Power Inverter OTP MCU

HT45R7130

Revision: V1.10 Date: March 16, 2026

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**
Internally Connected Signals 12
Absolute Maximum Ratings **12**
D.C. Characteristics **13**
Operating Voltage Characteristics 13
Operating Current Characteristics 13
Standby Current Characteristics 13
A.C. Characteristics **14**
Internal High Speed Oscillator – HIRC – Frequency Accuracy 14
Internal Low Speed Oscillator Characteristics – LIRC 15
Operating Frequency Characteristic Curves 15
System Start Up Time Characteristics 15
Input/Output Characteristics **16**
Input/Output (without Multi-power) D.C. Characteristics 16
Input/Output (with Multi-power) D.C. Characteristics 17
Memory Electrical Characteristics **18**
LVR Electrical Characteristics **18**
A/D Converter Characteristics **19**
Over/Under Voltage Protection Electrical Characteristics **20**
Over Current Protection Electrical Characteristics **21**
Divider Resistor Electrical Characteristics **21**
Delay Lock Loop Electrical Characteristics **22**
Gate-driver Electrical Characteristics **22**
Power-on Reset Characteristics **23**
System Architecture **23**
Clocking and Pipelining 23
Program Counter 24
Stack 25
Arithmetic and Logic Unit – ALU 26
OTP Program Memory **26**
Structure 26
Special Vectors 27
Look-up Table 27

Table Program Example 28
In Circuit Programming – ICP 29
On-Chip Debug Support – OCDS 30
OTP ROM Parameter Program – ORPP 30
Data Memory 33
Structure 33
General Purpose Data Memory 33
Special Purpose Data Memory 33
Special Function Register Description 35
Indirect Addressing Registers – IAR0, IAR1 35
Memory Pointers – MP0, MP1 35
Accumulator – ACC 36
Program Counter Low Byte Register – PCL 36
Look-up Table Registers – TBLP, TBHP, TBLH 36
Status Register – STATUS 36
Oscillators 38
Oscillator Overview 38
System Clock Configurations 38
Internal High Speed RC Oscillator – HIRC 38
Internal 32kHz Oscillator – LIRC 39
Operating Modes and System Clocks 39
System Clocks 39
System Operation Modes 40
Control Registers 41
Operating Mode Switching 42
Standby Current Considerations 46
Wake-up 46
Watchdog Timer 47
Watchdog Timer Clock Source 47
Watchdog Timer Control Register 47
Watchdog Timer Operation 48
Reset and Initialisation 49
Reset Functions 49
Reset Initial Conditions 51
Input/Output Ports 55
Pull-high Resistors 55
Port A Wake-up 56
I/O Port Control Register 56
I/O Port Power Source Control 56
Pin-shared Functions 57
I/O Pin Structures 60
Programming Considerations 60

Timer Modules – TM 61
Introduction 61
TM Operation 61
TM Clock Source 61
TM Interrupts 61
TM External Pins 62
Programming Considerations 62
Compact Type TM – CTM 64
Compact Type TM Operation 64
Compact Type TM Register Description 64
Compact Type TM Operation Modes 68
Periodic Type TM – PTM 74
Periodic Type TM Operation 74
Periodic Type TM Register Description 74
Periodic Type TM Operation Modes 78
Analog to Digital Converter 87
A/D Converter Overview 87
A/D Register Description 88
A/D Converter Reference Voltage 91
A/D Converter Input Signals 91
A/D Conversion Operation 92
Conversion Rate and Timing Diagram 93
Summary of A/D Conversion Steps 94
Programming Considerations 94
A/D Transfer Function 95
A/D Programming Examples 95
Over Current Protection – OCP 97
Over Current Protection Operation 97
Over Current Protection Registers 98
Input Voltage Range 100
Input Offset Calibration 100
Over/Under Voltage Protection – OUVP 102
Over Voltage Protection Operation 102
Under Voltage Protection Operation 102
Over/Under Voltage Protection Registers 103
Comparator Input Offset Calibration 106
Divider Resistors 107
Serial Peripheral Interface – SPI 108
SPI Interface Operation 108
SPI Registers 109
SPI Communication 112
SPI Bus Enable/Disable 114
SPI Operation Steps 114
Error Detection 115

UART Interface **116**
UART External Pins 117
UART Single Wire Mode 117
UART Data Transfer Scheme 117
UART Status and Control Registers 118
Baud Rate Generator 125
Baud Rate Correction Example 126
UART Setup and Control 126
UART Transmitter 128
UART Receiver 129
Managing Receiver Errors 131
UART Interrupt Structure 132
UART Power Down and Wake-up 133
Auto-adjust High Resolution PWM with Delay Lock Loop/Dead Time **134**
Functional Description 134
High Resolution PWM Register Description 134
PWM Generator 140
PWM Counting Modes 140
Dead Time Insert 141
Output Mode Selection 142
Delay Lock Loop 143
Auto-adjust Circuit 145
Interrupts **146**
Interrupt Registers 146
Interrupt Operation 150
External Interrupts 151
Over Current Protection Interrupt 151
Over Voltage Protection Interrupt 151
Under Voltage Protection Interrupt 152
UART Interrupt 152
SPI Interrupt 152
Time Base Interrupt 152
PWM Interrupts 153
A/D Converter Interrupt 154
Multi-function Interrupts 154
TM Interrupts 154
Interrupt Wake-up Function 155
Programming Considerations 155
Configuration Options **156**
Application Circuits **157**
Instruction Set **158**
Introduction 158
Instruction Timing 158
Moving and Transferring Data 158

Arithmetic Operations 158
Logical and Rotate Operation 159
Branches and Control Transfer 159
Bit Operations 159
Table Read Operations 159
Other Operations 159
Instruction Set Summary 160
Table Conventions 160
Instruction Definition 162
Package Information 172
16-pin NSOP (150mil) Outline Dimensions 173

Features

CPU Features

- Operating voltage
  - V_{DD}
    - f_{SYS}=8MHz: 2.2V~5.5V
    - f_{SYS}=12MHz: 2.7V~5.5V
    - f_{SYS}=16MHz: 3.3V~5.5V
  - V_{CC}=3V~28V
- Up to 0.25μs instruction cycle with 16MHz system clock
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 8/12/16MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- OTP Program Memory: (2K-16)×16
- Data Memory: 128×8
- OTP ROM Parameter Program – ORPP
- Watchdog Timer function
- 9 bidirectional I/O lines
- Single external interrupt line shared with I/O pin
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
- Auto-adjust High Resolution PWM with Delay Lock Loop/Dead Time
- Over Current Protection – OCP
- Over/Under Voltage Protection – OUVP
- Single Time-Base function for generation of fixed time interrupt signals
- Serial Peripheral Interface – SPI
- Fully-duplex / Half-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- 6 external channel 12-bit resolution A/D converter with internal reference voltage V_{VR}
- Integrated divider resistor circuits
- Low voltage reset function
- Integrated 5V/30mA LDO with ±1.5% accuracy
- Two level shift functions for driving external N-channel MOSFETs
- Package type: 16-pin NSOP

General Description

The device is an OTP type 8-bit high performance RISC architecture microcontroller, which includes multiple gate-driver circuits for driving N-channel MOSFETs, designed for Power Inverter applications.

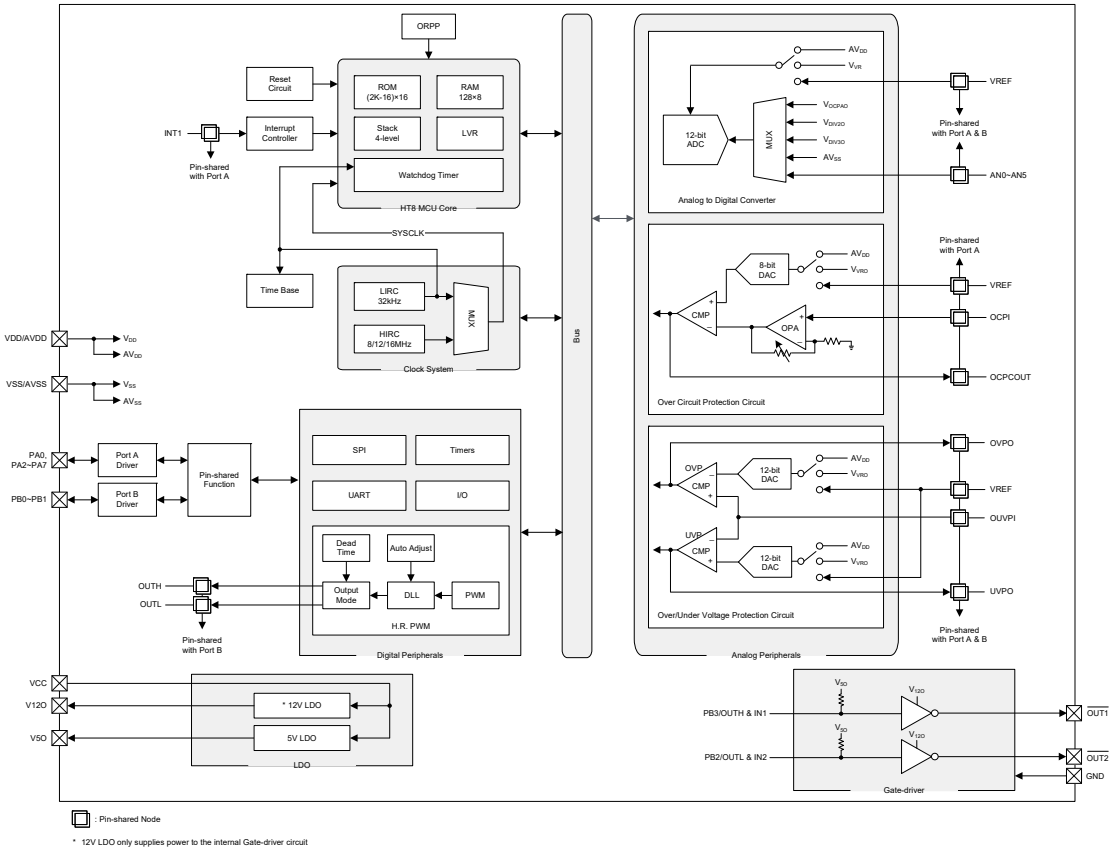
For memory features, the device is supplied with One-Time Programmable, OTP memory. Other memory includes an area of RAM Data Memory.

Analog features include a multi-channel 12-bit A/D converter, an Over Current Protection function, an Over/Under Voltage Protection function and multiple internal input divider resistor circuits. With regard to internal timers, the device includes multiple and extremely flexible Timer Modules providing functions for timing, pulse generation and PWM generation operations. Communication with the outside world is catered for by the inclusion of fully integrated SPI and UART interface functions, two popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

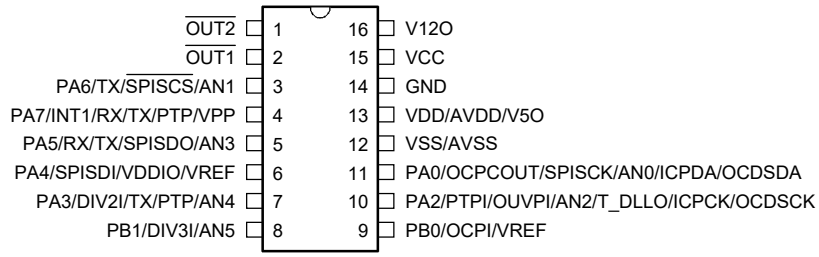
The device also includes fully integrated high and low speed oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

The inclusion of flexible I/O programming features, Time-Base function, LDO along with many other features further enhance device functionality and flexibility. The device also includes an 8-bit high resolution PWM output, which together with the over voltage/under voltage protection circuit can automatically adjust the duty cycle, to implement an auto-adjust closed loop control system. The PWM output can select either complementary output or push-pull output to meet different application requirements, especially suitable for Power Inverter boost control.

Block Diagram



Pin Assignment



HT45R7130/HT45EV7130
16 NSOP-A

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

2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the HT45EV7130 device (Flash type) which is the OCDS EV chip for the HT45R7130 device (OTP type).

3. There are several unbounded and not internally used pins, which should be properly configured to avoid unwanted power consumption resulting from floating input conditions. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.

4. The VPP pin is the High Voltage input for OTP programming and only available for the HT45R7130 device.

Pin Description

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

Pin Name	Function	OPT	I/T	O/T	Description
PA0/OCPCOUT/SPISCK/AN0/ICPDA/OCDSDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OCPCOUT	PAS0	—	CMOS	OCP Comparator output (before debounce)
	SPISCK	PAS0 IFS	ST	CMOS	SPI serial clock
	AN0	PAS0	AN	—	A/D Converter external input channel 0
	ICPDA	—	ST	CMOS	ICP data/address pin
	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
PA2/PTPI/OUVPI/AN2/ICPCK/OCDSCK	PA2	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTPI	PAS0	ST	—	PTM capture input
	OUVPI	PAS0	AN	—	OUVP input
	AN2	PAS0	AN	—	A/D Converter external input channel 2
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/DIV2I/TX/PTP/AN4	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	DIV2I	PAS0	AN	—	DIV2 external input
	TX	PAS0	—	CMOS	UART serial data output
	PTP	PAS0	—	CMOS	PTM output
	AN4	PAS0	AN	—	A/D Converter external input channel 4
PA4/SPISDI/VDDIO/VREF	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SPISDI	PAS1 IFS	ST	—	SPI serial data input
	VDDIO	PAS1	PWR	—	PA5~PA6 pin power supply
	VREF	PAS1	AN	—	ADC external reference voltage input
PA5/RX/TX/SPISDO/AN3	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	RX/TX	PAS1 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input / output in Single Wire Mode communication
	SPISDO	PAS1	—	CMOS	SPI serial data output
	AN3	PAS1	AN	—	A/D Converter external input channel 3
PA6/TX/$\overline{\text{SPISCS}}$/AN1	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	TX	PAS1	—	CMOS	UART serial data output
	$\overline{\text{SPISCS}}$	PAS1 IFS	ST	CMOS	SPI slave select
	AN1	PAS1	AN	—	A/D Converter external input channel 1

Pin Name	Function	OPT	I/T	O/T	Description
PA7/INT1/RX/TX/PTP/VPP	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up. This I/O is pin-shared with VPP and could result in increased current consumption if it is set to output high.
	INT1	INTEG INTC1 PAS1	ST	—	External interrupt 1 input
	RX/TX	PAS1 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input / output in Single Wire Mode communication
	PTP	PAS1	—	CMOS	PTM output
	VPP	PAS1	PWR	—	High Voltage input for OTP programming, not available for EV chip
PB0/OCPI/VREF	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	OCPI	PBS0	AN	—	OCP input
	VREF	PBS0	AN	—	ADC external reference voltage input
PB1/DIV3I/AN5	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	DIV3I	PBS0	AN	—	DIV3 external input
	AN5	PBS0	AN	—	A/D Converter external input channel 5
$\overline{\text{OUT1}}$	$\overline{\text{OUT1}}$	—	—	AN	Gate-driver 1 output
$\overline{\text{OUT2}}$	$\overline{\text{OUT2}}$	—	—	AN	Gate-driver 2 output
VCC	VCC	—	PWR	—	LDO and Gate-driver power supply
V12O	V12O	—	—	AN	12V LDO output (V12O only supplies power to the internal gate-driver circuit)
VDD/AVDD/V5O	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
	V5O	—	—	AN	5V LDO output
VSS/AVSS	VSS	—	PWR	—	Digital negative power supply
	AVSS	—	PWR	—	Analog negative power supply
GND	GND	—	PWR	—	Gate-driver circuit negative power supply, ground

Legend: I/T: Input type; O/T: Output type;
PWR: Power; OPT: Optional by register option;
CMOS: CMOS output; ST: Schmitt Trigger input;
AN: Analog signal

Internally Connected Signals

Several signals are not connected to external package pins. These signals are interconnection lines between the MCU and the Gate-driver and are listed in the following table. Users should properly configure the relevant pin-shared control bits and I/O control bits to implement correct interconnection.

MCU Signal Name	Gate-driver Signal Name	Function	Description
PB2/OUTL	IN2	PB2	General purpose I/O Internally connected to the gate-driver IN2
		OUTL	PWM output Internally connected to the gate-driver IN2
		IN2	Control input for OUT2 gate drive. Low active Internally connected to the MCU PB2/OUTL
PB3/OUTH	IN1	PB3	General purpose I/O Internally connected to the gate-driver IN1
		OUTH	PWM output Internally connected to the gate-driver IN1
		IN1	Control input for OUT1 gate drive. Low active Internally connected to the MCU PB3/OUTH

Note: The On/Off time of the PB2 and PB3 outputs should be configured according to the t_{ON} and t_{OFF} parameters in the Input/Output Characteristics.

Absolute Maximum Ratings

MCU Supply Voltage ..... V_{SS}-0.3V to 6.0V

MCU Input Voltage ..... V_{SS}-0.3V to V_{DD}+0.3V

V_{CC}, $\overline{OUT1}$, $\overline{OUT2}$ ..... V_{SS}-0.3V to 32V

Storage Temperature ..... -60°C to 150°C

Operating Temperature ..... -40°C to 105°C

I_{OH} Total ..... -80mA

I_{OL} Total 80mA

Total Power Dissipation 500mW

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

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	f_{SYS}=8MHz	2.2	—	5.5	V
		f_{SYS}=12MHz	2.7	—	5.5	
		f_{SYS}=16MHz	3.3	—	5.5	
	Operating Voltage – LIRC	f_{SYS}=32kHz	2.2	—	5.5	V

Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2.2V	f_{SYS}=32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	2.2V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	
		2.7V	f_{SYS}=12MHz	—	1.0	1.4	mA
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	
		3.3V	f_{SYS}=16MHz	—	1.5	3.0	mA
		5V		—	2.5	5.0	

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

1. Any digital inputs are setup in a non-floating condition and the I/O pin-shared with VPP is not setup in an output high condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Operating Current values are measured using a continuous NOP instruction program loop.

Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.2V	WDT off	—	0.45	0.80	7.00	μA
		3V		—	0.45	0.90	8.00	
		5V		—	0.5	2.0	10.0	
		2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3	5	6	
	IDLE0 Mode – LIRC	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	IDLE1 Mode – HIRC	2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	540	750	900	
		5V		—	800	1200	1440	
		3.3V	f_{SUB} on, f_{SYS}=16MHz	—	0.80	1.20	1.44	mA
		5V		—	1.4	2.0	2.4	

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

1. Any digital inputs are setup in a non-floating condition and the I/O pin-shared with VPP is not setup in an output high condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

A.C. Characteristics

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

Internal High Speed Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at a user selected HIRC frequency and user selected voltage of either 3V or 5V.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-20°C~60°C	-3%	8	+3%	
			-40°C~85°C	-4.5%	8	+4.5%	
		2.2V~5.5V	25°C	-3.5%	8	+3.5%	
			-20°C~60°C	-4.5%	8	+4.5%	
			-40°C~85°C	-5%	8	+5%	
	12MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	12	+1%	MHz
			-20°C~60°C	-1.5%	12	+1.1%	
			-40°C~85°C	-2%	12	+2%	
		2.7V~5.5V	25°C	-2.5%	12	+2.5%	
			-20°C~60°C	-2.8%	12	+2.8%	
			-40°C~85°C	-3%	12	+3%	
	16MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	16	+1%	MHz
			-20°C~60°C	-1.5%	16	+1.1%	
			-40°C~85°C	-2%	16	+2%	
		3.3V~5.5V	25°C	-2.5%	16	+2.5%	
			-20°C~60°C	-2.8%	16	+2.8%	
			-40°C~85°C	-3%	16	+3%	

Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

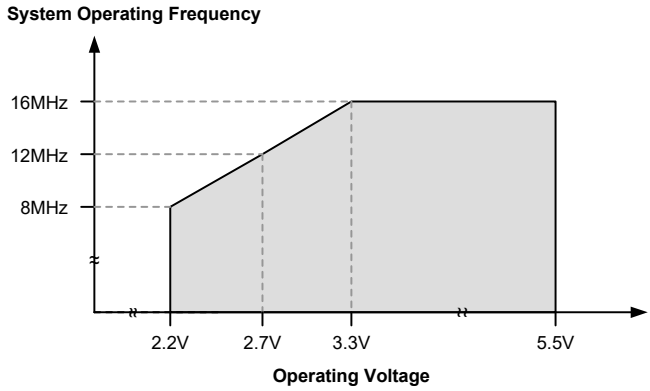
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

Internal Low Speed Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	2.2V~5.5V	25°C	-20%	32	+20%	kHz
			-40°C~85°C	-50%	32	+60%	
t_{START}	LIRC Start up Time	—	-40°C~85°C	—	—	500	μs

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time (Wake-up from Condition where f_{SYS} is off)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time (Wake-up from Condition where f_{SYS} is on)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	System Speed Switch Time (FAST to Slow Mode or SLOW to FAST Mode)	—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
t_{RSTD}	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR_{POR}=5V/ms	10	16	24	ms
	System Reset Delay Time (LVRC/WDTC Software Reset)	—	—				
	System Reset Delay Time (Reset Source from WDT Overflow Reset)	—	—	10	16	24	ms
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	180	μs

Note: 1. For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.

2. The time units, shown by the symbols t_{HIRC}, t_{SYS} etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example, $t_{HIRC}=1/f_{HIRC}$, $t_{SYS}=1/f_{SYS}$ etc.

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

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

Input/Output Characteristics

Input/Output (without Multi-power) D.C. Characteristics

For PA0, PA2~PA4, PA7 and PB0~PB1 pins.

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Ports	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Ports	3V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	
R_{PH}	Pull-high Resistance for I/O Ports(1)	3V	—	20	60	100	kΩ
		5V		10	30	50	
I_{LEAK}	Input Leakage Current for I/O Ports	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{INT}	External Interrupt Input Pin Minimum Pulse Width	—	—	10	—	—	μs
t_{TPI}	PTM Capture Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
f_{TMCLK}	PTM Maximum Timer Clock Source Frequency	5V	—	—	—	1	f_{SYS}
t_{CPW}	PTM Minimum Capture Pulse Width	—	—	t_{CPW}(2)	—	—	μs
PB2/OUTL and PB3/OUTH Output Timing Characteristics(3)							
t_{ON}	Output On Time	—	—	300	—	—	ns
t_{OFF}	Output Off Time	—	—	300	—	—	ns

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

2. If PTCAPTS=0, then $t_{CPW}=\max(2\times t_{TMCLK}, t_{TPI})$

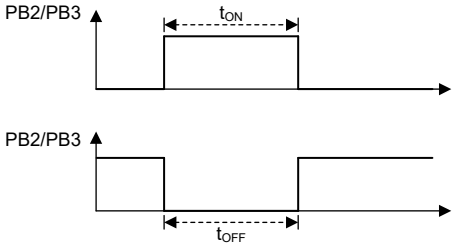
Ex1: If PTCAPTS=0, f_{TMCLK}=16MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.125μs, 0.3μs)=0.3μs

Ex2: If PTCAPTS=0, f_{TMCLK}=8MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.25μs, 0.3μs)=0.3μs

Ex3: If PTCAPTS=0, f_{TMCLK}=4MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.5μs, 0.3μs)=0.5μs

Where $t_{TMCLK}=1/f_{TMCLK}$.

3. The timing diagram is shown below.



Input/Output (with Multi-power) D.C. Characteristics

For PA5~PA6 pins.

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Power Supply – V_{DD0}	—	—	2.2	5.0	5.5	V
V_{DDIO}	Power Supply – V_{DD1}	—	—	2.2	—	V_{DD}	V
V_{IL}	Input Low Voltage for I/O Ports	5V	Pin power=V_{DDn}, V_{DDn}=V_{DD}, n=0~1	0	—	1.5	V
		—	Pin power=V_{DDn}, n=0~1	0	—	0.2V_{DDn}	
V_{IH}	Input High Voltage for I/O Ports	5V	Pin power=V_{DDn}, V_{DDn}=V_{DD}, n=0~1	3.5	—	5.0	V
		—	Pin power=V_{DDn}, n=0~1	0.8V_{DDn}	—	V_{DDn}	
I_{OL}	Sink Current for I/O Ports	3V	V_{OL}=0.1V_{DDn}, V_{DDn}=V_{DD}, n=0~1	16	32	—	mA
		5V	V_{OL}=0.1V_{DDn}, V_{DDn}=V_{DD}, n=0~1	32	65	—	
			V_{OL}=0.1V_{DDn}, V_{DDn}=3V, n=0~1	20	40	—	
I_{OH}	Source Current for I/O Ports	3V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1	-4	-8	—	mA
		5V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1	-8	-16	—	
			V_{OH}=0.9V_{DDn}, V_{DDn}=3V, n=0~1	-2.5	-5.0	—	
R_{PH}	Pull-high Resistance for I/O Ports (1)	3V	V_{DDn}=V_{DD}, n=0~1	20	60	100	kΩ
		5V	V_{DDn}=V_{DD}, n=0~1	10	30	50	
			V_{DDn}=3V, n=0~1	36	110	180	
I_{LEAK}	Input Leakage Current for I/O Ports	5V	V_{IN}=V_{SS} or V_{IN}=V_{DDn}, n=0~1	—	—	±1	μA

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

2. The actual V_{DDn} in the "Conditions" column, which can be V_{DD} or V_{DDIO}, is determined by the "V_{DD}" column value and the individual V_{DDn} voltage range.

