executed in order to implement an ADSTR triggered A/D conversion process.

- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCK2~ADCK0 in the ADCR1 register.
- Step 2
  Enable the A/D converter by clearing the ADOFF bit in the ADCR0 register to zero.
- Step 3
  Select which signal is to be connected to the internal A/D converter by correctly configuring the ACS3~ACS0 bits in the ADCR0 register. If external channel ANn is selected, its bypass unity-gain buffer function can be enabled or disabled by correctly configuring the ADBYPS register according to application requirements.
- Step 4
  Select which pin is to be used as A/D input and configure it by correctly programming the corresponding bit in the pin-shared control register.
- Step 5
  Select A/D converter resolution and data format by setting the ADCRL_SEL bit in the ADCR2 register and the ADRFS bit in the ADCR0 register.
- Step 6
  If A/D conversion interrupt is used, the 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, AEOCE, and the associated interrupt priority control bit, Int_prinE, must all be set high in advance.
- Step 7
  The A/D conversion procedure can now be initialised by setting the ADSTR bit from low to high and then low again. Note that this bit should have been originally cleared to zero.
- Step 8
  If A/D conversion is in progress, the EOCB flag will be set high. After the A/D conversion process is complete, the EOCB flag will go low and then the output data can be read from ADRH and ADRL registers.

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



12-bit A/D Conversion Timing



10-bit A/D Conversion Timing

DLSTR Triggered A/D Conversion Steps

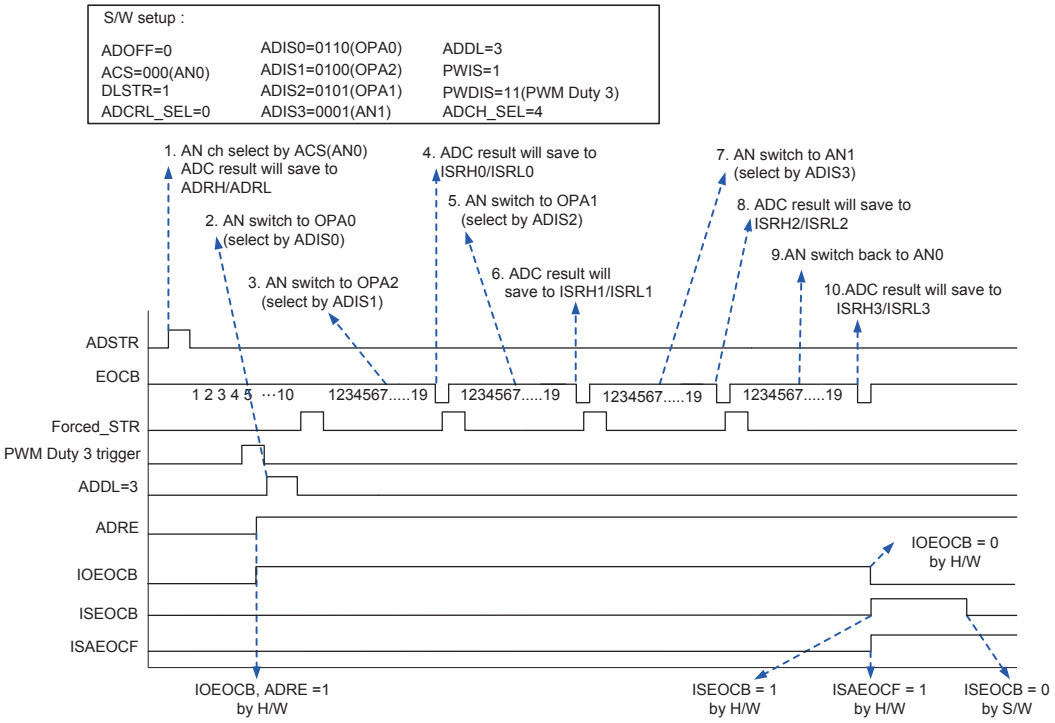
The following summarises the individual steps that should be executed in order to implement a DLSTR triggered A/D conversion process.

- Step 1
  Select which pins are to be used as A/D inputs and configure them by correctly programming the corresponding bits in the pin-shared control registers.
- Step 2
  Select the required A/D conversion clock by correctly programming bits ADCK2~ADCK0 in the ADCR1 register. Select the PWM interrupt signal source by setting the PWIS bit and PWDIS1~PWDIS0 bits. Select the number of channels to be converted by setting the ADCH_SEL bit in the ADCR2 register. And select which signals are to be orderly connected to the internal A/D converter by correctly configuring the ADISn3~ADISn0 bits in the ADISG1 and ADISG2 registers. If external channel ANn is selected, its bypass unity-gain buffer function can be enabled or disabled by correctly configuring the ADBYPS register according to application requirements.
- Step 3
  Enable the A/D by clearing the ADOFF bit in the ADCR0 register to zero.
- Step 4
  Enable PWM function by setting the PWMON bit in the PWMC register high.

- Step 5
  Select A/D converter resolution and data format by setting the ADCRL_SEL bit in the ADCR2 register and the ADRFS bit in the ADCR0 register.
- Step 6
  If A/D conversion interrupt is used, the 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, ISAEOCE, and the associated interrupt priority control bit, Int_prinE, must all be set high in advance.
- Step 7
  After the A/D conversion process is complete, the ISEOCB flag will go high and then the output data can be read from ISRHn and ISRLn registers. After reading the converted results, the ISEOCB bit should be cleared to zero by application program.

Note: When checking for the end of the conversion process, if the method of polling the ISEOCB bit in the ADCR2 register is used, the interrupt enable step above can be omitted.

Considerations for simutaniously using both trigger methods



A/D Convertion Timing for Two Trigger Modes

The following are some important notes for simutaniously using both trigger methods:

- The analog channel to be converted by the ADSTR triggered A/D conversion is determined by the ACS bit field.
- The analog channels to be converted by the DLSTR triggered A/D conversion is determined by the ADCH_SEL and ADISn bit fields. Up to four channels can be converted in one DLSTR triggered A/D conversion.
- The DLSTR triggered A/D conversion has a higher priority than the ADSTR triggered A/D conversion thus the former can interrupt the later. If this happens the ongoing ADSTR triggered A/D conversion will be abandoned and the ADRE bit will be set high by hardware to inform the system that the current result in the ADRH and ADRL registers is invalid.
- When entering the DLSTR trigger mode, the A/D converter will automatically switch to the first channel defined by the ADIS0 bie field. The system will wait for a delay time defined by ADDL and start to convert the first channel. Note that only before the first channel conversion the ADDL delay time is required.
- During each channel conversion the system will simutaniously check whether the current channel is the last one. If no the converter will switch to the next channel. If yes the converter will switch back to the channel defined by the ACS bit field.
- The converted result of channel defined by ADISn bit field is strored in the ISRHn and ISRLn registers.
- When the last channel conversion has been finished, the ISEOCB bit will be set high and the ISAEOCF interrupt will be generated.
- When switching back to the ACS defined channel, the converter will not re-sample the interrupted A/D conversion channel. If users need to re-sample the input channel, set the ADSTR bit from low to high and low again.

Programming Considerations

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

A/D Conversion Function

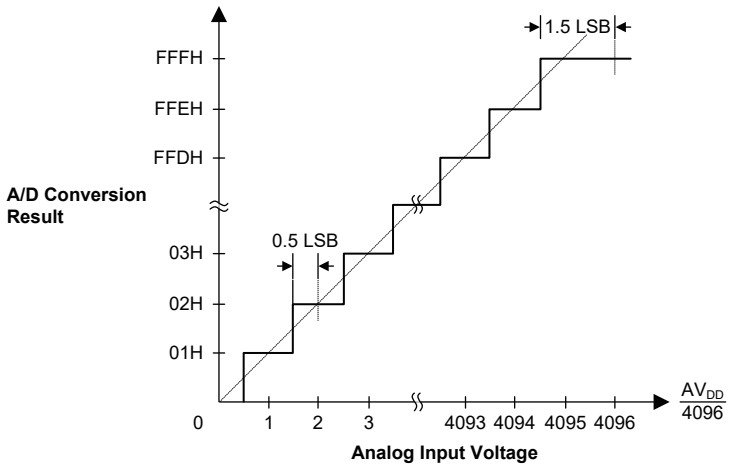
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to AV_{DD}, this gives a single bit analog input value of AV_{DD} divided by 4096.

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

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

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

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the AV_{DD} level. Note that for the 10-bit A/D conversion, 1 LSB= $AV_{DD} \div 1024$.



Ideal A/D Conversion Function

A/D Conversion Programming Examples

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

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

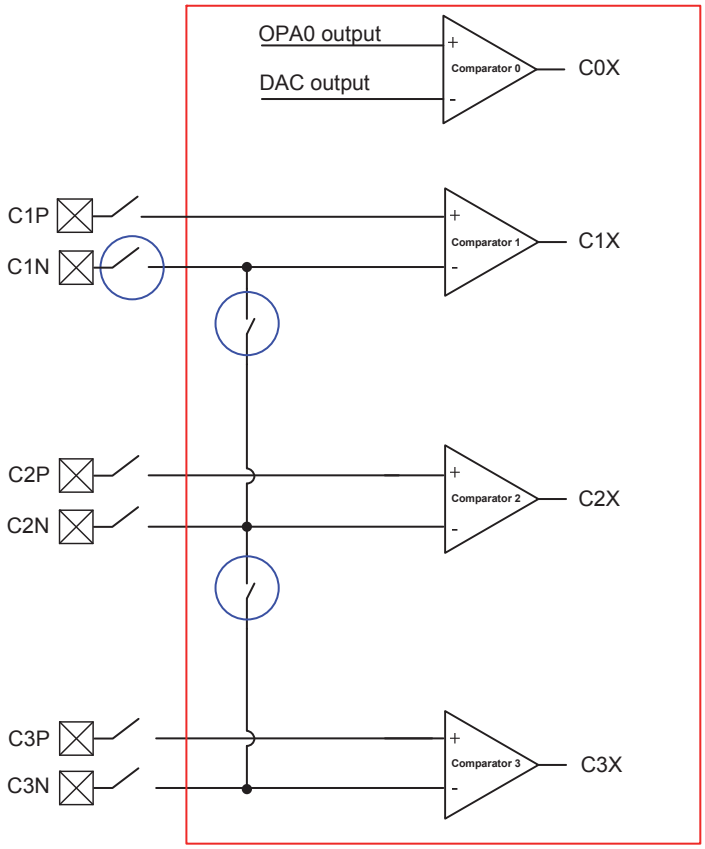
```
clr  AEOCE              ; disable A/D interrupt
mov  a,03H
mov  ADCR1,a            ; select fSYS/8 as A/D clock
clr  ADOFF
mov  a,01h              ; setup PDPS0 to configure pin AN0
mov  PDPS0,a
mov  a,00h
mov  ADCR0,a            ; enable and connect AN0 channel to A/D converter
:
:
start_conversion:
clr  ADSTR              ; high pulse on start bit to initiate conversion
set  ADSTR              ; reset A/D
clr  ADSTR              ; start A/D
polling_EOC:
sz   EOCB               ; poll the ADCR0 register EOCB bit to detect end of A/D conversion
jmp  polling_EOC        ; continue polling
mov  a,ADRL             ; read low byte conversion result value
mov  ADRL_buffer,a      ; save result to user defined register
mov  a,ADRH             ; read high byte conversion result value
mov  ADRH_buffer,a      ; save result to user defined register
:
:
jmp  start_conversion   ; start next A/D conversion
```

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

```
clr  AEOCE               ; disable A/D interrupt
mov  a,03H
mov  ADCR1,a             ; select fSYS/8 as A/D clock
clr  ADOFF
mov  a,01h               ; setup PDPS0 to configure pin AN0
mov  PDPS0,a
mov  a,00h
mov  ADCR0,a             ; enable and connect AN0 channel to A/D converter
:
:
start_conversion:
clr  ADSTR               ; high pulse on start bit to initiate conversion
set  ADSTR               ; reset A/D
clr  ADSTR               ; start A/D
clr  AEOCF               ; clear A/D interrupt request flag
set  Int_prinF           ; clear related interrupt priority request flag
set  AEOCE               ; enable A/D interrupt
set  Int_prinE           ; enable related interrupt priority
set  EMI                 ; enable global interrupt
:
:
; A/D interrupt service routine
ADC_ISR:
mov  acc_stack,a         ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a      ; save STATUS to user defined memory
:
:
mov  a,ADRL              ; read low byte conversion result value
mov  ADRL_buffer,a       ; save result to user defined register
mov  a,ADRH              ; read high byte conversion result value
mov  ADRH_buffer,a       ; save result to user defined register
:
:
EXIT_INT_ISR:
mov  a,status_stack
mov  STATUS,a            ; restore STATUS from user defined memory
mov  a,acc_stack         ; restore ACC from user defined memory
reti
```

Comparators

Four independent analog comparators are contained within 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.



Note: When the PB3S1~PB3S0 bits in the PBPS0 register are configured to 10B, the switches circled in the block diagram will be turned on, in which case C1N, C2N and C3N is shorted internally and connected to the CPN pin.

Comparators Block Diagram

Comparator Operation

The device contains four comparator functions which are used to compare two analog voltages and provide an output based on their difference. Any pull-high resistors connected to the shared comparator input pins will be automatically disconnected when the comparator is enabled. As the comparator inputs approach their switching level, some spurious output signals may be generated on the comparator output due to the slow rising or falling nature of the input signals. This can be minimised by selecting the hysteresis function will apply a small amount of positive feedback to the comparator. Ideally the comparator should switch at the point where the positive and negative inputs signals are at the same voltage level. However, unavoidable input offsets introduce some uncertainties here. The hysteresis function, if enabled, also increases the switching offset value.

Comparator Register

The CMPC control register controls othe hysteresis functions and on/off control of the four comparators. As the comparator 0 is used in the over current detector, there will be more description about it in the associated chapter.

CMPC Register

Bit	7	6	5	4	3	2	1	0
Name	C3HYEN	C2HYEN	C1HYEN	C0HYEN	C3EN	C2EN	C1EN	C0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	0	0	0	0

Bit 7 **C3HYEN**: Comparator 3 hysteresis control
0: Disable
1: Enable
This is the comparator 3 hysteresis control bit and if set high will apply a limited amount of hysteresis to the comparator, as specified in the Comparators Electrical Characteristics table. The positive feedback induced by hysteresis reduces the effect of spurious switching near the comparator threshold.

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

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

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

Bit 3 **C3EN**: Comparator 3 on/off control
0: Off
1: On
This is the Comparator 3 on/off control bit. If the bit is zero the comparator 3 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 3 is not used or before the device enters the SLEEP or IDLE mode.

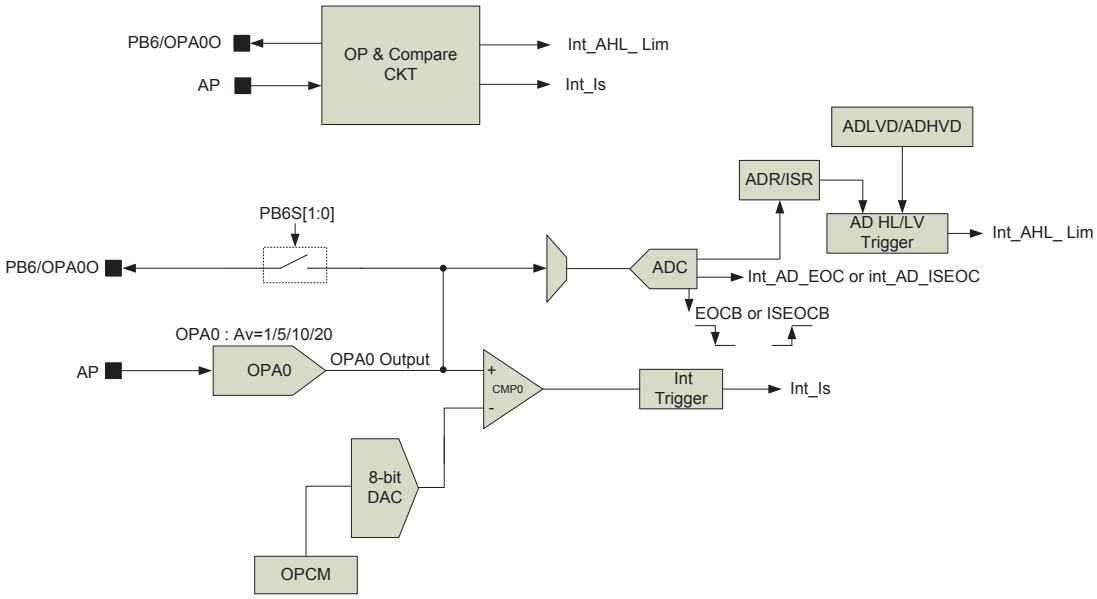
Bit 2 **C2EN**: Comparator 2 on/off control
0: Off
1: On
This is the Comparator 2 on/off control bit. If the bit is zero the comparator 2 will be switched off and no power consumed even if analog voltages are applied to its inputs. For power sensitive applications this bit should be cleared to zero if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

Bit 1 **C1EN**: Comparator 1 on/off control
0: Off
1: On
This is the Comparator 1 on/off control bit. If the bit is zero the comparator 1 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 1 is not used or before the device enters the SLEEP or IDLE mode.

Bit 0 **C0EN**: Comparator 0 on/off control
0: Off
1: On
This is the Comparator 0 on/off control bit. If the bit is zero the comparator 0 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 0 is not used or before the device enters the SLEEP or IDLE mode.

Over Current Detection

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



Over Current Detector Block Diagram

Over Current Detect Functional Description

The over current functional block includes an amplifier OPA0, 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 are generated which can be used for over current detection.

- A/D Converter interrupt – Int_AHL_Lim
- Comparator 0 interrupt – Int_Is

Over Current Detect Register

There are three registers to control the function and operation of the over current detection circuits, known as OPOMS, OPCM and OPA0CAL. These 8-bit registers define functions such as comparator 0 interrupt edge control, OPA0 operation mode selection and OPA0 calibration. The OPCM register is an 8-bit DAC register used for OPA0 comparison.

OPOMS Register

Bit	7	6	5	4	3	2	1	0
Name	CMP0_EG1	CMP0_EG0	—	—	—	OPA0VS2	OPA0VS1	OPA0VS0
R/W	R/W	R/W	—	—	—	R/W	R/W	R/W
POR	0	0	—	—	—	0	0	0

Bit 7~6 **CMP0_EG1~CMP0_EG0**: Interrupt edge control for Comparator 0
00: Comparator 0 and D/A converter disabled
01: Rising edge trigger
10: Falling edge trigger
11: Dual edge trigger

Bit 5~4 Unimplemented, read as "0"

Bit 2~0 **OPA0VS2~OPA0VS0**: OPA0 gain selection
000: OPA0 disabled
001: Av=5
010: Av=10
011: Av=20
100~110: Undefined
111: Av=1
Note that when the OPA0 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	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**: 8-bit D/A converter Register bit 7 ~ bit 0

OPA0CAL Register

Bit	7	6	5	4	3	2	1	0
Name	—	A0RS	A0OFM	A0OF4	A0OF3	A0OF2	A0OF1	A0OF0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	1	0	0	0	0

Bit 7 Unimplemented, read as "0"

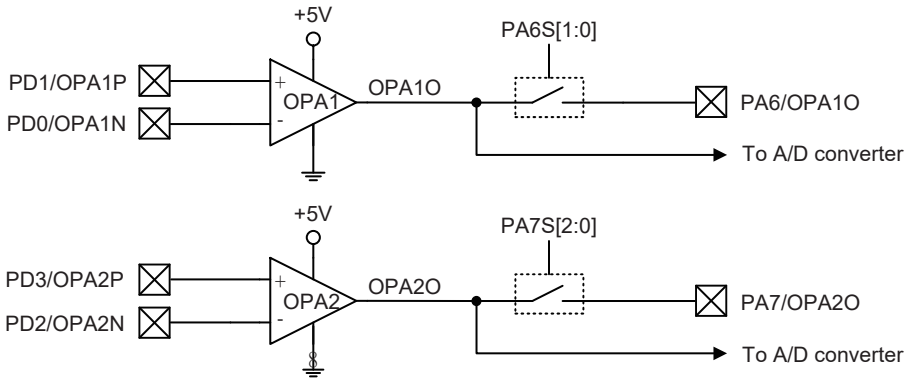
Bit 6 **A0RS**: Operational Amplifier input offset calibration on/off control
0: Off
1: On

Bit 5 **A0OFM**: Normal or Calibration Mode selection
0: Normal Mode
1: Offset Calibration Mode

Bit 4~0 **A0OF4~A0OF0**: Operational Amplifier input offset voltage calibration control
00000: Minimum
10000: Center
11111: Maximum

Phase Current Detection

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



Phase Current Detector Block Diagram

Phase Current Detect Functional Description

The phase current detect function can be implemented using OPA1 and OPA2. The OPA1 and OPA2 outputs can be measured by connecting to the A/D converter. These two signals can also be output on PA6 and PA7 respectively by properly configuring the associated pin-shared control bits. A kind of A/D converter interrupt, i.e, Int_AHL_Lim, can be used for phase current detection, which will be introduced more in the A/D Converter chapter.