3. When V_{DDn} is used as the I/O power, a 0.1μF bypass capacitor should be added close to the VDDn pin.

Memory Electrical Characteristics

Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
OTP Program Memory							
V_{DD}	V_{DD} for Read – ORPP	—	—	2.2	5.0	5.5	V
	V_{DD} for Write – ORPP	—	—	4.5	5.0	5.5	V
V_{PP}	V_{PP} for Write – ORPP	—	—	8.25	8.50	8.75	V
t_{WR}	Write Cycle Time – ORPP	—	—	—	300	450	μs
E_P	Cell Endurance – ORPP	—	—	1	—	—	W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
Flash Program Memory – for HT45EV7130 only							
t_{WR}	Write Cycle Time	—	—	—	2.2	2.7	ms
t_{ACTV}	ROM Activation Time	—	Wake-up from Power Down Mode	32	—	64	μs
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

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

LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	-3%	2.1	+3%	V
			LVR enable, voltage select 2.55V		2.55		
			LVR enable, voltage select 3.15V		3.15		
			LVR enable, voltage select 3.8V		3.8		
t_{LVR}	Minimum Low Voltage Width to Reset	—	TLVR[1:0]=00B	120	240	480	μs
			TLVR[1:0]=01B	0.5	1.0	2.0	ms
			TLVR[1:0]=10B	1	2	4	
			TLVR[1:0]=11B	2	4	8	
I_{LVR}	Additional Current for LVR Enable	5V	—	—	—	14	μA

A/D Converter Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	1.8	—	V_{DD}	V
N_R	Resolution	—	—	—	—	12	Bit
DNL	Differential Non-linearity	2.2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V_{REF}=V_{DD}, t_{ADCK}=0.5μs	-3	—	3	LSB
		3V					
		5V					
		2.2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V_{REF}=V_{DD}, t_{ADCK}=10μs				
		3V					
		5V					
INL	Integral Non-linearity	2.2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V_{REF}=V_{DD}, t_{ADCK}=0.5μs	-4	—	4	LSB
		3V					
		5V					
		2.2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V_{REF}=V_{DD}, t_{ADCK}=10μs				
		3V					
		5V					
I_{ADC}	Additional Current Consumption for A/D Converter Enable	2.2V	No load, t_{ADCK}=2.0μs	—	280	400	μA
		3V	No load, t_{ADCK}=0.5μs	—	450	600	
		5V		—	850	1000	
t_{ADCK}	Clock Period	2.2V~5.5V	—	0.5	—	10.0	μs
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t_{ADS}	Sampling Time	—	—	—	4	—	t_{ADCK}
t_{ADC}	Conversion Time (Including A/D Sample and Hold Time)	—	—	—	16	—	t_{ADCK}
I_{PGA}	Additional Current Consumption for PGA Enable	2.55V	No load, PGAIS=1	—	500	1000	μA
		3V		—	600	1200	
		5V		—	700	1400	
V_{OR}	PGA Maximum Output Voltage Range	2.2V	—	V_{SS}+0.1	—	V_{DD}-0.1	V
		3V		V_{SS}+0.1	—	V_{DD}-0.1	
		5V		V_{SS}+0.1	—	V_{DD}-0.1	
V_{VR}	PGA Fix Output Voltage	2.55V~5.5V	V_{RI}=V_{BGREF} (PGAIS=1)	-1.5%	2.4	+1.5%	V

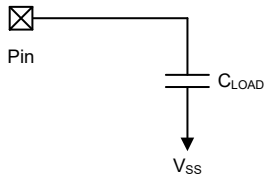
Over/Under Voltage Protection Electrical Characteristics

Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Supply Voltage	—	—	2.2	—	5.5	V
I_{OUVP}	Operating Current	3V	UVPEN=1, OVPEN=1, DAC V_{REF}=2.5V	—	520	700	μA
		5V		—	550	750	
V_{OS}	Input Offset Voltage	3V	With calibration	-2	—	2	mV
		5V		-2	—	2	
		3V	Without calibration (OVPCOF[4:0]=10000B)	-10	—	10	
			Without calibration (UVPCOF[4:0]=10000B)	-10	—	10	
		5V	Without calibration (OVPCOF[4:0]=10000B)	-10	—	10	
			Without calibration (UVPCOF[4:0]=10000B)	-10	—	10	
V_{HYS}	Hysteresis	3V	—	10	25	45	mV
		5V		10	25	45	
V_{CM}	Common Mode Voltage Range	3V	—	V_{SS}	—	V_{DD} -1.4	V
		5V		V_{SS}	—	V_{DD} -1.4	
DNL	Differential Non-linearity	3V	DAC V_{REF}=V_{DD} @ -40°C	—	—	±16	LSB
			DAC V_{REF}=V_{DD} @ 25°C	—	—	±3	
			DAC V_{REF}=V_{DD} @ 85°C	—	—	±3	
		5V	DAC V_{REF}=V_{DD} @ -40°C	—	—	±16	
			DAC V_{REF}=V_{DD} @ 25°C	—	—	±3	
			DAC V_{REF}=V_{DD} @ 85°C	—	—	±3	
INL	Integral Non-linearity	3V	DAC V_{REF}=V_{DD} @ -40°C	—	—	±16	LSB
			DAC V_{REF}=V_{DD} @ 25°C	—	—	±4	
			DAC V_{REF}=V_{DD} @ 85°C	—	—	±4	
		5V	DAC V_{REF}=V_{DD} @ -40°C	—	—	±16	
			DAC V_{REF}=V_{DD} @ 25°C	—	—	±4	
			DAC V_{REF}=V_{DD} @ 85°C	—	—	±4	
t_{RP}	Response Time	3V	With 100mV overdrive (Note)	—	200	250	ns
		5V		—	230	300	

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

Load Condition



Over Current Protection Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OCP}	Operating Current	3V	OCPEN[1:0]=01, OCPVRS[1:0]=10, OCPCHY=1, G[2:0]=111	—	300	500	μA
		5V		—	450	600	
V_{REF}	DAC Reference Voltage	3V	OCPVRS[1:0]=01	2.0	—	V_{DD}	V
		5V		2.0	—	V_{DD}	
V_{OS_CMP}	Comparator Input Offset Voltage	3V	Without calibration (OCPCOF[4:0]=10000B)	-15	—	15	mV
		5V		-15	—	15	
		3V	With calibration	-2	—	2	
		5V		-2	—	2	
V_{HYS}	Hysteresis	3V	—	10	40	60	mV
		5V		10	40	60	
V_{CM_CMP}	Comparator Common Mode Voltage Range	3V	—	V_{SS}	—	V_{DD}-1.0	V
		5V		V_{SS}	—	V_{DD}-1.0	
V_{OS_OPA}	OPA Input Offset Voltage	3V	Without calibration (OCPOOF[5:0]=100000B)	-15	—	15	mV
		5V		-15	—	15	
		3V	With calibration	-2	—	2	
		5V		-2	—	2	
V_{CM_OPA}	OPA Common Mode Voltage Range	3V	—	V_{SS}	—	V_{DD}-1.4	V
		5V		V_{SS}	—	V_{DD}-1.4	
V_{OR}	OPA Maximum Output Voltage Range	3V	—	V_{SS}+0.1	—	V_{DD}-0.1	V
		5V		V_{SS}+0.1	—	V_{DD}-0.1	
Ga	PGA Gain Accuracy	3V	All gains	-5	—	5	%
		5V		-5	—	5	
Ro	R2R Output Resistance	3V	—	—	10	—	kΩ
		5V		—	10	—	
DNL	Differential Non-linearity	3V	DAC V_{REF}=V_{DD}	—	—	±1.5	LSB
		5V		—	—	±1	
INL	Integral Non-linearity	3V	DAC V_{REF}=V_{DD}	—	—	±2	LSB
		5V		—	—	±1.5	

Divider Resistor Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
R_{DIV2}	The Sum of DIV2 R1 and R2	3V	—	2	4	6	kΩ
		5V	—	2	4	6	
R_{RDIV2}	The Ratio of DIV2 R1 and R2	3V	—	-1%	1:1	+1%	—
		5V	—	-1%	1:1	+1%	
R_{DIV3}	The Sum of DIV3 R1, R2 and R3	3V	—	1.5	3	4.5	kΩ
		5V	—	1.5	3	4.5	
RR_{DIV3}	The Ratio of DIV3 R1, R2 and R3	3V	—	-1%	2:1	+1%	—
		5V	—	-1%	2:1	+1%	

Delay Lock Loop Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DLL}	Operating Current	3V	DLLEN=1	—	1.2	1.5	mA
		5V		—	1.3	1.8	

Gate-driver Electrical Characteristics

V_{CC}=15V, Ta=25°C, unless otherwise specified

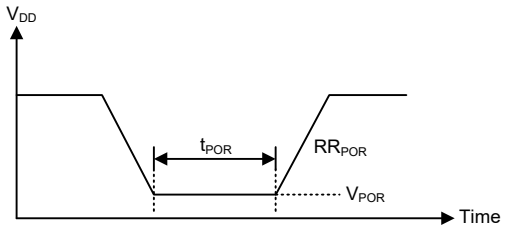
Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
V_{CC}	Supply Voltage	—	3	—	28	V
I_{CC}	Supply Standby Current	V_{CC}=12V, V5O with no load	—	6	11	μA
V_{5O} Regulator						
V_{5O}	V_{5O} Output Voltage	V_{CC}=V_{5O}+1V, I_{LOAD}=1mA	4.9	5.0	5.1	V
I_{LOAD} (Note)	V_{5O} Output Current	V_{CC}=V_{5O}+1V to 28V (w/o thermal limited)	30	—	—	mA
ΔV_{5O}	V_{5O} Load Regulation	V_{CC}=V_{5O}+1V, I_{LOAD}=0 to 30mA	—	16	27	mV
V_{5O_DROP}	V_{5O} Dropout Voltage	V_{CC}=V_{5O}+1V, I_{LOAD}=1mA	—	—	20	mV
$\frac{\Delta V_{5O}}{\Delta V_{CC} \times V_{5O}}$	V_{5O} Line Regulation	V_{CC} from (V_{5O}+1) to 28V	—	0.2	0.3	%/V
$\frac{\Delta V_{5O}}{\Delta Ta \times V_{5O}}$	V_{5O} Temperature Coefficient	V_{CC}=V_{5O}+1V, I_{LOAD}=10mA. Ta=-40°C~85°C	—	100	200	ppm/°C
$\frac{\Delta V_{5O}}{\Delta Ta}$	V_{5O} Temperature Coefficient	V_{CC}=V_{5O}+1V, I_{LOAD}=10mA. Ta=-40°C~85°C	—	±0.9	±2	mV/°C
PSRR	V_{5O} Power Supply Rejection Ratio	I_{LOAD}=30mA	—	60	—	dB
Noise	V_{5O} Output Noise	I_{LOAD}=30mA, BW=10~100kHz	—	50	—	μV_{RMS}
V_{12O} Regulator (V_{12O} only supplies power to the internal gate-driver circuit)						
V_{12O}	V_{12O} Output Voltage	V_{CC}=2.5V to 13V. I_{V12O}=10mA	V_{CC}-1	V_{CC}-0.5	—	V
		V_{CC}=13V to 28V. I_{V12O}=10mA	11.4	12	12.6	V
Gate Drive						
I_{source}	OUTx Peak Source Gate Current	C_{LOAD}=200nF	0.45	0.73	—	A
I_{sink}	OUTx Peak Source Gate Current	C_{LOAD}=200nF	0.45	0.54	—	A

Note: Output current is determined by the output voltage might keep 2% voltage drop compared to the output voltage when 1mA load current.

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



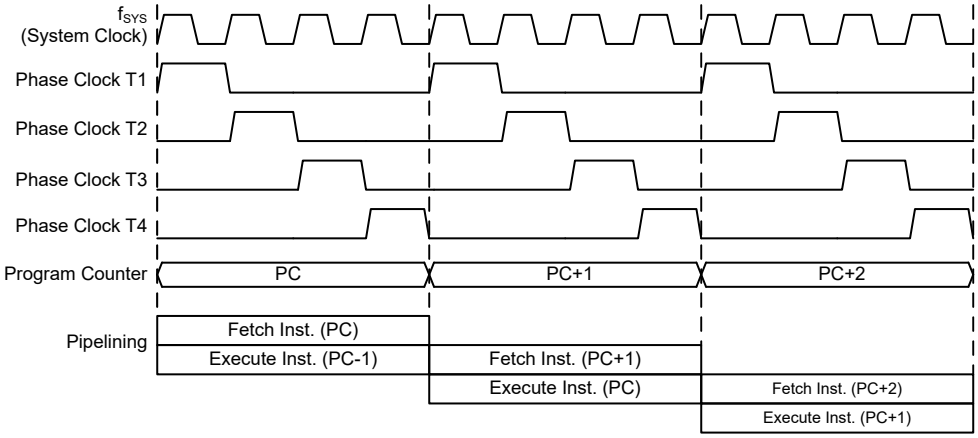
System Architecture

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

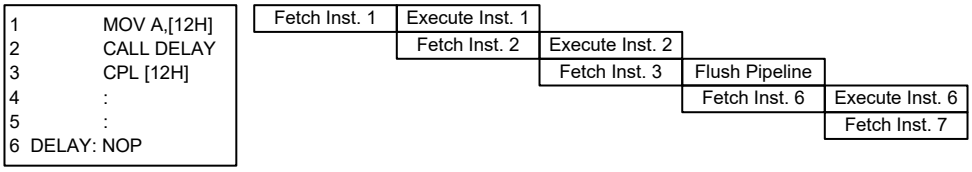
Clocking and Pipelining

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

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



System Clocking and Pipelining



Instruction Fetching

Program Counter

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

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

Program Counter	
High Byte	**Low Byte (PCL)**
PC10~PC8	PCL7~PCL0

Program Counter

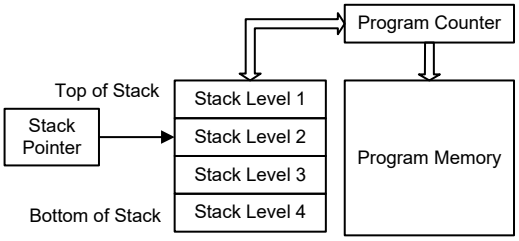
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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 4 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, STKPTR[1:0]. 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.



- **STKPTR Register**

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

Bit 7 **OSF**: Stack overflow flag
0: No stack overflow occurred
1: Stack overflow occurred
When the stack is full and a CALL instruction is executed or when the stack is empty and a RET instruction is executed, the OSF bit will be set high. The OSF bit is cleared only by software and cannot be reset automatically by hardware.

Bit 6~2 Unimplemented, read as "0"

Bit 1~0 **D1~D0**: Stack pointer register bit 1 ~ bit 0

The following example shows how the Stack Pointer and Stack Overflow Flag change when program branching conditions occur.

(1) When the CALL subroutine instruction is executed 5 times continuously and the RET instruction is not executed during the period, the corresponding changes of the STKPTR[1:0] and OSF bits are as follows:

CALL Execution Times	0	1	2	3	4	5
STKPTR[1:0] Bit Value	0	1	2	3	0	1
OSF Bit Value	0	0	0	0	0	1

(2) When the OSF bit is set high and not cleared, it will remain high no matter how many times the RET instruction is executed.

(3) When the stack is empty, the RET instruction is executed 4 times continuously, the corresponding changes of the STKPTR[1:0] and OSF bits are as follows:

RET Execution Times	0	1	2	3	4
STKPTR[1:0] Bit Value	0	3	2	1	0
OSF Bit Value	0	1	1	1	1

Arithmetic and Logic Unit – ALU

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

- Arithmetic operations:

  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations:

  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation:

  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement:

  INCA, INC, DECA, DEC
- Branch decision:

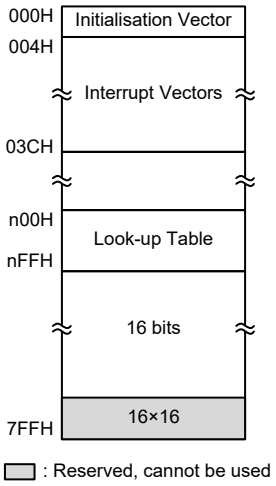
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

OTP Program Memory

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

Structure

The Program Memory has a capacity of (2K-16)×16 bits. Note that the subtractive 16×16 bits space is reserved and cannot be used. 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 registers, TBLP and TBHP. These registers define the total address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the Program Memory using the "TABRD [m]" or "TABRDL[m]" instruction. 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.

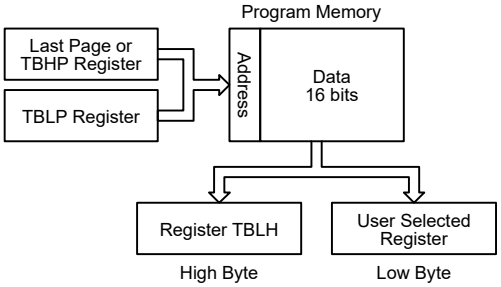


Table Program Example

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 "0700H" which refers to the start address of the last page within the 2K 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 "0706H" 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 TBHP and TBLP registers if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

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

Table Read Program Example

```
tempreg1 db ?     ; temporary register #1
tempreg2 db ?     ; temporary register #2
:
:
mov a, 06h        ; initialise low table pointer - note that this address
                  ; is referenced
mov tblp, a       ; to the last page or the page that tbhp pointed
mov a, 07h        ; initialise high table pointer
mov tbhp, a
:
:
tabrd tempreg1    ; transfers value in table referenced by table pointer data at
                  ; program memory address "0706H" 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 "0705H" transferred to
                  ; tempreg2 and TBLH in this example the data "1AH" is
                  ; transferred to tempreg1 and data "0FH" to register tempreg2
:
:
org 0700h         ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

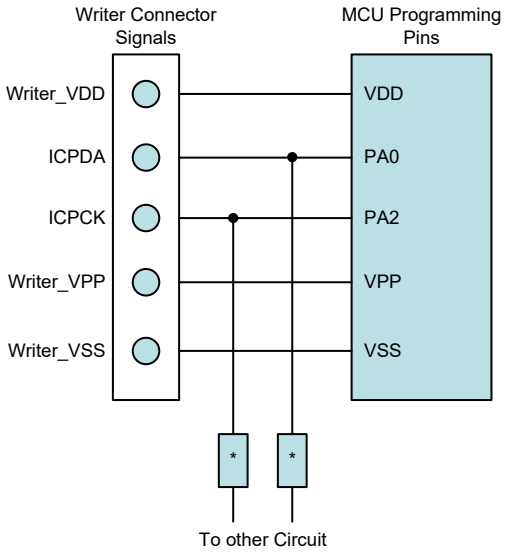
In Circuit Programming – ICP

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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VPP	VPP	Programming OTP ROM power supply (8.5V)
VDD	VDD	Power Supply
VSS	VSS	Ground

The Program Memory can be programmed serially in-circuit using this 5-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Three additional lines are required for the power supply. The technical details regarding the in-circuit programming of the device are 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 EV chip named HT45EV7130 which is used to emulate the HT45R7130 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 is 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 device for debugging, other pin functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Program Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

OTP ROM Parameter Program – ORPP

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

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

ORPP Registers

Three registers control the overall operation of the internal ORPP function. These are data registers ODL and ODH, and a control register OCR.

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

ORPP Register List

- **ODL Register**

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

Bit 7~0 **D7~D0**: ORPP Program Memory data bit 7~bit 0

• **ODH Register**

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

Bit 7~0 **D15~D8**: ORPP Program Memory data bit 15~bit 8

• **OCR Register**

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

Bit 7~4 Unimplemented, read as "0"

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

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

Bit 1~0 Unimplemented, read as "0"

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

ORPP Writing Data to the OTP Program Memory

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

ORPP Reading Data from the OTP Program Memory

For ORPP read operation the desired address should first be placed in the TBLP register. Then the data can be retrieved from the program memory using the "TABRDL [m]" instruction. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

Programming Considerations

Care must be taken that data is not inadvertently written to the OTP Program Memory. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process.

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

Programming Examples

ORPP Reading data from the OTP Program Memory

```
Tempreg1 db?            ; temporary register
MOV A, 03H
MOV TBLP, A             ; set read address 03H
TABRDL Tempreg1         ; transfers value in table (last page)
                        ; referenced by table pointer,
                        ; data at program memory address "0703H"
                        ; transferred to tempreg1 and TBLH
```

ORPP Writing Data to the OTP Program Memory