Phase Current Detect Register

There are two registers to control the on/off function of the phase current detection circuits, known as OPA1CAL and OPA2CAL.

OPA1CAL Register

Bit	7	6	5	4	3	2	1	0
Name	OPA1ON	—	—	—	—	—	—	—
R/W	R/W	—	—	—	—	—	—	—
POR	0	—	—	—	—	—	—	—

Bit 7 **OPA1ON**: OPA1 on/off control
0: Disable, OPA1 off
1: Enable, OPA1 on

Bit 6~0 Unimplemented, read as "0"

OPA2CAL Register

Bit	7	6	5	4	3	2	1	0
Name	OPA2ON	—	—	—	—	—	—	—
R/W	R/W	—	—	—	—	—	—	—
POR	0	—	—	—	—	—	—	—

Bit 7 **OPA2ON**: OPA2 on/off control
0: Disable, OPA2 off
1: Enable, OPA2 on

Bit 6~0 Unimplemented, read as "0"

BLDC Motor Control Circuit

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

Functional Description

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

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

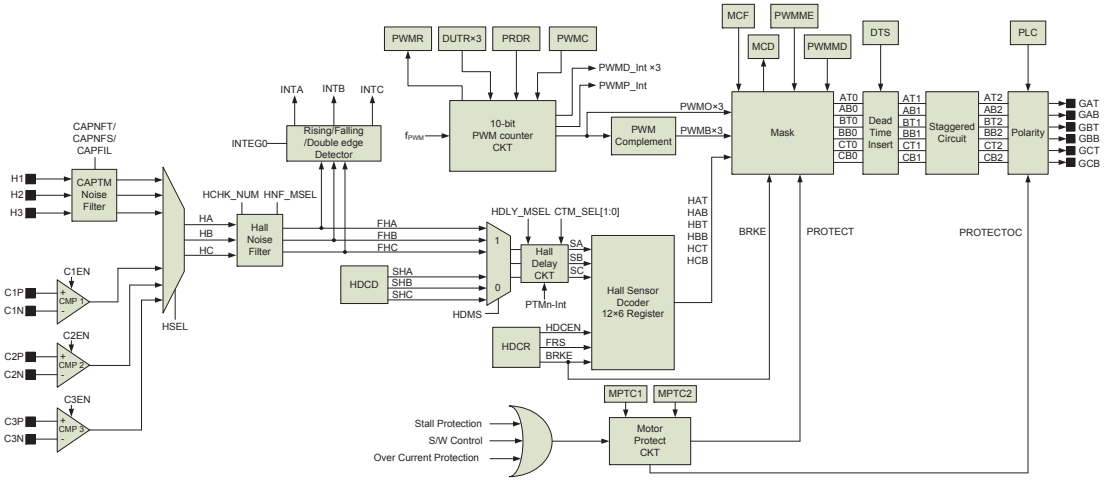
The Dead-Time insertion circuit is used to ensure the upper and lower Gate Driver Transistor Pairs are not enabled simultaneously to prevent the occurrence of a virtual power short circuit. The dead time is selected under software control.

The Staggered circuit can force all the outputs to an off status if the software detects an error condition which could be due to external factors such as ESD problems or both upper and lower external Gate Driver Transistor pairs being simultaneously on.

The Polarity circuit can select the output polarity of the BLDC motor output control port to support many different types of external MOS gate drive device circuit combinations.

The Motor Protect circuit includes many detection circuits for functions such as a motor stall condition, over current condition, etc.

The Hall Sensor Decoder circuit is a six-step system which can be used control the motor direction. Twelve registers, each using 6 bits, are used to control the direction of the motor. The motor forward, backward, brake and free running functions are controlled by the HDCD/HDCR registers. The HA/HB/HC or SHA/SHB/SHC can be selected as the Hall Sensor Decoder circuit inputs.

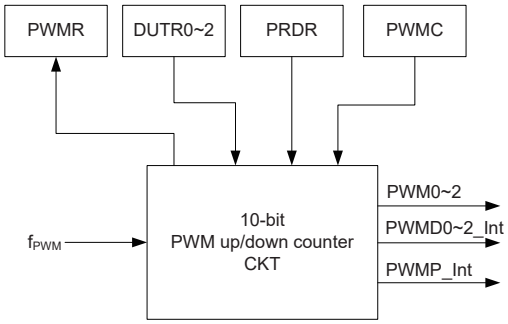


Note: GAT, GAB, GBT, GBB, GCT, GCB == PWM0H, PWM0L, PWM1H, PWM1L, PWM2H, PWM2L.

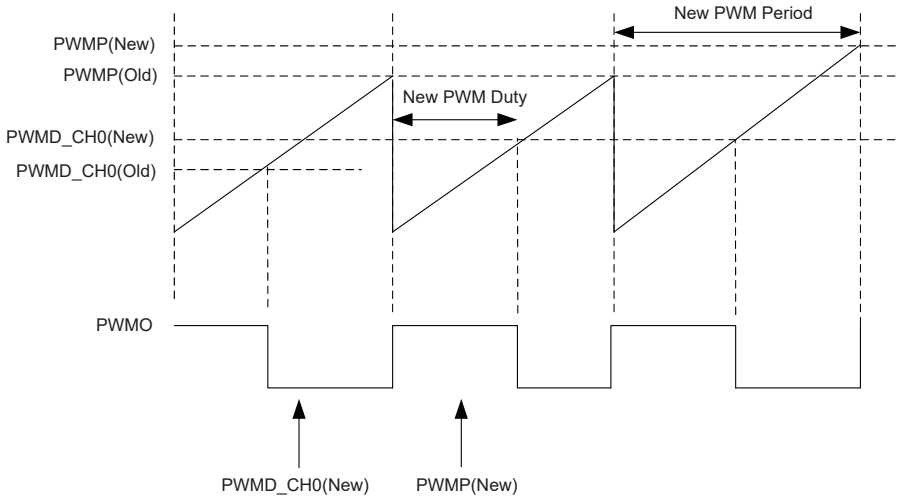
BLDC Motor Control Block Diagram

PWM Counter Control Circuit

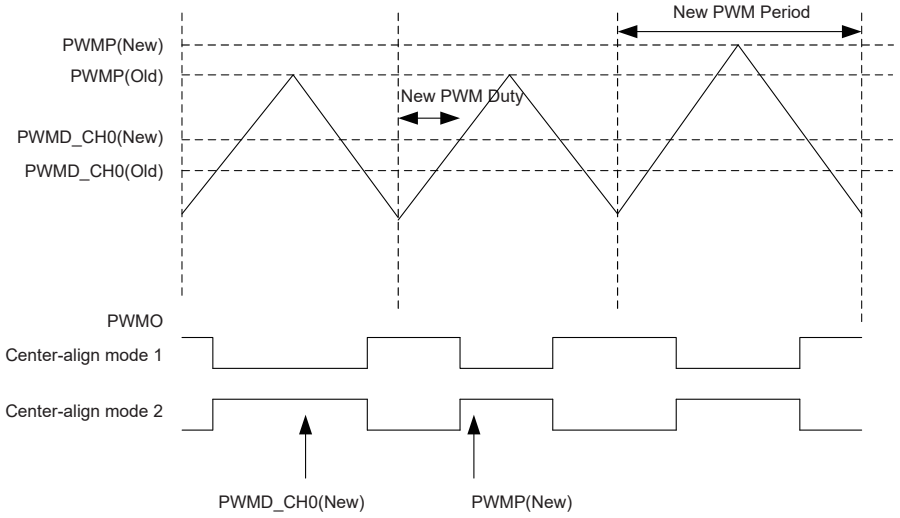
The device includes a 10-bit PWM generator. The PWM signal has both adjustable duty cycle and frequency that can be setup by programming 10-bit values into the corresponding PWM registers.



PWM Block Diagram



PWM 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 three PWM duty values. This will generate undesired PWM duty outputs and lead to control errors. To improve this problem, two methods are provided for synchronous update of three PWM duty values.

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

PWM Register Description

Overall PWM operation is controlled by a series of registers. The DUTRnL/DUTRnH register pair is used for PWM duty control for adjustment of the motor output power. The PRDRL/PRDRH register pair are used together to form a 10-bit value to setup the PWM period for PWM frequency adjustment. Being able to change the PWM frequency is useful for motor characteristic matching for problems such as noise reduction and resonance. The PWMRL/PWMRH registers are used to monitor the PWM counter dynamically. The PWMON bit in the PWMC register is the 10-bit PWM counter on/off bit. The PWM clock source for the PWM counter can be selected by PCKS1~PCKS0 bits in the PWMC register. The PWMMS bit field in the PWMC register determines the PWM alignment type, which can be either edge or center type. The PWMCS register is used for PWM DUTR0~DUTR2 duty synchronisation control. Note that the DUTR3L/DUTR3H register pair does not have an actual PWM output path, it only provides an additional PWM duty for BLDC motor sine-wave calculation. The order of writing data to PWM register is high byte first and then low byte.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PWMC	PWMMS1	PWMMS0	PCKS1	PCKS0	PWMON	ITCMS1	ITCMS0	PWMLD
PWMCS	—	—	—	—	—	PWMSUF	PWMSU	PWMSV
DUTR0L	D7	D6	D5	D4	D3	D2	D1	D0
DUTR0H	—	—	—	—	—	—	D9	D8
DUTR1L	D7	D6	D5	D4	D3	D2	D1	D0
DUTR1H	—	—	—	—	—	—	D9	D8
DUTR2L	D7	D6	D5	D4	D3	D2	D1	D0
DUTR2H	—	—	—	—	—	—	D9	D8
DUTR3L	D7	D6	D5	D4	D3	D2	D1	D0
DUTR3H	—	—	—	—	—	—	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 Registers 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 mode selection
0x: Edge-aligned mode
10: Center-aligned mode 1
11: Center-aligned mode 2

Bit 5~4 **PCKS1~PCKS0**: PWM counter clock source selection (f_{PWM} based on f_H)
00: f_{PWM}
01: $f_{PWM}/2$
10: $f_{PWM}/4$
11: $f_{PWM}/8$
PWM frequency=1/{PWMR×[1/(f_{PWM}/n)]}

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

Bit 2~1 **ITCMS1~ITCMS0**: PWM center-aligned mode duty interrupt control
00: Disable center-aligned mode duty interrupt
01: Center-aligned mode duty interrupt only in count up condition
10: Center-aligned mode duty interrupt only in count down condition
11: Center-aligned mode duty interrupt in count up or down condition

Bit 0 **PWMLD**: PWM PRDR and DUTRn (n=0~3) register update control
0: The register values of PRDR and DUTRn (n=0~3) are never loaded to counter and comparator registers
1: The PRDR register value will be loaded to counter register after counter underflow, and hardware will clear it by next clock cycle

- **PWMCS Register**

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

Bit 7~3 Unimplemented, read as "0"

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

Bit 1 **PWMSU**: PWM DUTR0~2 duty synchronous update control
0: Disable
1: Enable

Bit 0 **PWMSV**: PWM duty written to DUTR0 also written to DUTR1 and DUTR2
0: Disable, DUTR0~2 are configured seperately
1: Enable, PWM duty written to DUTR0 is also written to DUTR1 and DUTR2

- **DUTRnL Register (n=0~3)**

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

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

- **DUTRnH Register (n=0~3)**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **D9~D8**: 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	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**: 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	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

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

- **PWMRL Register**

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

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

- **PWMRH Register**

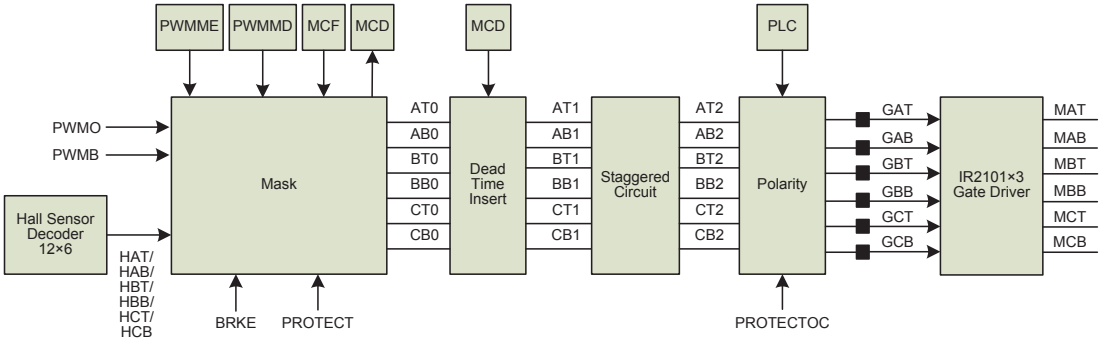
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

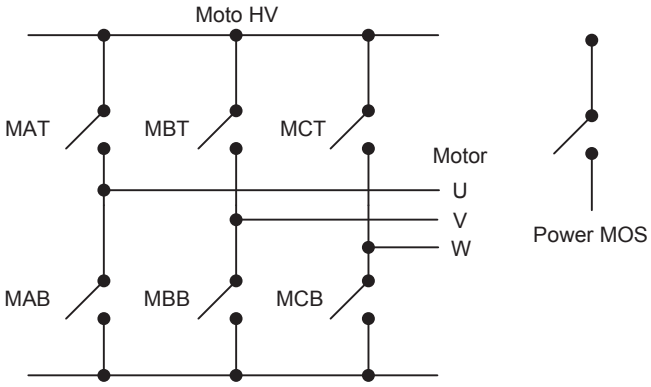
Bit 1~0 **D9~D8**: 10-bit PWM Counter High Byte Register bit 1 ~ bit 0
10-bit PWM Counter bit 9 ~ bit 8

Mask Function

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



Mask Function Block Diagram



Mask Switching

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

Normal Mode

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

- When PWMS =0, the bottom port PWM output selects transistor pair bottom arm GAB/ GBB/ GCB.
- When PWMS =1, the top port PWM output selects transistor pair top arm, GAT/ GBT/ GCT.
- When MPWE =0, the PWM output is disabled and AT0/BT0/CT0/AB0/BB0/CB0 are all on.
- When MPWE =1, the PWM output is enabled and AT0/BT0/CT0/AB0/BB0/CB0 can output a variable PWM signal for speed control.
- When MPWMS=0, the PWM has a Complementary output.
- When MPWMS=1, the PWM has a Non-complementary output.
- 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 PWM Mode selection
0: Complementary output
1: Non-complementary output
This bit selection is invalid when in the Software Mask Mode where the PWM mode selection is determined by the PWMME register.

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

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

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

- **MCD Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	GAT	GAB	GBT	FHC	FHB	FHA
R/W	—	—	R	R	R	R	R	R
POR	—	—	0	0	0	x	x	x

"x": unknown

Bit 7~6 Unimplemented, read as "0"

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

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

Hardware Mask Mode

- **Complementary control, MPWMS=0**

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

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

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

- **Non-complementary control, MPWMS=1**

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

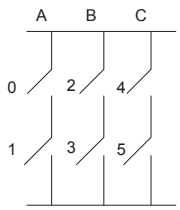
	HBT	HBB	BT0	BB0		HBT	HBB	BT0	BB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	0	PWMO	PWMS=1	0	1	0	1
	1	0	1	0		1	0	PWMO	0
	1	1	0	0		1	1	0	0

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

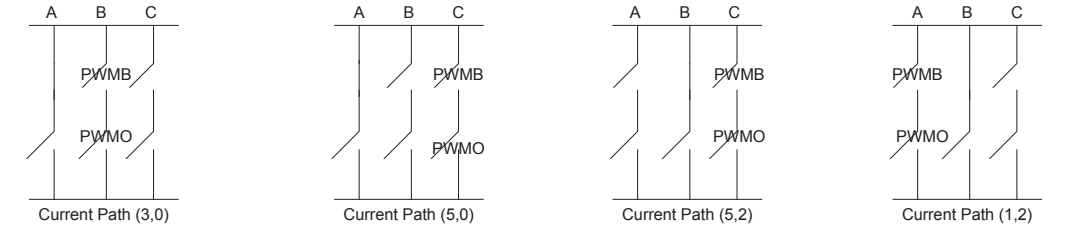
Software Mask Mode

To control the software Mask circuit, two registers known as PWMME and PWMMD are provided. PWMME register is used to control PWM signal and PWMMD is used to determine the MOS Gate Driver Circuit is on or off.

3-phase inverter Symbol



Mask Complement Mode Example



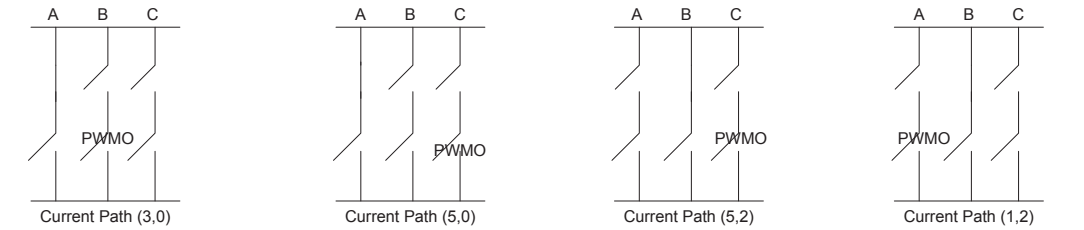
PMEN			PMD		
1	0	1	1	x	0
1	0	1	0	x	0

PMEN			PMD		
1	1	0	1	0	x
1	1	0	0	0	x

PMEN			PMD		
1	1	0	0	1	x
1	1	0	0	0	x

PMEN			PMD		
0	1	1	x	1	0
0	1	1	x	0	0

Mask Independent Mode Example



PMEN			PMD		
1	1	1	1	0	0
1	0	1	0	x	0

PMEN			PMD		
1	1	1	1	0	0
1	1	0	0	0	x

PMEN			PMD		
1	1	1	0	1	0
1	1	0	0	0	x

PMEN			PMD		
1	1	1	0	1	0
0	1	1	x	0	0

Software Mask Mode Circuit

Note: 1. If the mask mode is enabled, when GxT and GxB (x=A, B or C) are masked simultaneously, the two lines of each pair, PMD0 and PMD1, PMD2 and PMD3, PMD4 and PMD5, can not be set to "1" simultaneously. If they are all high in the same time, switch 2n and switch 2n+1 will output "0".

2. If PWM and complementary PWM are enabled simultaneously, one of GxT and GxB (x=A, B or C) outputs PWM and the other one can not be masked to "1" but outputs "0" automatically by hardware.
3. If the GxT and GxB (x=A, B or C) pins are configured as I/O function, then PWM Mask function will be invalid.

- **PWMME Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **PMEn**: PWM mask enable bit (n=0~5)
0: PWM generator signal is output to the next stage
1: PWM generator signal is masked and PMDn is output to the next stage

- **PWMMD Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **PMDn**: PWM mask data bit when PMEn=1 (n=0~5)
0: Output logic low
1: Output logic high

Brake Mode

The brake mode has the highest priority than other modes. When this mode is activated, the external gate driver transistor pair top arm will be off and the bottom arm will be on. The brake truth decode table is shown below.

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

Motor Protect Mode

When the motor protect mode is activated, the external gate driver transistor pair can select brake, where the top arm is off and the bottom arm is on, or select free running where the top and tottom arm are both off. The protection decode table is shown below.

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

Other Functions

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

Dead Time Function

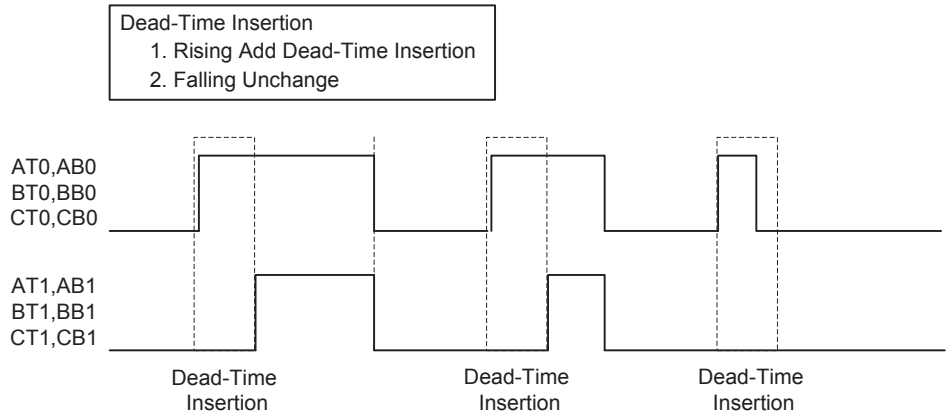
During transistor pair switching, the Dead Time function is used to prevent both upper and lower transistor pairs from conducting at the same time thus preventing a virtual short circuit condition from occurring. The actual dead time value can be setup to be within a value from 0.3μs to 5μs which is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

The Dead Time Insertion circuit requires six independent output circuits:

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

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

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



Dead Time Insertion Timing

A single regiter, DTS, is delicated for use by the Dead Time function.

- **DTS Register**

Bit	7	6	5	4	3	2	1	0
Name	DTCKS1	DTCKS0	DTE	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **DTCKS1~DTCKS0**: Dead Time clock source selection (f_{SYS}=f_H)
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 insertion control
0: Disable
1: Enable

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

Staggered Function

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

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

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

Polarity Control Function

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

- **PLC Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PCBC**: C pair Bottom port gate output control
0: Non-inverse output
1: Inverse output

Bit 4 **PCTC**: C pair Top port gate output control
0: Non-inverse output
1: Inverse output

Bit 3 **PBBC**: B pair Bottom port gate output control
0: Non-inverse output
1: Inverse output

Bit 2 **PBTC**: B pair Top port gate output control
0: Non-inverse output
1: Inverse output

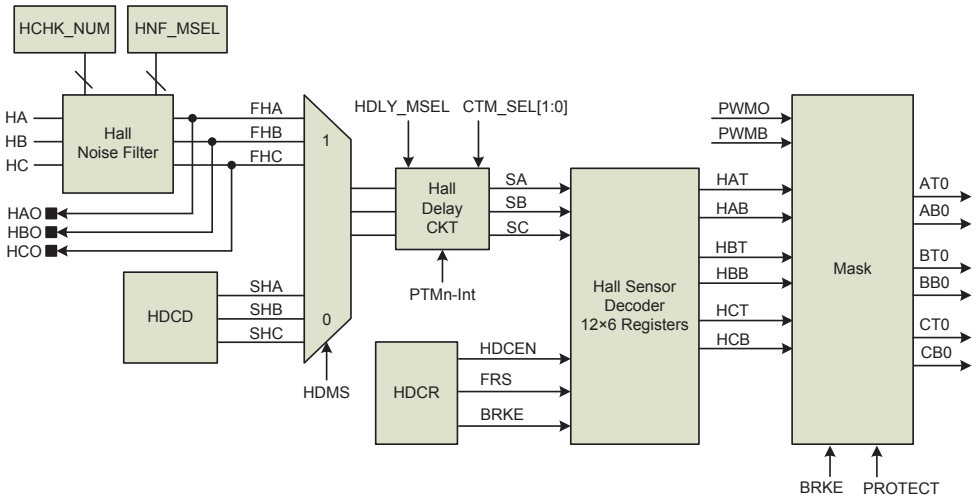
Bit 1 **PABC**: A pair Bottom port gate output control
0: Non-inverse output
1: Inverse output

Bit 0 **PATC**: A pair Top port gate output control
0: Non-inverse output
1: Inverse output

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

Hall Sensor Decoder

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

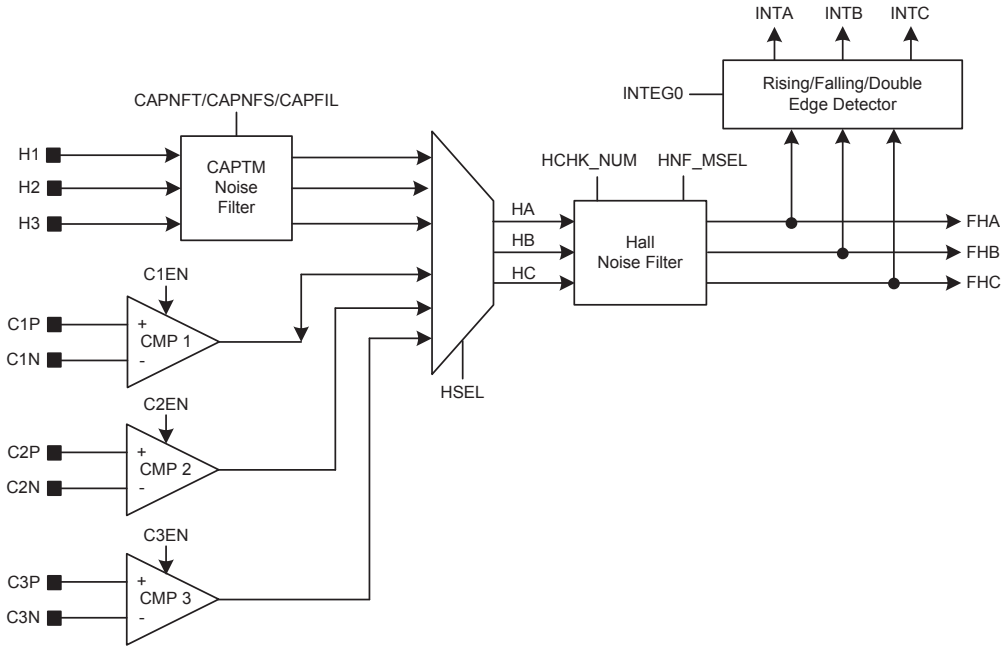


Hall Sensor Decoder Block Diagram

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

Hall Sensor Noise Filter

This device includes a Hall Noise Filter function to filter out the effects of noise generated by the large switching currents of the motor driver. This generated noise may affect the Hall Sensor inputs (H1/H2/H3), which in turn may result in incorrect Hall Sensor output decoding.



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

Hall Sensor Noise Fliter Blick Diagram

Several registers are used to control the noise filter. The HNF_EN bit in the HNF_MSEL register is used as the overall enable/disable bit for the noise filter. It is necessary to enable CMP1, CMP2 and CMP3 hysteries function before the camparators are used during motor control sensorless applications.

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

Hall Sensor Noise Filter Enable

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

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

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

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

Hall Sensor Delay Function

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

- Step 1
  Set the Hall Decode table to select either the phase forward or phase backward function.
- Step 2
  Select which TM is used to generate the Delay Time by programming the CTM_SEL1~CTM_SEL0 bits and set the selected TM to run in the Compare Match Output Mode.
- Step 3
  Use the HDLY_MSEL bit to select the Hall Delay circuit operating mode. The default value of HDLY_MSEL is zero which will disable the Hall Delay circuit. If the HDLY_MSEL bit is set high, then the Hall Delay circuit will be enabled.
- Step 4
  Enable the Hall Decoder using the HDCEN bit.

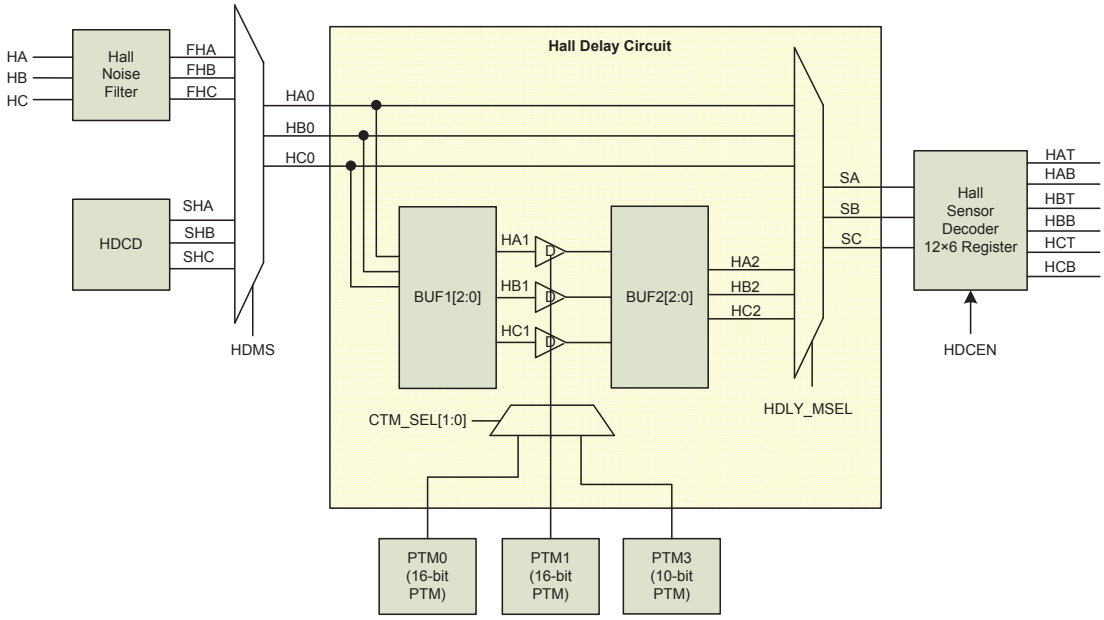
The following points should be noted regarding the HDLY_MSEL bit.

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

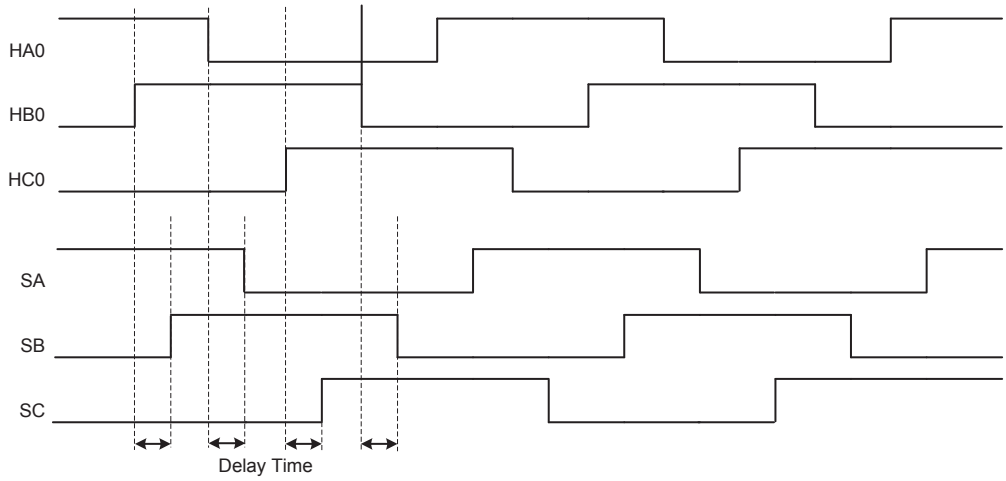
With regard to the selected PTM functions the following notes should be taken before the Delay function is enabled.

- Keep PTnON=0 and PTnPAU=0.
- The PTM should be set in the Compare Match Output Mode.
- Set PTnCCLR=1, therefore the PTM counter is cleared with a comparator A match condition.
- Setup the Delay time by properly setting PTMn CCRA and TCKn.

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



Delay Function Block Diagram



Delay Function Timing

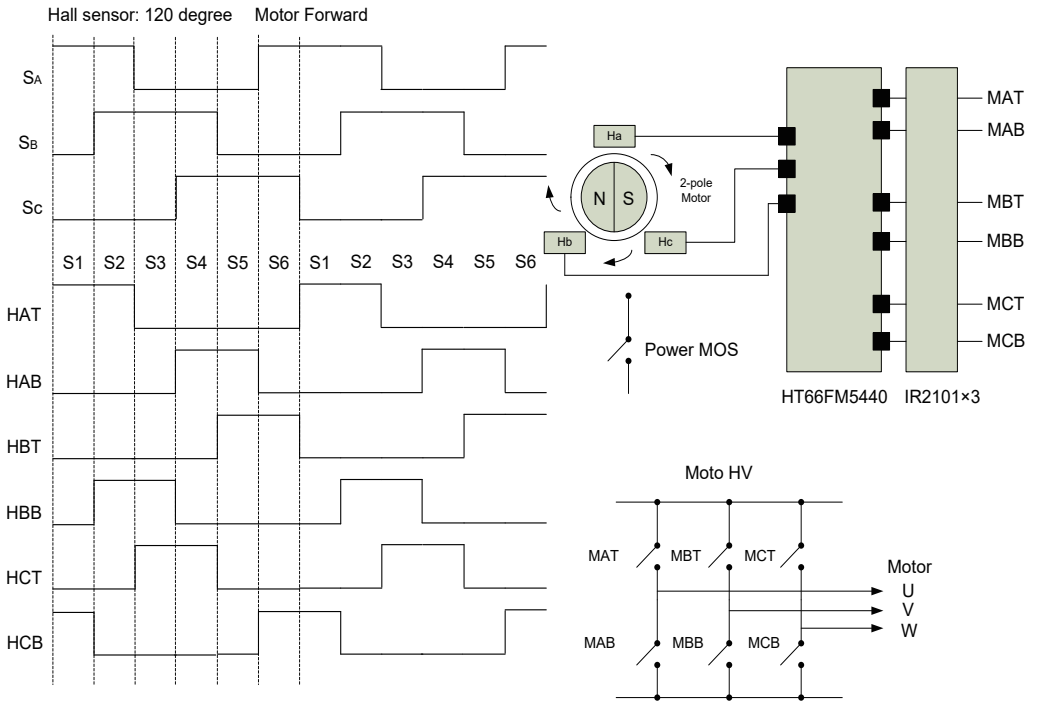
Motor Control Drive Signals

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

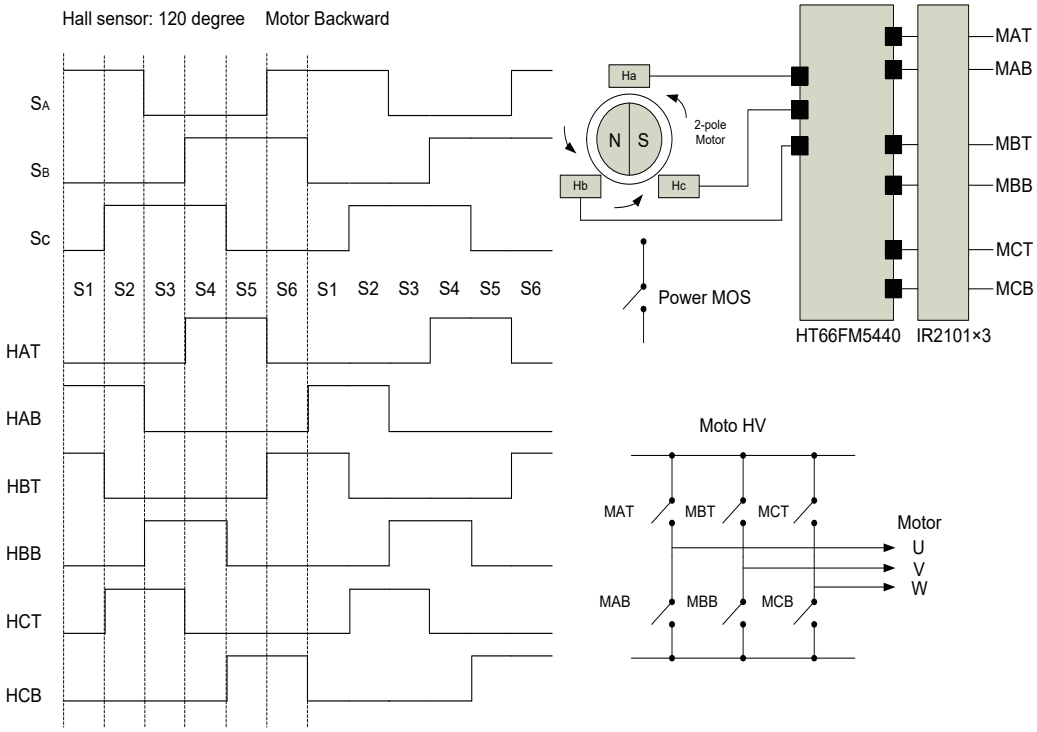
The accompanying tables show the truth tables for each of the registers.

	60 Degree			120 Degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Forward (HDCEN=1, FRS=0, BRKE=0)	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
	1	0	0	1	0	0	HDCT0[5:0]					
	1	1	0	1	1	0	HDCT1[5:0]					
	1	1	1	0	1	0	HDCT2[5:0]					
	0	1	1	0	1	1	HDCT3[5:0]					
	0	0	1	0	0	1	HDCT4[5:0]					
	0	0	0	1	0	1	HDCT5[5:0]					
Backword (HDCEN=1, FRS=1, BRKE=0)	60 Degree			120 Degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
	1	0	0	1	0	0	HDCT6[5:0]					
	1	1	0	1	1	0	HDCT7[5:0]					
	1	1	1	0	1	0	HDCT8[5:0]					
	0	1	1	0	1	1	HDCT9[5:0]					
	0	0	1	0	0	1	HDCT10[5:0]					
	0	0	0	1	0	1	HDCT11[5:0]					
Brake (BRKE=1, HDCEN=X, FRS=X)	60 Degree			120 Degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
	V	V	V	V	V	V	0	1	0	1	0	1
Hall Decoder Disable (HDCEN=0) Free Running	60 Degree			120 Degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
	V	V	V	V	V	V	0	0	0	0	0	0
Hall Decoder Error (HDCEN=X)	60 Degree			120 Degree			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
	1	0	1	1	1	1	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0

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



Motor Drive Signal Timing Diagram – Forward Direction



Motor Drive Signal Timing Diagram – Backward Direction

Hall Sensor Decoder Register Description

The HDCR register is the Hall Sensor Decoder control register, HDCD is the Hall Sensor Decoder input data register, and HDCT0~HDCT11 are the Hall Sensor Decoder tables. The HCHK_NUM register is the Hall Noise Filter check number register and HNF_MSEL is the Hall Noise Filter Mode select register. The INTEG0 register is Hall Noise Filter input source selection and output interrupt edge control register.