```
MOV A, ORPP_ADRES       ; user defined address
MOV TBLP, A
MOV A, ORPP_DATA_L      ; user defined data
MOV ODL, A
MOV A, ORPP_DATA_H
MOV ODH, A
MOV A, 00H
MOV OCR, A
CLR EMI
SET WREN                ; set WREN bit, enable write operation
SET WR                  ; start Write Cycle - set WR bit - executed immediately
                        ; after setting WREN bit
SET EMI
BACK:
SZ  WR                  ; check for write cycle end
JMP BACK
NOP
```

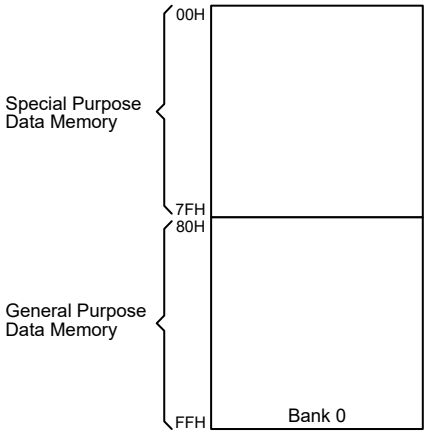
Data Memory

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

Structure

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

The start address of the Data Memory for the device is 00H. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the address range of the General Purpose Data Memory is from 80H to FFH.



Data Memory Structure

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Bank 0
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	
0CH	
0DH	
0EH	
0FH	RSTFC
10H	
11H	
12H	IFS
13H	PMPS
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	PB
19H	PBC
1AH	PBPU
1BH	INTEG
1CH	INTC0
1DH	INTC1
1EH	INTC2
1FH	INTC3
20H	MFI
21H	STKPTR
22H	
23H	
24H	
25H	
26H	
27H	
28H	SCC
29H	HIRCC
2AH	LVRC
2BH	TLVRC
2CH	WDTC
2DH	PAS0
2EH	PAS1
2FH	PBS0
30H	
31H	
32H	
33H	USR
34H	UCR1
35H	UCR2
36H	UCR3
37H	BRDH
38H	BRDL
39H	UFCR
3AH	TXR_RXR
3BH	RxCNT
3CH	SPIC0
3DH	SPIC1
3EH	SPID
3FH	

	Bank 0
40H	
41H	
42H	
43H	CTMC0
44H	CTMC1
45H	CTMDL
46H	CTMDH
47H	CTMAL
48H	CTMAH
49H	PTMC0
4AH	PTMC1
4BH	PTMDL
4CH	PTMDH
4DH	PTMAL
4EH	PTMAH
4FH	PTMRPL
50H	PTMRPH
51H	PSCR
52H	TBC
53H	SADC0
54H	SADC1
55H	SADC2
56H	SADOL
57H	SADOH
58H	VBGRC
59H	DIV2SC
5AH	DIV3SC
5BH	
5CH	OCR
5DH	ODL
5EH	ODH
5FH	
60H	OUVPC0
61H	OUVPC1
62H	OUVPC2
63H	OUVPC3
64H	OVPDAL
65H	OVPDAH
66H	UVPDAL
67H	UVPDAH
68H	OCPC0
69H	OCPC1
6AH	OCPDA
6BH	OCPOCAL
6CH	OCPCCAL
6DH	
6EH	
6FH	
70H	PWMP
71H	PWMDL
72H	PWMDH
73H	PWMC0
74H	PWMC1
75H	
76H	ADJDT
77H	ADJS
78H	ADJC
79H	ADJMAXL
7AH	ADJMAXH
7BH	ADJMINL
7CH	ADJMINH
7DH	ADJBL
7EH	ADJBH
7FH	DLLC

: Unused, read as 00H : Reserved, cannot be changed

Special Purpose Data Memory Structure

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0/IAR1 and MP0/MP1 can together access data from Bank 0. 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, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0/MP1, together with Indirect Addressing Register, IAR0/IAR1, are used to access data from Bank 0.

The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.

Indirect Addressing Program Example

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

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

Accumulator – ACC

The Accumulator is central to the operation of any microcontroller and is closely related with operations carried out by the ALU. The Accumulator is the place where all intermediate results from the ALU are stored. Without the Accumulator it would be necessary to write the result of each calculation or logical operation such as addition, subtraction, shift, etc., to the Data Memory resulting in higher programming and timing overheads. Data transfer operations usually involve the temporary storage function of the Accumulator; for example, when transferring data between one user-defined register and another, it is necessary to do this by passing the data through the Accumulator as no direct transfer between two registers is permitted.

Program Counter Low Byte Register – PCL

To provide additional program control functions, the low byte of the Program Counter is made accessible to programmers by locating it within the Special Purpose area of the Data Memory. By manipulating this register, direct jumps to other program locations are easily implemented. Loading a value directly into this PCL register will cause a jump to the specified Program Memory location, however, as the register is only 8-bit wide, only jumps within the current Program Memory page are permitted. When such operations are used, note that a dummy cycle will be inserted.

Look-up Table Registers – TBLP, TBHP, TBLH

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP and TBHP are the table pointers and indicate the location where the table data is located. Their value must be setup before any table read commands are executed. Their value can be changed, for example using the “INC” or “DEC” instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition, operations related to the status register may give different results due to the different instruction operations. The TO flag can be affected only by a system power-up, a WDT time-out or by executing the “CLR WDT” or “HALT” instruction. The PDF flag is affected only by executing the “HALT” or “CLR WDT” instruction or during a system power-up.

The Z, OV, AC and C flags generally reflect the status of the latest operations.

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.

- PDF is cleared by a system power-up or executing the “CLR WDT” instruction. PDF is set by executing the “HALT” instruction.
- TO is cleared by a system power-up or executing the “CLR WDT” or “HALT” instruction. TO is set by a WDT time-out.

In addition, on entering an interrupt sequence or executing a subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status registers are important and if the subroutine can corrupt the status register, precautions must be taken to correctly save it.

• STATUS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R	R	R/W	R/W	R/W	R/W
POR	—	—	0	0	x	x	x	x

“x”: unknown

Bit 7~6 Unimplemented, read as “0”

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

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

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

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

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

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

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

Oscillators

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

Oscillator Overview

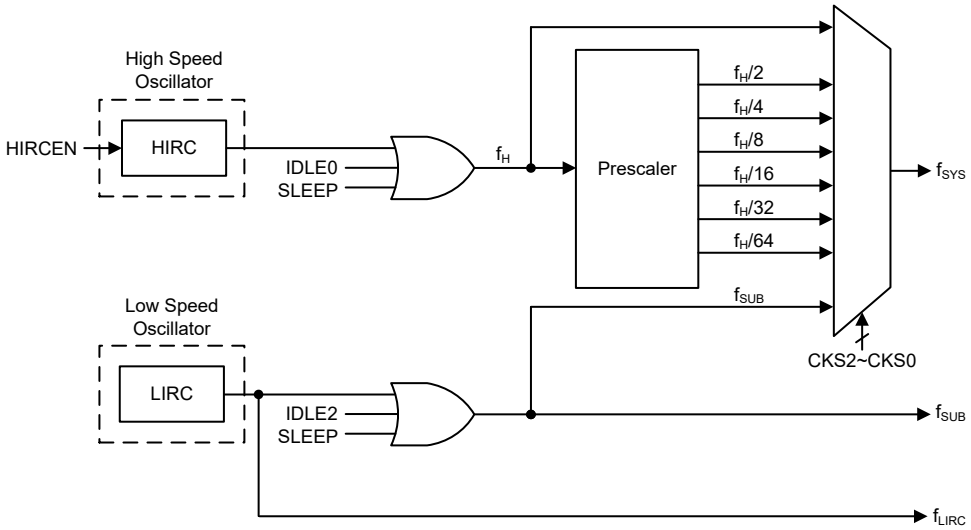
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupt. Two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators.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

Type	Name	Frequency
Internal High Speed RC	HIRC	8/12/16MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two methods of generating the system clock, a high speed oscillator and a low speed oscillator. The high speed oscillator is the internal 8/12/16MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.



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 three fixed frequencies of 8MHz, 12MHz and 16MHz, which are selected by the HIRC1~HIRC0 bits in the HIRCC register. These bits must also be setup to match

the selected configuration option frequency to ensure that the HIRC frequency accuracy specified in the A.C. Characteristics is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is a fully integrated RC oscillator with a typical frequency of 32kHz, requiring no external components for its implementation.

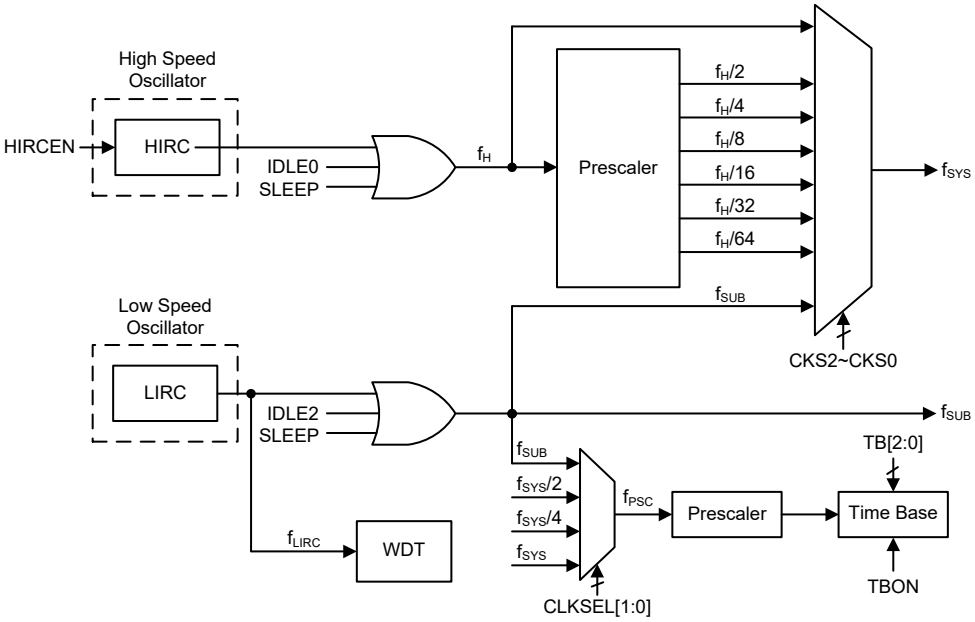
Operating Modes and System Clocks

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

System Clocks

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

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



Device Clock Configurations

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

System Operation Modes

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

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

"x": don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.
2. The f_{LIRC} clock can be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

FAST Mode

This is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come 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 CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

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

SLEEP Mode

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

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

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

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

System Operating Mode Control Register List

- **SCC Register**

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

Bit 7~5 **CKS2~CKS0**: System clock selection
000: f_H
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: f_{SUB}

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from f_H or f_{SUB}, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4~2 Unimplemented, read as “0”

Bit 1 **FHIDEN**: High Frequency oscillator control when CPU is switched off
0: Disable
1: Enable

This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing a “HALT” instruction.

Bit 0 **FSIDEN**: Low Frequency oscillator control when CPU is switched off
0: Disable
1: Enable

This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing a “HALT” instruction.

Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.

Clock switching delay time=$4\times t_{SYS}+[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$, where $t_{Curr.}$ indicates the current clock period, $t_{Tar.}$ indicates the target clock period and t_{SYS} indicates the current system clock period.

• HIRCC Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection
00: 8MHz
01: 12MHz
10: 16MHz
11: 8MHz

When the HIRC oscillator is enabled or the HIRC frequency selection is changed by the application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1.

It is recommended that the HIRC frequency selected by these two bits should be the same with the frequency determined by the configuration option to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable

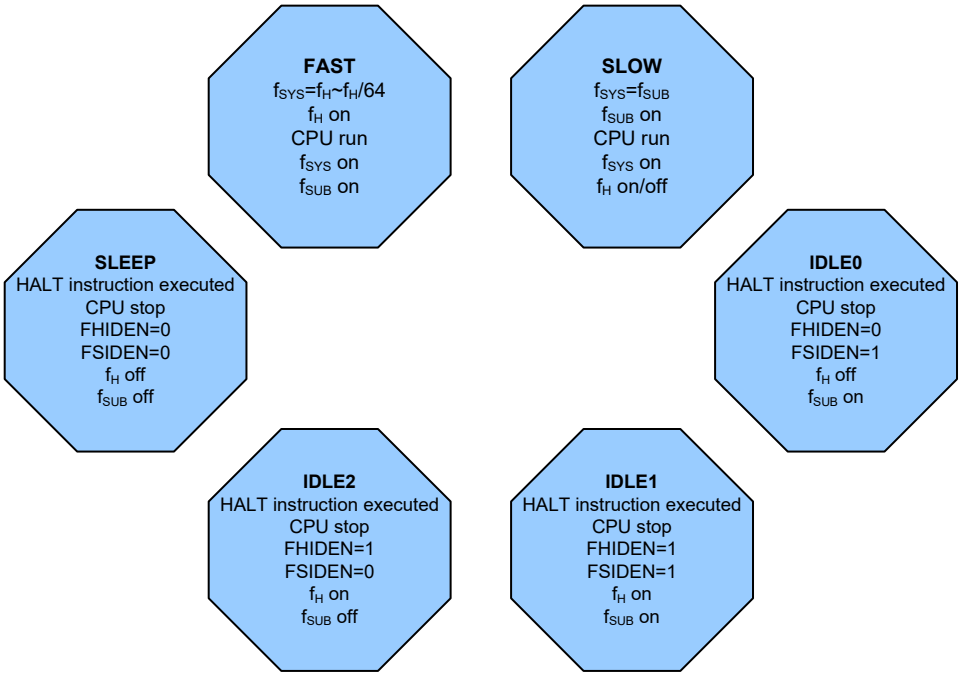
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator or the HIRC frequency selection is changed by the application program, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

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

Operating Mode Switching

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

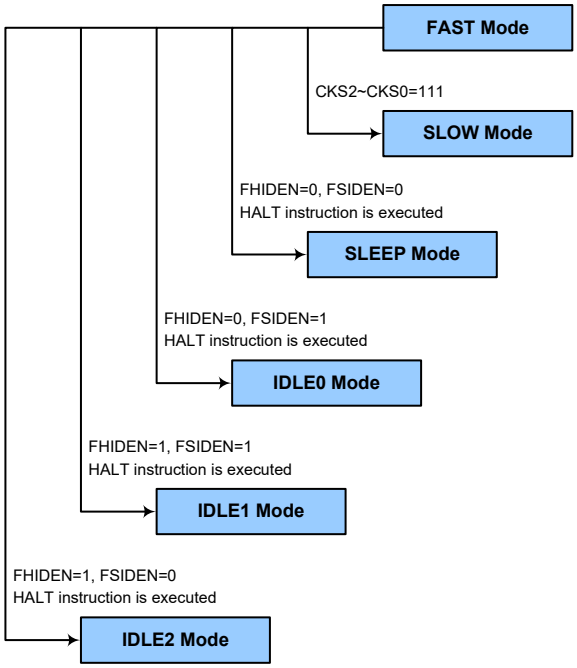
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



FAST Mode to SLOW Mode Switching

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

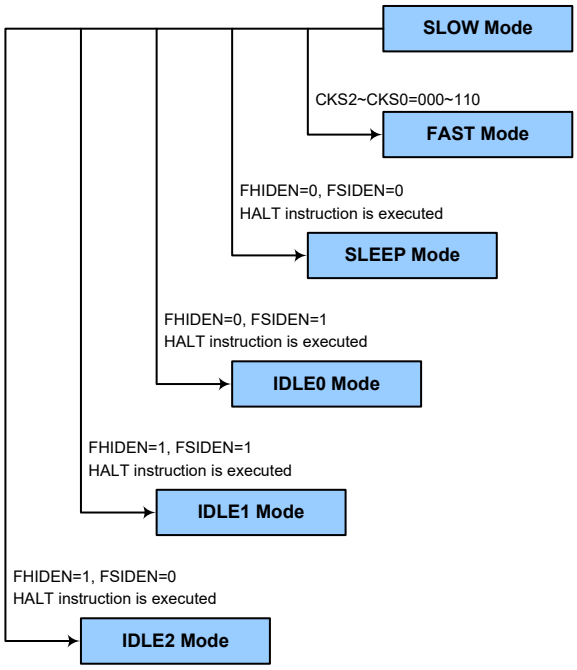
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

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

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



Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

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

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

Entering the IDLE2 Mode

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

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

Standby Current Considerations

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

Care must also be taken with the loads, which are connected to I/O pins, which are 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 and IDLE2 Mode the high speed oscillator is on, if the peripheral function clock source is derived from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

To minimise power consumption the device can enter the SLEEP or any IDLE Mode, where the CPU will be switched off. However, when the device is woken up again, 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

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

Each pin on Port A can be 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_{LIRC} which is sourced from 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.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the WDT enable/disable and software reset MCU operation.

- **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 enable control
10101: Disable
01010: Enable
Other values: Reset MCU

If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set to 1.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$

These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period.

- **RSTFC Register**

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

"x": unknown

Bit 7~3 Unimplemented, read as "0"

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

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

Bit 0 **WRF**: WDT control register software reset flag
0: Not occurred
1: Occurred
This bit is set 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 instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control of the Watchdog Timer and MCU software reset control. The WDT function will be enabled when the WE4~WE0 bits are set to a value of 01010B while the WDT function will be disabled if the WE4~WE0 bits are equal to 10101B. If the WE4~WE0 bits are set to any other values rather than 01010B and 10101B, it will reset the device after a delay time, t_{SRESET}. After power on these bits will have a value of 01010B.

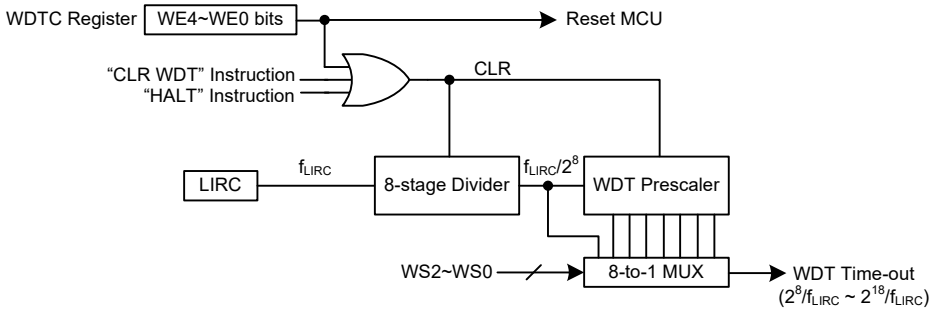
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

Watchdog Timer Function Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO 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 field, the second is using the Watchdog Timer software clear instruction and the third is via a HALT instruction.

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

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.

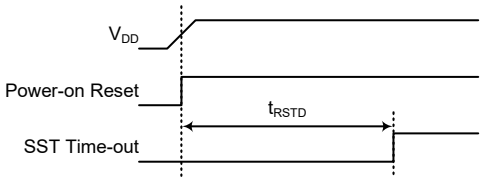
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All 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

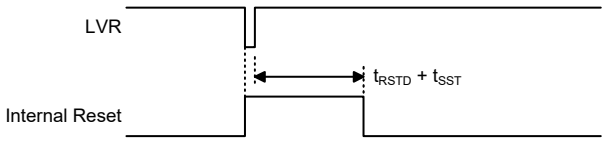
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. 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 can be enabled or disabled by the LVRC control register. If the LVRC control register is configured to enable the LVR function, the LVR function will be always enabled in the FAST or SLOW mode with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set 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 Electrical Characteristics. If the duration of the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual t_{LVR} value can be selected by the TLVR1~TLVR0 bits in the TLVRC register. The actual V_{LVR} value can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits have any other values, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set 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
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
11110000: LVR disable
Other values: Generates a MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage values above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps for greater than a t_{LVR} time. The actual t_{LVR} value can be selected by the TLVR1~TLVR0 bits in the TLVRC register. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than 11110000B and the four defined LVR voltage values above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time, t_{SRESET}. However in this situation the register contents will be reset to the POR value.

- **TLVRC Register**

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

Bit 7~2 Unimplemented, read as “0”

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

- **RSTFC Register**

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

“x”: unknown

Bit 7~3 Unimplemented, read as “0”

Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred

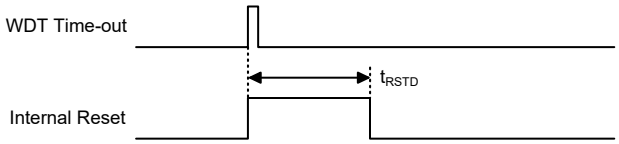
This bit is set high when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

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

Bit 0 **WRF**: WDT control register software reset flag
Refer to the Watchdog Timer Control Register section.

Watchdog Time-out Reset during Normal Operation

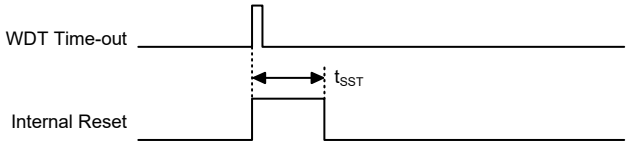
The Watchdog time-out Reset during normal operation in the FAST or SLOW Mode is the same as the hardware Low Voltage Reset except that the Watchdog time-out flag TO will be set to "1".



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



WDT Time-out Reset during SLEEP or IDLE Mode Timing Chart

Reset Initial Conditions

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

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

"u": unchanged

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

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Base	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 the microcontroller internal registers.

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	xxxx xxxx	uuuu uuuu	uuuu uuuu
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu
IAR1	xxxx xxxx	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	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	---- -xxx	---- -uuu	---- -uuu
STATUS	--00 xxxx	--1u uuuu	--11 uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu
IFS	---- 0000	---- 0000	---- uuuu
PMPS	---- --00	---- --00	---- --uu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
INTEG	---- 0000	---- 0000	---- uuuu
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
MFI	--00 --00	--00 --00	--uu --uu
STKPTR	0--- --00	0--- --00	u--- --00
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- 0001	---- 0001	---- uuuu
LVRC	0101 0101	0101 0101	uuuu uuuu
TLVRC	---- --01	---- --01	---- --uu
WDTC	0101 0011	0101 0011	uuuu uuuu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 0000	0000 0000	uuuu uuuu
USR	0000 1011	0000 1011	uuuu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
UCR1	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	uuuu uuuu
UCR3	---- ---0	---- ---0	---- ---u
BRDH	0000 0000	0000 0000	uuuu uuuu
BRDL	0000 0000	0000 0000	uuuu uuuu
UFCR	--00 0000	--00 0000	--uu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
RxCNT	---- -000	---- -000	---- -uuu
SPIC0	000- --00	000- --00	uuu- --uu
SPIC1	--00 0000	--00 0000	--uu uuuu
SPID	xxxx xxxx	xxxx xxxx	uuuu uuuu
CTMC0	0000 0000	0000 0000	uuuu uuuu
CTMC1	0000 0000	0000 0000	uuuu uuuu
CTMDL	0000 0000	0000 0000	uuuu uuuu
CTMDH	---- --00	---- --00	---- --uu
CTMAL	0000 0000	0000 0000	uuuu uuuu
CTMAH	---- --00	---- --00	---- --uu
PTMC0	0000 0---	0000 0---	uuuu u---
PTMC1	0000 0000	0000 0000	uuuu uuuu
PTMDL	0000 0000	0000 0000	uuuu uuuu
PTMDH	---- --00	---- --00	---- --uu
PTMAL	0000 0000	0000 0000	uuuu uuuu
PTMAH	---- --00	---- --00	---- --uu
PTMRPL	0000 0000	0000 0000	uuuu uuuu
PTMRPH	---- --00	---- --00	---- --uu
PSCR	---- -000	---- -000	---- -uuu
TBC	0--- -000	0--- -000	u--- -uuu
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 -000	0000 -000	uuuu -uuu
SADC2	0--0 00--	0--0 00--	u--u uu--
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
	---- xxxx	---- xxxx	---- uuuu (ADRFS=1)
VBGRC	---- ---0	---- ---0	---- ---u
DIV2SC	---- --00	---- --00	---- --uu
DIV3SC	---- --00	---- --00	---- --uu
OCR	---- 00--	---- 00--	---- uu--
ODL	0000 0000	0000 0000	uuuu uuuu
ODH	0000 0000	0000 0000	uuuu uuuu
OUVPC0	--00 0000	--00 0000	--uu uuuu
OUVPC1	--00 0000	--00 0000	--uu uuuu
OUVPC2	0001 0000	0001 0000	uuuu uuuu
OUVPC3	0001 0000	0001 0000	uuuu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
OVPDAL	0000 0000	0000 0000	uuuu uuuu
OVPDAH	---- 0000	---- 0000	---- uuuu
UVPDAL	0000 0000	0000 0000	uuuu uuuu
UVPDAH	---- 0000	---- 0000	---- uuuu
OCPC0	0000 0--0	0000 0--0	uuuu u--u
OCPC1	--00 0000	--00 0000	--uu uuuu
OCPDA	0000 0000	0000 0000	uuuu uuuu
OCPOCAL	0010 0000	0010 0000	uuuu uuuu
OCPCCAL	0001 0000	0001 0000	uuuu uuuu
PWMP	0000 0000	0000 0000	uuuu uuuu
PWMDL	0000 0000	0000 0000	uuuu uuuu
PWMDH	---- 0000	---- 0000	---- uuuiu
PWMC0	0000 0-00	0000 0-00	uuuu u-uu
PWMC1	---0 1000	---0 1000	---u uuuu
ADJDT	--00 0000	--00 0000	--uu uuuu
ADJS	0000 0000	0000 0000	uuuu uuuu
ADJC	000- xx--	000- xx--	uuu- uu--
ADJMAXL	0000 0000	0000 0000	uuuu uuuu
ADJMAXH	---- 0000	---- 0000	---- uuuu
ADJMINL	0000 0000	0000 0000	uuuu uuuu
ADJMINH	---- 0000	---- 0000	---- uuuu
ADJBL	0000 0000	0000 0000	uuuu uuuu
ADJBH	---- 0000	---- 0000	---- uuuu
DLLC	10-- ---0	10-- ---0	uu-- ---u

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1*	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1**	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1**	PAWU0
PB	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

"—": Unimplemented, read as "0"

I/O Logic Function Register List

Note: The control bit with an asterisk * should be cleared to 0 and the control bit with an asterisk ** should remain the POR value after power-on reset. This can set these unbonded and not internally used lines as outputs to prevent additional power consumption.

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the PAPU~PBPU registers, and are implemented using weak PMOS transistors.

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

- **PxPU Register**

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

PxPUn: I/O Port x Pin pull-high function control
0: Disable
1: Enable

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