- **INTEG0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	HSEL	INTCS1	INTCS0	INTBS1	INTBS0	INTAS1	INTAS0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **HSEL**: HA/HB/HC source selection
0: H1/H2/H3
1: CMP1/CMP2/CMP3 output

Bit 5~4 **INTCS1~INTCS0**: FHC Interrupt edge control for INTC
00: Disable
01: Rising edge trigger
10: Falling edge trigger
11: Dual edge trigger

Bit 3~2 **INTBS1~INTBS0**: FHB Interrupt edge control for INTB
00: Disable
01: Rising edge trigger
10: Falling edge trigger
11: Dual edge trigger

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

- **HDCR Register**

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

Bit 7~6 **CTM_SEL1~CTM_SEL0**: TM select for the Hall Delay Circuit
00: PTM0 (16-bit PTM)
01: PTM1 (16-bit PTM)
10: PTM3 (10-bit PTM)
11: Unused

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

Bit 4 **HALS**: Hall Sensor Decoder Degree selection
0: Hall Sensor 60 degree
1: Hall Sensor 120 degree

Bit 3 **HDMS**: Hall Sensor Decoder Mode selection
0: Software Mode (SHA/SHB/SHC in the HDCD register)
1: Hall Sensor Mode (FHA/FHB/FHC outputs from noise filter)

Bit 2 **BRKE**: Motor brake control
0: AT/BT/CT/AB/BB/CB=V
1: AT/BT/CT=0, AB/BB/CB=1

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

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

- **HDCD Register**

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

Bit 7~3 Unimplemented, read as "0"

Bit 2 **SHC**: Software Hall C

Bit 1 **SHB**: Software Hall B

Bit 0 **SHA**: Software Hall A

- **HDCTn Register (n=0~11)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	HATDn	HABDn	HBTDn	HBBDn	HCTDn	HCBDn
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **HATDn**: AT output state control
0: Output 0
1: Output 1

Bit 4 **HABDn**: AB output state control
0: Output 0
1: Output 1

Bit 3 **HBTDn**: BT output state control
0: Output 0
1: Output 1

Bit 2 **HBBDn**: BB output state control
0: Output 0
1: Output 1

Bit 1 **HCTDn**: CT output state control
0: Output 0
1: Output 1

Bit 0 **HCBDn**: CB output state control
0: Output 0
1: Output 1

- **HCHK_NUM Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	HCK_N4	HCK_N3	HCK_N2	HCK_N1	HCK_N0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **HCK_N4~HCK_N0**: Hall Noise Filter check times

- **HNF_MSEL Register**

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

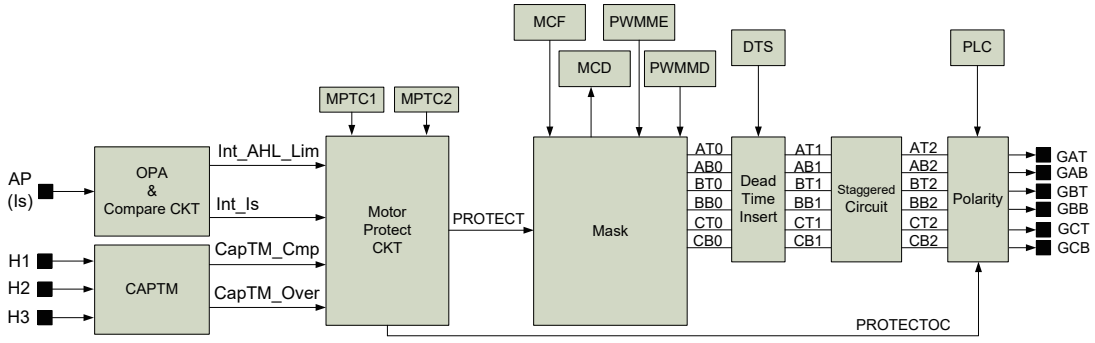
Bit 7~4 Unimplemented, read as "0"

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

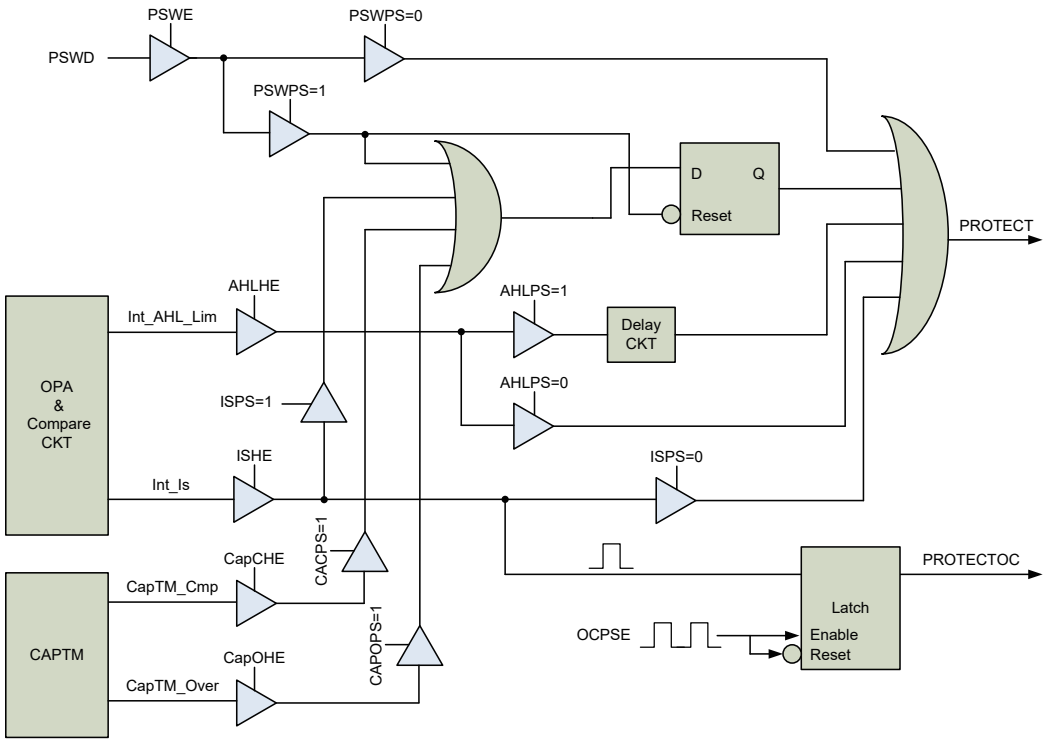
Bit 2~0 **HFR_SEL2~HFR_SEL0**: Hall Noise Filter clock source selection (f_{SYS}=f_H)
000: f_{SYS}/2
001: f_{SYS}/4
010: f_{SYS}/8
011: f_{SYS}/16
100: f_{SYS}/32
101: f_{SYS}/64
110: f_{SYS}/128
111: Unuse

Motor Protection Function

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



Protection Function Block Diagram



Protection Function Control

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 using software

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

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

Additionally, the PROTECTOC mechanism is used to immediately switch off the drive signals. If an over current condition occurs, the drive signals will be switched off directly at the polarity stage and the preset OCPS protection signals will be output. When the PROTECTOC protection is finished, it needs to be disabled by software.

Current Protection Function

The device contains an internal OPA0, a high speed (2μs) 12-bit A/D Converter, an 8-bit D/A Converter and a comparator to measure the motor current and to detect excessive current values□If an over current situation should occur, then the external drive circuit can be shut down immediately to prevent motor damage. More details are provided in the Over Current Detection chapter.

As the motor driver PCB will have rather large amounts of noise, and as this noise will be amplified by the OPA0, this can easily lead to false triggering. For this reason the Fault Mode must be used.

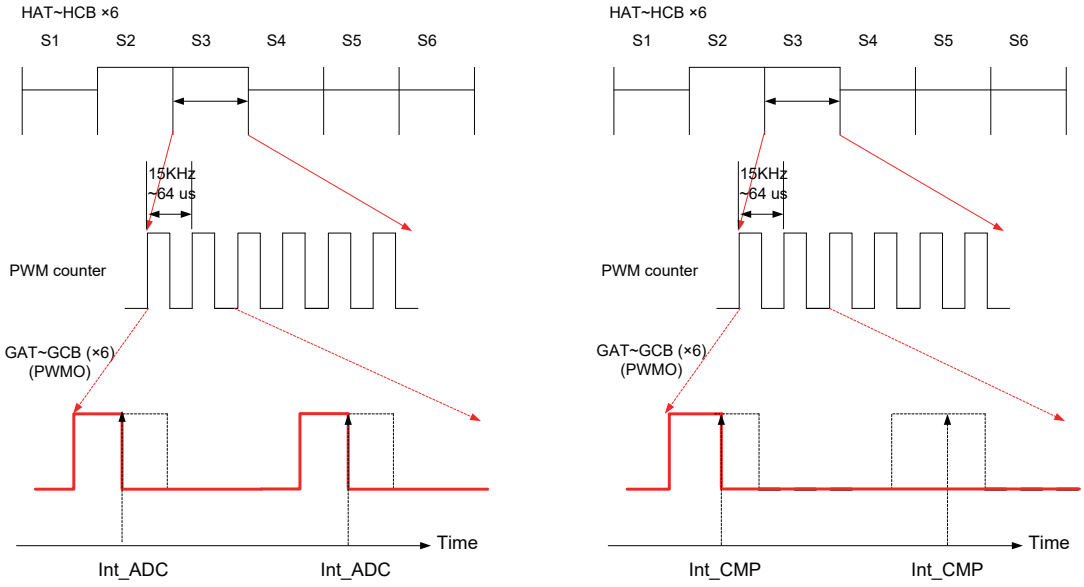
The PROTECTOC mechanism provides a more immediate current protection. Ensure that the ISHE bit has been set to select the hardware over current protection before setting the OCPSE bit high. After this the over current compare interrupt int_Is can be used to directly switch off the drive signals. Since Int_Is is a pulse signal, a latch component must be used to latch the over current trigger source. 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 remove 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 polarity drive signals will be recovered.

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

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

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

For the MOS over current mechanism Int_Is, when ISHE=0 the hardware mode is disabled. When ISHE=1 the hardware mode is enabled and the protect signals are generated according to the CMP0_EG[1:0] bits in the OPOMS register. It should be noted that only when CMP0_EG[1:0]=11 a interrupt will be generated, but no protection is activated. When ISPS=0, the Fault Mode is selected, when ISPS=1, the Pause Mode is selected.



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

Motor Stall Detection Function

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

Stall Detrect Mechanism CapTM_Cmp: when CapCHE=0 disable the hardware mode, and when CapCHE=1 enable the hadware mode. The stall detect mechanism must use the Pause Mode. CAPOPS=1, then select the Pause Mode.

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

Motor Protection 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	—	OCPSE
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 **PSWD**: Motor Protection Software Mode data
0: PSWD=0
1: PSWD=1

Bit 6 **PSWE**: Motor Protection Software Mode control
0: Disable
1: Enable
When the motor protection software mode has been enabled and triggered, is should be removed by setting this bit from high to low and then high again.

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

Bit 4 **CapCHE**: CapTM_Cmp Hardware Mode control
0: Disable
1: Enable

Bit 3 **ISHE**: Int_Is Hardware Mode control
0: Disable
1: Enable

Bit 2 **AHLHE**: Int_AHL_Lim Hardware Mode control
0: Disable
1: Enable

Bit 1 Unimplemented, read as "0"

Bit 0 **OCPSE**: PROTECTOC over current protection control
0: Disable
1: Enable
When the PROTECTOC protection mode has been enabled and triggered, it should be removed by clearing the OCPSE bit to zero and then setting it from low to high.

- **MPTC2 Register**

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

Bit 7~5 Unimplemented, read as "0"

Bit 4 **PSWPS**: Pause/Fault Mode selection in Software Mode
0: Fault Mode
1: Pause Mode

Bit 3 **AHLPS**: Int_AHL_Lim Pause/Fault Mode selection
0: Protection circuit allows immediate restart of PWM output when the Int_AHL_Lim interrupt has been reset.
1: Protection circuit 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
In the Pause Mode, to remove the Int_Is over current condition, users must setup in the order of PSWE=1 → PSWPS=1 → PSWE=0.

Bit 1 **CAPCPS**: CapTM_Cmp Pause Mode selection
0: Undefined
1: Pause Mode
Note that this bit must be set to 1 to select the Pause Mode for valid configuration when required.

Bit 0 **CAPOPS**: CapTM_Over Pause Mode selection
0: Undefined
1: Pause Mode
Note that this bit must be set to 1 to select the Pause Mode for valid configuration when required.

- **OCPS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	OCPCB	OCPCT	OCPBB	OCPBT	OCPAB	OCPAT
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **OCPCB**: C pair Bottom port gate output selection
0: Output 0
1: Output 1

Bit 4 **OCPCT**: C pair Top port gate output selection
0: Output 0
1: Output 1

Bit 3 **OCPBB**: B pair Bottom port gate output selection
0: Output 0
1: Output 1

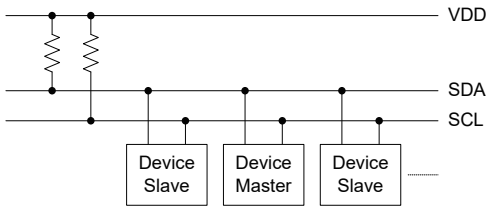
Bit 2 **OCPBT**: B pair Top port gate output selection
0: Output 0
1: Output 1

Bit 1 **OCPAB**: A pair Bottom port gate output selection
0: Output 0
1: Output 1

Bit 0 **OCPAT**: A pair Top port gate output selection
0: Output 0
1: Output 1

I²C Interface

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



I²C Master/Slave Bus Connection

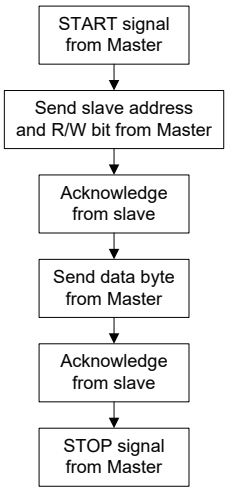
I²C Interface Operation

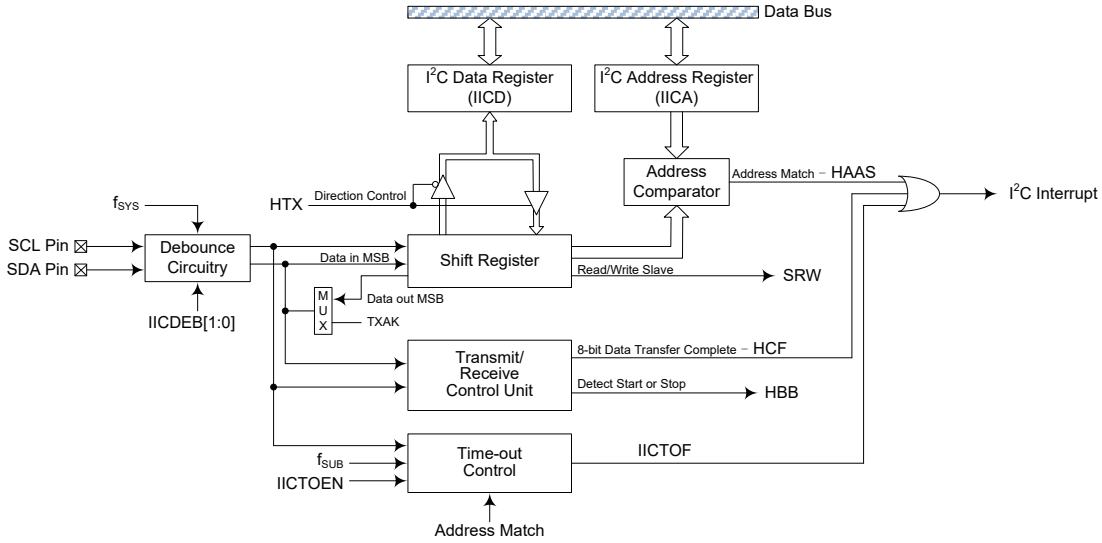
The I²C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs.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

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

It is suggested that the user should not allow the device to enter IDLE or SLEEP mode by application program during processing I²C communication.

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





I²C Block Diagram

The IICDEB1 and IICDEB0 bits determine the debounce time of the I²C interface. This uses the system clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required I²C data transfer speed, there exists a relationship between the system clock, f_{SYS}, and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

I²C Debounce Time Selection	I²C Standard Mode (100kHz)	I²C Fast Mode (400kHz)
No Debounce	f_{SYS} > 2 MHz	f_{SYS} > 5 MHz
2 system clock debounce	f_{SYS} > 4 MHz	f_{SYS} > 10 MHz
4 system clock debounce	f_{SYS} > 8 MHz	f_{SYS} > 20 MHz

I²C Minimum f_{SYS} Frequency

I²C Registers

There are four control registers associated with the I²C bus, IICC0, IICC1, IICA and IICTOC, and one data register, IICD. Further explanation on each of the bits is given below. The IICTOC register is described in the I²C Time-out Control section.

Register Name	Bit							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
IICC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
IICD	D7	D6	D5	D4	D3	D2	D1	D0
IICA	A6	A5	A4	A3	A2	A1	A0	—
IICTOC	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0

I²C Registers List

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

IICD 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	x	x	x	x	x	x	x	x

"x": unknown

Bit 7~0 **D7~D0**: I²C Data Buffer bit 7~bit 0

The IICA register is the location where the 7-bit slave address of the slave device is stored. When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the IICA register, the slave device will be selected.

IICA Register

Bit	7	6	5	4	3	2	1	0
Name	A6	A5	A4	A3	A2	A1	A0	—
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 7~1 **A6~A0**: I²C slave address

A6~A0 is the I²C slave address bit 6 ~ bit 0. Bits 7~ 1 of the IICA register define the device slave address. Bit 0 is not defined. When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the IICA register, the slave device will be selected.

Bit 0 Unimplemented, read as "0"

There are two control registers for the I²C interface, IICC0 and IICC1. The register IICC0 is used for I²C communication settings. The IICC1 register contains the relevant flags which are used to indicate the I²C communication status.

IICC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **IICDEB1~IICDEB0**: I²C Debounce Time Selection

00: No debounce
01: 2 system clock debounce
10: 4 system clock debounce
11: 4 system clock debounce

Bit 1 **IICEN**: I²C enable

0: Disable
1: Enable

The bit is the overall on/off control for the I²C interface. When the IICEN bit is cleared to zero to disable the I²C interface, the SDA and SCL lines will lose their I²C function and the I²C operating current will be reduced to a minimum value. When the bit is high the I²C interface is enabled. If the IICEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 Unimplemented, read as "0"

IICC1 Register

Bit	7	6	5	4	3	2	1	0
Name	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **HCF**: I²C Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer

The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Below is an example of the flow of a two-byte I²C data transfer.

First, I²C slave device receive a start signal from I²C master and then HCF bit is automatically cleared to zero.

Second, I²C slave device finish receiving the 1st data byte and then HCF bit is automatically set to one.

Third, user read the 1st data byte from IICD register by the application program and then HCF bit is automatically cleared to zero.

Fourth, I²C slave device finish receiving the 2nd data byte and then HCF bit is automatically set to one and so on.

Finally, I²C slave device receive a stop signal from I²C master and then HCF bit is automatically set to one.

Bit 6 **HAAS**: I²C Bus address match flag
0: Not address match
1: Address match

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

Bit 5 **HBB**: I²C Bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy

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

Bit 4 **HTX**: Select I²C slave device is transmitter or receiver
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK**: I²C Bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag

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

Bit 2 **SRW**: I²C Slave Read/Write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode

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

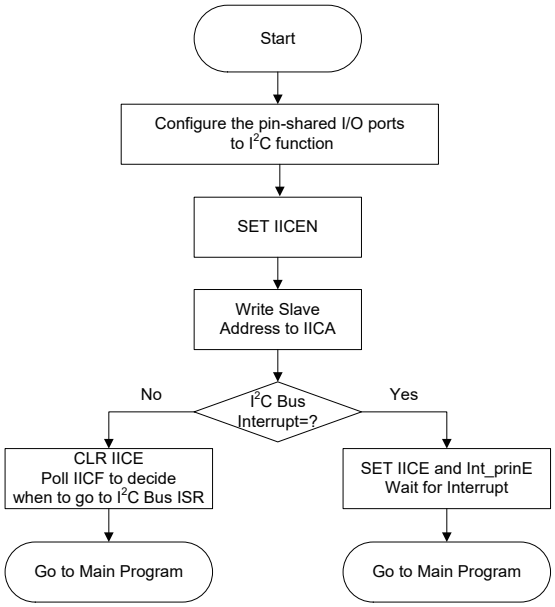
Bit 1 **IAMWU**: I²C Address Match Wake Up Control
0: Disable
1: Enable
This bit should be set to 1 to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

Bit 0 **RXAK**: I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave do not receive acknowledge flag
The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

I²C Bus Communication

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

- Step 1
  Configure the pin-shared I/O ports to I²C function pins and set the IICEN bit in the IICC0 register to "1" to enable the I²C bus.
- Step 2
  Write the slave address of the device to the I²C bus address register IICA.
- Step 3
  Set IICE and the corresponding interrupt priority enable bit of the interrupt control registers to enable the I²C interrupt and the related interrupt priority.