Note that for the PB2~PB3 lines, which are internally connected to the Gate-driver inputs, the corresponding control bits should remain unchanged after power-on reset.

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 is selected as a general purpose input and the MCU enters the IDLE or SLEEP 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 Register

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

- **PxC Register**

Bit	7	6	5	4	3	2	1	0
Name	PxC7	PxC6	PxC5	PxC4	PxC3	PxC2	PxC1	PxC0
R/W	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" is the Port name which can be A or B. However, the actual available bits for each I/O Port may be different.

Note that for the PB2~PB3 lines, which are internally connected to the Gate-driver inputs, the corresponding PxCn bits should be cleared to zero to set these I/O lines as outputs after power-on reset, thus controlling the Gate-driver properly.

I/O Port Power Source Control

The device supports different I/O port power source selections for PA5~PA6. With the exception of $\overline{\text{RES}}$/OCDS, the multi-power function is only effective when the pin is set to have a digital input or output function.

The port power can come from either the power pin VDD or VDDIO, which is determined using the

PMPS1~PMPS0 bits in the PMPS register. The VDDIO power pin function should first be selected using the corresponding pin-shared function selection bits if the port power is supposed to come from the VDDIO pin.

An important point to know is that the input power voltage on the VDDIO pin should be equal to or less than the device supply power voltage V_{DD} when the VDDIO pin is selected as the port power supply pin.

- **PMPS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PMPS1	PMPS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PMPS1~PMPS0**: PA5~PA6 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}
If the PA4 pin is switched to the VDDIO function, and the PMPS1~PMPS0 bits are set to "1x", the VDDIO pin input voltage can be used for PA5~PA6 pin power.

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

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as INT1, PTPI, 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 input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	D3	D2	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
IFS	—	—	—	—	SPISCKPS	SPISDIPS	SPISCSBPS	RXTXPS

Pin-shared Function Selection Register List

- **PAS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	PAS05	PAS04	D3	D2	PAS01	PAS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PAS07~PAS06**: PA3 pin-shared function selection
00: PA3/DIV2I
01: TX/DIV2I
10: PTP/DIV2I
11: AN4/DIV2I

Bit 5~4 **PAS05~PAS04**: PA2 pin-shared function selection
00: PA2/PTPI
01: OUVPI
10: AN2
11: Reserved, cannot be used

Note: If the PA2 pin is setup as the OUVPI pin function, then the OUVPI will be connected together with the internal DIV3 Divider Resistor circuit output signal V_{DIV3O}, at which piont special consideration should be given to the interaction between internal and external circuits and the corresponding control bits should be properly configured according to the application requirements.

Bit 3~2 **D3~D2**: Reserved bits, should remain unchanged after power-on reset

Bit 1~0 **PAS01~PAS00**: PA0 pin-shared function selection
00: PA0
01: OCPCOUT
10: SPISCK
11: AN0

- **PAS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PAS17~PAS16**: PA7 pin-shared function selection
00: PA7/INT1/VPP
01: RX/TX
10: PTP
11: PA7/INT1/VPP

Bit 5~4 **PAS15~PAS14**: PA6 pin-shared function selection
00: PA6
01: TX
10: $\overline{\text{SPISCS}}$
11: AN1

Bit 3~2 **PAS13~PAS12**: PA5 pin-shared function selection
00: PA5
01: RX/TX
10: SPISDO
11: AN3

Bit 1~0 **PAS11~PAS10**: PA4 pin-shared function selection
00: PA4
01: SPISDI
10: VDDIO
11: VREF

• **PBS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
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 **PBS07~PBS06**: PB3 pin-shared function selection
00: PB3
01: Reserved
10: Reserved
11: OUTH

Note: The PB3/OUTH line is internally connected to the gate-driver input, INHC, these bit shold be set to 11 when the gate-driver is used.

Bit 5~4 **PBS05~PBS04**: PB2 pin-shared function selection
00: PB2
01: Reserved
10: Reserved
11: OUTL

Note: The PB2/OUTL line is internally connected to the gate-driver input, $\overline{\text{INLC}}$, these bit shold be set to 11 when the gate-driver is used.

Bit 3~2 **PBS03~PBS02**: PB1 pin-shared function selection
00: PB1/DIV3I
01: AN5/DIV3I
10: PB1/DIV3I
11: PB1/DIV3I

Bit 1~0 **PBS01~PBS00**: PB0 pin-shared function selection
00: PB0
01: OCPI
10: VREF
11: PB0

• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SPISCKPS	SPISDIPS	SPISCSBPS	RXTXPS
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as “0”

Bit 3 **SPISCKPS**: SPISCK input source pin selection
0: PA0
1: Reserved

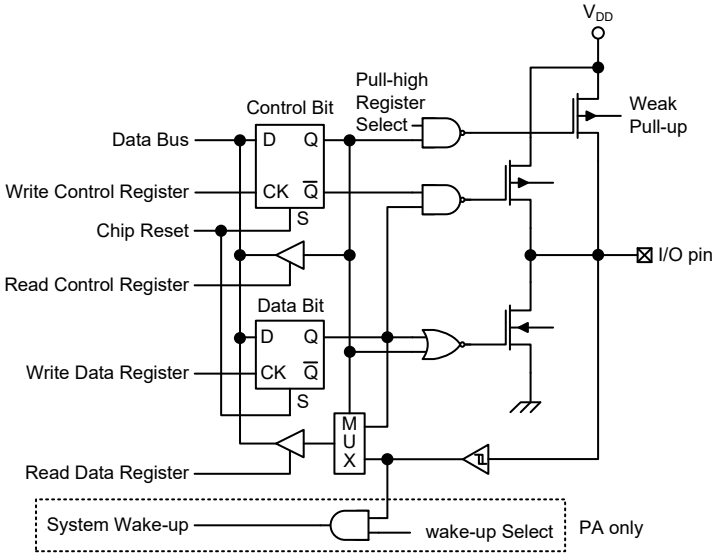
Bit 2 **SPISDIPS**: SPISDI input source pin selection
0: PA4
1: Reserved

Bit 1 **SPISCSBPS**: $\overline{\text{SPISCS}}$ input source pin selection
0: PA6
1: Reserved

Bit 0 **RXTXPS**: RX/TX input source pin selection
0: PA5
1: PA7

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 devices is the ability to control and measure time. To implement time related functions the device includes several Timer Modules, generally 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 interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

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

Introduction

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

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

TM Function Summary

TM Operation

The different types of TM offer a diverse range of functions, from simple timing operations to PWM signal generation. The key to understanding how the TM operates is to see it in terms of a free running count-up counter whose value is then compared with the value of pre-programmed internal comparators. When the free running count-up 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 xTCK2~xTCK0 bits in the xTM control registers, where “x” stands for C or P type TM. 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.

TM Interrupts

The Compact or Periodic type TM each has two internal interrupt, 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

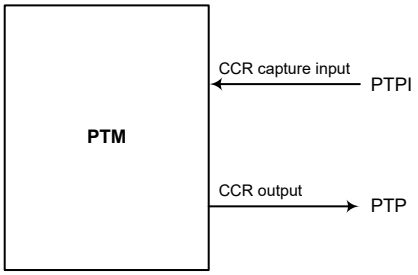
The Compact type TM has no input pins. The Periodic type TM has one input pin with the label PTPI. The PTM input pin, PTPI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTIO1~PTIO0 bits in the PTMC1 register.

The Compact type TM has no output pins. The Periodic type TM has one output pin with the label PTP. 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 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 register. The details of the pin-shared function selection are described in the pin-shared function section.

CTM		PTM	
Input	Output	Input	Output
—	—	PTPI	PTP

TM External Pins

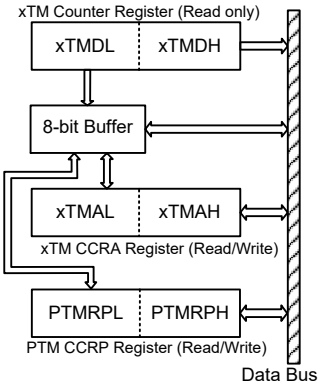


PTM Function Pin Block Diagram

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA 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 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 and CCRP low byte registers, named xTMAL and PTMRPL, using the following access procedures. Accessing the CCRA or CCRP low byte registers without following these access procedures will result in unpredictable values.

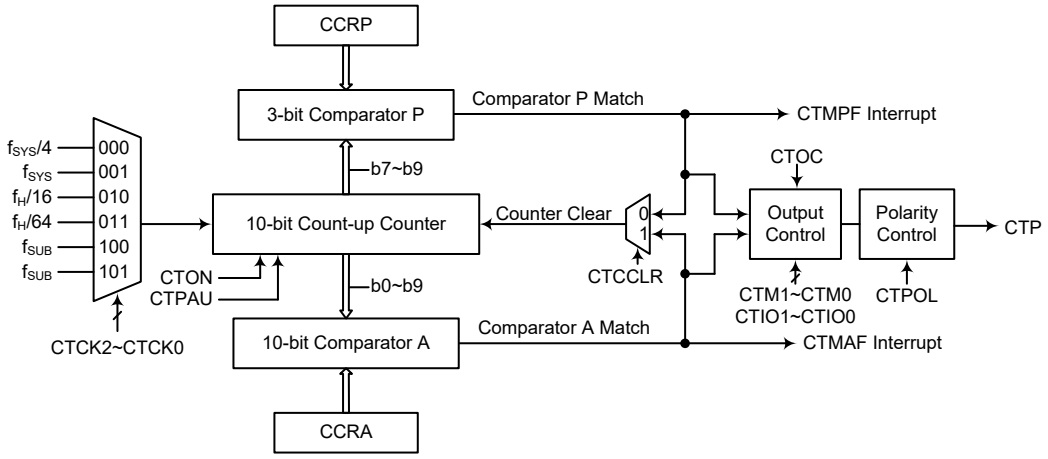


The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - Step 1. Write data to Low Byte xTMAL or PTMRPL
    - Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte xTMAH or PTMRPH
    - Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers, CCRA or CCRP
  - Step 1. Read data from the High Byte xTMDH, xTMAH or PTMRPH
    - Here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - Step 2. Read data from the Low Byte xTMDL, xTMAL or PTMRPL
    - This step reads data from the 8-bit buffer.

Compact Type TM – CTM

The Compact Type TM contains three operating modes, which are Compare Match Output, Timer/ Event Counter and PWM Output modes.



Note: The CTP is not externally bounded and unavailable.

10-Bit Compact Type TM Block Diagram

Compact Type TM Operation

The size of Compact Type TM is 10-bit wide and its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP comparator is 3-bit wide whose value is compared with the highest 3 bits in the counter while the CCRA is the 10 bits and therefore compares all counter bits.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the CTON 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 CTM interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources and can also control outputs. All operating setup conditions are selected using relevant internal registers.

Compact Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
CTMC0	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
CTMC1	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
CTMDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMDH	—	—	—	—	—	—	D9	D8
CTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMAH	—	—	—	—	—	—	D9	D8

10-Bit Compact Type TM Register List

- **CTMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
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 **CTPAU**: CTM 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 CTM 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 **CTCK2~CTCK0**: CTM Counter clock selection
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: Reserved
111: Reserved

These three bits are used to select the clock source for the CTM. 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 Operating Modes and System Clocks section.

Bit 3 **CTON**: CTM counter on/off control
0: Off
1: On

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

Bit 2~0 **CTRP2~CTRP0**: CTM CCRP 3-bit register, compared with the CTM counter bit 9~bit 7

Comparator P match period =
000: 1024 CTM clocks
001: 128 CTM clocks
010: 256 CTM clocks
011: 384 CTM clocks

100: 512 CTM clocks
101: 640 CTM clocks
110: 768 CTM clocks
111: 896 CTM clocks

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

- **CTMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
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 **CTM1~CTM0**: CTM operating mode selection
00: Compare Match Output Mode
01: Undefined
10: PWM Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the CTM. To ensure reliable operation the CTM should be switched off before any changes are made to the CTM1 and CTM0 bits. In the Timer/Counter Mode, the CTM output state is undefined.

Bit 5~4 **CTIO1~CTIO0**: CTM output function selection

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

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

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the CTIO1 and CTIO0 bits determine how the CTM output changes state when a compare match occurs from the Comparator A. The CTM output 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 CTM output should be setup using the CTOC bit in the CTMC1 register. Note that the output level requested by the CTIO1 and CTIO0 bits must be different from the initial value setup using the CTOC bit otherwise no change will occur on the CTM output when a compare match occurs. After the CTM output changes state, it can be reset to its initial level by changing the level of the CTON bit from low to high.

In the PWM Output Mode, the CTIO1 and CTIO0 bits determine how the CTM output 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 CTIO1 and CTIO0 bits only after the CTM has been switched off. Unpredictable PWM outputs will occur if the CTIO1 and CTIO0 bits are changed when the CTM is running.

Bit 3 **CTOC**: CTM CTP output control

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

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

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

Bit 2 **CTPOL**: CTM CTP output polarity control
0: Non-invert
1: Invert

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

Bit 1 **CTDPX**: CTM PWM duty/period control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

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

Bit 0 **CTCCLR**: CTM counter clear condition selection
0: CTM Comparator P match
1: CTM Comparator A match

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

- **CTMDL 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**: CTM counter low byte register bit 7 ~ bit 0
CTM 10-bit counter bit 7 ~ bit 0

• **CTMDH 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**: CTM counter high byte register bit 1~ bit 0
CTM 10-bit counter bit 9 ~ bit 8

• **CTMAL 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**: CTM CCRA low byte register bit 7 ~ bit 0
CTM 10-bit CCRA bit 7 ~ bit 0

• **CTMAH 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**: CTM CCRA high byte register bit 7 ~ bit 0
CTM 10-bit CCRA bit 9 ~ bit 8

Compact Type TM Operation Modes

The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTM1 and CTM0 bits in the CTMC1 register.

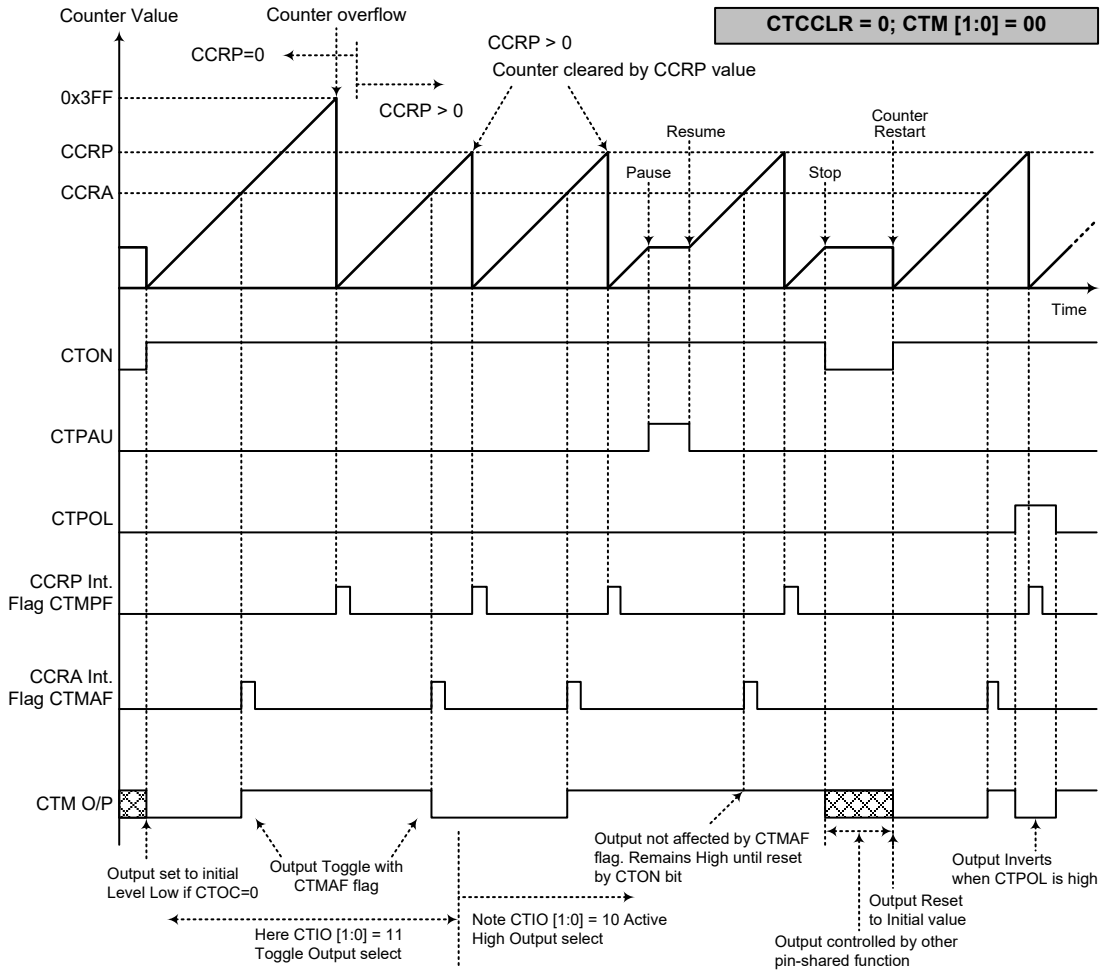
Compare Match Output Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 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 CTCCLR 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 CTMAF and CTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the CTCCLR bit in the CTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTCCLR is high no CTMPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value. However, here the CTMAF interrupt request flag will not be generated.

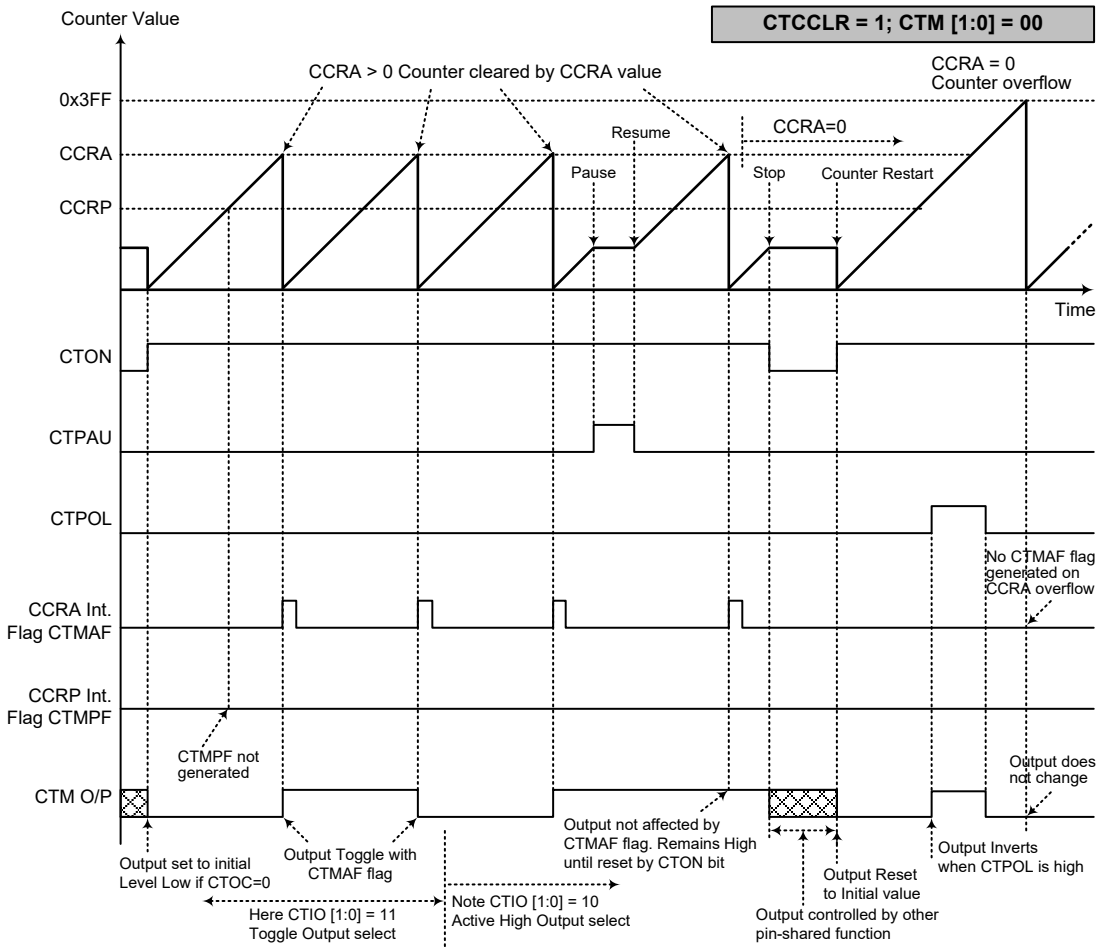
As the name of the mode suggests, after a comparison is made, the CTM output, will change state. The CTM output condition however only changes state when a CTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMPF interrupt request flag,

generated from a compare match occurs from Comparator P, will have no effect on the CTM output. The way in which the CTM output changes state are determined by the condition of the CTIO1 and CTIO0 bits in the CTMC1 register. The CTM output can be selected using the CTIO1 and CTIO0 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 CTM output, which is setup after the CTON bit changes from low to high, is setup using the CTOC bit. Note that if the CTIO1 and CTIO0 bits are zero then no output change will take place.



Compare Match Output Mode – CTCCLR=0

Note: 1. With CTCCLR=0 a Comparator P match will clear the counter
2. The CTM output is controlled only by the CTMAF flag
3. The output is reset to its initial state by a CTON bit rising edge



Compare Match Output Mode – CTCCLR=1

Note: 1. With CTCCLR=1 a Comparator A match will clear the counter
2. The CTM output is controlled only by the CTMAF flag
3. The output is reset to its initial state by a CTON bit rising edge
4. A CTMPF flag is not generated when CTCCLR=1

Timer/Counter Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 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 CTM output is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function.

PWM Output Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 register should be set to 10 respectively. The PWM function within the CTM 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 CTM output, 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 CTCCLR bit has no effect as the PWM period. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTDPX bit in the CTMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The CTOC bit in the CTMC1 register is used to select the required polarity of the PWM waveform while the two CTIO1 and CTIO0 bits are used to enable the PWM output or to force the CTM output to a fixed high or low level. The CTPOL bit is used to reverse the polarity of the PWM output waveform.

- **10-bit CTM, PWM Output Mode, Edge-aligned Mode, CTDPX=0**

CCRP	1~7	0
Period	CCRP×128	1024
Duty	CCRA	

If f_{SYS}=8MHz, CTM clock source is $f_{SYS}/4$, CCRP=4 and CCRA=128,

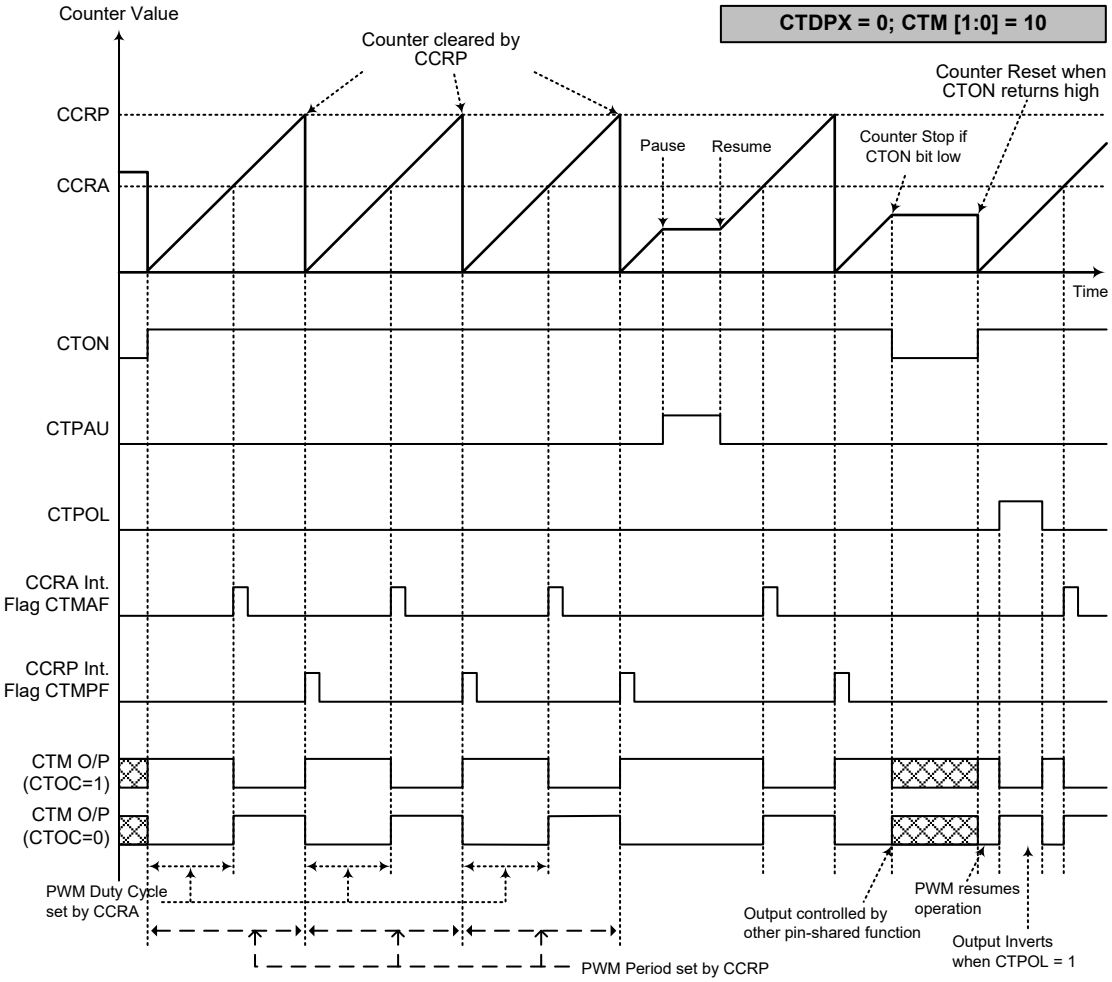
The CTM PWM output frequency=$(f_{SYS}/4)/(4\times128)=f_{SYS}/2048$=4kHz, duty=128/(4×128)=25%.

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

- **10-bit CTM, PWM Output Mode, Edge-aligned Mode, CTDPX=1**

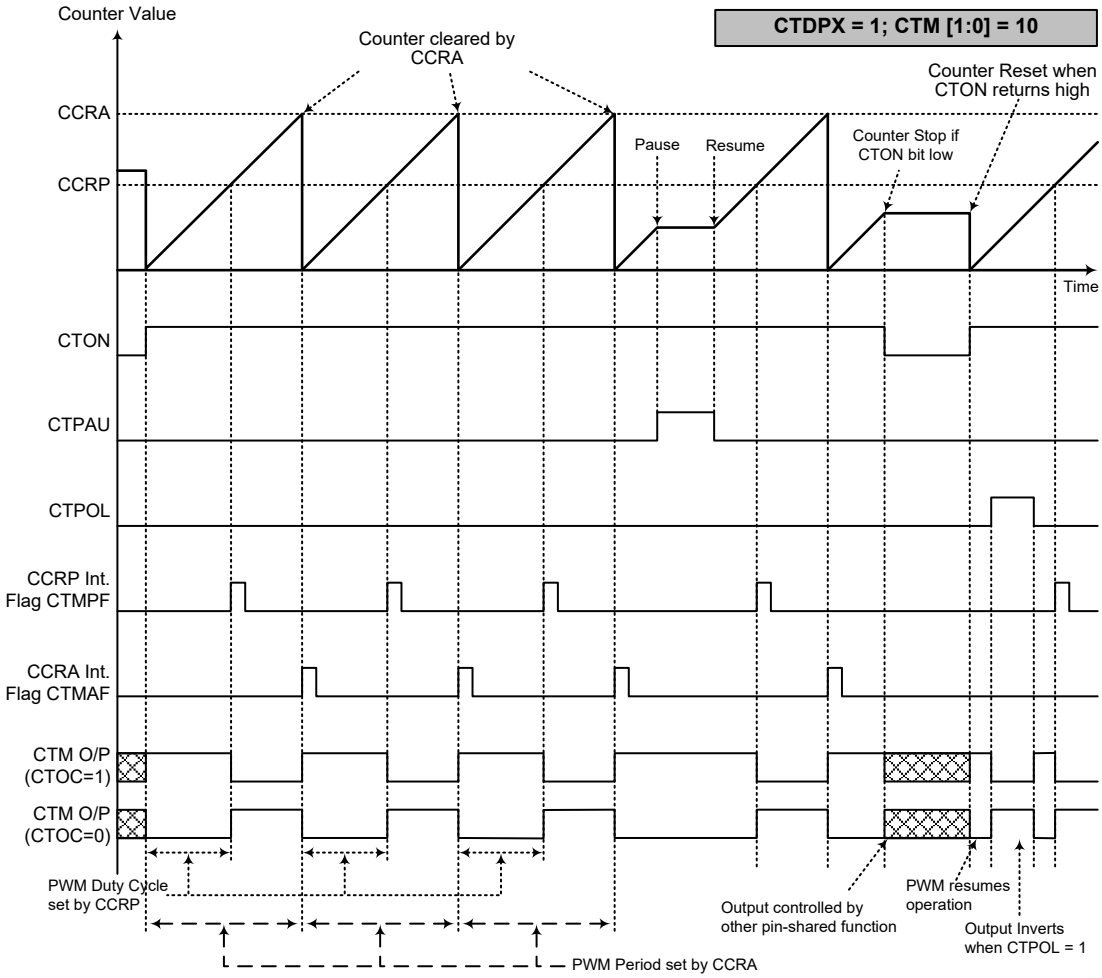
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

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



PWM Output Mode – CTDPX=0

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

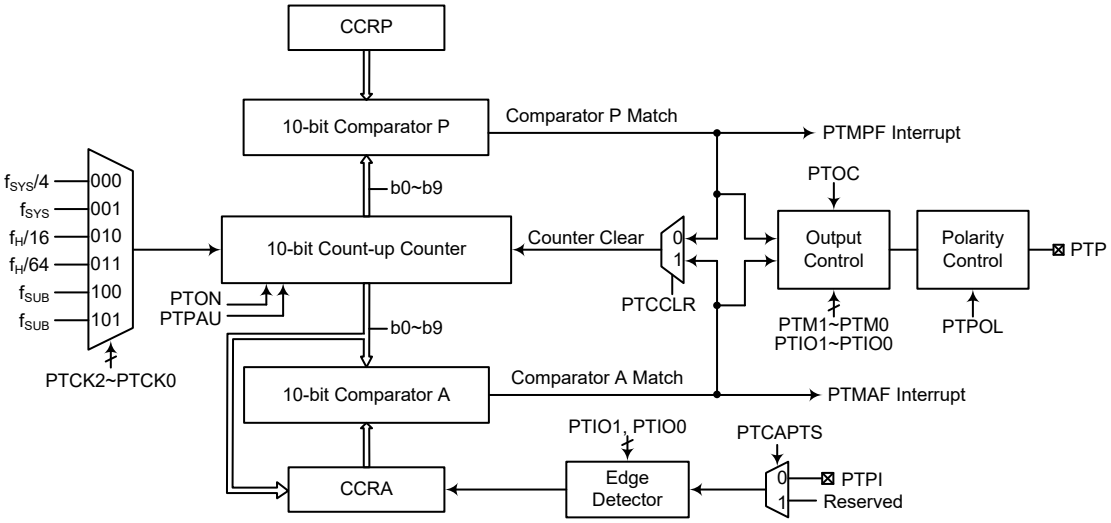


PWM Output Mode – CTDPX=1

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

Periodic Type TM – PTM

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



Note: The PTM external pins are pin-shared with other functions, therefore before using the PTM function the pin-shared function registers must be set properly to enable the PTM pin function. The PTPI pin, if used, must also be set as an input by setting the corresponding bits in the port control register.

10-bit Periodic Type TM Block Diagram

Periodic Type TM Operation

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

The only way of changing the value of the 10-bit counter using the application program is to clear the counter by changing the PTON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a PTM interrupt signal will also usually be generated. The Periodic Type TM can operate in a number of different operational modes, can be driven by different clock sources and can also control an 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 value, while two read/write register pairs exist to store the internal 10-bit CCRA and CCRP value. The remaining two registers are control registers which setup the different operating and control modes.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	—	—	—	—	—	—	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	—	—	—	—	—	—	D9	D8
PTMRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMRPH	—	—	—	—	—	—	D9	D8

10-bit Periodic Type TM Register List

- **PTMC0 Register**

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

Bit 7 **PTPAU**: PTM counter pause control
0: Run
1: Pause

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

Bit 6~4 **PTCK2~PTCK0**: PTM counter clock selection
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: Reserved
111: Reserved

These three bits are used to select the clock source for the PTM. 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 “Operating Modes and System Clocks” section.

Bit 3 **PTON**: PTM counter on/off control
0: Off
1: On

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

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

Bit 2~0 Unimplemented, read as “0”

• **PTMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PTM1~PTM0**: PTM operating mode selection
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the PTM. To ensure reliable operation the PTM should be switched off before any changes are made to the PTM1 and PTM0 bits. In the Timer/Counter Mode, the PTM output pin state is undefined.

Bit 5~4 **PTIO1~PTIO0**: PTM external pin function selection

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

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

Capture Input Mode
00: Input capture at rising edge of PTPI
01: Input capture at falling edge of PTPI
10: Input capture at rising/falling edge of PTPI
11: Input capture disabled

Timer/Counter Mode
Unused

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

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

In the PWM Output Mode, the PTIO1 and PTIO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PTM output function is modified by changing these two bits. It is necessary to only change the values of the PTIO1 and PTIO0 bits only after the PTM has been switched off. Unpredictable PWM outputs will occur if the PTIO1 and PTIO0 bits are changed when the PTM is running.

Bit 3 **PTOC**: PTM PTP output control

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

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

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

Bit 2 **PTPOL**: PTM PTP output polarity control
0: Non-inverted
1: Inverted

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

Bit 1 **PTCAPTS**: PTM capture trigger source selection
0: From PTPI pin
1: Reserved

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

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

- **PTMDL Register**

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

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

- **PTMDH 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**: PTM Counter High Byte Register bit 1 ~ bit 0
PTM 10-bit Counter bit 9 ~ bit 8

• **PTMAL Register**

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

Bit 7~0 **D7~D0**: PTM CCRA Low Byte Register bit 7 ~ bit 0
PTM 10-bit CCRA bit 7 ~ bit 0

• **PTMAH 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**: PTM CCRA High Byte Register bit 1 ~ bit 0
PTM 10-bit CCRA bit 9 ~ bit 8

• **PTMRPL Register**

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

Bit 7~0 **D7~D0**: PTM CCRP Low Byte Register bit 7 ~ bit 0
PTM 10-bit CCRP bit 7 ~ bit 0

• **PTMRPH 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**: PTM CCRP High Byte Register bit 1 ~ bit 0
PTM 10-bit CCRP bit 9 ~ bit 8

Periodic Type TM Operation Modes

The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTM1 and PTM0 bits in the PTMC1 register.

Compare Match Output Mode

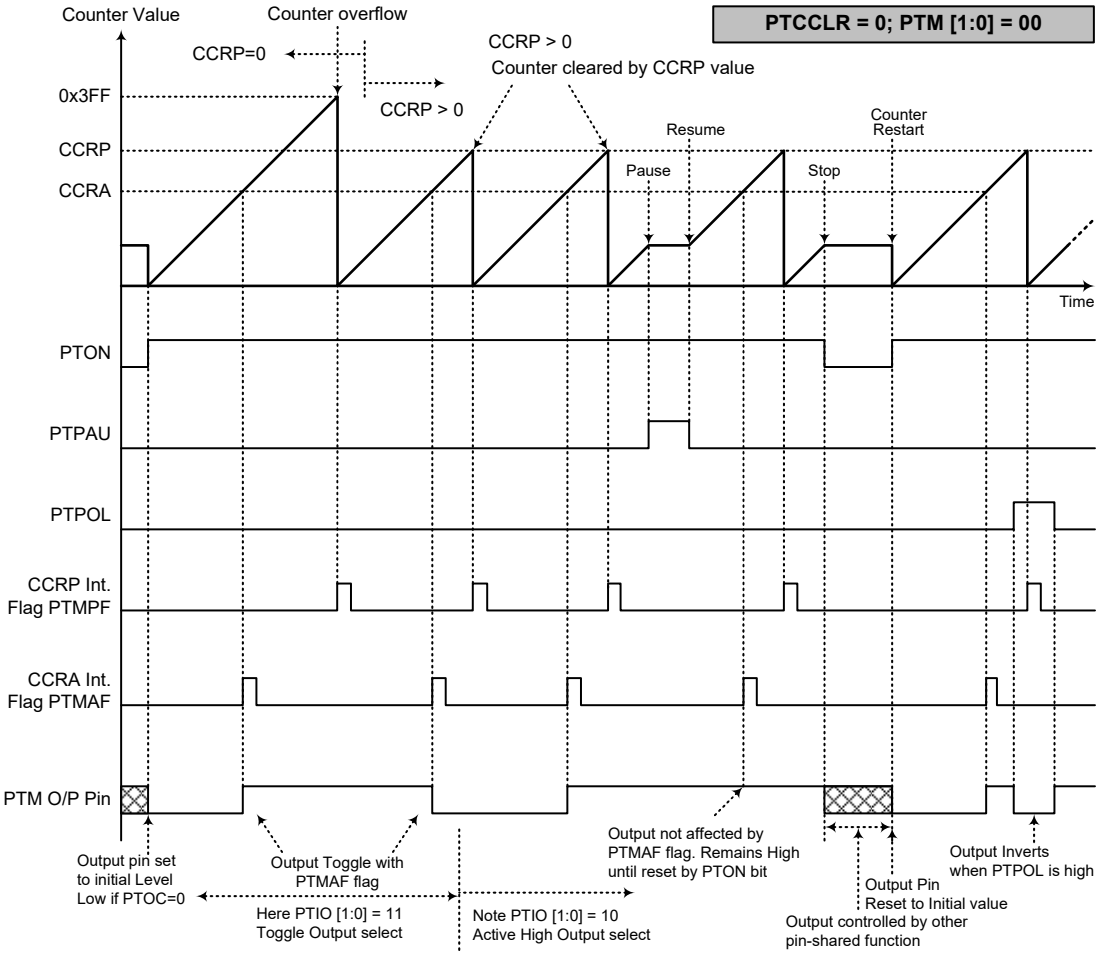
To select this mode, bits PTM1 and PTM0 in the PTMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the PTCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both PTMAF and PTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the PTCCLR bit in the PTMC1 register is high then the counter will be cleared when a compare

match occurs from Comparator A. However, here only the PTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTCCLR is high no PTMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be cleared to “0”.

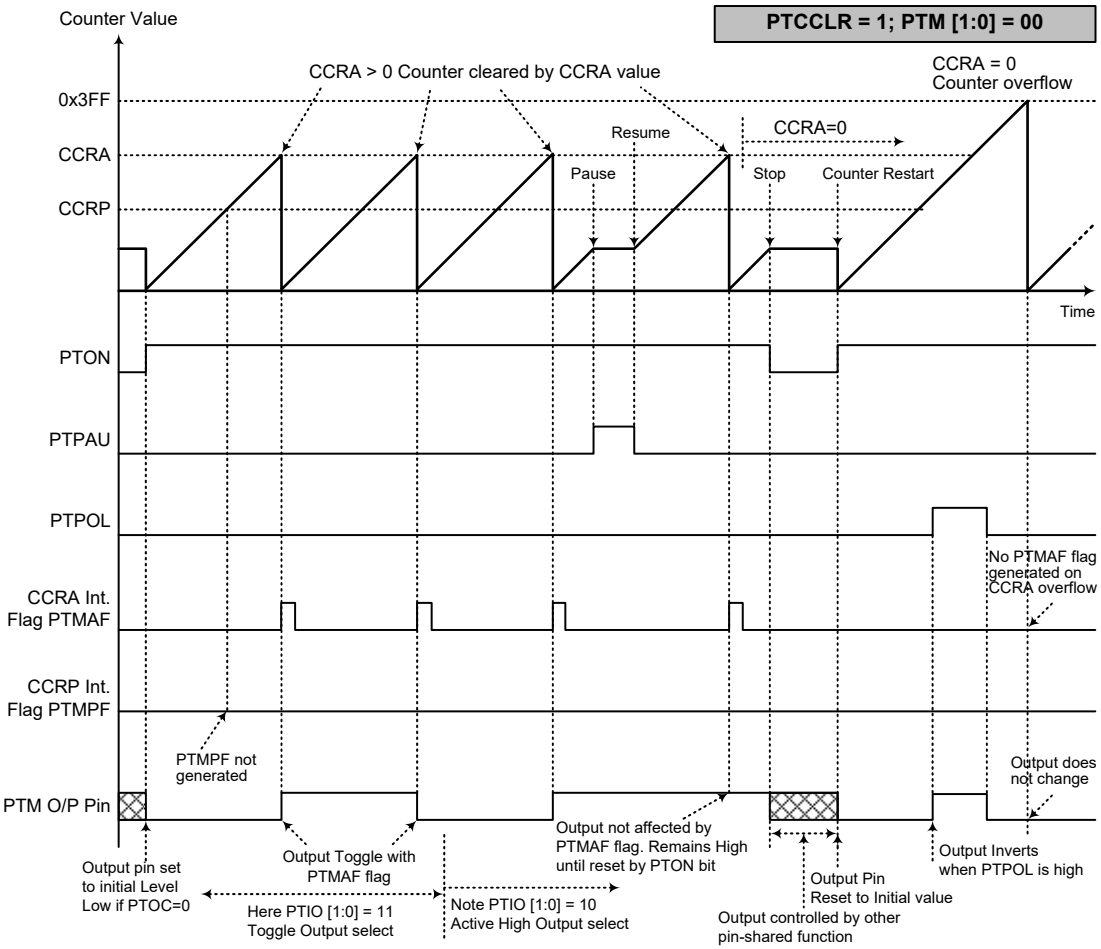
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the PTMAF interrupt request flag will not be generated.

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



Compare Match Output Mode – PTCCLR=0

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



Compare Match Output Mode – PTCCLR=1

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

Timer/Counter Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the PTM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the PTM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively. The PWM function within the PTM is useful for applications which require functions such as motor control, heating control, illumination control, etc. By providing a signal of fixed frequency but of varying duty cycle on the PTM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

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

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The PTOC bit in the PTMC1 register is used to select the required polarity of the PWM waveform while the two PTIO1 and PTIO0 bits are used to enable the PWM output or to force the PTM output pin to a fixed high or low level. The PTPOL bit is used to reverse the polarity of the PWM output waveform.

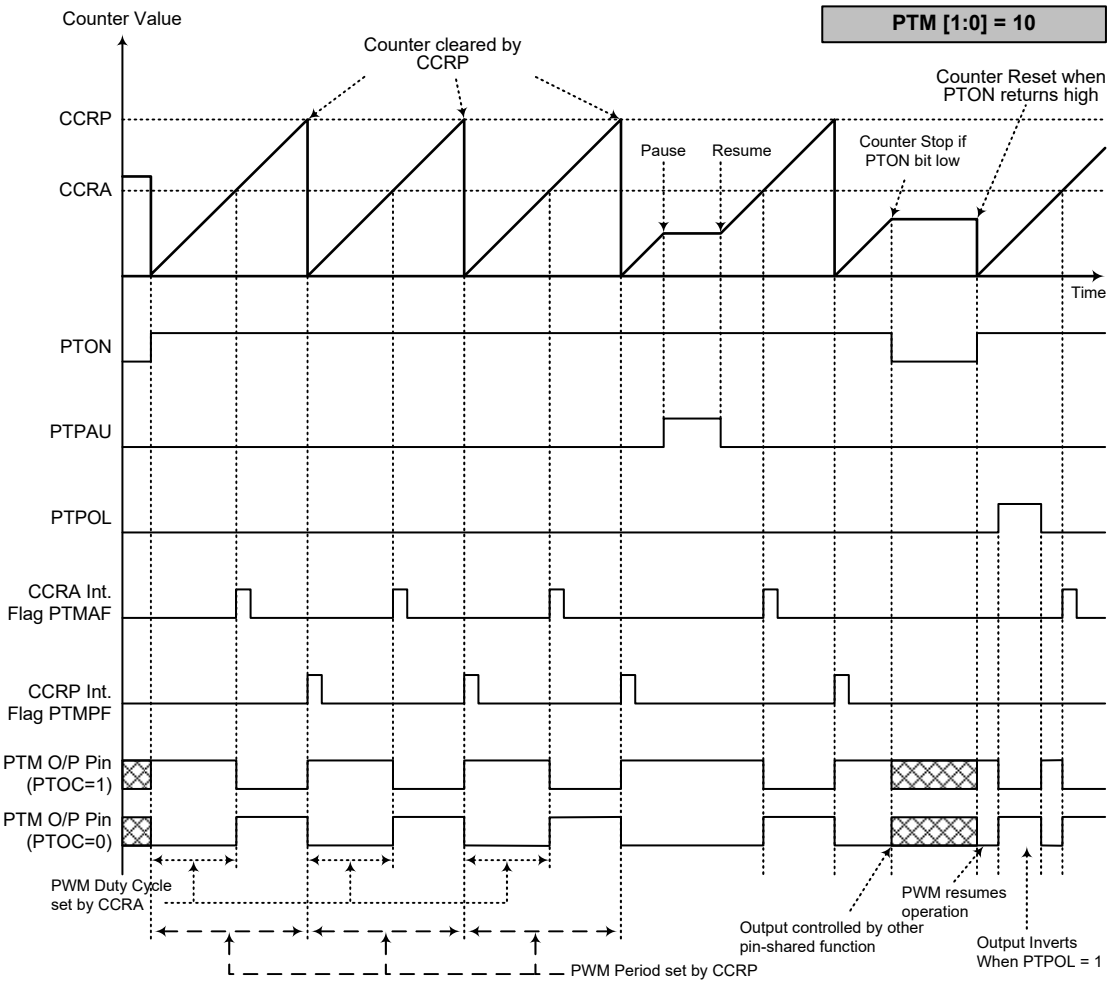
- **10-bit PTM, PWM Output Mode, Edge-aligned Mode**

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

If f_{SYS}=16MHz, PTM clock source select $f_{SYS}/4$, CCRP=512 and CCRA=128,

The PTM PWM output frequency=($f_{SYS}/4$)/512=$f_{SYS}/2048$=8kHz, duty=128/512=25%,

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



PWM Output Mode

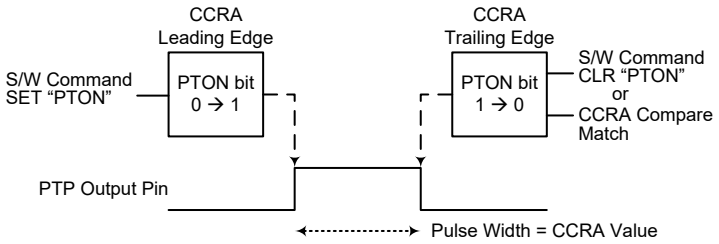
Note: 1. The counter is cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTIO[1:0]=00 or 01
4. The PTCCLR bit has no influence on PWM operation

Single Pulse Output Mode

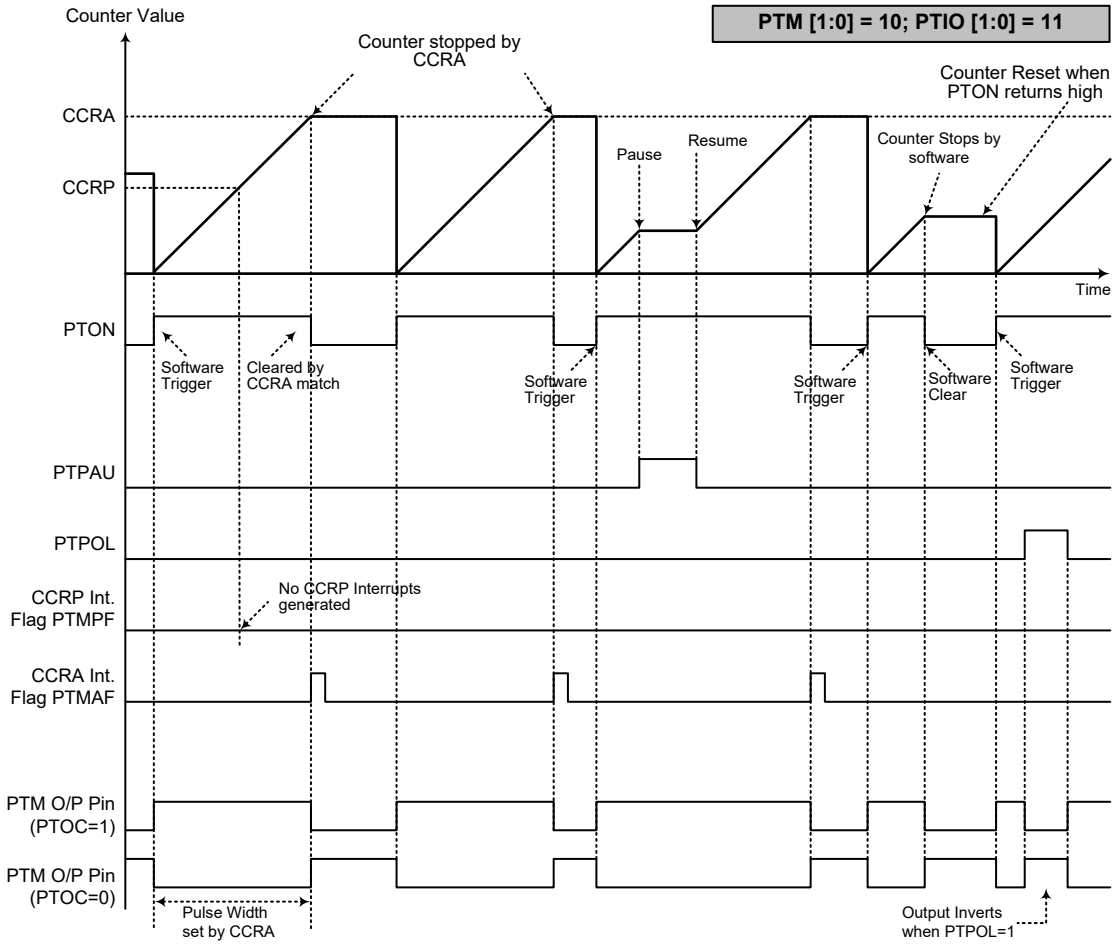
To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

The trigger for the pulse output leading edge is a low to high transition of the PTON bit, which can be implemented using the application program. When the PTON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The PTON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the PTON bit is cleared to zero, which can be implemented using the application program or when a compare match occurs from Comparator A.

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



Single Pulse Generation



Single Pulse Output Mode

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

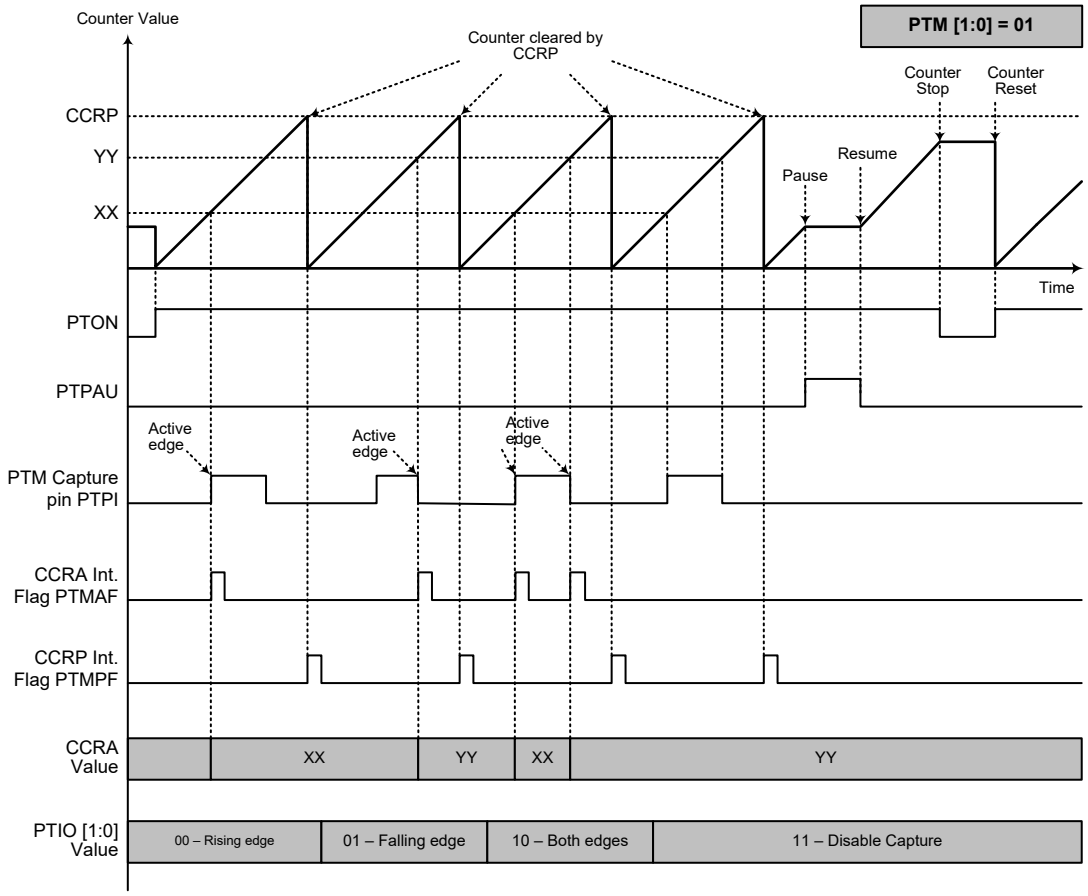
Capture Input Mode

To select this mode bits PTM1 and PTM0 in the PTMC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPI pin, selected by the PTCAPTS bit in the PTMC1 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 PTIO1 and PTIO0 bits in the PTMC1 register. The counter is started when the PTON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the PTPI pin the present value in the counter will be latched into the CCRA registers and a PTM interrupt generated. Irrespective of what events occur on the PTPI pin the counter will continue to free run until the PTON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTIO1 and PTIO0 bits can select the active trigger edge on the PTPI pin to be a rising edge, falling edge or both edge types. If the PTIO1 and PTIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPI pin, however it must be noted that the counter will continue to run.

There are some considerations that should be noted. If the captured pulse width is less than 2 timer clock periods, it may be ignored by hardware. After the counter value is latched to the CCRA registers by an active capture edge, the PTMAF flag will be set high after 0.5 timer clock periods. The delay time from the active capture edge received to the action of latching counter value to CCRA registers is less than 1.5 timer clock periods.

The PTCCLR, PTOC and PTPOL bits are not used in this Mode.



Capture Input Mode

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

Analog to Digital Converter

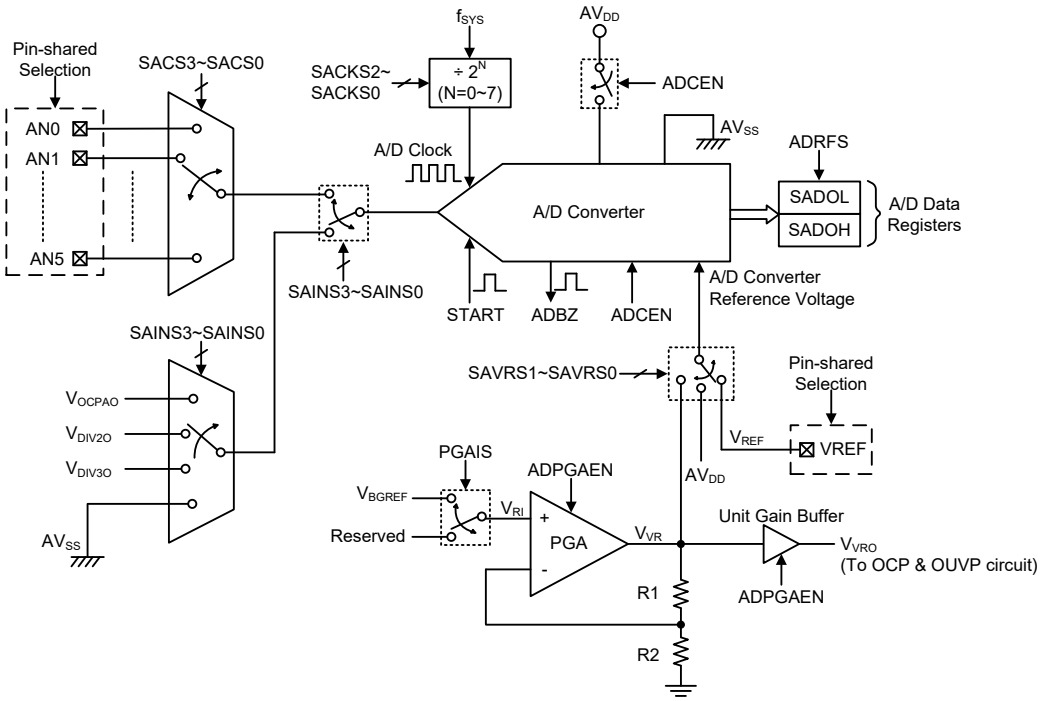
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the OCP operational amplifier output signal, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS3~SAINS0 and SACS3~SACS0 bits. Note that when the internal analog signal is selected to be converted using the SAINS3~SAINS0 bits, the external channel analog input will automatically be switched off. More detailed information about the A/D input signal selection will be described in the "A/D Converter Input Signals" section.

External Input Channels	Internal Signal	A/D Signal Select
AN0~AN5	V_{OCPAO}, V_{DIV2O}, V_{DIV3O}, AV_{SS}	SAINS3~SAINS0 SACS3~SACS0

The accompanying block diagram shows the internal structure of the A/D converter together with its associated registers and control bits.



Note: 1. When the V_{DIV3O} is selected to be converted, if the PA2 pin is setup as the OUVPI pin function, then the OUVPI will be connected together with the internal DIV3 Divider Resistor circuit output signal V_{DIV3O}, at which piont special consideration should be given to the interaction between internal and external circuits and the corresponding control bits should be properly configured according to the application requirements.

2. The unit gain buffer output V_{VRO} can be used as the D/A converter reference input of the OCP and OUVP functions.

A/D Converter Structure

A/D Register Description

Overall operation of the A/D converter is controlled using six registers. A read only register pair exists to store the A/D Converter data 12-bit value. Three registers, SADC0, SADC1 and SADC2, are the control registers which setup the operating conditions and control function of the A/D converter. The VBGRC register contains the VBGREN bit to control the bandgap reference voltage.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
SADC2	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	—	—
VBGRC	—	—	—	—	—	—	—	VBGREN

A/D Converter Register List

A/D Converter Data Registers – SADOL, SADOH

As the device contains an internal 12-bit A/D converter, it requires two data registers to store the converted value. These are a high byte register, known as SADOH, and a low byte register, known as SADOL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 12 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRFS bit in the SADC0 register as shown in the accompanying table. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero. The A/D data registers contents will be unchanged if the A/D converter is disabled.

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

A/D Converter Data Registers

A/D Converter Control Registers – SADC0, SADC1, SADC2

To control the function and operation of the A/D converter, three control registers known as SADC0, SADC1 and SADC2 are provided. These 8-bit registers define functions such as the selection of which analog signal 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 busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SAINS3~SAINS0 bits in the SADC1 register and SACS3~SACS0 bits in the SADC0 register are used to determine which analog signal derived from the external or internal signals will be connected to the A/D converter. The A/D converter also contains a programmable gain amplifier, PGA, to generate the A/D converter internal reference voltage. The overall operation of the PGA is controlled using the SADC2 register.

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. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

- **SADC0 Register**

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

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

Bit 6 **ADBZ**: A/D Converter busy flag
0: No A/D conversion is in progress
1: A/D conversion is in progress
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.

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

Bit 4 **ADRFS**: A/D conversion data format selection
0: A/D converter data format → SADOH=D [11:4]; SADOL=D [3:0]
1: A/D converter data format → SADOH=D [11:8]; SADOL=D [7:0]
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D converter data register section.

Bit 3~0 **SACS3~SACS0**: A/D converter external analog input channel selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110~1111: Non-existed channel, input floating if selected

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
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 7~4 **SAINS3~SAINS0**: A/D converter input signal selection
0000: External source – External analog channel input, ANn
0001: Internal source – Internal OCP operational amplifier output, V_{OCPAO}
0010: Internal source – Internal DIV2 divider resistor circuit output, V_{DIV2O}
0011: Internal source – Internal DIV3 divider resistor circuit output, V_{DIV3O}
0100: Internal source – Connected to the ground, AV_{SS}
0101~1111: External source – External analog channel input, ANn

When the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off regardless of the SACS3~SACS0 bit values. It will prevent the external channel input from being connected together with the internal analog signal.

Bit 3 Unimplemented, read as "0"

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

• **SADC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	—	—
R/W	R/W	—	—	R/W	R/W	R/W	—	—
POR	0	—	—	0	0	0	—	—

Bit 7 **ADPGAEN**: A/D converter PGA enable/disable control
0: Disable
1: Enable

This bit is used to control the A/D converter internal PGA function. When the PGA output voltage is selected as A/D input or A/D reference voltage, the PGA needs to be enabled by setting this bit high. Otherwise the PGA needs to be disabled by clearing the ADPGAEN bit to zero to conserve power.

Bit 6~5 Unimplemented, read as "0"

Bit 4 **PGAIS**: PGA input voltage selection
0: Reserved
1: From internal reference voltage V_{BGREF}

When the internal independent reference voltage V_{BGREF} is selected as the PGA input, the internal bandgap reference V_{BGREF} should be enabled by setting the VBGREN bit in the VBGRC register to "1".

Bit 3~2 **SAVRS1~SAVRS0**: A/D converter reference voltage selection
00: Internal A/D converter power, AV_{DD}
01: External VREF pin
10~11: Internal PGA output voltage, V_{VR}

These bits are used to select the A/D converter reference voltage source. When the internal reference voltage source is selected, the reference voltage derived from the external VREF pin will automatically be switched off.

Bit 1~0 Unimplemented, read as "0"

Bandgap Referenc Voltage Control Register – VBGRC

A high performance bandgap voltage reference is included in the device. It has an accurate voltage reference output, V_{BGREF}, when input supply voltage change or temperature variation. The VBGRC register is used to control the bandgap reference voltage circuit enable or disable.

- **VBGRC Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **VBGREN**: Bandgap reference voltage control
0: Disable
1: Enable

This bit is used to enable the internal Bandgap reference circuit. The internal Bandgap reference circuit should first be enabled before the V_{BGREF} voltage is selected to be used. A specific start-up time is necessary for the Bandgap circuit to become stable and accurate. When this bit is cleared to 0, the Bandgap voltage output V_{BGREF} is in a low state.

A/D Converter Reference Voltage

The actual reference voltage supply to the A/D Converter can be supplied from the internal A/D converter power, AV_{DD}, an external reference source supplied on pin VREF or an internal reference source derived from the PGA output V_{VR}. The desired selection is made using the SAVRS1~SAVRS0 bits in the SADC2 register. The internal reference voltage V_{VR} is an amplified signal through a programmable gain amplifier, PGA, which is controlled by the ADPGAEN bit in the SADC2 register. The PGA input is sourced from an internal Bandgap reference voltage, V_{BGREF}, set by the PGAIS bit in the SADC2 register. The internal Bandgap reference circuit should first be enabled before the V_{BGREF} is selected to be used.

As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage pin, the VREF pin-shared function selection bits should first be properly configured to disable other pin-shared functions. However, if the internal reference signal is selected as the reference source, the external reference input from the VREF pin will automatically be switched off by hardware.

Note that the analog input values must not be allowed to exceed the value of the selected reference voltage.

SAVRS[1:0]	Reference Source	Description
00	AV_{DD}	Internal A/D converter power supply
01	VREF Pin	External A/D converter reference pin
10~11	V_{VR}	Internal A/D converter PGA output

A/D Converter Reference Voltage Selection

A/D Converter Input Signals

All of the external A/D analog input pins are pin-shared with the I/O pins as well as other functions. The corresponding pin-shared function selection bits in the PxS1 and PxS0 registers, determine whether the external input pins are setup as A/D converter analog channel inputs or whether they have other functions. If the corresponding pin is setup to be an A/D converter analog channel input, the original pin function will be disabled. In this way, pins can be changed under program control

to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the relevant A/D input function selection bits enable an A/D input, the status of the port control register will be overridden.

As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SAINS3~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the external channel input or internal analog signal. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. If the SAINS3~SAINS0 bits are set to "0000" or "0101~1111", the external channel input will be selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected.

When the SAINS3~SAINS0 bits are set to the value of "0001~0100", the internal analog signal will be selected. If the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off regardless of the SACS3~SACS0 bit values. It will prevent the external channel input from being connected together with the internal analog signal.

SAINS[3:0]	SACS[3:0]	Input Signals	Description
0000, 0101~1111	0000~0101	AN0~AN5	External channel analog input ANn
	0110~1111	—	Floating, no channel is selected
0001	xxxx	V_{OCPAO}	Internal OCP operational amplifier output
0010	xxxx	V_{DIV2O}	Internal DIV2 divider resistor circuit output
0011	xxxx	V_{DIV3O}	Internal DIV3 divider resistor circuit output
0100	xxxx	AV_{SS}	Connected to the ground

"x": don't care

A/D Converter Input Signal Selection

A/D Conversion Operation

The START bit in the SADC0 register is used to start the AD conversion. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated.

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in progress or not. This bit will be automatically set to 1 by the microcontroller after an A/D conversion is successfully initiated. When the A/D conversion is complete, the ADBZ bit will be cleared to 0. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an 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 ADBZ bit in the SADC0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the SACKS2~SACKS0 bits in the SADC1 register. Although the A/D clock source is determined by the system clock f_{SYS} and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 8MHz, the SACKS2~SACKS0 bits should not be set to 000, 001 or 111. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion

values. Refer to the following table for examples, where values marked with an asterisk * special care must be taken.

f_{SYS}	A/D Clock Period (t_{ADCK})							
	SACKS[2:0] =000 (f_{SYS})	SACKS[2:0] =001 (f_{SYS}/2)	SACKS[2:0] =010 (f_{SYS}/4)	SACKS[2:0] =011 (f_{SYS}/8)	SACKS[2:0] =100 (f_{SYS}/16)	SACKS[2:0] =101 (f_{SYS}/32)	SACKS[2:0] =110 (f_{SYS}/64)	SACKS[2:0] =111 (f_{SYS}/128)
1MHz	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *	128μs *
2MHz	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *
4MHz	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *
8MHz	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *
12MHz	83ns *	167ns *	333ns *	667ns	1.33μs	2.67μs	5.33μs	10.67μs *
16MHz	62.5ns *	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs

A/D Clock Period Examples

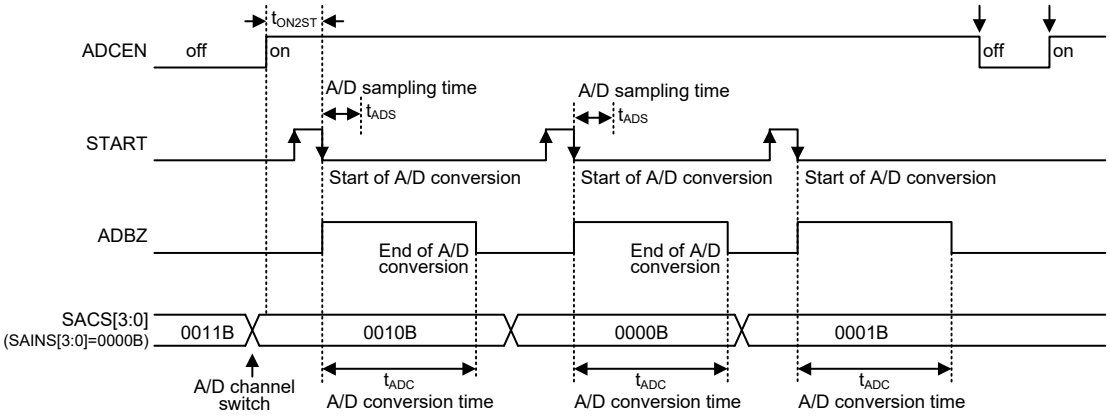
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry, a certain delay as indicated in the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

Conversion Rate and Timing Diagram

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

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

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



A/D Conversion Timing – External Channel Input

Summary of A/D Conversion Steps

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

- Step 1

  Select the required A/D conversion clock by properly programming the SACKS2~SACKS0 bits in the SADC1 register.

- Step 2

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

- Step 3

  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS3~SAINS0 and SACS3~SACS0 bits.

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

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

- Step 4

  If the SAINS3~SAINS0 bits are set to “0000” or “0101~1111”, the external channel input can be selected. The desired external channel input is selected by configuring the SACS3~SACS0 bits. When the A/D input signal comes from the external channel input, the corresponding pin should be configured as an A/D input function by selecting the relevant pin-shared function control bits. Then go to Step 6.

- Step 5

  If the SAINS3~SAINS0 bits are set to “0001~0100”, the relevant internal analog signal will be selected. When the internal analog signal is selected to be converted, the external channel analog input will automatically be disconnected. Then go to Step 6.

- Step 6

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

- Step 7

  Select the A/D converter reference voltage source by configuring the SAVRS1~SAVRS0 bits.

- Step 8

  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, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.

- Step 9

  The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.

- Step 10

  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

Note: When checking for the end of the conversion process, if the method of polling the ADBZ bit in the SADC0 register is used, the interrupt enable step above can be omitted.

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by setting bit ADCEN low in the

SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/O pins, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Transfer Function

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

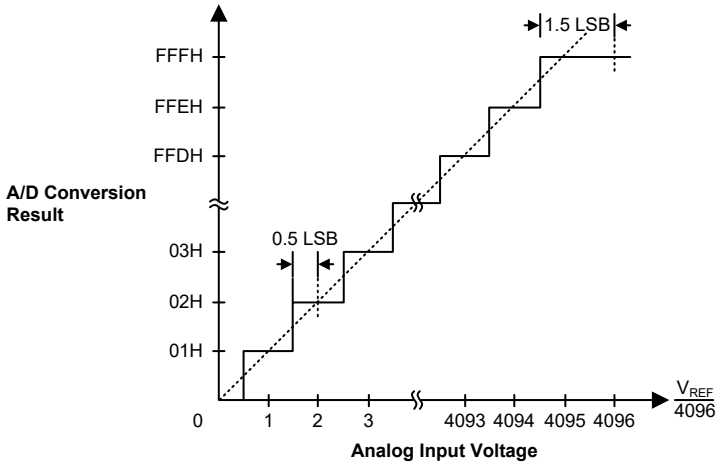
$$1\ LSB = V_{REF}/4096$$

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

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

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

Note that here the V_{REF} voltage is the actual A/D converter reference voltage determined by the SAVRS1~SAVRS0 bits.



Ideal A/D Transfer Function

A/D Programming Examples

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the ADBZ bit in the SADC0 register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

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