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

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

I²C Slave Address

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

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

I²C Bus Read/Write Signal

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

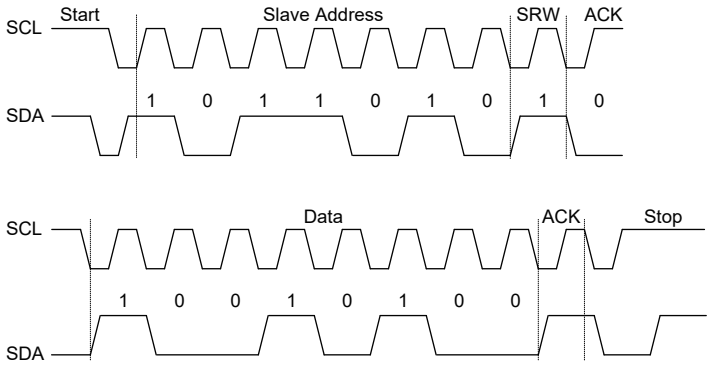
I²C Bus Slave Address Acknowledge Signal

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

I²C Bus Data and Acknowledge Signal

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

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

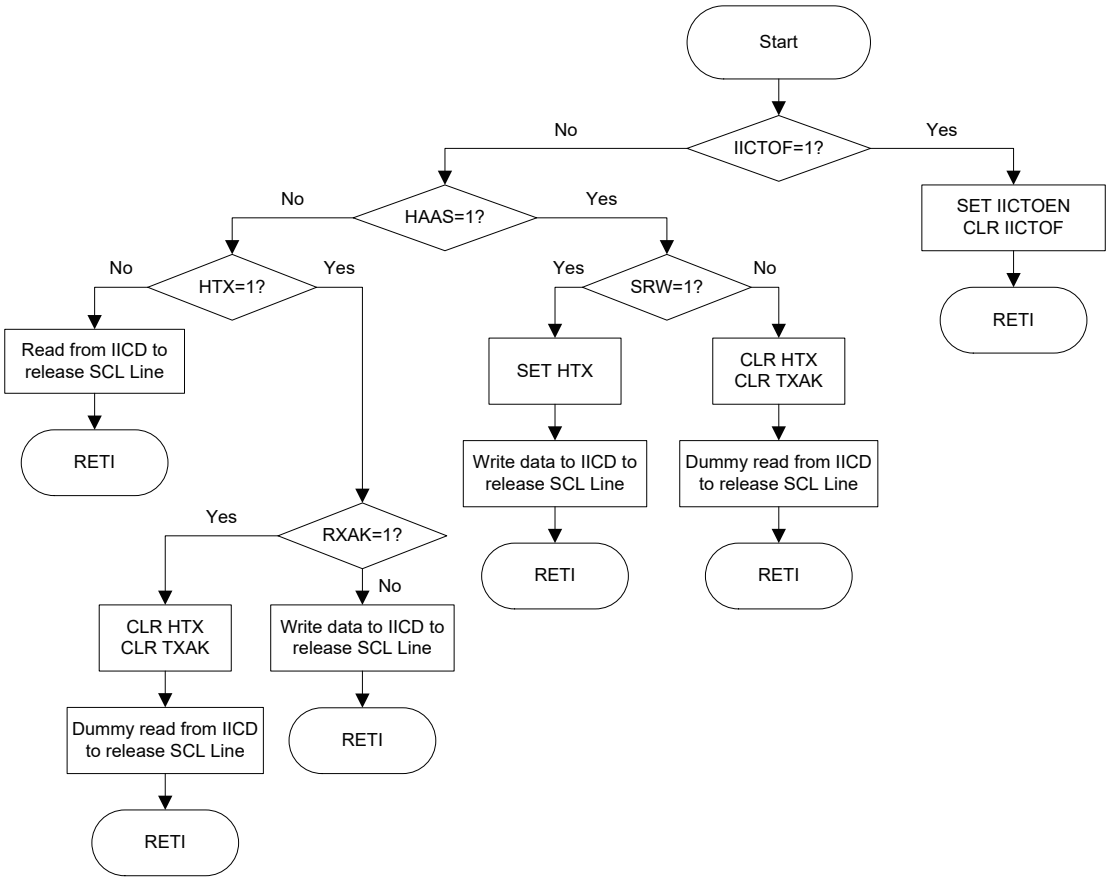


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	·····	S	SA	SR	M	D	A	D	A	·····	P

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

I²C Communication Timing Diagram

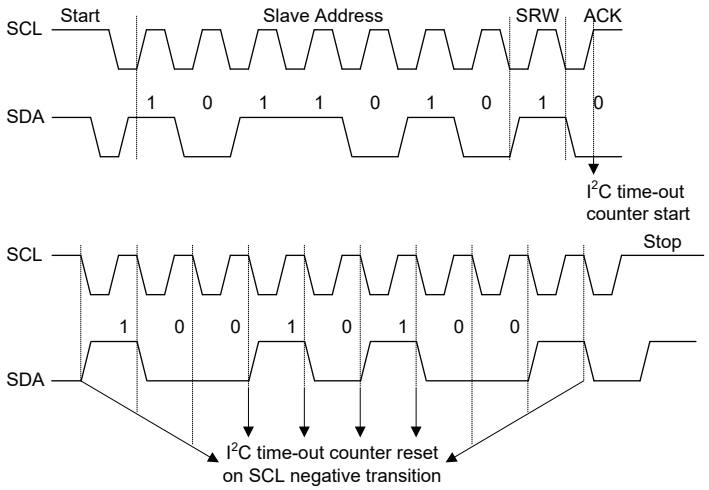


I²C Bus ISR Flow Chart

I²C Time-out Control

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

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



I²C Time-out

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

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

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

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

This gives a range of about 1ms to 64ms.

IICTOC Register

Bit	7	6	5	4	3	2	1	0
Name	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0
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 **IICTOEN**: I²C Time-out Control
0: Disable
1: Enable

Bit 6 **IICTOF**: Time-out flag
0: No time-out
1: Time-out occurred

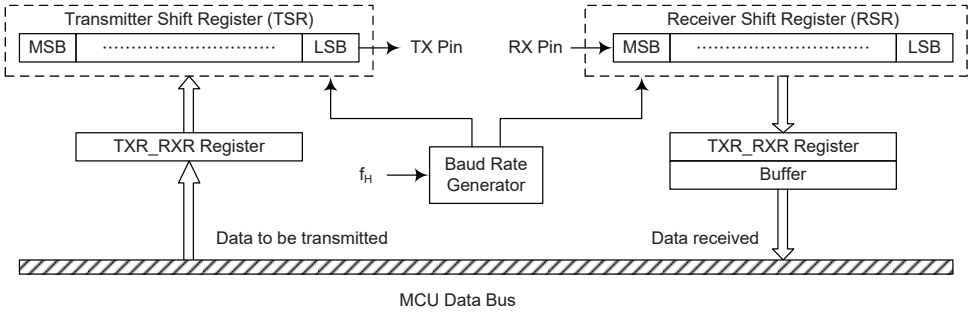
Bit 5~0 **IICTOS5~IICTOS0**: Time-out Definition
I²C time-out clock source is $f_{SUB}/32$.
I²C time-out time is given by: (IICTOS[5:0]+1) × $(32/f_{SUB})$

UART Interface

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

The integrated UART function contains the following features:

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



UART Data Transfer Block Diagram

UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as TX and RX. The TX and RX pins are the UART transmitter and receiver pins respectively. The TX and RX 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 setup these pins to their respective TX output and RX input conditions and disable any pull-high resistor option which may exist on the TX and RX pins. When the TX or RX pin function is disabled by clearing the UARTEN, TXEN or RXEN bit, the TX or RX pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the TX or RX pin or not is determined by the corresponding I/O pull-high function control bit.

UART Data Transfer Scheme

The above 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 TX 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 RX pin, from where it is shifted in, LSB first, to the Receiver Shift Register at a rate controlled by the Baud Rate Generator. When the shift register is full, the data will then be transferred from the shift register to the internal 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 in the Data Memory. This shared register known as the TXR_RXR register is used for both data transmission and data reception.

UART Status and Control Registers

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

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

UART Registers 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 Parity mode (odd or even) is selected. The flag can also be cleared by a software sequence which involves a read to the status register USR followed by an access to the 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 RX 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
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. When the contents of the shift register are transferred to the TXR_RXR register, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag will eventually be 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 more new 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 TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared by reading the USR register with TIDLE set and then writing to the TXR_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 register are the two UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length etc. Further explanation on each of the bits is given below:

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

"x": unknown

Bit 7 **UARTEN**: UART function enable control
0: Disable UART. TX and RX pins are in a floating state
1: Enable UART. TX and RX pins function as UART pins
The UARTEN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX pin as well as the TX pin will be set in a floating state. When the bit is equal to "1", the UART will be enabled and the TX and RX pins will function as defined by the TXEN and RXEN enable control bits.

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

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

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

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

This 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. Replace the most significant bit position with a parity bit.

Bit 4 **PRT**: Parity type selection bit
0: Even parity for parity generator
1: Odd parity for parity generator

This bit is the parity type selection bit. When this bit is equal to "1", odd parity type will be selected. If the bit is equal to "0", then even parity type will be selected.

Bit 3 **STOPS**: Number of Stop bits selection
0: One stop bit format is used
1: Two stop bits format is used

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

Bit 2 **TXBRK**: Transmit break character
0: No break character is transmitted
1: Break characters transmit

The TXBRK bit is the Transmit Break Character bit. When this bit is "0", there are no break characters and the TX 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 interrupt sources. The register also serves to control the baud rate speed, 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	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TXEN**: UART Transmitter 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 TX pin will be set in a floating state.

If the TXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be 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 RX 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 RX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX pin will be set in a floating state.

Bit 5 **BRGH**: Baud Rate speed selection
0: Low speed baud rate
1: High speed baud rate

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

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 RX7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of 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**: RX pin wake-up UART function enable control
0: RX pin wake-up UART function is disabled
1: RX pin wake-up UART function is enabled
This bit is used to control the wake-up UART function when a falling edge on the RX pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no RX pin wake-up UART function if the UART clock (f_H) exists. If the WAKE bit is set to 1 as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the RX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX 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 can not resume even if there is a falling edge on the RX 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.

TXR_RXR Register

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

"x": unknown

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

BRG Register

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

"x": unknown

Bit 7~0 **BRG7~BRG0**: Baud Rate values
By programming the BRGH bit in UCR2 Register which allows selection of the related formula described above and programming the required value in the BRG register, the required baud rate can be setup.
Note: Baud rate= f_H / [64 × (N+1)] if BRGH=0.
Baud rate= f_H / [16 × (N+1)] if BRGH=1.

Baud Rate Generator

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

UCR2 BRGH Bit	0	1
Baud Rate (BR)	f_H / [64 (N+1)]	f_H / [16 (N+1)]

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

Calculating the Baud Rate and Error Values

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

From the above table the desired baud rate BR=f_H / [64 (N+1)]

Re-arranging this equation gives N=[f_H / (BR×64)] - 1

Giving a value for N=[4000000 / (4800×64)] - 1=12.0208

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

Therefore the error is equal to (4808 - 4800) / 4800=0.16%

UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding BNO, PRT, PREN, and STOPS bits in the UCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the 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 TX output pin and RX input pin respectively. If no data is being transmitted on the TX pin, then it will default to a logic high value.

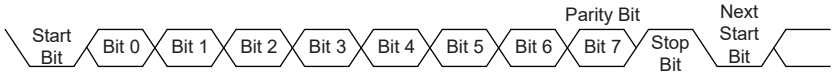
Clearing the UARTEN bit will disable the TX and RX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding 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 being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2 and BRG registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

Data, Parity and Stop Bit Selection

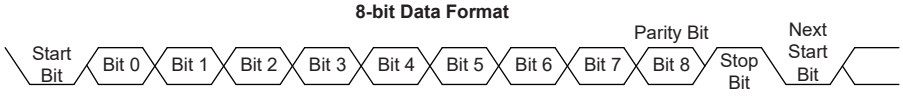
The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1 register. The BNO bit controls the number of data bits which can be set to either 8 or 9, the PRT bit controls the choice of odd or even 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. 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 used for the transmitter. There is only one stop bit for the receiver.

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

Transmitter Receiver Data FormatThe 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 TX 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 TX 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, PRT, PREN and STOPS bits to define the required word length, parity type and number of stop bits.
- Setup the BRG register to select the desired baud rate.
- Set the TXEN bit to ensure that the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR_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.

Transmit Break

If the TXBRK bit is set 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 one or 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 recognised.

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 RX external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

Receiving Data

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

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

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

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

- The RXIF bit in the USR register will be set when the TXR_RXR register has data available. There will be at most one more character available before an overrun error occurs.
- When the contents of the shift register have been transferred to the TXR_RXR register, then if the RIE bit is set, 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

Receive 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 stop bit. 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 stop bit. 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 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.

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 two byte deep FIFO data buffer, where two bytes can be held in the FIFO register, while a third byte can continue to be received. Before this third 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.

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 a 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

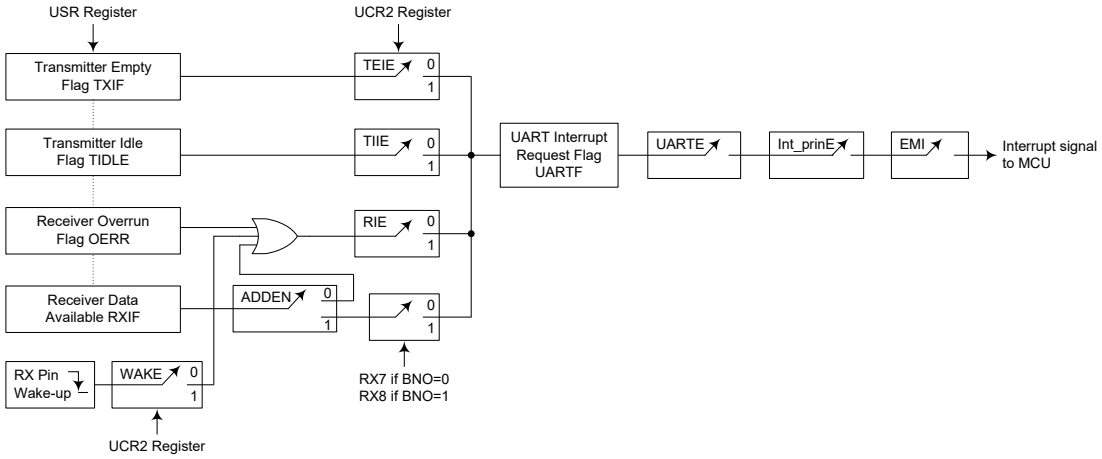
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd or even is selected. The read only PERR flag and the received data will be recorded in the 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 data available, receiver overrun, address detect and an RX pin wake-up. When any of these conditions are created, if the global interrupt enable bit, interrupt priority 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 RX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the 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 RX 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 UART interrupt can be disabled or enabled by the related interrupt enable control bits in the interrupt control registers of the microcontroller to decide whether the interrupt requested by the UART module is masked out or allowed.



UART Interrupt Structure

Address Detect Mode

Setting the Address Detect 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 Int_prinE, UARTE 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	Bit 9 if BNO=1, Bit 8 if BNO=0	UART Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

ADDEN Bit Function

UART Power Down and Wake-up

When the UART clock, f_H, is switched off, the UART will cease to function. If the MCU switches off the UART clock, f_H, and enters the power down mode while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU switches off the UART clock f_H and enters the power down mode by executing the "HALT" instruction while receiving data, then the reception of data will likewise be paused. When the MCU enters the power down mode, note that the USR, UCR1, UCR2, transmit and receive registers, as well as the BRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the power down mode.

The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the MCU enters the power down mode with the UART clock f_H being switched off, then a falling edge on the RX pin will initiate an RX 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 RX pin will be ignored.

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

Low Voltage Detector – LVD

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

LVD Register

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

LVDC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	VBGEN	VLVD2	VLVD1	VLVD0
R/W	—	—	R	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 **LVDO**: LVD Output flag
0: No Low Voltage Detected
1: Low Voltage Detected

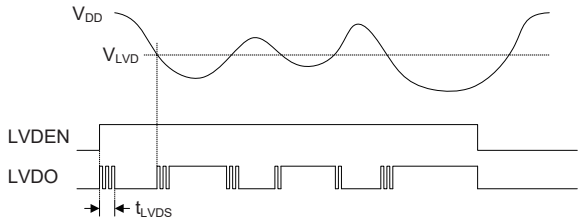
Bit 4 **LVDEN**: Low Voltage Detector Enable control
0: Disable
1: Enable

Bit 3 **VBGEN**: Bandgap Voltage Output Enable control
0: Disable
1: Enable
Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VBGEN bit is set to 1.

Bit 2~0 **VLVD2~VLVD0**: LVD Voltage selection
000: 4.0V
001: 4.0V
010: 4.0V
011: 4.0V
100: 4.0V
101: 4.0V
110: 4.0V
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 pre-sepcified voltage level has a fixed value of 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. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. 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.

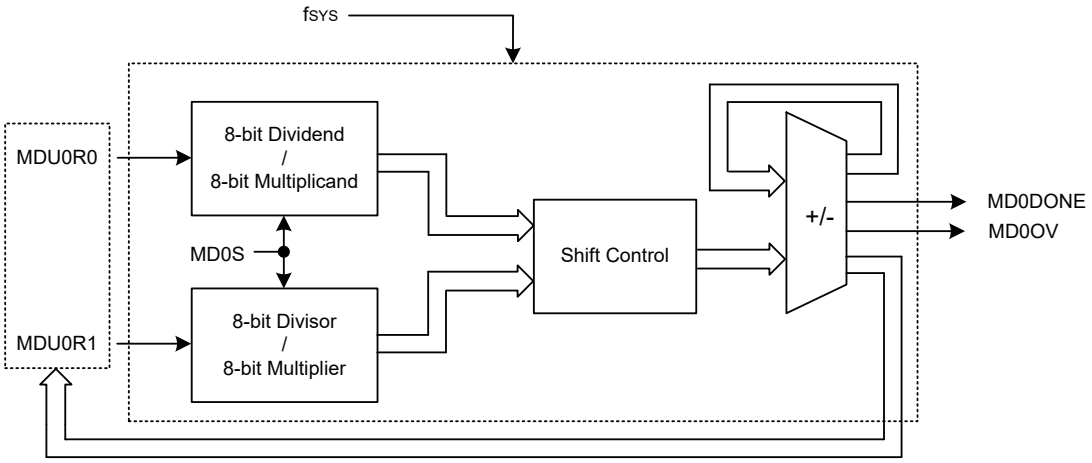


LVD Operation

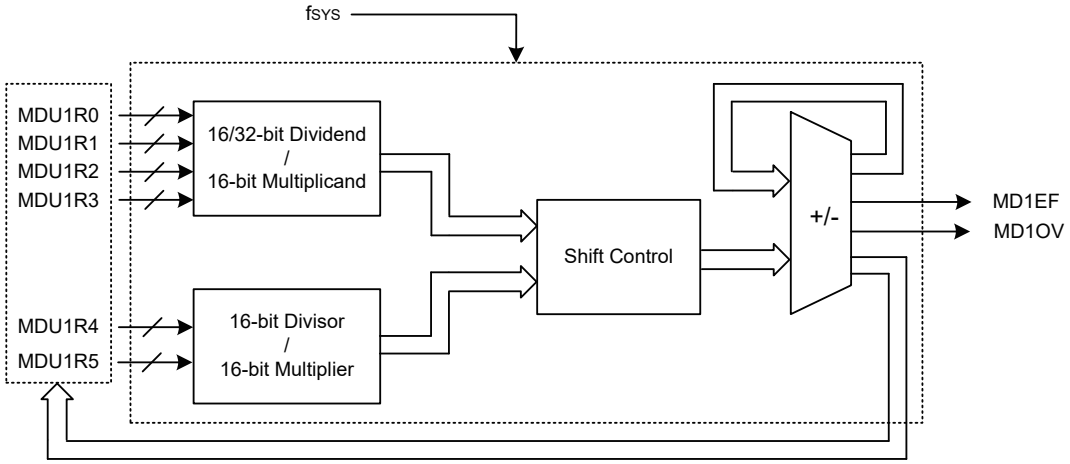
The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set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

Multiplication Division Unit – MDU

The device has an 8-bit and a 16-bit Multiplication Division Units, named MDU0 and MDU1 respectively. The MDU0 integrates an 8-bit unsigned multiplier and divider. The MDU1 integrates a 16-bit unsigned multiplier and a 32-bit/16-bit divider. They can be used to calculate the motor electric rotation angle. The MDUs, 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. They also reduce the overall microcontroller loading and results in the overall system performance improvements.