```
clr ADE             ; disable ADC interrupt
mov a, 03H          ; select fSYS/8 as A/D clock and A/D input
mov SADC1, a        ; signal comes from external channel
mov a, 00H          ; select AVDD as the A/D reference voltage source
mov SADC2, a
mov a, 03H          ; setup PAS0 to configure pin AN0
mov PAS0, a
mov a, 20H          ; enable A/D converter and select AN0 as
```

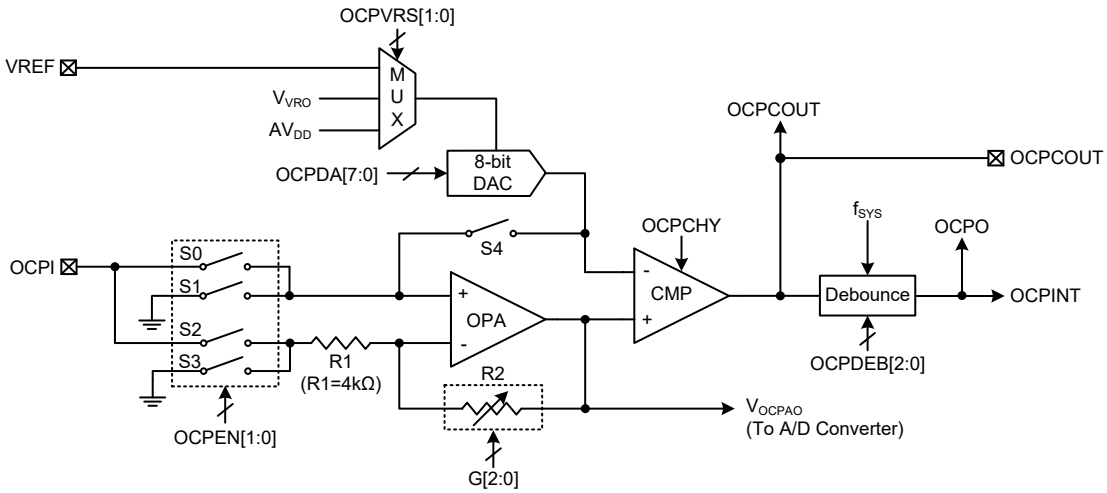
```
mov SADC0, a          ; the A/D external channel input
:
start_conversion:
clr START             ; high pulse on start bit to initiate conversion
set START             ; reset A/D
clr START             ; start A/D
:
polling_EOC:
sz  ADBZ              ; poll the SADC0 register ADBZ bit to detect end of A/D
                      ; conversion
jmp polling_EOC       ; continue polling
:
mov a, SADOL          ; read low byte conversion result value
mov SADOL_buffer, a   ; save result to user defined register
mov a, SADOH          ; read high byte conversion result value
mov SADOH_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 ADE               ; disable ADC interrupt
mov a, 03H            ; select fSYS/8 as A/D clock and A/D input
mov SADC1, a          ; signal comes from external channel
mov a, 00H            ; select AVDD as the A/D reference voltage source
mov SADC2, a
mov a, 03H            ; setup PAS0 to configure pin AN0
mov PAS0, a
mov a, 20H            ; enable A/D converter and select AN0 as
mov SADC0, a          ; the A/D external channel input
:
start_conversion:
clr START             ; high pulse on START bit to initiate conversion
set START             ; reset A/D
clr START             ; start A/D
clr ADF               ; clear ADC interrupt request flag
set ADE               ; enable ADC interrupt
set EMI               ; enable global interrupt
:
:
ADC_ISR:              ; ADC interrupt service routine
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, SADOL          ; read low byte conversion result value
mov SADOL_buffer, a   ; save result to user defined register
mov a, SADOH          ; read high byte conversion result value
mov SADOH_buffer, a   ; save result to user defined register
:
EXIT_INT_ISR:
mov a, status_stack
mov STATUS, a         ; restore STATUS from user defined memory
mov a, acc_stack      ; restore ACC from user defined memory
reti
```

Over Current Protection – OCP

The device includes an over current protection function which provides a current protection mechanism for applications. The current on the OCPI pin is converted to a relevant voltage level according to the current value using the OCP operational amplifier. It is then compared with a reference voltage generated by an 8-bit D/A converter. When an over current event occurs, an OCP interrupt will be generated if the corresponding interrupt control bit is enabled if the corresponding interrupt control bit is enabled.



Note: 1. As the OCP function relevant external pins are pin-shared with general I/O or other functions, ensure that the corresponding pin-shared function registers have been configured properly before using the OCP function.

2. The V_{VRO} is sourced from the A/D converter unit gain buffer output and the V_{OCPAO} can be selected as the A/D converter input signal.

Over Current Protection Circuit

Over Current Protection Operation

The OCP circuit is used to prevent the input current from exceeding a specific level. The current on the OCPI input is converted to a voltage and then amplified by the OCP Programmable Gain Amplifier (PGA) with a programmable gain from 5 to 50 selected by the G2~G0 bits in the OCPC1 register. The PGA consists of an operational amplifier and two resistors; the PGA gain can be positive or negative determined by input voltage connecting to positive input or negative input of the PGA. This PGA can also be configured to operate in the non-inverting, inverting or input offset calibration mode determined by the OCPEN1~OCPEN0 bits in the OCPC0 register. After the current is converted and amplified to a specific voltage level, it will be compared with a reference voltage provided by an 8-bit D/A converter. The 8-bit D/A converter reference voltage can be supplied by AV_{DD}, V_{VRO} or the external VREF pin, which is selected by the OCPVRS1~OCPVRS0 bits in the OCPC0 register. The comparator output, OCPCOUT, will first be filtered with a certain de bounce time period selected by the OCPDEB2~OCPDEB0 bits in the OCPC1 register. Then a filtered OCP comparator digital comparator output, OCPO, is obtained to indicate whether an over current condition occurs or not. The OCPO bit will be set to 1 if an over current condition occurs. Otherwise, the OCPO bit is zero. Once an over current event occurs, i.e., the converted voltage of the OCPI input current is greater than the reference voltage, the corresponding interrupt will be generated if the relevant interrupt control bit is enabled.

Note that the debounce clock, f_{DEB}, comes from the system clock, f_{SYS}. The operational amplifier output voltage can be read out by the A/D converter through an A/D internal input channel. The D/A converter output voltage is controlled by the OCPDA register.

Over Current Protection Registers

Overall operation of the over current protection is controlled using several registers. The OCPDA register is used to provide the reference voltage for the over current protection circuit. The OCPOCAL and OCPCCAL registers are used to cancel out the operational amplifier and comparator input offset. The OCPC0 and OCPC1 registers are control registers which control the OCP function, D/A converter reference voltage selection, PGA gain selection, comparator debounce time together with the hysteresis function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OCPC0	OCPEN1	OCPEN0	OCPVRS1	OCPVRS0	OCPCHY	—	—	OCPO
OCPC1	—	—	G2	G1	G0	OCPDEB2	OCPDEB1	OCPDEB0
OCPDA	D7	D6	D5	D4	D3	D2	D1	D0
OCPOCAL	OCPOOFM	OCPORSP	OCPOOF5	OCPOOF4	OCPOOF3	OCPOOF2	OCPOOF1	OCPOOF0
OCPCCAL	OCPCOUT	OCPCOFM	OCPCRSP	OCPCOF4	OCPCOF3	OCPCOF2	OCPCOF1	OCPCOF0

Over Current Protection Register List

• OCPC0 Register

Bit	7	6	5	4	3	2	1	0
Name	OCPEN1	OCPEN0	OCPVRS1	OCPVRS0	OCPCHY	—	—	OCPO
R/W	R/W	R/W	R/W	R/W	R/W	—	—	R
POR	0	0	0	0	0	—	—	0

Bit 7~6 **OCPEN1~OCPEN0**: OCP function operating mode selection
00: OCP function is disabled, S1 and S3 on, S0 and S2 off
01: Non-inverting mode, S0 and S3 on, S1 and S2 off
10: Inverting mode, S1 and S2 on, S0 and S3 off
11: Calibration mode, S1 and S3 on, S0 and S2 off

Bit 5~4 **OCPVRS1~OCPVRS0**: OCP D/A converter reference voltage selection
00: From AV_{DD}
01: From VREF pin
10: From V_{VRO}
11: From AV_{DD}

Bit 3 **OCPCHY**: OCP comparator hysteresis function control
0: Disable
1: Enable

Bit 2~1 Unimplemented, read as “0”

Bit 0 **OCPO**: OCP comparator digital output bit (after debounce)
0: No over current situation occurred
1: Over current situation occurred

• **OCPC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	G2	G1	G0	OCPDEB2	OCPDEB1	OCPDEB0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **G2~G0**: OCP PGA R2/R1 ratio selection
000: Reserved
001: R2/R1=5
010: R2/R1=10
011: R2/R1=15
100: R2/R1=20
101: R2/R1=30
110: R2/R1=40
111: R2/R1=50

These bits are used to select the R2/R1 ratio to obtain various gain values for inverting and non-inverting mode. The calculating formula of the PGA gain for the inverting and non-inverting mode is described in the "Input Voltage Range" section.

Bit 2~0 **OCPDEB2~OCPDEB0**: OCP output filter debounce time selection
000: Bypass, without debounce
001: $(1\sim2)\times t_{DEB}$
010: $(3\sim4)\times t_{DEB}$
011: $(7\sim8)\times t_{DEB}$
100: $(15\sim16)\times t_{DEB}$
101: $(31\sim32)\times t_{DEB}$
110: $(63\sim64)\times t_{DEB}$
111: $(127\sim128)\times t_{DEB}$

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

• **OCPDA 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**: OCP D/A converter output voltage control bits
OCP D/A converter output=(D/A converter reference voltage/256)×D[7:0]

• **OCPOCAL Register**

Bit	7	6	5	4	3	2	1	0
Name	OCPOOFM	OCPORSP	OCPOOF5	OCPOOF4	OCPOOF3	OCPOOF2	OCPOOF1	OCPOOF0
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	0

Bit 7 **OCPOOFM**: OCP operational amplifier input offset calibration mode enable control
0: Input offset calibration mode disable
1: Input offset calibration mode enable

This bit is used to control the OCP operational amplifier input offset Calibration function. The OCPEN1 and OCPEN0 bits must first be set to "11" and then the OCPOOFM bit must be set to 1 followed by the OCPCOFM bit being setting to 0, then the operational amplifier input offset Calibration mode will be enabled. Refer to the "Operational Amplifier Input Offset Calibration" section for the detailed offset Calibration procedures.

Bit 6 **OCPORSP**: OCP operational amplifier input offset calibration reference selection
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 5~0 **OCPOOF5~OCPOOF0**: OCP operational amplifier input offset calibration control
This 6-bit field is used to perform the operational amplifier input offset Calibration operation and the value for the OCP operational amplifier input offset Calibration can be restored into this bit field. More detailed information is described in the "Operational Amplifier Input Offset Calibration" section.

- **OCPCCAL Register**

Bit	7	6	5	4	3	2	1	0
Name	OCPCOUT	OCPCOFM	OCPCRSP	OCPCOF4	OCPCOF3	OCPCOF2	OCPCOF1	OCPCOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **OCPCOUT**: OCP comparator output, positive logic (before debounce)
0: Positive input voltage < Negative input voltage
1: Positive input voltage > Negative input voltage

Bit 6 **OCPCOFM**: OCP comparator input offset calibration mode enable control
0: Input offset calibration mode disable
1: Input offset calibration mode enable
This bit is used to control the OCP comparator input offset Calibration function. The OCPEN1 and OCPEN0 bits must first be set to "11" and then the OCPCOFM bit must be set to 1 followed by the OCPOOFM bit being setting to 0, then the comparator input offset calibration mode will be enabled. Refer to the "Comparator Input Offset Calibration" section for the detailed offset calibration procedures.

Bit 5 **OCPCRSP**: OCP comparator input offset calibration reference selection
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 4~0 **OCPCOF4~OCPCOF0**: OCP comparator input offset calibration control
This 5-bit field is used to perform the comparator input offset calibration operation and the value for the OCP comparator input offset calibration can be restored into this bit field. More detailed information is described in the "Comparator Input Offset Calibration" section.

Input Voltage Range

Together with different PGA operating modes, the input voltage can be positive or negative for flexible operation. The PGA output for the positive or negative input voltage is calculated based on different formulas and described as follows.

- For input voltages $V_{IN} > 0$, the PGA operates in the non-inverting mode and the PGA output is obtained using the formula below:

$$V_{OUT}=(1+R2/R1)\times V_{IN}$$

- For input voltages $0 > V_{IN} > -0.2V$, the PGA operates in the inverting mode and the PGA output is obtained using the formula below. Note that if the input voltage is negative, it cannot be lower than -0.2V which will result in current leakage.

$$V_{OUT}=-(R2/R1)\times V_{IN}$$

Input Offset Calibration

The OCP circuit has four operating modes controlled by the OCPEN[1:0] bit field, one of them is calibration mode. In calibration mode, operational amplifier and comparator offset can be calibrated.

Operational Amplifier Input Offset Calibration

Step 1. Set OCPEN[1:0]=11, OCPOOFM=1, OCPCOFM=0 and OCPORSP=1, the OCP will operate in the operational amplifier input offset calibration mode.

Step 2. Set OCPOOF[5:0]=000000 and then read the OCPCOUT bit.

Step 3. Increase the OCPOOF[5:0] value by 1 and then read the OCPCOUT bit.

If the OCPCOUT bit state has not changed, then repeat Step 3 until the OCPCOUT bit state has changed.

If the OCPCOUT bit state has changed, record the OCPOOF value as V_{OOS1} and then go to Step 4.

Step 4. Set OCPOOF[5:0]=111111 and read the OCPCOUT bit.

Step 5. Decrease the OCPOOF[5:0] value by 1 and then read the OCPCOUT bit.

If the OCPCOUT bit state has not changed, then repeat Step 5 until the OCPCOUT bit state has changed.

If the OCPCOUT bit state has changed, record the OCPOOF value as V_{OOS2} and then go to Step 6.

Step 6. Restore the operational amplifier input offset calibration value V_{OOS} into the OCPOOF[5:0] bit field. The offset Calibration procedure is now finished.

Where $V_{OOS}=(V_{OOS1}+V_{OOS2})/2$

Note: S4 is off. In this mode, the operational amplifier outputs to OCPCOUT bypassing the comparator.

Comparator Input Offset Calibration

Step 1. Set OCPEN[1:0]=11, OCPCOFM=1 and OCPOOFM=0, the OCP will now operate in the comparator input offset calibration mode.

Step 2. Set OCPCOF[4:0]=00000 and read the OCPCOUT bit.

Step 3. Increase the OCPCOF[4:0] value by 1 and then read the OCPCOUT bit.

If the OCPCOUT bit state has not changed, then repeat Step 3 until the OCPCOUT bit state has changed.

If the OCPCOUT bit state has changed, record the OCPCOF value as V_{COS1} and then go to Step 4.

Step 4. Set OCPCOF[4:0]=11111 and then read the OCPCOUT bit.

Step 5. Decrease the OCPCOF[4:0] value by 1 and then read the OCPCOUT bit.

If the OCPCOUT bit state has not changed, then repeat Step 5 until the OCPCOUT bit state has changed.

If the OCPCOUT bit state has changed, record the OCPCOF value as V_{COS2} and then go to Step 6.

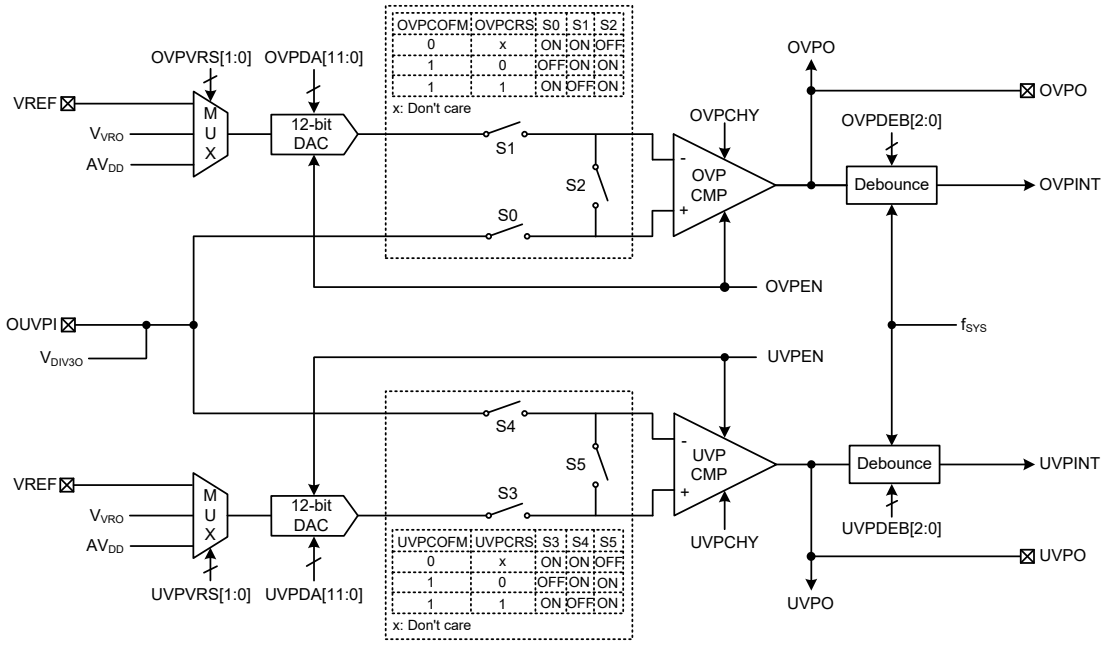
Step 6. Restore the comparator input offset calibration value V_{COS} into the OCPCOF[4:0] bit field. The offset Calibration procedure is now finished.

Where $V_{COS}=(V_{COS1}+V_{COS2})/2$

Note: S4 is on and the D/A converter is off. This situation is only available for comparator calibration procedure. In the normal operation mode, S4 is off.

Over/Under Voltage Protection – OUVP

The device includes an Over/Under Voltage Protection (OUVP) function which provides a voltage protection mechanism for applications. The input voltage on the OUVPI pin is compared with two reference voltages generated by two 12-bit D/A converters respectively. When an over or under voltage condition occurs, an OVP or UVP interrupt will be generated if the corresponding interrupt control bit is enabled.



Note: 1. As the OUVP function relevant external pins are pin-shared with general I/O or other functions, ensure that the corresponding pin-shared function registers have been configured properly before using the OVP function.

2. As the OUVPI pin is connected together with the internal DIV3 Divider Resistor circuit output signal V_{DIV3O}, special consideration should be given to the interaction between internal and external circuits and the corresponding control bits should be properly configured according to the application requirements.

3. The V_{VRO} is sourced from the A/D converter unit gain buffer output.

Over/Under Voltage Protection Circuit

Over Voltage Protection Operation

The OVP circuit is used to prevent the input voltage from exceeding a specific level. The OUVPI pin input voltage is compared with a reference voltage provided by a 12-bit D/A converter. The 12-bit D/A converter reference input signal is supplied by AV_{DD}, V_{VRO} or the external VREF pin, which is selected by the OVPVRS1~OVPVRS0 bits. The comparator output, OVPO, will be filtered with a certain debounce time period selected by the OVPDEB1~OVPDEB0 bits in the OUVPC0 register. Then a filtered OVP digital comparator output, OVPINT, is obtained to indicate whether an over voltage condition occurs or not, which will further trigger the OVP interrupt if the corresponding interrupt control bit is enabled. The comparator in the OVP circuit also has hysteresis function controlled by the OVPCHY bit.

Under Voltage Protection Operation

The UVP circuit is used to prevent the input voltage from being less than a specific level. The OUVPI pin input voltage is compared with a reference voltage provided by a 12-bit D/A converter. The 12-bit D/A converter reference input signal is supplied by AV_{DD}, V_{VRO} or the external VREF

pin, which is selected by the UVPVRS1~UVPVRS0 bits. The comparator output, UVPO, will be filtered with a certain debounce time period selected by the UVPDEB1~UVPDEB0 bits in the OUVPC1 register. Then a filtered UVP digital comparator output, UVPINT, is obtained to indicate whether an under voltage condition occurs or not, which will further trigger the UVP interrupt if the corresponding interrupt control bit is enabled. The comparator in the UVP circuit also has hysteresis function controlled by the UVPCHY bit.

Over/Under Voltage Protection Registers

Overall operation of the OUVP function is controlled using several registers. The OVPDAH/OVPDAL and UVPDAH/UVPDAL registers are used to provide reference voltage for the OVP and UVP circuits respectively. The OUVPC0~OUVPC3 control registers are used to control the OVP/UVP function, D/A converter reference voltage selection, comparator debounce time selection, comparator hysteresis function and comparator output polarity control, etc.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OVPDAL	D7	D6	D5	D4	D3	D2	D1	D0
OVPDAH	—	—	—	—	D11	D10	D9	D8
UVPDAL	D7	D6	D5	D4	D3	D2	D1	D0
UVPDAH	—	—	—	—	D11	D10	D9	D8
OUVPC0	—	—	OVPEN	OVPCHY	OVPVRS1	OVPVRS0	OVPDEB1	OVPDEB0
OUVPC1	—	—	UVPEN	UVPCHY	UVPVRS1	UVPVRS0	UVPDEB1	UVPDEB0
OUVPC2	OVPO	OVPCOFM	OVPCRS	OVPCOF4	OVPCOF3	OVPCOF2	OVPCOF1	OVPCOF0
OUVPC3	UVPO	UVPCOFM	UVPCRS	UVPCOF4	UVPCOF3	UVPCOF2	UVPCOF1	UVPCOF0

Over/Under Voltage Protection Register List

• OVPDAH & OVPDAL Registers

Register	OVPDAH								OVPDAL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8	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	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0	0	0	0	0	0	0	0	0

"—": Unimplemented, read as "0"

D11~D8: OVP D/A converter reference output voltage control high byte

D7~D0: OVP D/A converter reference output voltage control low byte

OVP D/A converter output=(OVP D/A converter reference voltage/4096)×D[11:0]

Note: Each time when the OVPDAH register is written, the whole 12-bit data will be loaded into the D/A converter and a conversion cycle will be initiated

• UVPDAH & UVPDAL Registers

Register	UVPDAH								UVPDAL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8	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	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0	0	0	0	0	0	0	0	0

"—": Unimplemented, read as "0"

D11~D8: UVP D/A converter reference output voltage control high byte

D7~D0: UVP D/A converter reference output voltage control low byte

UVP D/A converter output=(UVP D/A converter reference voltage/4096)×D[11:0]

Note: Each time when the UVPDAH register is written, the whole 12-bit data will be loaded into the D/A converter and a conversion cycle will be initiated.

• **OUVPC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	OVPEN	OVPCHY	OVPVRS1	OVPVRS0	OVPDEB1	OVPDEB0
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 **OVPEN**: OVP function enable control
0: Disable
1: Enable
If the OVPEN bit is cleared to 0, the over voltage protection function is disabled and no power will be consumed. This results in the comparator and D/A converter of the OVP both being switched off.

Bit 4 **OVPCHY**: OVP comparator hysteresis enable control
0: Disable
1: Enable

Bit 3~2 **OVPVRS1~OVPVRS0**: OVP D/A converter reference voltage selection
00: From AV_{DD}
01: From VREF pin
10: From V_{VRO}
11: From AV_{DD}

Bit 1~0 **OVPDEB1~OVPDEB0**: OVP comparator debounce time selection
00: No debounce
01: $(7\sim8)\times t_{DEB}$
10: $(15\sim16)\times t_{DEB}$
11: $(31\sim32)\times t_{DEB}$
Note: $t_{DEB}=1/f_{SYS}$.

• **OUVPC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	UVPEN	UVPCHY	UVPVRS1	UVPVRS0	UVPDEB1	UVPDEB0
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 **UVPEN**: UVP function enable control
0: Disable
1: Enable
If the UVPEN bit is cleared to 0, the under voltage protection function is disabled and no power will be consumed. This results in the comparator and D/A converter of the UVP both being switched off.

Bit 4 **UVPCHY**: UVP comparator hysteresis enable control
0: Disable
1: Enable

Bit 3~2 **UVPVRS1~UVPVRS0**: UVP D/A converter reference voltage selection
00: From AV_{DD}
01: From VREF pin
10: From V_{VRO}
11: From AV_{DD}

Bit 1~0 **UVPDEB1~UVPDEB0**: UVP comparator debounce time selection
00: No debounce
01: $(7\sim8)\times t_{DEB}$
10: $(15\sim16)\times t_{DEB}$
11: $(31\sim32)\times t_{DEB}$
Note: $t_{DEB}=1/f_{SYS}$.

• **OUVPC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	OVPO	OVPCOFM	OVPCRS	OVPCOF4	OVPCOF3	OVPCOF2	OVPCOF1	OVPCOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **OVPO**: OVP comparator output bit
0: Positive input voltage < Negative input voltage
1: Positive input voltage > Negative input voltage

Bit 6 **OVPCOFM**: OVP comparator operating mode selection
0: Normal operation mode
1: Input offset voltage calibration mode

This bit is used to select the OVP comparator operating mode. To select the comparator input offset cancellation mode, the OVPCOFM bit must be set to 1. Refer to the "OVP Comparator Calibration" section for the detailed offset cancellation procedures.

Bit 5 **OVPCRS**: OVP comparator input offset voltage calibration reference selection
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 4~0 **OVPCOF4~OVPCOF0**: OVP comparator input offset voltage calibration control

This 5-bit field is used to perform the comparator input offset cancellation operation and the value for the OVP comparator input offset cancellation can be restored into this bit field. More detailed information is described in the "OVP Comparator Calibration" section.

• **OUVPC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	UVPO	UVPCOFM	UVPCRS	UVPCOF4	UVPCOF3	UVPCOF2	UVPCOF1	UVPCOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **UVPO**: UVP comparator output bit
0: Positive input voltage < Negative input voltage
1: Positive input voltage > Negative input voltage

Bit 6 **UVPCOFM**: UVP comparator operating mode selection
0: Normal operation mode
1: Input offset voltage calibration mode

This bit is used to select the UVP comparator operating mode. To select the comparator input offset cancellation mode, the UVPCOFM bit must be set to 1. Refer to the "UVP Comparator Calibration" section for the detailed offset cancellation procedures.

Bit 5 **UVPCRS**: UVP comparator input offset voltage calibration reference selection
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 4~0 **UVPCOF4~UVPCOF0**: UVP comparator input offset voltage calibration control