8-Bit MDU0 Block Diagram



16-Bit MDU1 Block Diagram

MDU Registers

For the 8-bit MDU0 the multiplication and division operations are determined by the MD0S bit and implemented by writing the required operand to the corresponding MDU0 data register. For the 16-bit MDU1 the multiplication and division operations are implemented in a specific way, a specific write access sequence of a series of MDU1 data registers. The status register, MDUnCTRL, 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
MDU0R0	D7	D6	D5	D4	D3	D2	D1	D0
MDU0R1	D7	D6	D5	D4	D3	D2	D1	D0
MDU0CTRL	—	—	—	—	—	MD0DONE	MD0OV	MD0S

8-Bit MDU0 Registers List

Register Name	Bit							
	7	6	5	4	3	2	1	0
MDU1R0	D7	D6	D5	D4	D3	D2	D1	D0
MDU1R1	D7	D6	D5	D4	D3	D2	D1	D0
MDU1R2	D7	D6	D5	D4	D3	D2	D1	D0
MDU1R3	D7	D6	D5	D4	D3	D2	D1	D0
MDU1R4	D7	D6	D5	D4	D3	D2	D1	D0
MDU1R5	D7	D6	D5	D4	D3	D2	D1	D0
MDU1CTRL	MD1EF	MD1OV	—	—	—	—	—	—

16-Bit MDU1 Registers List

MDU0Rn Register – n=0~1

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**: 8-bit MDU0 data register n

MDU0CTRL Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	MD0DONE	MD0OV	MD0S
R/W	—	—	—	—	—	R	R	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **MD0DONE**: 8-bit MDU0 operation complete flag
0: Not completed
1: Completed
This bit will be set to 1 if the MDU0 operation is completed.

Bit 1 **MD0OV**: 8-bit MDU0 overflow flag
0: No overflow occurs
1: Multiplication product > FFH 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 1 **MD0S**: 8-bit MDU0 multiplication or division operation selection
0: Division operation
1: Multiplication operation

MDU1Rn 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 MDU1 data register n

MDU1CTRL Register

Bit	7	6	5	4	3	2	1	0
Name	MD1EF	MD1OV	—	—	—	—	—	—
R/W	R	R	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

Bit 7 **MD1EF**: 16-bit MDU1 error flag
0: Normal
1: Abnormal
This bit will be set to 1 if the data register MDU1Rn is written or read as the MDU1 operation is executing. This bit should be cleared to 0 by reading the MDU1CTRL register if it is equal to 1 and the MDU1 operation is completed.

Bit 6 **MD1OV**: 16-bit MDU1 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

The 8-bit MDU0 operation and the 16-bit MDU1 operation are carried out in their own specific way and will be introduced separately.

8-bit MDU0 Operation

For this MDU the multiplication or division operation is carried out in a specific way and is determined by the MD0S bit in the MDU0CTRL register. Before a multiplication or division operation starts, write the multiplicand/dividend into the MDU0R0 register and the multiplier/divisor into the MDU0R1 register. After the multiplication or division operation is completed, the MDU0R0 register stores the multiplication product low byte or the division quotient, the MDU0R1 register stores the multiplication product high byte or the division remainder. All MDU0 operations will be executed after the MDU0CTRL register is write-accessed. The necessary calculation time for one MDU0 operation, either multiplication or division, is $4 \times t_{SYS}$. After the completion of each operation, it is necessary to check the operation status in the MDU0CTRL register to make sure that whether the operation is completed or not and correct or not.

The overall important points for the MDU0 read/write access and calculation time are summarised in the following table.

Operations / Items	8-bit / 8-bit Division	8-bit × 8-bit Multiplication
Write Access	Dividend written to MDU0R0 Divisor written to MDU0R1	Multiplicand written to MDU0R0 Multiplier written to MDU0R1
Calculation Time	$4 \times t_{SYS}$	$4 \times t_{SYS}$
Read Access	Quotient read from MDU0R0 Remainder read from MDU0R1	Product Low Byte read from MDU0R0 Product High Byte read from MDU0R1

8-Bit MDU0 Operations Summary

16-bit MDU1 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 MDU1 data registers, MDU1R0~MDU1R5. The low byte data, regardless of the dividend, multiplicand, divisor or multiplier, must first be written into the corresponding MDU1 data register followed by the high byte data. All MDU1 operations will be executed after the MDU1R5 register is write-accessed together with the correct specific write access sequence of the MDU1Rn. Note that it is not necessary to consecutively write data into the MDU1 data registers but must be in a correct write access sequence. Therefore, a non-write MDU1Rn 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 MDU1 operation is shown in the following.

- 32-bit/16-bit division operation: Write data sequentially into the six MDU1 data registers from MDU1R0 to MDU1R5.
- 16-bit/16-bit division operation: Write data sequentially into the specific four MDU1 data registers in a sequence of MDU1R0, MDU1R1, MDU1R4 and MDU1R5 with no write access to MDU1R2 and MDU1R3.
- 16-bit×16-bit multiplication operation: Write data sequentially into the specific four MDU1 data registers in a sequence of MDU1R0, MDU1R4, MDU1R1 and MDU1R5 with no write access to MDU1R2 and MDU1R3.

After the specific write access sequence is determined, the MDU1 will start to perform the corresponding operation. The calculation time necessary for these MDU1 operations are different. During the calculation time any read/write access to the six MDU1 data registers is forbidden. After the completion of each operation, it is necessary to check the operation status in the MDU1CTRL register to make sure that whether the operation is correct or not. Then the operation result can be read out from the corresponding MDU1 data registers in a specific read access sequence if the operation is correctly finished. The necessary calculation time for different MDU1 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 MDU1 data registers and should be read out from the MDU1 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 MDU1 data registers but must be in a correct read access sequence. Therefore, a non-read MDU1Rn 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 MDU1 operation is shown in the following.

- 32-bit/16-bit division operation: Read the quotient from MDU1R0 to MDU1R3 and remainder from MDU1R4 and MDU1R5 sequentially.
- 16-bit/16-bit division operation: Read the quotient from MDU1R0 and MDU1R1 and remainder from MDU1R4 and MDU1R5 sequentially.
- 16-bit×16-bit multiplication operation: Read the product sequentially from MDU1R0 to MDU1R3.

The overall important points for the MDU1 read/write access sequence and calculation time are summarised in the following table.

Items \ Operations	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 MDU1R0 Dividend Byte 1 written to MDU1R1 Dividend Byte 2 written to MDUWR2 Dividend Byte 3 written to MDU1R3 Divisor Byte 0 written to MDU1R4 Divisor Byte 1 written to MDU1R5	Dividend Byte 0 written to MDU1R0 Dividend Byte 1 written to MDU1R1 Divisor Byte 0 written to MDU1R4 Divisor Byte 1 written to MDU1R5	Multiplicand Byte 0 written to MDU1R0 Multiplier Byte 0 written to MDU1R4 Multiplicand Byte 1 written to MDU1R1 Multiplier Byte 1 written to MDU1R5
Calculation Time	17 × t_{SYS}	9 × t_{SYS}	11 × t_{SYS}
Read Sequence First read ↓ ↓ ↓ ↓ Last read	Quotient Byte 0 read from MDU1R0 Quotient Byte 1 read from MDU1R1 Quotient Byte 2 read from MDU1R2 Quotient Byte 3 read from MDU1R3 Remainder Byte 0 read from MDU1R4 Remainder Byte 1 read from MDU1R5	Quotient Byte 0 read from MDU1R0 Quotient Byte 1 read from MDU1R1 Remainder Byte 0 read from MDU1R4 Remainder Byte 1 read from MDU1R5	Product Byte 0 read from MDU1R0 Product Byte 1 read from MDU1R1 Product Byte 2 read from MDU1R2 Product Byte 3 read from MDU1R3

16-Bit MDU1 Operations Summary

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external H1, H2, H3, NFIN and INT1 pins, while the internal interrupts are generated by various internal functions including the TMs, Comparator 0, 16-bit CAPTM, Time Base, UART, I^2C, LVD and the A/D converter.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers falls into four categories. The first is the INTEG0 and INTEG1 registers which setup the interrupt trigger edge type for certain interrupts. The second is the INTC0~INTC3 registers which setup the interrupt priority n, the third is the MFI0~MFI7 registers which setup the primary interrupts. Finally there are eight registers, Pri_name0 ~ Pri_name7, to setup the priority for each interrupt number, which will be described in the Interrupt Priority Configuration section.

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.

For interrupts without their own interrupt enable bit and request flag, they can use the corresponding interrupt priority enable bit and request flag after being configured with the desired interrupt priority level. Refer to the following description for more details.

Interrupt Name	Enable Bit	Request Flag	Interrupt Number
Multi-function 0 (Hall Sensor A/B/C interrupts)	HALAE	HALAF	1
	HALBE	HALBF	
	HALCE	HALCF	
Comparator 0: Int_Is	—	—	2
Multi-function 1 (PWM Period and Duty 0~2)	PWMPE	PWMPF	3
	PWMD0E	PWMD0F	
	PWMD1E	PWMD1F	
	PWMD2E	PWMD2F	
PWM Duty 3	—	—	4
CAPTM Capture: CapTM_Over	—	—	5
CAPTM Compare: CapTM_Cmp	—	—	6
A/D Converter: Int_AHL_Lim	—	—	7
Multi-function 2 (A/D Converter_IEOCB, A/D Converter_EOCB)	ISAEOCE	ISAEOCF	8
	AEOCE	AEOCF	
Multi-function 3 (10 bit PTM2)	TM2PE	TM2PF	9
	TM2AE	TM2AF	
Multi-function 4 (16-bit PTM0)	TM0PE	TM0PF	10
	TM0AE	TM0AF	
Multi-function 5 (16-bit PTM1)	TM1PE	TM1PF	11
	TM1AE	TM1AF	
External Interrupt 1: INT1 pin	—	—	12
Noise Filter Interrupt: NFIN pin	—	—	13
Multi-function 6 (10-bit PTM3)	TM3PE	TM3PF	14
	TM3AE	TM3AF	
Multi-function 7 (I^2C, UART, LVD,Time Base)	IICE	IICF	15
	UARTE	UARTF	
	LVE	LVF	
	TBE	TBF	

Interrupt Name/Interrupt Register Bit Name/Interrupt Number

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG0	—	HSEL	INTCS1	INTCS0	INTBS1	INTBS0	INTAS1	INTAS0
INTEG1	—	—	—	—	INTPRI1	INTPRI0	INT1S1	INT1S0
INTC0	—	Int_pri3F	Int_pri2F	Int_pri1F	Int_pri3E	Int_pri2E	Int_pri1E	EMI
INTC1	Int_pri7F	Int_pri6F	Int_pri5F	Int_pri4F	Int_pri7E	Int_pri6E	Int_pri5E	Int_pri4E
INTC2	Int_pri11F	Int_pri10F	Int_pri9F	Int_pri8F	Int_pri11E	Int_pri10E	Int_pri9E	Int_pri8E
INTC3	Int_pri15F	Int_pri14F	Int_pri13F	Int_pri12F	Int_pri15E	Int_pri14E	Int_pri13E	Int_pri12E
MFI0	—	HALCF	HALBF	HALAF	—	HALCE	HALBE	HALAE
MFI1	PWMPF	PWMD2F	PWMD1F	PWMD0F	PWMPE	PWMD2E	PWMD1E	PWMD0E
MFI2	—	—	AEOCF	ISAEOCF	—	—	AEOCE	ISAEOCE
MFI3	—	—	TM2AF	TM2PF	—	—	TM2AE	TM2PE
MFI4	—	—	TM0AF	TM0PF	—	—	TM0AE	TM0PE
MFI5	—	—	TM1AF	TM1PF	—	—	TM1AE	TM1PE
MFI6	—	—	TM3AF	TM3PF	—	—	TM3AE	TM3PE
MFI7	TBF	LVF	UARTF	IICF	TBE	LVE	UARTE	IICE

Interrupt Registers List

INTEG0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	HSEL	INTCS1	INTCS0	INTBS1	INTBS0	INTAS1	INTAS0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **HSEL**: HA/HB/HC source selection
Described elsewhere

Bit 5~4 **INTCS1~INTCS0**: FHC Interrupt edge control for INTC
00: Disable
01: Rising edge trigger
10: Falling edge trigger
11: Dual edge trigger

Bit 3~2 **INTBS1~INTBS0**: FHB Interrupt edge control for INTB
00: Disable
01: Rising edge trigger
10: Falling edge trigger
11: Dual edge trigger

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

INTEG1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INTPRI1	INTPRI0	INT1S1	INT1S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INTPRI1~INTPRI0**: Interrupt Vector 04H and 08H priority preempt control
Described elsewhere

Bit 1~0 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	Int_pri3F	Int_pri2F	Int_pri1F	Int_pri3E	Int_pri2E	Int_pri1E	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 **Int_pri3F**: Interrupt priority 3 request flag
0: No request
1: Interrupt request

Bit 5 **Int_pri2F**: Interrupt priority 2 request flag
0: No request
1: Interrupt request

Bit 4 **Int_pri1F**: Interrupt priority 1 request flag
0: No request
1: Interrupt request

Bit 3 **Int_pri3E**: Interrupt priority 3 control
0: Disable
1: Enable

Bit 2 **Int_pri2E**: Interrupt priority 2 control
0: Disable
1: Enable

Bit 1 **Int_pri1E**: Interrupt priority 1 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	Int_pri7F	Int_pri6F	Int_pri5F	Int_pri4F	Int_pri7E	Int_pri6E	Int_pri5E	Int_pri4E
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 **Int_pri7F**: Interrupt priority 7 request flag
0: No request
1: Interrupt request

Bit 6 **Int_pri6F**: Interrupt priority 6 request flag
0: No request
1: Interrupt request

Bit 5 **Int_pri5F**: Interrupt priority 5 request flag
0: No request
1: Interrupt request

Bit 4 **Int_pri4F**: Interrupt priority 4 request flag
0: No request
1: Interrupt request

Bit 3 **Int_pri7E**: Interrupt priority 7 control
0: Disable
1: Enable

Bit 2 **Int_pri6E**: Interrupt priority 6 control
0: Disable
1: Enable

Bit 1 **Int_pri5E**: Interrupt priority 5 control
0: Disable
1: Enable

Bit 0 **Int_pri4E**: Interrupt priority 4 control
0: Disable
1: Enable

INTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	Int_pri11F	Int_pri10F	Int_pri9F	Int_pri8F	Int_pri11E	Int_pri10E	Int_pri9E	Int_pri8E
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 **Int_pri11F**: Interrupt priority 11 request flag
0: No request
1: Interrupt request

Bit 6 **Int_pri10F**: Interrupt priority 10 request flag
0: No request
1: Interrupt request

Bit 5 **Int_pri9F**: Interrupt priority 9 request flag
0: No request
1: Interrupt request

Bit 4 **Int_pri8F**: Interrupt priority 8 request flag
0: No request
1: Interrupt request

Bit 3 **Int_pri11E**: Interrupt priority 11 control
0: No request
1: Interrupt request

Bit 2 **Int_pri10E**: Interrupt priority 10 control
0: Disable
1: Enable

Bit 1 **Int_pri9E**: Interrupt priority 9 control
0: Disable
1: Enable

Bit 0 **Int_pri8E**: Interrupt priority 8 control
0: Disable
1: Enable

INTC3 Register

Bit	7	6	5	4	3	2	1	0
Name	Int_pri15F	Int_pri14F	Int_pri13F	Int_pri12F	Int_pri15E	Int_pri14E	Int_pri13E	Int_pri12E
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 **Int_pri15F**: Interrupt priority 15 request flag
0: No request
1: Interrupt request

Bit 6 **Int_pri14F**: Interrupt priority 14 request flag
0: No request
1: Interrupt request

Bit 5 **Int_pri13F**: Interrupt priority 13 request flag
0: No request
1: Interrupt request

Bit 4 **Int_pri12F**: Interrupt priority 12 request flag
0: No request
1: Interrupt request

Bit 3 **Int_pri15E**: Interrupt priority 15 control
0: No request
1: Interrupt request

Bit 2 **Int_pri14E**: Interrupt priority 14 control
0: Disable
1: Enable

Bit 1 **Int_pri13E**: Interrupt priority 13 control
0: Disable
1: Enable

Bit 0 **Int_pri12E**: Interrupt priority 12 control
0: Disable
1: Enable

MFI0 Register

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

Bit 7 Unimplemented, read as "0"

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

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

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

Bit 3 Unimplemented, read as "0"

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

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

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

MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	PWMPF	PWMD2F	PWMD1F	PWMD0F	PWMPE	PWMD2E	PWMD1E	PWMD0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PWMPF**: PWM Period match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **PWMD2F**: PWM2 Duty match interrupt request flag
0: No request
1: Interrupt request

Bit 5 **PWMD1F**: PWM1 Duty match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PWMD0F**: PWM0 Duty match interrupt request flag
0: No request
1: Interrupt request

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

Bit 2 **PWMD2E**: PWM2 Duty match interrupt control
0: Disable
1: Enable

Bit 1 **PWMD1E**: PWM1 Duty match interrupt control
0: Disable
1: Enable

Bit 0 **PWMD0E**: PWM0 Duty match interrupt control
0: Disable
1: Enable

MFI2 Register

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **AEOCF**: A/D normal conversion interrupt request flag
0: No request
1: Interrupt request

Bit 4 **ISAEOCF**: A/D auto-scan interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

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

Bit 0 **ISAEOCE**: A/D auto-scan interrupt control
0: Disable
1: Enable

MFI3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TM2AF	TM2PF	—	—	TM2AE	TM2PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TM2AF**: PTM2 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TM2PF**: PTM2 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **TM2AE**: PTM2 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **TM2PE**: PTM2 Comparator P match interrupt control
0: Disable
1: Enable

MFI4 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TM0AF	TM0PF	—	—	TM0AE	TM0PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TM0AF**: PTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TM0PF**: PTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **TM0AE**: PTM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **TM0PE**: PTM0 Comparator P match interrupt control
0: Disable
1: Enable

MFI5 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TM1AF	TM1PF	—	—	TM1AE	TM1PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TM1AF**: PTM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TM1PF**: PTM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **TM1AE**: PTM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **TM1PE**: PTM1 Comparator P match interrupt control
0: Disable
1: Enable

MFI6 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TM3AF	TM3PF	—	—	TM3AE	TM3PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TM3AF**: PTM3 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TM3PF**: PTM3 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **TM3AE**: PTM3 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **TM3PE**: PTM3 Comparator P match interrupt control
0: Disable
1: Enable

MFI7 Register

Bit	7	6	5	4	3	2	1	0
Name	TBF	LVF	UARTF	IICF	TBE	LVE	UARTE	IICE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

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

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

Bit 5 **UARTF**: UART interrupt request flag
0: No request
1: Interrupt request

Bit 4 **IICF**: I^2C interrupt request flag
0: No request
1: Interrupt request

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

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

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

Bit 0 **IICE**: I^2C interrupt control
0: Disable
1: Enable

Interrupt Priority Configuration

Some interrupt sources have their own individual interrupt number while others share the same interrupt number, as shown in the Interrupt Registers section. All interrupts are categorised into fifteen priority levels, from the higheset interrupt priority 1 (04H) to the lowest interrupt priority 15 (3CH). Each interrupt priority level has its own interrupt enable bit and request flag. The actual priority for each interrupt source is configured by writing its corresponding interrupt number into the target bit field in the Pri_name0 ~ Pri_name7 registers.

If an interrupt number is not configured into any interrupt priority level, when the interrupt source within this interrupt number occurs, only its interrupt request flag will be set high without further interrupt response. If an interrupt number has been configured into multiple interrupt priority levels, when the interrupt source within this interrupt number occurs, a subroutine call to each preset interrupt vector will take place according to the priority order. Note that after the interrupt priority initialisation, all the interrupt priority flags must be cleared once.

Vector	Interrupt Priority	Enable Bit	Request Flag	Note
04H	Interrupt priority 1	Int_pri1E	Int_pri1F	Highest Priority
08H	Interrupt priority 2	Int_pri2E	Int_pri2F	↓
0CH	Interrupt priority 3	Int_pri3E	Int_pri3F	↓
10H	Interrupt priority 4	Int_pri4E	Int_pri4F	↓
14H	Interrupt priority 5	Int_pri5E	Int_pri5F	↓
18H	Interrupt priority 6	Int_pri6E	Int_pri6F	↓
1CH	Interrupt priority 7	Int_pri7E	Int_pri7F	↓
20H	Interrupt priority 8	Int_pri8E	Int_pri8F	↓
24H	Interrupt priority 9	Int_pri9E	Int_pri9F	↓
28H	Interrupt priority 10	Int_pri10E	Int_pri10F	↓
2CH	Interrupt priority 11	Int_pri11E	Int_pri11F	↓
30H	Interrupt priority 12	Int_pri12E	Int_pri12F	↓
34H	Interrupt priority 13	Int_pri13E	Int_pri13F	↓
38H	Interrupt priority 14	Int_pri14E	Int_pri14F	↓
3CH	Interrupt priority 15	Int_pri15E	Int_pri15F	Lowest Priority

Interrupt Priority Order

Register Name	Bit							
	7	6	5	4	3	2	1	0
Pri_name0	IP2R3	IP2R2	IP2R1	IP2R0	IP1R3	IP1R2	IP1R1	IP1R0
Pri_name1	IP4R3	IP4R2	IP4R1	IP4R0	IP3R3	IP3R2	IP3R1	IP3R0
Pri_name2	IP6R3	IP6R2	IP6R1	IP6R0	IP5R3	IP5R2	IP5R1	IP5R0
Pri_name3	IP8R3	IP8R2	IP8R1	IP8R0	IP7R3	IP7R2	IP7R1	IP7R0
Pri_name4	IP10R3	IP10R2	IP10R1	IP10R0	IP9R3	IP9R2	IP9R1	IP9R0
Pri_name5	IP12R3	IP12R2	IP12R1	IP12R0	IP11R3	IP11R2	IP11R1	IP11R0
Pri_name6	IP14R3	IP14R2	IP14R1	IP14R0	IP13R3	IP13R2	IP13R1	IP13R0
Pri_name7	—	—	—	—	IP15R3	IP15R2	IP15R1	IP15R0

Interrupt Priority Configuration Registers List

Pri_name0 Register

Bit	7	6	5	4	3	2	1	0
Name	IP2R3	IP2R2	IP2R1	IP2R0	IP1R3	IP1R2	IP1R1	IP1R0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	1

Bit 7~4 **IP2R3~IP2R0**: Setup the required interrupt number in interrupt priority 2

Bit 3~0 **IP1R3~IP1R0**: Setup the required interrupt number in interrupt priority 1

Pri_name1 Register

Bit	7	6	5	4	3	2	1	0
Name	IP4R3	IP4R2	IP4R1	IP4R0	IP3R3	IP3R2	IP3R1	IP3R0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	0	0	0	1	1

Bit 7~4 **IP4R3~IP4R0**: Setup the required interrupt number in interrupt priority 4

Bit 3~0 **IP3R3~IP3R0**: Setup the required interrupt number in interrupt priority 3

Pri_name2 Register

Bit	7	6	5	4	3	2	1	0
Name	IP6R3	IP6R2	IP6R1	IP6R0	IP5R3	IP5R2	IP5R1	IP5R0
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	0	1

Bit 7~4 **IP6R3~IP6R0**: Setup the required interrupt number in interrupt priority 6

Bit 3~0 **IP5R3~IP5R0**: Setup the required interrupt number in interrupt priority 5

Pri_name3 Register

Bit	7	6	5	4	3	2	1	0
Name	IP8R3	IP8R2	IP8R1	IP8R0	IP7R3	IP7R2	IP7R1	IP7R0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	0	0	1	1	1

Bit 7~4 **IP8R3~IP8R0**: Setup the required interrupt number in interrupt priority 8

Bit 3~0 **IP7R3~IP7R0**: Setup the required interrupt number in interrupt priority 7

Pri_name4 Register

Bit	7	6	5	4	3	2	1	0
Name	IP10R3	IP10R2	IP10R1	IP10R0	IP9R3	IP9R2	IP9R1	IP9R0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	1	0	1	0	0	1

Bit 7~4 **IP10R3~IP10R0**: Setup the required interrupt number in interrupt priority 10

Bit 3~0 **IP9R3~IP9R0**: Setup the required interrupt number in interrupt priority 9

Pri_name5 Register

Bit	7	6	5	4	3	2	1	0
Name	IP12R3	IP12R2	IP12R1	IP12R0	IP11R3	IP11R2	IP11R1	IP11R0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	0	0	1	0	1	1

Bit 7~4 **IP12R3~IP12R0**: Setup the required interrupt number in interrupt priority 12

Bit 3~0 **IP11R3~IP11R0**: Setup the required interrupt number in interrupt priority 11

Pri_name6 Register

Bit	7	6	5	4	3	2	1	0
Name	IP14R3	IP14R2	IP14R1	IP14R0	IP13R3	IP13R2	IP13R1	IP13R0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	1	1	0	1

Bit 7~4 **IP14R3~IP14R0**: Setup the required interrupt number in interrupt priority 14

Bit 3~0 **IP13R3~IP13R0**: Setup the required interrupt number in interrupt priority 13

Pri_name7 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IP15R3	IP15R2	IP15R1	IP15R0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **IP15R3~IP15R0**: Setup the required interrupt number in interrupt priority 15

Interrupt Preempt Function

Regarding the interrupt vector 04H and 08H, each of them, with its priority preempt function being enabled by properly setting the INTPRI1~INTPRI0 bits in the INTEG1 register, can suspend any other ongoing interrupt subroutines and immediately enter its interrupt subroutine if the stack is not full. However the interrupt vector 08H can not suspend the vector 04H subroutine even when the 04H vector preempt function is disabled. It should be noted that when a preempt interrupt occurs and the stack is full, the requested preempt interrupt will not be immediately serviced until the stack is not full. After the current higher priority interrupt has been serviced, the program will return to the last suspended interrupt subroutine using a RETI instruction. For interrupts that require immediate response, this interrupt priority preempt function can be properly used.

INTEG1 Register

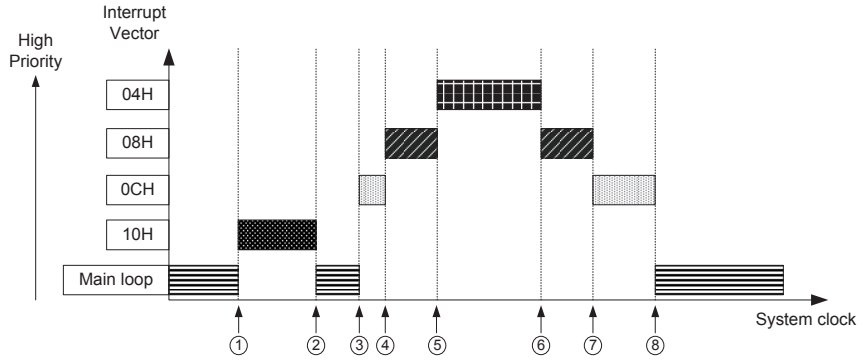
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INTPRI1	INTPRI0	INT1S1	INT1S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INTPRI1~INTPRI0**: Interrupt Vector 04H and 08H priority preempt control
00: Interrupt Vector 04H and 08H priority preempt disabled
01: Interrupt Vector 04H priority preempt enabled, Vector 08H priority preempt disabled
10: Interrupt Vector 04H priority preempt disabled, Vector 08H priority preempt enabled
11: Interrupt Vector 04H and 08H priority preempt enabled

Bit 1~0 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
Described elsewhere

The following example explains how the program runs when both preempt functions are enabled by setting the INTPRI1~INTPRI0 bits to 11B.



1 Vector 10H preempts the main loop.

2 Exits vector 10H subroutine and returns to the main loop.

3 Vector 0CH preempts the main loop.

4 The second highest priority 08H preempts the 0CH subroutine.

5 The first highest priority 04H preempts the 08H subroutine.

6 Exits vector 04H subroutine by "RETI" and go no to execute the 08H subroutine.

7 Exits vector 08H subroutine by "RETI" and go no to execute the 0CH subroutine.

8 Exits vector 0CH subroutine by "RETI" and returns to the main loop.

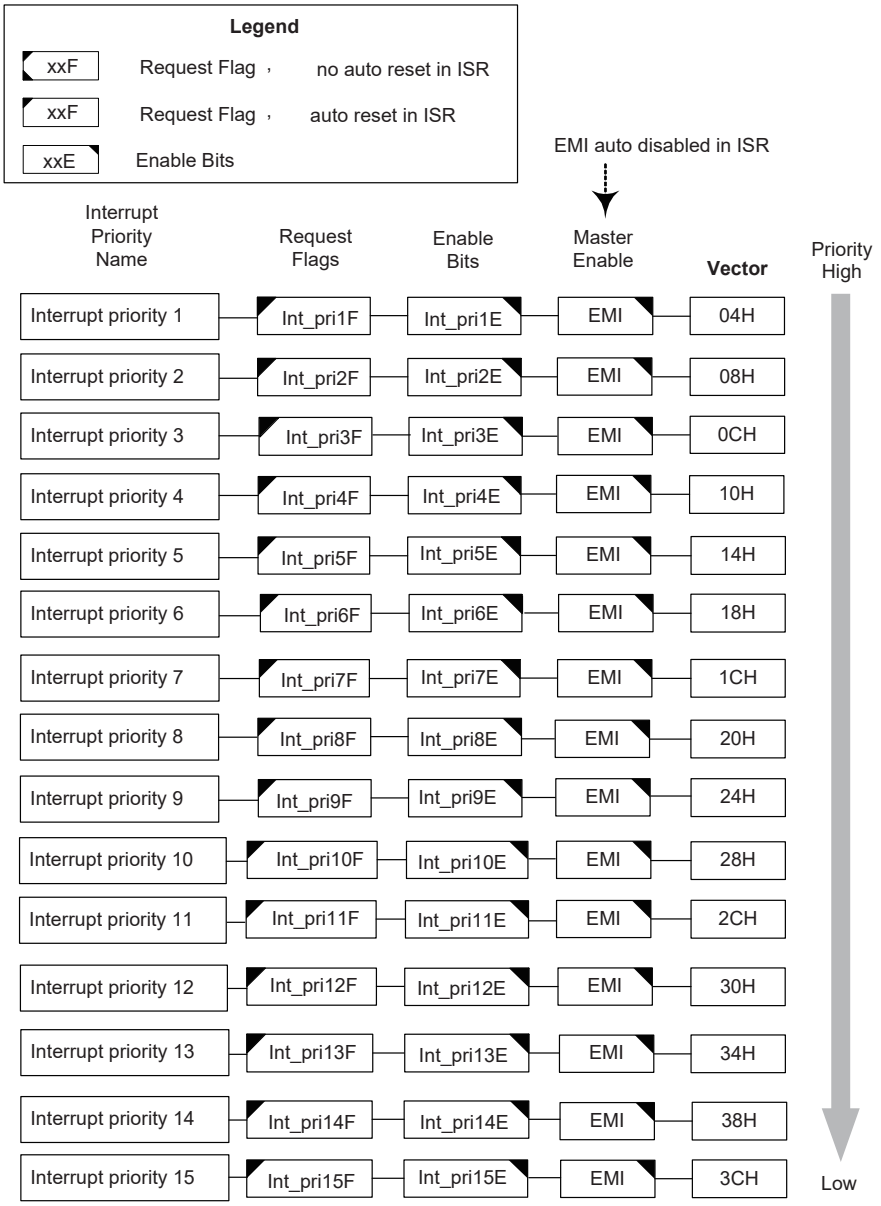
Interrupt Operation

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

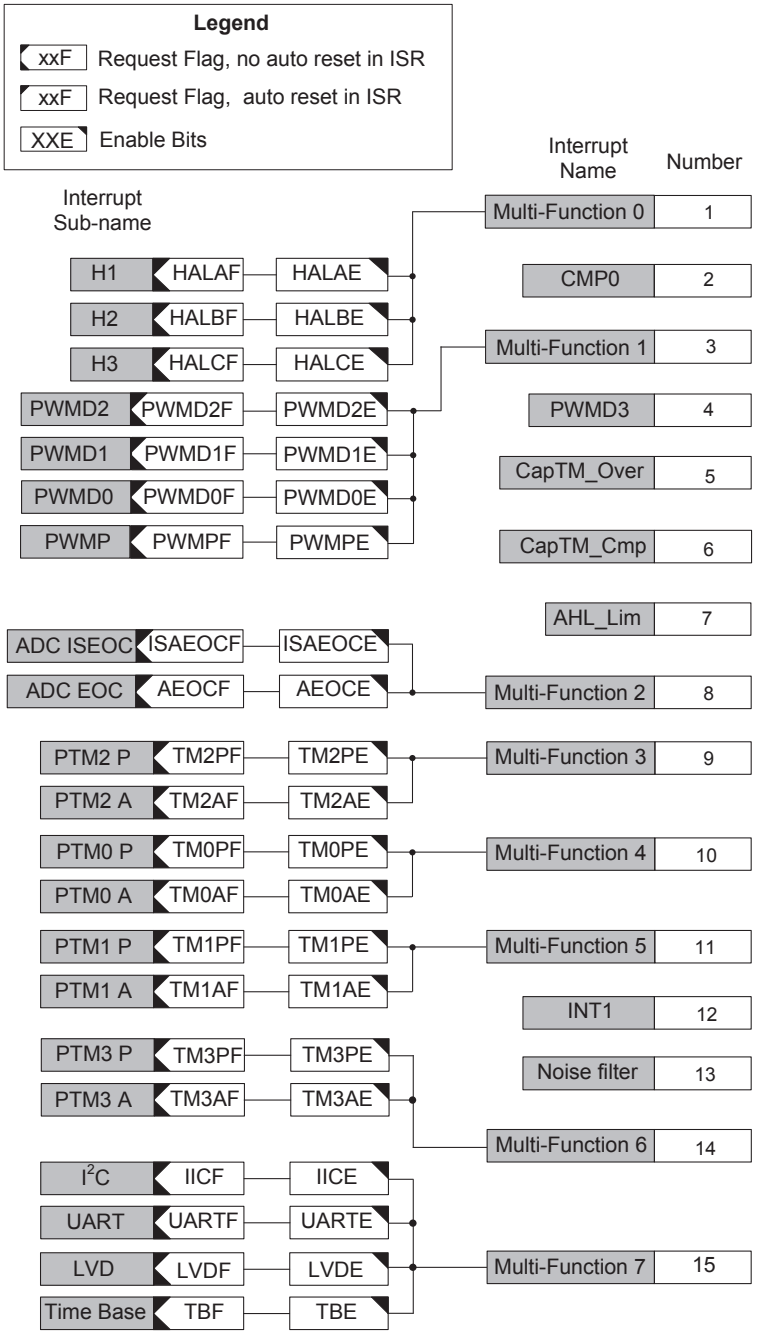
When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will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

Some interrupt sources have their own individual interrupt number while others share the same interrupt number. Once an interrupt subroutine is serviced, all the other interrupts except the interrupt vector 04H or 08H with their interrupt preempt function enabled, will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

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



Interrupt Priority Structure



Interrupt Structure

Hall Sensor Interrupts

The Hall sensor interrupts are contained within the multi-function interrupt 0 thus sharing the same interrupt number. They are controlled by signal transistions on the Hall sensor noise filter output signals, FHA, FHB and FHC. After being configured with a desired interrupt priority level, a Hall sensor interrupt request will take place when the Hall sensor interrupt request flag, HALAF, HALBF or HALCF, and the corresponding interrupt priority request flag are set, which will occur when a transition appears on the Hall sensor noise filter outputs. To allow the program to branch to its prespective interrupt vector address, the global interrupt enable bit, EMI, the Hall sensor interrupt enable bit, HALAE, HALBE or HALCE, and the relevant interrupt priority enable bit, Int_prinE, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG0 register. If the Hall noise filter inputs are selected to come from the external input pins, H1, H2 and H3, which are pin-shared with I/O pins, they should be configured as an external Hall sensor input pins using the corresponding pin-shared control bits before the Hall sensor interrupt functions are enabled. When the interrupt is enabled, the stack is not full and either one of the Hall sensor interrupts occurs, a subroutine call to the relevant interrupt vector will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the related interrupt priority request flag, Int_prinF, will be automatically reset, but the Hall sensor interrupt request flags, HALAF, HALBF and HALCF, must be manually cleared by the application program.

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

External Interrupt 1

The external interrupt 1 has its own independent interrupt number and is controlled by signal transitions on the INT1 pin. After being configured with a desired interrupt priority level, an external interrupt request will take place when its interrupt priority request flag, Int_prinF, is set, which will occur when a transition, whose type is chosen by the edge select bits in the INTEG1 register, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the relevant interrupt priority enable bit, Int_prinE, must first be set. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if its interrupt priority enable bit has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, its interrupt priority request flag, Int_prinF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pin will remain valid even if the pin is used as an external interrupt input.

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

Noise Filter Input Interrupt

The external noise filter intput interrupt has its own independent interrupt number and is controlled by signal transitions on the NFIN pin. After being configured with a desired interrupt priority level, an external noise filtered intput interrupt request will take place when the relevant interrupt priority request flag, Int_prinF, is set, which will occur when a transition, whose type is chosen by the edge select bits NFIS1 and NFIS0 in the NF_VIL register, appears on the NFIN pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the relevant interrupt priority enable bit, Int_prinE, must first be set. As the external noise filter interrupt pin is pin-shared with I/O pin, it can only be configured as the noise filter interrupt pin if its interrupt priority enable bit has been set and the noise filter interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the noise filter interrupt pin, a subroutine call to the noise filter interrupt vector will take place. When the interrupt is serviced, its interrupt priority request flag, Int_prinF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the NFIN pin will remain valid even if the pin is used as a noise filter interrupt input.

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

Comparator Interrupt

The comparator interrupt has its own independent interrupt number and is controlled by the internal comparator 0. After being configured with a desired interrupt priority level, a comparator interrupt request will take place when the relevant interrupt priority request flag, Int_prinF, is set, a situation that will occur when the comparator output changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the relevant interrupt priority enable bit, Int_prinE, must first be set. When the interrupt is enabled, the stack is not full and the comparator inputs generate a comparator output transition, a subroutine call to the comparator interrupt vector, will take place. When the interrupt is serviced, the relevant interrupt priority request flag, Int_prinF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

CAPTM Interrupts

The CAPTM module has two interrupts and each has an independent interrupt number. These are the capture overflow interrupt, CapTM_Over, and the compare match interrupt, CapTM_Cmp. After being configured with a desired interrupt priority level, a CAPTM capture interrupt request or compare match interrupt will take place when the relevant interrupt prioriy request flag, Int_prinF, is set, which occurs when CAPTM capture overflows or compare matches. To allow the program to branch to their respective interrupt vector address, the global interrupt enable bit, EMI, and the relevant interrupt priority enable bit, Int_prinE, must first be set. When the interrupt is enabled, the stack is not full and CAPTM capture overflows or compare matches, a subroutine call to the respectivel interrupt vector will take place. When the CAPTM interrupt is serviced, the relevant interrupt priority request flag, Int_prinF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Multi-function Interrupts

Within the device there are up to seven Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the Hall sensor input interrupts, PWM period and duty interrupts, TM Interrupts, A/D Converter Interrupts, UART Interrupt, I^2C interrupt and LVD Interrupt.