This 5-bit field is used to perform the comparator input offset cancellation operation and the value for the UVP comparator input offset cancellation can be restored into this bit field. More detailed information is described in the "UVP Comparator Calibration" section.

Comparator Input Offset Calibration

As the OUVPI input is pin-shared with other pin functions, it should be configured as the OUVPI input pin function first by configuring the relevant pin-shared control bits. It should be noted that before offset calibration, the hysteresis voltage should be zero by clearing the OVPCHY or UVPCHY bit to zero. For comparator input offset calibration, the procedures are summarised as follows.

OVP Comparator Calibration

Step 1. Set OVPCOFM=1, OVPCRS=1, the OVP is now in the comparator offset calibration mode, S0 and S2 are on. To make sure V_{OS} as minimised as possible after calibration, the input reference voltage in calibration mode should be the same as input DC operating voltage in normal mode operation.

Step 2. Set OVPCOF[4:0]=00000 and then read the OVPO bit status.

Step 3. Increase the OVPCOF[4:0] value by 1 and then read the OVPO bit status.

If the OVPO state has not changed, repeat Step3 until the OVPO bit state has changed.

If the OVPO state has changed, record the OVPCOF[4:0] value as V_{OS1} and then go to Step4.

Step 4. Set OVPCOF[4:0]=11111 and then read the OVPO bit status.

Step 5. Decrease the OVPCOF[4:0] value by 1 and then read the OVPO bit status.

If the OVPO state has not changed, repeat Step5 until the OVPO bit state has changed.

If the OVPO state has changed, record the OVPCOF[4:0] value as V_{OS2} and then go to Step6.

Step 6. Restore $V_{OS}=(V_{OS1}+V_{OS2})/2$ to the OVPCOF[4:0] bits. The calibration is finished.

If $(V_{OS1}+V_{OS2})/2$ is not integral, discard the decimal. Residue $V_{OS}=V_{OUT}-V_{IN}$.

UVP Comparator Calibration

Step 1. Set UVPCOFM=1, UVPCRS=0, the UVP is now in the comparator offset calibration mode, S4 and S5 are on. To make sure V_{OS} as minimised as possible after calibration, the input reference voltage in calibration mode should be the same as input DC operating voltage in normal mode operation.

Step 2. Set UVPCOF[4:0]=00000 and then read the UVPO bit status.

Step 3. Increase the UVPCOF[4:0] value by 1 and then read the UVPO bit status.

If the UVPO state has not changed, repeat Step3 until the UVPO bit state has changed.

If the UVPO state has changed, record the UVPCOF[4:0] value as V_{OS1} and then go to Step4.

Step 4. Set UVPCOF[4:0]=11111 and then read the UVPO bit status.

Step 5. Decrease the UVPCOF[4:0] value by 1 and then read the UVPO bit status.

If the UVPO state has not changed, repeat Step5 until the UVPO bit state has changed.

If the UVPO state has changed, record the UVPCOF[4:0] value as V_{OS2} and then go to Step6.

Step 6. Restore $V_{OS}=(V_{OS1}+V_{OS2})/2$ to the UVPCOF[4:0] bits. The calibration is finished.

If $(V_{OS1}+V_{OS2})/2$ is not integral, discard the decimal. Residue $V_{OS}=V_{OUT}-V_{IN}$.

Divider Resistors

The device includes multiple integrated divider resistor circuits. The DIVxSC (x=2 or 3) register can be used to select whether the input signal on the DIVxI pin passes through the integrated divider resistor circuit, directly enters the MCU or is cut off.

Users should take care that the control bits are configured properly to ensure that the results are as expected.

- If the signal input is required to be cut off, the corresponding control switches should be set as follows:

  DIVxS1=0, DIVxS0=0

- If the internal divider resistors are required, the corresponding control switches should be set as follows:

  DIVxS1=0, DIVxS0=1

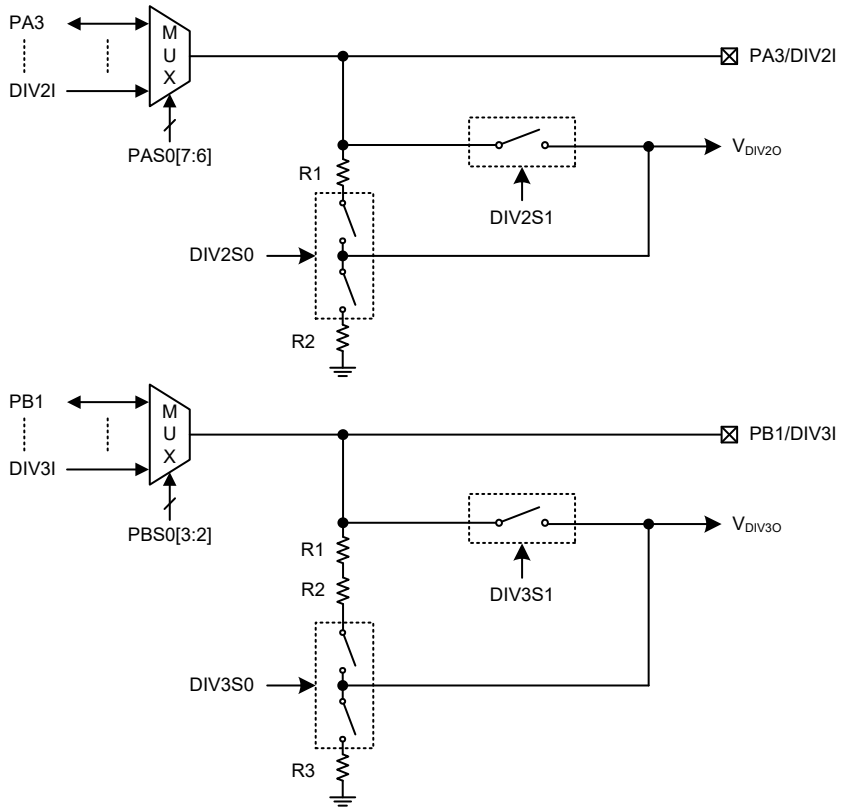
- If the input signal is allowed to directly enter the MCU, the corresponding control switches should be set as follows:

  DIVxS1=1, DIVxS0=0

When the pin is used as other pin-shared functions, the corresponding DIVx switches should be switched off to avoid impact.

When the pin is used as the DIVxI function, other unused functions should be off to avoid impact.

If the V_{DIVxO} is connected to the A/D converter internal input, the A/D sampling of the V_{DIVxO} under conditions of DIVxS1=1 and DIVxS0=0 should be avoided due to the influence of the switch circuits, so as to avoid measurement errors and non-compliance with specifications.



Divider Resistor Circuits

• **DIV2SC Register – HT45R7130**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	DIV2S1	DIV2S0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"
Bit 1 **DIV2S1**: Integrated divider resistor Switch DIV2S1 on/off control
0: Off
1: On
Bit 0 **DIV2S0**: Integrated divider resistor Switch DIV2S0 on/off control
0: Off
1: On

• **DIV3SC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	DIV3S1	DIV3S0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"
Bit 1 **DIV3S1**: Integrated divider resistor Switch DIV3S1 on/off control
0: Off
1: On
Bit 0 **DIV3S0**: Integrated divider resistor Switch DIV3S0 on/off control
0: Off
1: On

Serial Peripheral Interface – SPI

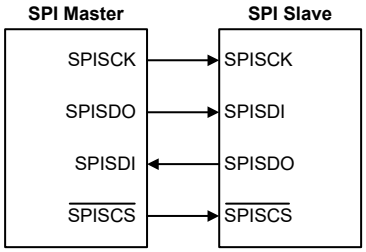
The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the programming requirements when communicating with external hardware devices.

The communication is full duplex and operates as a slave/master type, where the device can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, however the device provides only one $\overline{\text{SPISCS}}$ pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pins to select the slave devices.

SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SPISDI, SPISDO, SPISCK and $\overline{\text{SPISCS}}$. Pins SPISDI and SPISDO are the Serial Data Input and Serial Data Output lines, the SPISCK pin is the Serial Clock line and $\overline{\text{SPISCS}}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins, the SPI interface must first be enabled by configuring the corresponding selection bits in the pin-shared function selection registers. The SPI can be disabled or enabled using the SPIEN bit in the SPIC0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{\text{SPISCS}}$ pin only one slave device can be utilized. The pull-high resistors of the SPI pin-shared I/O are selected using pull-high control registers when the SPI function is enabled and the corresponding pins are used as SPI input pins.

The $\overline{\text{SPISCS}}$ pin is controlled by software, set SPICSEN bit to 1 to enable the $\overline{\text{SPISCS}}$ pin function, and clear SPICSEN bit to 0, the $\overline{\text{SPISCS}}$ pin will be floating state.

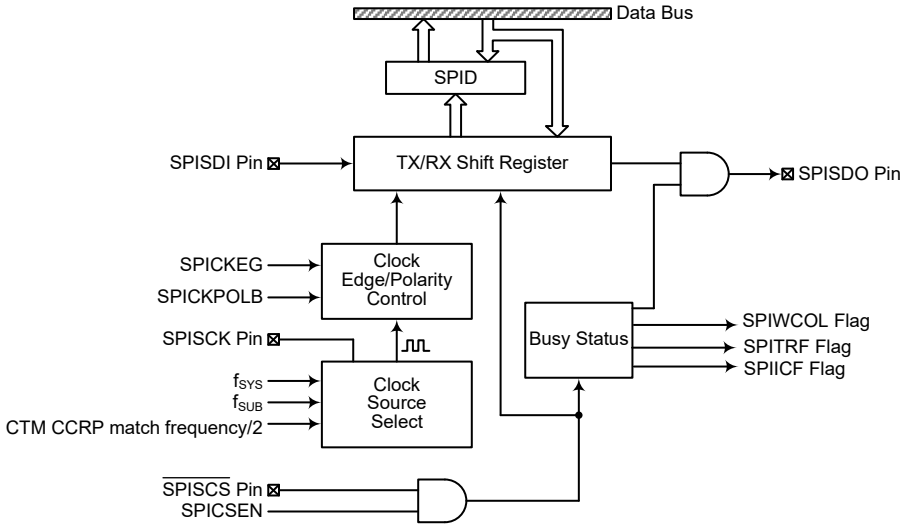


SPI Master/Slave Connection

The SPI function in the device offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes
- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge

The status of the SPI interface pins is determined by a number of factors such as whether the device is in the master or slave mode and upon the condition of certain control bits such as SPICSEN and SPIEN.



SPI Block Diagram

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SPID data register and two registers, SPIC0 and SPIC1.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SPIC0	SPIM2	SPIM1	SPIM0	—	—	—	SPIEN	SPIICF
SPIC1	—	—	SPICKPOLB	SPICKEG	SPIMLS	SPICSEN	SPIWCOL	SPITRF
SPID	D7	D6	D5	D4	D3	D2	D1	D0

SPI Register List

SPI Data Register

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

- **SPID 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**: SPI data register bit 7 ~ bit 0

SPI Control Registers

There are also two control registers for the SPI interface, SPIC0 and SPIC1. The SPIC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SPIC1 register is used for other control functions such as LSB/MSB selection, write collision flag etc.

- **SPIC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SPIM2	SPIM1	SPIM0	—	—	—	SPIEN	SPIICF
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	0	0	0	—	—	—	0	0

Bit 7~5 **SPIM2~SPIM0**: SPI operating mode control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is CTM CCRP match frequency/2
101: SPI slave mode
110: SPI master mode; SPI clock is $f_{SYS}/2$
111: SPI master mode; SPI clock is f_{SYS}

These bits are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from CTM and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4~2 Unimplemented, read as "0"

Bit 1 **SPIEN**: SPI enable control
0: Disable
1: Enable

The bit is the overall on/off control for the SPI interface. When the SPIEN bit is cleared to zero to disable the SPI interface, the SPISDI, SPISDO, SPISCK and $\overline{\text{SPISCS}}$ lines will lose their SPI function and the SPI operating current will be reduced to a minimum value. When the bit is high the SPI interface is enabled.

Bit 0 **SPIICF**: SPI incomplete flag
0: SPI incomplete condition is not occurred
1: SPI incomplete condition is occurred

This bit is only available when the SPI is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SPIEN and SPICSEN bits both being set high but the $\overline{\text{SPISCS}}$ line is pulled high by the external master device before the

SPI data transfer is completely finished, the SPIICF bit will be set high together with the SPITRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the SPITRF bit will not be set high if the SPIICF bit is set high by software application program.

- **SPIC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SPICKPOLB	SPICKEG	SPIMLS	SPICSEN	SPIWCOL	SPITRF
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 **SPICKPOLB**: SPI clock line base condition selection
0: The SPISCK line will be high when the clock is inactive
1: The SPISCK line will be low when the clock is inactive

The SPICKPOLB bit determines the base condition of the clock line, if the bit is high, then the SPISCK line will be low when the clock is inactive. When the SPICKPOLB bit is low, then the SPISCK line will be high when the clock is inactive.

Bit 4 **SPICKEG**: SPI SPISCK clock active edge type selection

SPICKPOLB=0
0: SPISCK has high base level with data capture on SPISCK rising edge
1: SPISCK has high base level with data capture on SPISCK falling edge

SPICKPOLB=1
0: SPISCK has low base level with data capture on SPISCK falling edge
1: SPISCK has low base level with data capture on SPISCK rising edge

The SPICKEG and SPICKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before a data transfer is executed otherwise an erroneous clock edge may be generated. The SPICKPOLB bit determines the base condition of the clock line, if the bit is high, then the SPISCK line will be low when the clock is inactive. When the SPICKPOLB bit is low, then the SPISCK line will be high when the clock is inactive. The SPICKEG bit determines active clock edge type which depends upon the condition of the SPICKPOLB bit.

Bit 3 **SPIMLS**: SPI data shift order control
0: LSB first
1: MSB first

This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2 **SPICSEN**: SPI $\overline{\text{SPISCS}}$ pin control
0: Disable
1: Enable

The SPICSEN bit is used as an enable/disable for the $\overline{\text{SPISCS}}$ pin. If this bit is low, then the $\overline{\text{SPISCS}}$ pin will be disabled and placed into a floating condition. If the bit is high the $\overline{\text{SPISCS}}$ pin will be enabled and used as a select pin.

Bit 1 **SPIWCOL**: SPI write collision flag
0: No collision
1: Collision

The SPIWCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SPID register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared to zero by the application program.

Bit 0 **SPITRF**: SPI Transmit/Receive complete flag
0: SPI data is being transferred
1: SPI data transmission is completed

The SPITRF bit is the Transmit/Receive Complete flag and is set "1" automatically

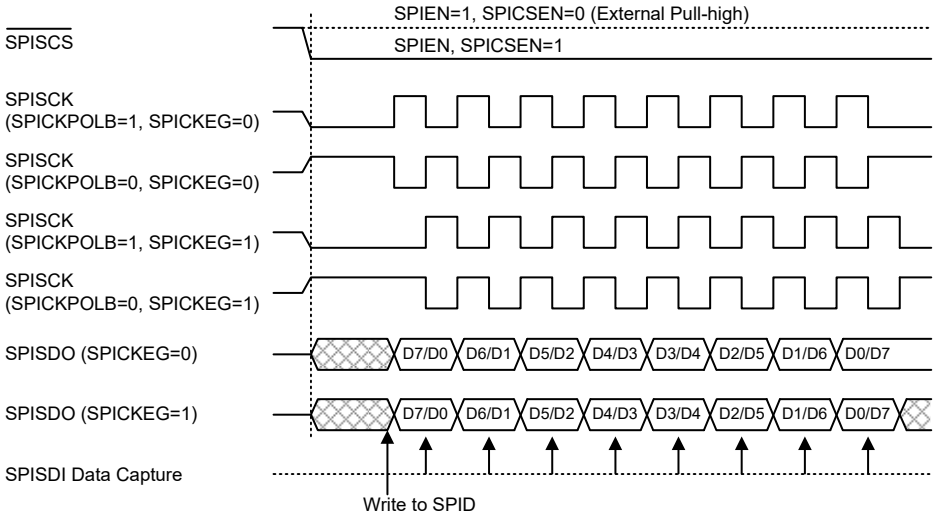
when an SPI data transmission is completed, but must cleared to zero by the application program. It can be used to generate an interrupt.

SPI Communication

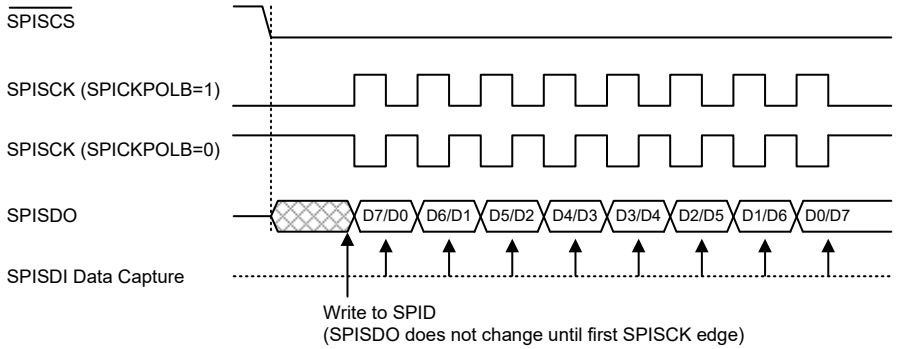
After the SPI interface is enabled by setting the SPIEN bit high, then in the Master Mode, when data is written to the SPID register, transmission/reception will begin simultaneously. When the data transfer is complete, the SPITRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SPID register will be transmitted and any data on the SPISDI pin will be shifted into the SPID register.

The master should output a $\overline{\text{SPISCS}}$ signal to enable the slave device before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the SPISCK signal depending upon the configurations of the SPICKPOLB bit and SPICKEG bit. The accompanying timing diagram shows the relationship between the slave data and SPISCK signal for various configurations of the SPICKPOLB and SPICKEG bits.

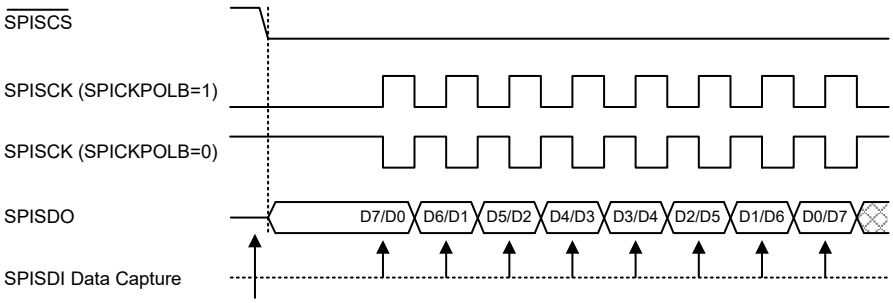
The SPI Master mode will continue to function if the SPI clock is running.



SPI Master Mode Timing



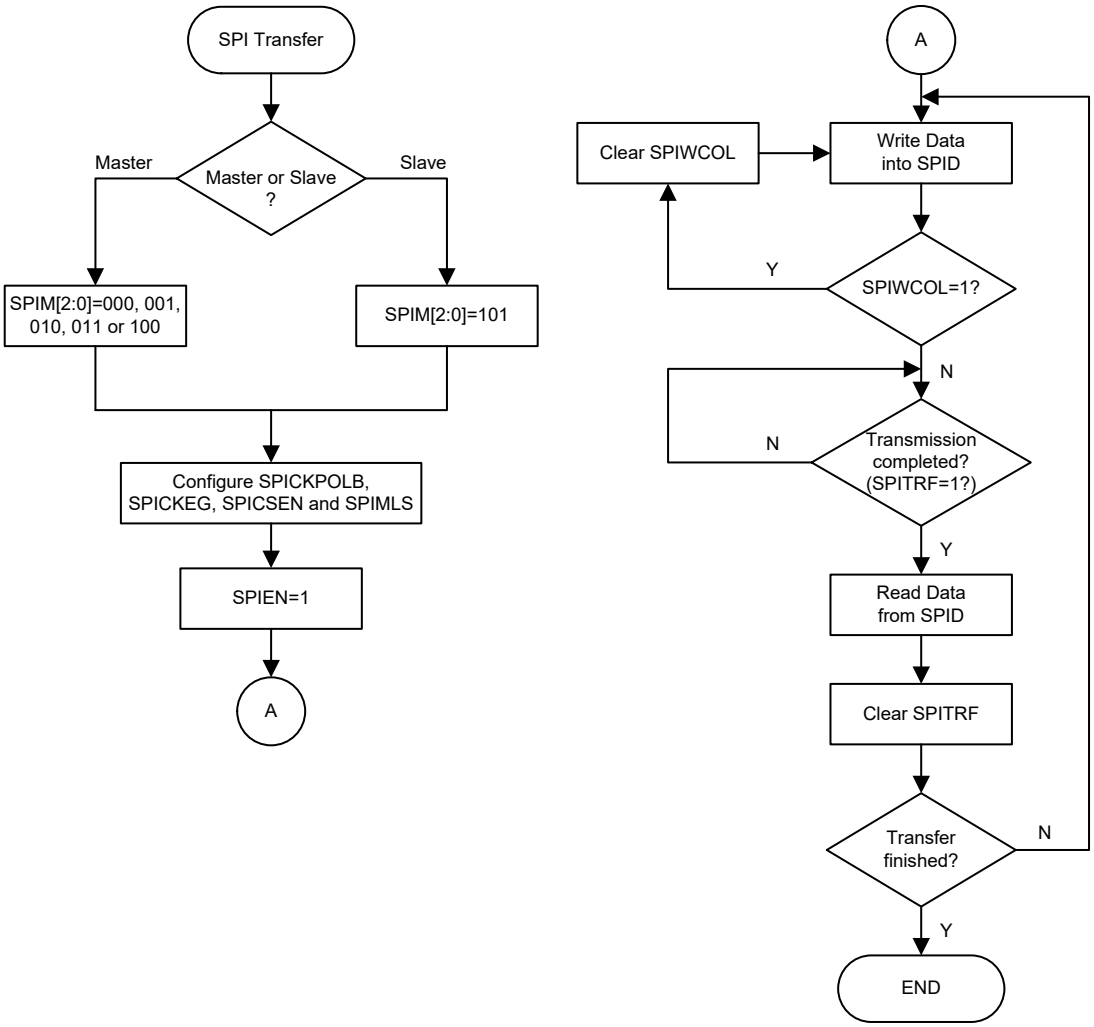
SPI Slave Mode Timing – SPICKEG=0



Write to SPID
(SPISDO changes as soon as writing occurs; SPISDO is floating if $\overline{\text{SPISCS}}$=1)

Note: For SPI slave mode, if SPIEN=1 and SPICSEN=0, SPI is always enabled and ignores the $\overline{\text{SPISCS}}$ level.

SPI Slave Mode Timing – SPICKEG=1



SPI Transfer Control Flowchart

SPI Bus Enable/Disable

To enable the SPI bus, set SPICSEN=1 and $\overline{\text{SPISCS}}$=0, then wait for data to be written into the SPID (TXRX buffer) register. For the Master Mode, after data has been written to the SPID (TXRX buffer) register, then transmission or reception will start automatically. When all the data has been transferred the SPITRF bit should be set. For the Slave Mode, when clock pulses are received on SPISCK, data in the TXRX buffer will be shifted out or data on SPISDI will be shifted in.

When the SPI bus is disabled, SPISCK, SPISDI, SPISDO, $\overline{\text{SPISCS}}$ will become I/O pins or the other pin-shared functions by configuring the corresponding pin-shared selection bits.

SPI Operation Steps

All communication is carried out using the 4-line interface for either Master or Slave Mode.

The SPICSEN bit in the SPIC1 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{\text{SPISCS}}$ line to be active, which can then be used to control the SPI interface. If the SPICSEN bit is low, the SPI interface will be disabled and the $\overline{\text{SPISCS}}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the SPICSEN bit and the SPIEN bit in the SPIC0 register are set high, this will place the SPISDI line in a floating condition and the SPISDO line high. If in Master Mode the SPISCK line will be either high or low depending upon the clock polarity selection bit SPICKPOLB in the SPIC1 register. If in Slave Mode the SPISCK line will be in a floating condition. If SPIEN is low then the bus will be disabled and $\overline{\text{SPISCS}}$, SPISDI, SPISDO and SPISCK will all become I/O pins or the other functions using the corresponding pin-shared function selection bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SPID register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode

- Step 1

  Select the SPI Master mode and clock source using the SPIM2~SPIM0 bits in the SPIC0 control register.

- Step 2

  Setup the SPICSEN bit and setup the SPIMLS bit to choose if the data is MSB or LSB first, this must be same as the Slave device.

- Step 3

  Setup the SPIEN bit in the SPIC0 control register to enable the SPI interface.

- Step 4

  For write operations: write the data to the SPID register, which will actually place the data into the TXRX buffer. Then use the SPISCK and SPISDO lines to output the data. After this go to step 5.

  For read operations: the data transferred in on the SPISDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPID register.

- Step 5

  Check the SPIWCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.

- Step 6

  Check the SPITRF bit or wait for an SPI serial bus interrupt.

- Step 7

  Read data from the SPID register.
- Step 8

  Clear SPITRF.
- Step 9

  Go to step 4.

Slave Mode

- Step 1

  Select the SPI Slave mode using the SPIM2~SPIM0 bits in the SPIC0 control register.
- Step 2

  Setup the SPICSEN bit and setup the SPIMLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master device.
- Step 3

  Setup the SPIEN bit in the SPIC0 control register to enable the SPI interface.
- Step 4

  For write operations: write the data to the SPID register, which will actually place the data into the TXRX buffer. Then wait for the master clock SPISCK and $\overline{\text{SPISCS}}$ signal. After this, go to step 5.

  For read operations: the data transferred in on the SPISDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPID register.
- Step 5

  Check the SPIWCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6

  Check the SPITRF bit or wait for an SPI serial bus interrupt.
- Step 7

  Read data from the SPID register.
- Step 8

  Clear SPITRF.
- Step 9

  Go to step 4.

Error Detection

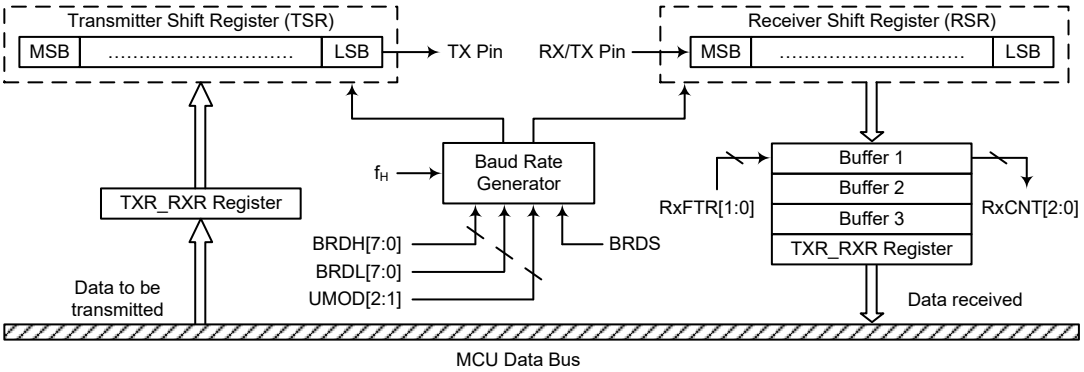
The SPIWCOL bit in the SPIC1 register is provided to indicate errors during data transfer. The bit is set by the SPI serial Interface but must be cleared by the application program. This bit indicates a data collision has occurred which happens if a write to the SPID register takes place during a data transfer operation and will prevent the write operation from continuing.

UART Interface

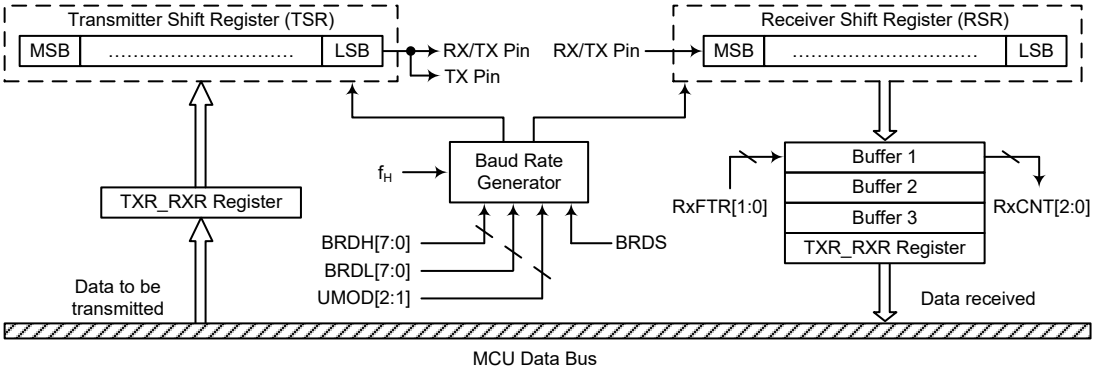
The device contains an integrated full-duplex or half-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data 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 or half-duplex (single wire mode), asynchronous communication
- 8 or 9 bits character length
- Even, odd, mark, space or no parity options
- One or two stop bits configurable for receiver
- Two stop bits for transmitter
- Baud rate generator with 16-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 4-byte Deep FIFO Receive Data Buffer
- 1-byte Deep FIFO Transmit Data Buffer
- RX/TX pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
  - Transmitter Empty
  - Transmitter Idle
  - Receiver reaching FIFO trigger level
  - Receiver Overrun
  - Address Mode Detect



UART Data Transfer Block Diagram – SWM=0



UART Data Transfer Block Diagram – SWM=1

UART External Pins

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

UART Single Wire Mode

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

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

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

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

UART Data Transfer Scheme

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

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

UART Status and Control Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
UCR3	—	—	—	—	—	—	—	SWM
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRDH	D7	D6	D5	D4	D3	D2	D1	D0
BRDL	D7	D6	D5	D4	D3	D2	D1	D0
UFCR	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
RxCNT	—	—	—	—	—	D2	D1	D0

UART Register List

• USR Register

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

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

Bit 7 **PERR**: Parity error flag
0: No parity error is detected
1: Parity error is detected

The PERR flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if the parity is enabled and the parity type (odd, even, mark or space) is selected. The flag can also be cleared by a software sequence which involves a read to the status register USR followed by an access to the TXR_RXR data register.

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

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

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

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

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

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

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

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

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

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

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

The TIDLE flag is known as the transmission complete flag. When this read only flag is “0”, it indicates that a transmission is in progress. This flag will be set high when the TXIF flag is “1” and when there is no transmit data or break character being transmitted. When TIDLE is equal to “1”, the 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 will also be set since the transmit data register is not yet full.

• UCR1 Register

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

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

"x": unknown

Bit 7 **UARTEN**: UART function enable control
0: Disable UART. TX and RX/TX pins are in a floating state
1: Enable UART. TX and RX/TX 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/TX 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/TX pins will function as defined by the SWM mode selection bit together with 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 as well as the RxCNT register will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

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

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

Note that the 9th bit of data if BNO=1, or the 8th bit of data if BNO=0, which is used as the parity bit, does not transfer to RX8 or TXRX7 respectively when the parity function is enabled.

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

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

Bit 2 **TXBRK**: Transmit break character
0: No break character is transmitted
1: Break characters transmit
The TXBRK bit is the Transmit Break Character bit. When this bit is "0", there are no break characters and the 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 receiver STOP bit number selection, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

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

Bit 7 **TXEN**: UART Transmitter enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled
The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to "0", the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the 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/TX 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/TX 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/TX pin will be set in a floating state.

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

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

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

The bit named ADDEN is the address detect function enable control bit. When this bit is equal to "1", the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXRX7 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/TX pin wake-up UART function enable control