The interrupt sources within each Multi-function interrupt share the same interrupt number. After being configured with the desired interrupt priority level, a multi-function request will take place when its relevant interrupt priority request flag, Int_prinF, is set, which occurs when any of its included functions generate an interrupt request. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the relevant interrupt priority enable bit, must first be set. When the interrupt is enabled and the stack is not full, and either one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related interrupt priority 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 priority request flags will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupts will not be automatically reset and must be manually reset by the application program.

PWM Module Interrupts

The PWM module has five interrupts, one PWM Period match interrupt known as PWMP and four PWM Duty match interrupts known as PWMDn (n=0~3). The PWMP and PWMD0~PWMD2 interrupts are contained in multi-function interrupt 1 and they share the same interrupt number. The PWMD3 interrupt has its independent number.

Regarding the PWMP and PWMD0~PWMD2 interrupts, after being configured with the desired interrupt priority level, a PWM interrupt request will take place when the PWM interrupt request flag, PWMPF or PWMDnF, and the corresponding interrupt priority request flag, are set, which occurs when the PWM Period or Duty matches. To allow the program to branch to its respectively interrupt vector address, the global intgerrupt enable bit, EMI, the PWM Period or Duty match interrupt enable bit, PWMPE or PWMDnE, and the relevant interrupt priority enable bit, must first be set. When the interrupt is enabled, the stack is not full and the PWM Period or Duty maches, a subroutine call to this vector loacation will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the interrupt priority request flag will be automatically reset, but the interrupt request flag, PWMDnF or PWMPF, must be manually cleared by the application program.

Regarding the PWMD3 interrupt, after being configured with the desired interrupt priority level, its interrupt request will take place when the associated interrupt priority request flag, Int_prinF, is set, which occurs when the PWM Duty 3 maches. To allow the program to branch to its interrupt vector address, the global intgerrupt enable bit, EMI, and the relevant interrupt priority enable bit must first be set. When the interrupt is enabled, the stack is not full and the PWM Duty 3 maches, a subroutine call to this vector loacation will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the interrupt priority request flag will be automatically reset.

A/D Converter Interrupts

The A/D Converter has three interrupts. One is the A/D converter boundary interrupt which is controlled by the comparison between the converted data registers, ADRH/ADRL or ISRHn/ISRLn, and the boundary register pairs, ADHVDH/ADHVDL and ADLVDH/ADLVDL. It has its independent interrupt number. The other two are the A/D normal conversion interrupt, which is controlled by the end of an normal A/D conversion process, and the A/D auto-scan interrupt, which is controlled by the end of A/D auto-scan process. They are contained within the multi-function 2 and share the same interrupt number.

After being configured with the desired interrupt priority level, the A/D converter boundary interrupt request will take place when the related interrupt priority request flag is set, which occurs when the preset comparison type defined in the ADCHVE and ADCLVE bits in the ADCR1 register takes place. To allow the program to branch to its interrupt bector address, the global interrupt enable bit, EMI, and the related interrupt priority enable bit must first be set. When the interrupt is enabled, the stack is not full and the preset A/D comparise type occurs, a subroutine call to its interrupt vector will take place. When the A/D converter boundary interrupt is serviced, the related interrupt priority request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

After being configured with the desired interrupt priority level, an A/D normal conversion interrupt request or A/D auto-scan interrupt request will take place when the A/D normal conversion interrupt request flag, AEOCF, or the A/D auto-scan interrupt request flag, ISAEOCF, and the corresponding interrupt priority request flag are set, which occurs when the A/D normal conversion process or the A/D auto-scan process finishes. To allow the program to branch to the respective interrupt vector address, the global interrupt enable bit, EMI, the A/D normal conversion interrupt enable bit, AEOCE, or the A/D auto-scan interrupt enable bit, ISAEOCE, and associated interrupt priority enable bit, must first be set. When the interrupt is enabled, the stack is not full and the A/D normal conversion process or auto-scan process has ended, a subroutine call to the interrupt vector will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the relevant interrupt priority request flag will be also automatically cleared. As the AEOCF flag and ISAEICF flag will not be automatically cleared, they have to be cleared by the application program.

TM Interrupts

The Periodic Type TMs each have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the multi-function interrupts and share the same interrupt number with other interrupts in the same group. For all of the TM types there are two interrupt request flags, TMnPF and TMnAF, and two enable control bits, TMnPE and TMnAE. After being configured with the desired interrupt priority level, a TM interrupt request will take place when any of the TM request flags together with the associated interrupt priority request flag are set, a situation which occurs when a TM comparator P or A match situation happens.

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

UART Interrupt

The UART interrupt is contained within the multi-function interrupt 7 sharing the same interrupt number with other interrupt sources in the same group. Several individual UART conditions can generate a UART interrupt. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. After being configured with the desired interrupt priority level, an UART interrupt request will take place when the UART interrupt request flag and the associated interrupt priority request flag are set, which happens when one of these conditions occurs. To allow the program to branch to the respective interrupt vector addresses, the global interrupt enable bit, EMI, interrupt priority enable bit, Int_prinE and UART interrupt enable bit, UARTE, 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 respective interrupt vector, will take place. When the UART interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the interrupt priority request flag will be also automatically cleared. As the UARTF flag will not be automatically cleared, it has to be cleared by the application program. 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.

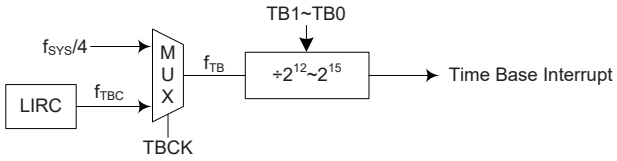
I²C Interrupt

The I²C interrupt is contained within the multi-function interrupt 7 sharing the same interrupt number with other interrupt sources in the same group. After being configured with the desired interrupt priority level, an I²C Interrupt request will take place when the I²C Interrupt request flag, IICF, and the associated interrupt priority request flag are set, which occurs when a byte of data has been received or transmitted by the I²C interface, I²C address match or I²C time-out occurs. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the I²C Interface Interrupt enable bit, IICE, and the realted interrupt priority enable bit must first be set. When the interrupt is enabled, the stack is not full and any of these situations occurs, a subroutine call to the I²C Interrupt vector, will take place. When the I²C Interface Interrupt is serviced, the interrupt priority request flag will be automatically reset and the EMI bit will be cleared to disable other interrupts. However, the interrupt request flag, IICF, has to be cleared by the application program.

Time Base Interrupt

The function of the Time Base Interrupt is to provide regular time signal in the form of an internal interrupt. It is controlled by the overflow signal from its timer function. The Time Base interrupt is contained within the multi-function interrupt 7 which means it shares the same interrupt number with other interrupt sources in the same group. After being configured with the desired interrupt priority level, a Time Base interrupt request will take place when the Time Base interrupt request flag, TBF, and the associated interrupt priority request flag are set, which occurs when the overflow condition happens. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, the Time Base enable bit, TBE, and the relevant interrupt priority enble bit 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 relevant interrupt priority request flag, Int_prinF, will be automatically reset and the EMI bit will be cleared to disable other interrupts. However, the interrupt request flag, TBF, must be cleared manually by application program.

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



Time Base Interrupt

TBC Register

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

Bit 7 **TBON**: Time Base Control
0: Disable
1: Enable

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

Bit 5~4 **TB1~TB0**: Select Time Base Time-out Period
00: $f_{TB}/2^{12}$
01: $f_{TB}/2^{13}$
10: $f_{TB}/2^{14}$
11: $f_{TB}/2^{15}$

Bit 3~0 Unimplemented, read as "0"

LVD Interrupt

The Low Voltage Detector Interrupt is contained within the multi-function interrupt 7 sharing the same interrupt number with other interrupt sources in the same group. After being configured with the desired interrupt priority level, an LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, and the associated interrupt priority request flag are set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVE, and associated interrupt priority enable bit must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to its interrupt vector will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the interrupt priority request flag will be also automatically cleared. As the LVF flag will not be automatically cleared, it has to be cleared by the application program.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt priority 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 priority request 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 priority 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. Where a certain interrupt, whether has its own request flag or not, has been configured to an interrupt priority level, then when the interrupt service routine is executed, as only the interrupt priority request flags, Int_prinF, will be automatically cleared, the individual request flag of the function needs to be cleared by the application program.

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

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

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

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

Application Circuits

Introduction

Holtek's HT66FM5440 is specially directed at three-phase BLDC motor control applications. Acting like a human brain, the device is the main power control core of the BLDC motor driving systems, which takes the responsibility to immediately collect and transmit information. The BLDC motor sensor driving solution relies on the Hall sensor to sense the magnetic field of the motor rotor and then drives the corresponding motor phase based on the Hall sensor signal, sequentially achieving phase change purpose. Therefore, it is necessary to first understand the relationship between the Hall sensor signal and motor phase. However, there is no such problem for the sensorless driving solution. Motor driving is divided into two types, square-wave control and sine-wave control, which are both supported by the device. In a general driving system, system voltage and current detection as well as speed command are necessary. Power component temperature and motor temperature detection, etc., may also be needed in accordance with different products. Basing on the basic functional requirements mentioned above, this chapter will introduce how to use the HT66FM5440 to implement three-phase BLDC motor control. The BLDC motor system is mainly composed of the power circuit, MCU control circuit, motor driving circuit and voltage/current detection circuit, which will be introduced in detail in the hardware block diagram section.

Functional Description

All necessary and important functions required for brushless motor driving are integrated in the HT66FM5440 device and are coordinated by the BLDC motor control circuit. These functions include a 10-bit motor dedicated PWM, mask function, Hall sensor decoder and motor protection function, which combined with the OCP, capture timer, etc., provide the advantage of fast system protection. From the perspective of a control system, the main points of each function will be explained in the following.

Hall Sensor/Sensorless

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

Square-wave / Sine-wave Driving

The square-wave driving and the sine-wave driving are different in the phase change algorithm and more importantly in the PWM configuration. The mask function is composed of mask circuit, dead-time circuit and polarity control circuit, which combined with the motor dedicated 10-bit PWM can implement square-wave driving. When the hardware mask mode is selected, square-wave driving phase change logic is implemented by the Hall sensor decoder, users can detect the signals on the H1~H3 pins for automatic phase change or use program for phase change. When the software mask mode is selected, users can flexibly configure the PWM signals on the motor control dedicated I/O ports and use the PWM center-align mode and dead-time control to implement PWM outputs required for sine-wave driving.

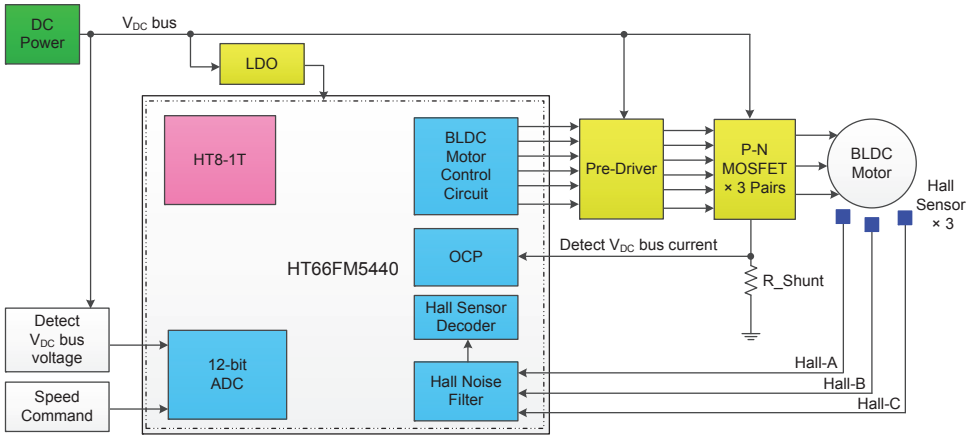
Detection and Protection Function Configuration

Brushless motor driving requires instant detection for system voltage, current and speed. Once an abnormal situation occurs, the driving circuit must be immediately switched off to protect the peripheral hardware thus ensuring system security. The motor dedicated 16-bit capture timer contains motor stalling protection related configurations and can be used to monitor the motor speed. When the stalling situation occurs, the PWM signals will be directly switched off to prevent large motor static current from occurring.

The OCP function of the HT66FM5440 is composed of an OPA, a CMP and a DAC. The DAC provides a reference value, when the motor current amplified by the OPA is larger than the reference value, it means that an over current situation has occurred, in which condition, the PWM signals can be switched off directly to prevent damage to the power transistor. Those important information including voltage, current and temperature, can be detected by the A/D auto-scan function at fixed internals. And then with the interrupt priority function and interrupt preempt function, users can configure the over current interrupt and A/D auto-scan interrupt into the highest two priority levels to achieve the most secure motor driving protection.

Hardware Block Diagram

Here use the low voltage and Hall sensor driving solution to introduce the BLDC motor control system.

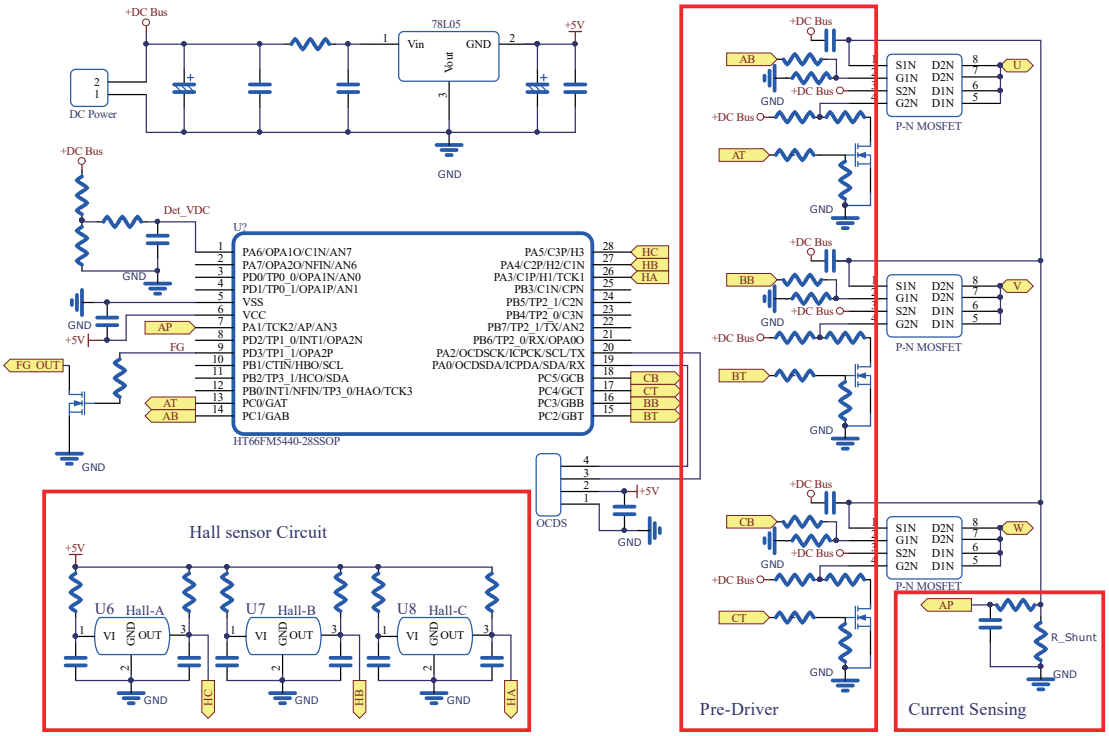


BLDC Motor Control with Hall Sensor

1. The DC power is a low voltage system which provides power for the LDO, VDC bus voltage detection circuit, pre-driver circuit and P-N MOSFET circuit.
2. The LDO outputs 5V standard power for the MCU, Hall sensor and speed command component.
3. The speed command component is a variable resistor. It outputs a voltage from 0V to 5V as the speed adjustment command.
4. The VDC is first decreased via the voltage detection circuit and then input to the MCU A/D channel, in which way the A/D converter can check the current system voltage.
5. The pre-driver circuit is controlled by the BLDC motor control circuit, and its outputs are used to drive the P-N MOSFET gate driver.
6. The inverter in the three-phase BLDC motor system consists of three pairs of top and bottom MOSFETs. In a low voltage and small current system, a P MOSFET is usually used for the top gate in order to eliminate the bootstrap component. Here the inverter uses the P-N MOSFETs to drive the motor.
7. When the motor is loaded, the Hall signals are susceptible to noise interference. Therefore the signals of the three Hall sensors, Hall-A, Hall-B and Hall-C should first be input to the Hall sensor noise filter. After being filtered, these signals will be transfer to the interrupt controller, capture timer and Hall sensor decoder.

8. The Hall sensor decoder contains a total of 12 phase change logic registers for motor forward and backward directions. To implement the required phase change logic, these registers should be established in advance and can be controlled by the hardware decoder or by the HDCD register using the application program. And then use the HDCR register to implement motor operating, motor brake and motor direction control.
9. Motor operating current sampling is achieved using the current sampling resistor, R_Shunt. When the current flows through the register, a tiny voltage signal will be produced. This signal will first be amplified by the OPA in the OCP circuit, and then read by the ADC or checked by the DAC and CMP to detect the over current situation, which once occurred will trigger the corresponding over current protection measures.
10. Lastly, by coordinating all the relevant peripheral circuits the BLDC motor control circuit will configure the motor driving PWM signals and phase change logic according to the Hall sensor signals and protection signals to implement motor operating control.

Hardware Circuit



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 1 system clock cycle, therefore in the case of an 16MHz system oscillator, most instructions would be implemented within 0.0625μs and branch or call instructions would be implemented within 0.125μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

The 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 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 up to three cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the "CLR WDT" instruction the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after the "CLR WDT" instructions is executed. Otherwise the TO and PDF flags remain unchanged.

Extended Instruction Set

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

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	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 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 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 up to four cycles are required, if no skip takes place two cycles is required.
2. Any extended instruction which changes the contents of the PCL register will also require three cycles for execution.

Instruction Definition

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

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

Operation ACC ← 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] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s): C

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

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

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

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 Data Memory is not 0

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

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

Operation Skip if [m]≠ 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 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 pair (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

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

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

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

Affected flag(s) None

ITABRD [m] Increment table pointer low byte first and read table to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

ITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Extended Instruction Definition

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

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

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

Operation ACC ← 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 $[m] \leftarrow \overline{[m]}$

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

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

Operation $ACC \leftarrow \overline{[m]}$

Affected flag(s) Z

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

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

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

LDEC [m] Decrement Data Memory

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

Operation $[m] \leftarrow [m] - 1$

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

LINC [m] Increment Data Memory

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

Operation $[m] \leftarrow [m] + 1$

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] + 1$

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [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 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

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 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

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

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

Operation ACC ← [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 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

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

LSNZ [m].i Skip if Data Memory is not 0

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

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

Operation Skip if [m] ≠ 0

Affected flag(s) None

LSUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

LSWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

LSZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

LITABRD [m] Increment table pointer low byte first and read table to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (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

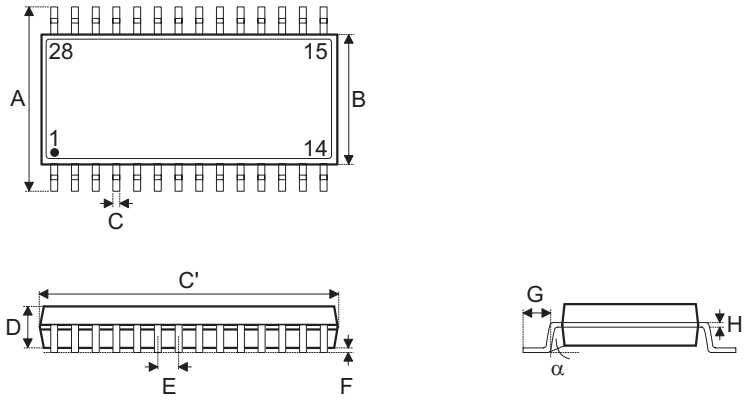
Package Information

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

28-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.390 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.0098
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.0 BSC	—
B	—	3.9 BSC	—
C	0.20	—	0.30
C’	—	9.9 BSC	—
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°