0: RX/TX pin wake-up UART function is disabled
1: RX/TX 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/TX pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no RX/TX 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/TX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX/TX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the RX/TX pin when the WAKE bit is cleared to 0.

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

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

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

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

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

• UCR3 Register

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

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

Bit 7~1 Unimplemented, read as "0"
Bit 0 **SWM**: Single Wire Mode enable control
0: Disable, the RX/TX pin is used as UART receiver function only
1: Enable, the RX/TX pin can be used as UART receiver or transmitter function controlled by the RXEN and TXEN bits
Note that when the Single Wire Mode is enabled, if both the RXEN and TXEN bits are high, the RX/TX pin will just be used as UART receiver input.

• TXR_RXR Register

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

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

"x": unknown

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

• BRDH Register

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

Bit 7~0 **D7~D0**: Baud rate divider high byte
The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.
Baud Rate=f_H/(BRD+UMOD/8)
BRD=16~65535 or 8~65535 depending on BRDS
Note: 1. The BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.
2. The BRDL must be written first and then BRDH, otherwise errors may occur.
3. The BRDH register should not be modified during data transmission process.

• **BRDL Register**

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

Bit 7~0 **D7~D0**: Baud rate divider low byte

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

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

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

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

• **UFCR Register**

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

Bit	7	6	5	4	3	2	1	0
Name	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5~3 **UMOD2~UMOD0**: UART Modulation Control bits

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

Bit 2 **BRDS**: BRD range selection
0: BRD range is from 16 to 65535
1: BRD range is from 8 to 65535

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

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

Bit 1~0 **RxFTR1~RxFTR0**: Receiver FIFO trigger level (bytes)
00: 4 bytes in Receiver FIFO
01: 1 or more bytes in Receiver FIFO
10: 2 or more bytes in Receiver FIFO
11: 3 or more bytes in Receiver FIFO

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

• **RxCNT Register**

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

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

Bit 7~3 Unimplemented, read as "0"

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

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

Baud Rate Generator

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

f_H/BR=Integer Part+Fractional Part

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

BRD=TRUNC (f_H/BR)

UMOD=ROUND[MOD(f_H/BR)×8]

Therefore, the actual baud rate is calculated as follows:

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

Calculating the Baud Rate and Error Values

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

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

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

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

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

Modulation Control Example

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

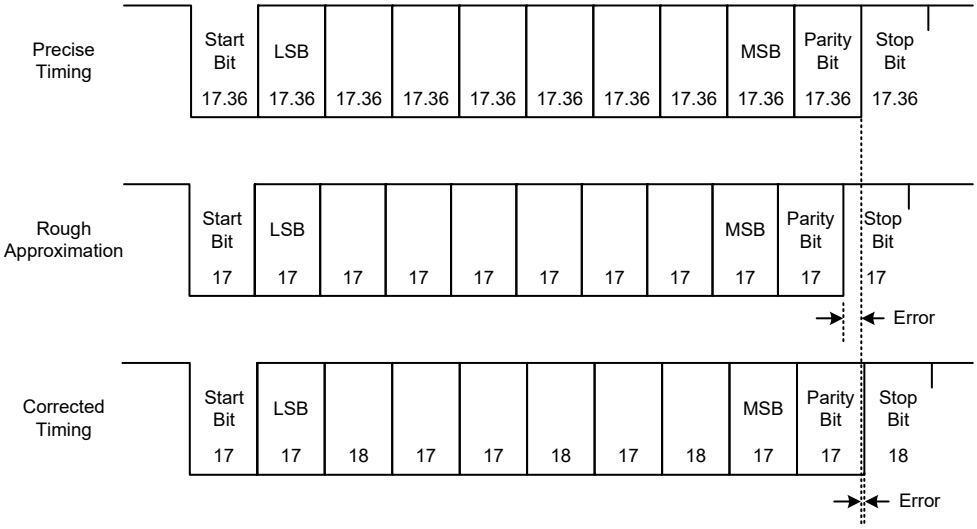
Fraction Addition	Carry to Bit 3	UART Bit Time Sequence	Extra UART Clock Cycle
0000b+0011b=0011b	No	Start bit	No
0011b+0011b=0110b	No	D0	No
0110b+0011b=1001b	Yes	D1	Yes
1001b+0011b=1100b	No	D2	No
1100b+0011b=1111b	No	D3	No
1111b+0011b=0010b	Yes	D4	Yes
0010b+0011b=0101b	No	D5	No
0101b+0011b=1000b	Yes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

Baud Rate Correction Example

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

The following figure shows three different frames:

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



UART Setup and Control

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

generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

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

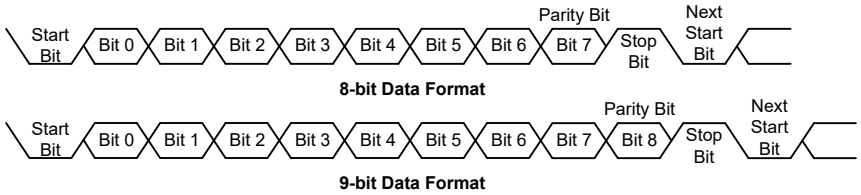
Data, Parity and Stop Bit Selection

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

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

Transmitter Receiver Data Format

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



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, PRT1~PRT0 and PREN bits to define the required word length and parity type. Two stop bits are used for the transmitter.
- Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits 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.

Transmitting Break

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

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the BNO bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8 bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX/TX 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/TX 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/TX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX/TX 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/TX input pin, LSB first. In the read mode, the TXR_RXR register forms a buffer between the internal bus and the receiver shift register. The TXR_RXR register is a four byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR_RXR before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERR will be subsequently indicated. For continuous multi-byte data transmission, it is strongly recommended that the receiver uses two stop bits to avoid a receiving error caused by the accumulated error of the receiver baud rate frequency.

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

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

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

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

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

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

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

Receiving Break

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

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

Idle Status

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

Receiver Interrupt

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

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

Managing Receiver Errors

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

Overrun Error – OERR

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

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

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

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

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

Noise Error – NF

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

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

Note that the NF flag is reset by 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

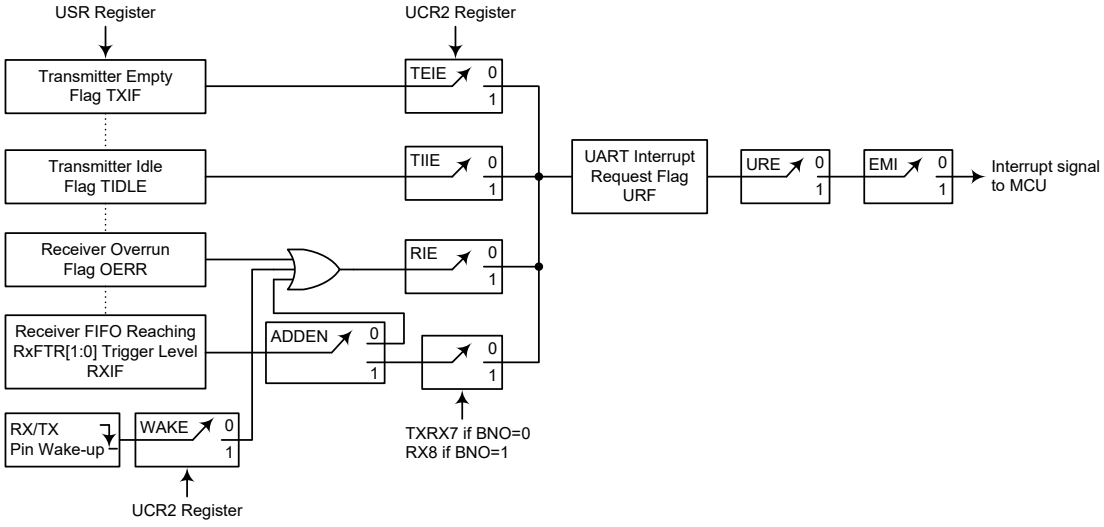
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd, even, mark or space, is selected. The read only PERR flag and the received data will be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

UART Interrupt Structure

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

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX/TX 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/TX 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 URE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to zero.

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

ADDEN Bit Function

UART Power Down and Wake-up

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

The UART function contains a receiver RX/TX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the UART clock (f_H) is off, then a falling edge on the RX/TX pin will trigger an RX/TX 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/TX pin will be ignored.

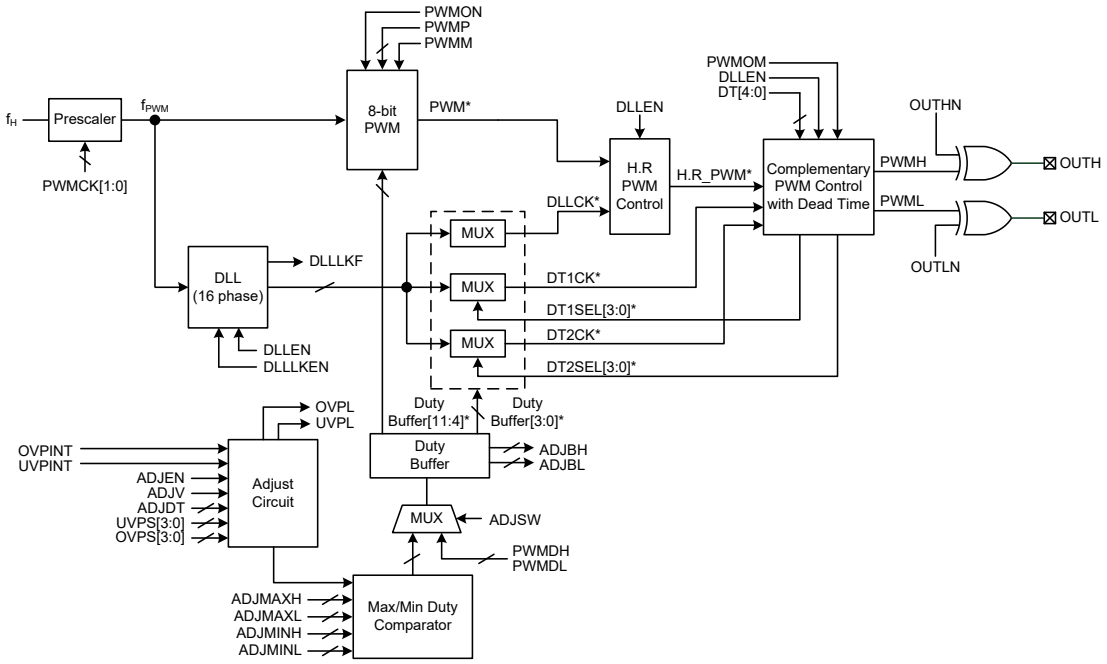
For a UART wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URE, must be set. If the EMI and URE bits are not set then only a wake up event will occur and no interrupt will be generated. Note also that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

Auto-adjust High Resolution PWM with Delay Lock Loop/Dead Time

The device contains a multi-feature fully integrated High Resolution PWM Generator which has complementary or push-pull output for maximum application flexibility.

Functional Description

The High Resolution 8-bit PWM circuit includes a PWM generator circuit, a delay lock loop circuit and PWM complementary outputs with dead time insertion. It can operate either in the Edge-aligned mode or Center-aligned mode. The device also provides the PWM duty adjusting control for high resolution PWM output.



H.R PWM Output Block Diagram

Note: 1. Symbols marked with an asterisk "*" are the internal signal name and not the Special Function Register bit.
Duty Buffer[11:0] are the calculated result of PWMDH/PWMDL or Auto-adjust system, which can be selected by the ADJSW bit.
DT1SEL[3:0] and DT2SEL[3:0] are calculated and selected automatically based on the DT[4:0] bits.
DT1CK is the OUTH DT reference signal.
DT2CK is the OUTL DT reference signal.
DLLCK is the H.R PWM reference signal.
2. H.R=High Resolution.
3. f_{PWM} is the DLL input frequency.
4. The OUTH and OUTL are not externally bounded but internally connected to the Gate-driver, the relevant control bits should be properly configured according to the application requirements.

High Resolution PWM Register Description

The basic operation of the High Resolution PWM is controlled using several registers. A PWM period register, PWMP, exists to store the desired 8-bit PWM period value. A 12-bit register pair, PWMDH/PWMDL, is used to store the PWM duty value and select the DLL circuit phase. The PWM function enable control, PWM counter clock selection, PWM counting mode selection, PWM output mode selection and PWM output signals inverting control are determined by the PWMC0

register. The dead time duration is determined by the PWMC1 register. The DLLC register is used for the DLL circuit enable control and the losing lock protection control.

There are also some registers used for the auto-adjust PWM function. The register ADJC is to control the auto-adjust function enable/disable, the PWM duty adjust operation and store the OUVP comparator output status. The ADJS register is to select the adjust steps when over voltage or under voltage condition occurs while the ADJDT is to select the auto-adjust function delay time after being triggered. The two register pairs, ADJMAXH/ADJMAXL and ADJMINH/ADJMINL, are used to set the maximum and minimum duty data. The register pair, ADJBH/ADJBL, is used to store the auto-adjust PWM buffer duty data.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PWMP	D7	D6	D5	D4	D3	D2	D1	D0
PWMDL	D7	D6	D5	D4	D3	D2	D1	D0
PWMDH	—	—	—	—	D11	D10	D9	D8
PWMC0	PWMCK1	PWMCK0	PWMON	PWMM	PWMOM	—	OUTHN	OUTLN
PWMC1	—	—	—	DT4	DT3	DT2	DT1	DT0
DLLC	DLLLKEN	DLLEN	—	—	—	—	—	DLLLKF
ADJDT	—	—	D5	D4	D3	D2	D1	D0
ADJS	OVPS3	OVPS2	OVPS1	OVPS0	UVPS3	UVPS2	UVPS1	UVPS0
ADJC	ADJEN	ADJV	ADJSW	—	OVPL	UVPL	—	—
ADJMAXL	D7	D6	D5	D4	D3	D2	D1	D0
ADJMAXH	—	—	—	—	D11	D10	D9	D8
ADJMINL	D7	D6	D5	D4	D3	D2	D1	D0
ADJMINH	—	—	—	—	D11	D10	D9	D8
ADJBL	D7	D6	D5	D4	D3	D2	D1	D0
ADJBH	—	—	—	—	D11	D10	D9	D8

High Resolution PWM Generator & Auto-adjust Register List

- **PWMP Register**

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

Bit 7~0 **D7~D0**: 8-bit PWM period register bit 7 ~ bit 0

In Edge-aligned mode:

PWM period=(PWMP[7:0]+1)/f_{PWM}

In Center-aligned mode:

PWM period=(PWMP[7:0]×2)/f_{PWM}

Note: The PWMP[7:0] setting must be greater than or equal to 03H in the Edge-aligned mode and 07H in the Center-aligned mode.

• **PWMDH & PWMDL Registers**

Register	PWMDH								PWMDL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8	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	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0	0	0	0	0	0	0	0	0

"—": Unimplemented, read as "0"

D11~D8: PWM duty high byte register bit 3 ~ bit 0

D7~D0: PWM duty low byte register bit 7 ~ bit 0

Note: 1. The PWMDH/PWMDL (PWMD) has 12 valid bits. D11~D4 are corresponding to the PWMP[7:0] bits, D3~D0 are corresponding to the DLL phase #0~#F Rising edge.

2. The PWMD value should meet the condition:

When DLLEN=1, DT[4:0]×2+1≤PWMD[11:0]≤PWMP[7:0]×16-DT[4:0]×2-1

When DLLEN=0, DT[4:3]+1≤PWMD[11:4]≤PWMP[7:0]-DT[4:3]-1

3. The output pulse width is calculated as follows:

When DLLEN=0:

P=PWMP[7:0], D=PWMD[11:4], DT=DT[4:3]

In the Edge-aligned mode, the pulse width time is obtained using the formula:

OUTH=(D+1-DT[4:3])/f_{PWM}

OUTL=(P-D-DT[4:3])/f_{PWM}

In the Center-aligned mode, the pulse width time is obtained using the formula:

OUTH=(2D-1-DT[4:3])/f_{PWM}

OUTL=[2(P-D)+1-DT[4:3]]/f_{PWM}

When DLLEN=1:

P=PWMP[7:0], D=PWMD[11:4], DLL=PWMD[3:0], DT=DT[4:0]×2

In the Edge-aligned mode, the pulse width time is obtained using the formula:

OUTH=[D+1+(DLL-DT[4:0]×2)/16]/f_{PWM}

OUTL=[P-D-(DLL+DT[4:0]×2)/16]/f_{PWM}

In the Center-aligned mode, the OUTH and OUTL after calculation should be greater than or equal to 4 f_{PWM} clocks.

• **PWMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PWMCK1	PWMCK0	PWMON	PWMM	PWMOM	—	OUTHN	OUTLN
R/W	R/W	R/W	R/W	R/W	R/W	—	R/W	R/W
POR	0	0	0	0	0	—	0	0

Bit 7~6 **PWMCK1~PWMCK0**: PWM counter clock source selection

00: f_H

01: $f_H/2$

10: $f_H/4$

11: $f_H/8$

Bit 5 **PWMON**: PWM function enable control

0: Disable, PWM counter=0

1: Enable

Bit 4 **PWMM**: PWM counting mode selection

0: Edge-aligned mode

1: Center-aligned mode (cannot be used when DLLEN=1)

Bit 3 **PWMOM**: PWM output mode selection

0: Complementary PWM output

1: Push-pull PWM output

Note: In the Push-pull PWM output mode, the OUTL pulse width is the same as OUTH.

Bit 2 Unimplemented, read as "0"

Bit 1 **OUTHN**: OUTH signal inverting control
0: Non-invert
1: Invert

Bit 0 **OUTLN**: OUTL signal inverting control
0: Non-invert
1: Invert

• **PWMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	DT4	DT3	DT2	DT1	DT0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	1	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **DT4~DT0**: PWM dead time selection
If DLLEN=1:
01000: Dead time is t_{DLL}×16 ~ t_{DLL}×17
…
11110: Dead time is t_{DLL}×60 ~ t_{DLL}×61
11111: Dead time is t_{DLL}×62 ~ t_{DLL}×63
If DLLEN=0:
01xxx: Dead time is t_{PWM}×1
10xxx: Dead time is t_{PWM}×2
11xxx: Dead time is t_{PWM}×3
This means that the lower 3 bit settings of DT[4:0] are invalid.
Note: 1. t_{DLL}=1/(f_{PWM}×16).
2. The DT[4:0] setting must be greater than or equal to 08H.

• **DLLC Register**

Bit	7	6	5	4	3	2	1	0
Name	DLLLKEN	DLLEN	—	—	—	—	—	DLLLKF
R/W	R/W	R/W	—	—	—	—	—	R/W
POR	1	0	—	—	—	—	—	0

Bit 7 **DLLLKEN**: DLL circuit Losing Lock protection function enable control
0: Disable
1: Enable
Note: When DLLLKEN=1 and DLL function is enabled, if a losing lock condition occurs, the DLLLKF bit is set high and the losing lock condition will be solved by DLL automatically. While when DLLLKEN=0, the DLLLKF bit is always zero even if a losing lock condition occurs, and the DLL will not solve the condition.

Bit 6 **DLLEN**: DLL function enable control
0: DLL disabled
1: DLL enabled

Bit 5~1 Unimplemented, read as "0"

Bit 0 **DLLLKF**: DLL circuit losing lock flag
0: No losing lock condition occurs
1: Losing lock occurs
This bit can be cleared to zero by software, but cannot be set high by software.
Note: When DLLLKEN=1 and DLL function is enabled, if no losing lock condition occurs, the DLLLKF bit is 0, if a losing lock condition occurs, the DLLLKF bit is set high which can only be cleared by software. If DLLLKEN=0, the DLLLKF bit is always zero.

• **ADJDT Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **D5~D0**: Auto-adjust PWM delay time selection
000000: Delay time=PWM Cycle×2
000001: Delay time=PWM Cycle×4
000010: Delay time=PWM Cycle×8
…
111111: Delay time=PWM Cycle×128

Delay time=(ADJDT[5:0]+1)×PWM Cycle×2

For example:

ADJDT=1 (4 cycles automatically adjust once)

When OVPL=1, the auto-adjust function will be enabled to trigger auto-adjust function, it is at Cycle 1 now. Then Cycle 5 needs to restart the auto-adjust function. The next auto-adjust function will start at Cycle 9 and so on until OVPL=0.

• **ADJS Register**

Bit	7	6	5	4	3	2	1	0
Name	OVPS3	OVPS2	OVPS1	OVPS0	UVPS3	UVPS2	UVPS1	UVPS0
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 **OVPS3~OVPS0**: OVP auto-adjust PWM duty steps selection
0000: 0 step
0001: 1 step
…
1111: 15 steps

Bit 3~0 **UVPS3~UVPS0**: UVP auto-adjust PWM duty steps selection
0000: 0 step
0001: 1 step
…
1111: 15 steps

• **ADJC Register**

Bit	7	6	5	4	3	2	1	0
Name	ADJEN	ADJV	ADJSW	—	OVPL	UVPL	—	—
R/W	R/W	R/W	R/W	—	R	R	—	—
POR	0	0	0	—	x	x	—	—

"x": unknown

Bit 7 **ADJEN**: Auto-adjust PWM duty function enable control
0: Disable
1: Enable

Bit 6 **ADJV**: Auto-adjust PWM duty action selection
0: OVPL increase duty, UVPL decrease duty
1: OVPL decrease duty, UVPL increase duty

Bit 5 **ADJSW**: PWM duty adjustment by S/W auto-adjust control
0: Disable, write into Duty Buffer from PWMDH+PWMDL registers by firmware
1: Enable, write into Duty Buffer by auto-adjust system

When the ADJEN is cleared to zero, the auto-adjust circuit is off, so this bit is always 0. When the ADJEN bit is set high, the auto-adjust circuit is on. Then if the OVPL or

UVPL is 1, this bit will be set high automatically to trigger the auto-adjust system to control the duty. When the auto duty adjustment is completed, the ADJSW bit should be cleared to zero manually. Note that only when the OVPL and UVPL bits are equal to 0, the ADJSW bit can be changed from 1 to 0 successfully. However, if the OVPL and UVPL bits are both 1, the auto-adjust system will not be triggered.

Bit 4 Unimplemented, read as "0"

Bit 3 **OVPL**: OVP comparator output status
0: Output low (no over voltage occurs)
1: Output high (over voltage occurs)

Bit 2 **UVPL**: UVP comparator output status
0: Output low (no under voltage occurs)
1: Output high (under voltage occurs)

Bit 1~0 Unimplemented, read as "0"

- **ADJMAXH & ADJMAXL Registers**

Register	ADJMAXH								ADJMAXL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8	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	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0	0	0	0	0	0	0	0	0

"—": Unimplemented, read as "0"

D11~D8: Auto-adjust PWM Maximum duty high byte

D7~D0: Auto-adjust PWM Maximum duty low byte

Note: D11~D4 are corresponding to the PWMD[11:4] bits, D3~D0 are corresponding to the PWMD[3:0] bits.

- **ADJMINH & ADJMINL Registers**

Register	ADJMINH								ADJMINL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8	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	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0	0	0	0	0	0	0	0	0

"—": Unimplemented, read as "0"

D11~D8: Auto-adjust PWM Minimum duty high byte

D7~D0: Auto-adjust PWM Minimum duty low byte

Note: D11~D4 are corresponding to the PWMD[11:4] bits, D3~D0 are corresponding to the PWMD[3:0] bits.

- **ADJBH & ADJBL Registers**

Register	ADJBH								ADJBL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8	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	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0	0	0	0	0	0	0	0	0

"—": Unimplemented, read as "0"

D11~D8: Auto-adjust PWM Buffer duty high byte

D7~D0: Auto-adjust PWM Buffer duty low byte

Note: D11~D4 are corresponding to the PWMD[11:4] bits, D3~D0 are corresponding to the PWMD[3:0] bits.

PWM Generator

The PWM signal generator is driven by the HIRC clock and can generate a PWM signal, with a variable duty and period cycles by configuring the PWMP and PWMDH/PWMDL registers. The PWM signal period is dependent upon the PWM counter clock source which is set by the PWMCK[1:0] bits in the PWMC0 register and determined by the PWMP register. The PWM signal duty cycle is set by the PWMDH/PWMDL registers. The DLL can generate 16 phase outputs in a counter frequency cycle to fine tune the PWM output to achieve a 4-bit increment in the PWM duty resolution. The PWMC0 register can set the PWM count clock frequency, PWM enable/disable, PWM operating in edge-aligned/center-aligned mode and PWM output as complementary/push-pull output. The PWMC1 register can be used to set the PWM output dead time width.

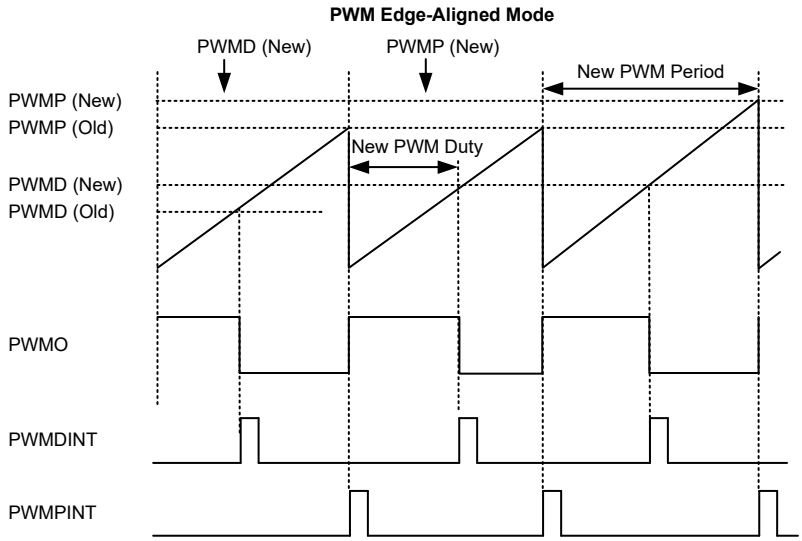
The PWM output frequency and duty are set by writing the PWMP and PWMDL/PWMDH registers. The PWMP is an 8-bit register, and the PWMDH/PWMDL (PWMD) are a 12-bit valid register pair, where the PWMP[7:0] bits are aligned with the PWMD[11:4] bits, the PWM output period and duty cycle time are determined by the f_{PWM}, and the PWMD[3:0] bits are used to set the DLL phase to fine tune the PWM output to improve the PWM duty adjustment resolution.

It should be noted that the written PWMDH/PWMDL contents will first be placed into the Duty Buffer and updated only when a new PWM period occurs. In addition, the current PWM output duty can be obtained by reading the ADJBH/ADJBL registers.

PWM Counting Modes

The PWM has two counting modes, known as Edge-aligned mode and Center-aligned mode, which can be selected by the PWMM bit.

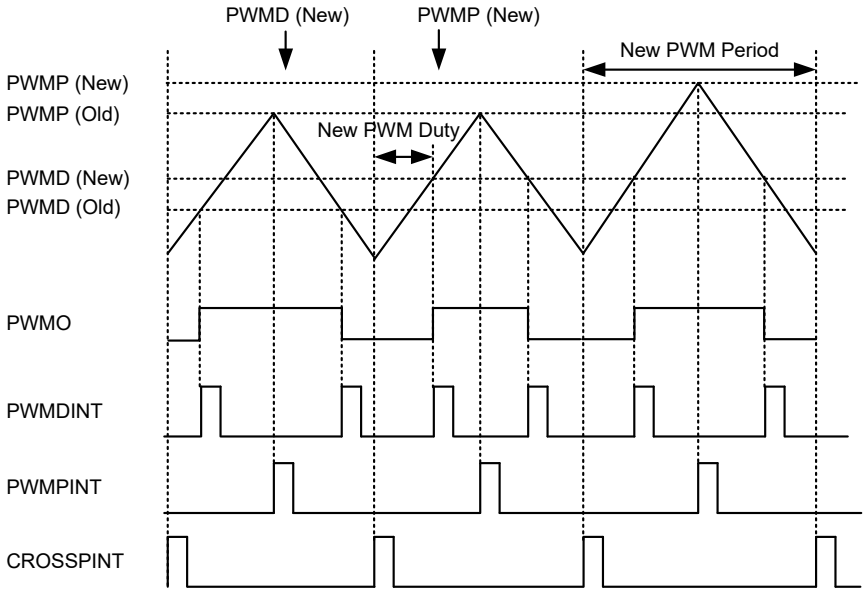
In the edge-aligned mode, the counter counts up until it is the same as the PWMP value, at which point it is cleared to zero. It can count up again at the next PWM clock trigger. In the edge-aligned mode, the PWM provides two interrupt signals, namely PWMDINT and PWMPINT. The PWMDINT interrupt signal will be triggered when the PWM counter value is the same as the PWMD value. When the PWM counter value is equal to the PWMP value, the PWM counter will be cleared and the PWMPINT interrupt signal will be triggered. In the edge-aligned mode, the CROSS interrupt enable bit should be turned off to avoid unpredictable behavior. The edge-aligned PWM timing diagram is shown as follows:



In the center-aligned mode, the counter counts up until it is the same as the PWMP value, and

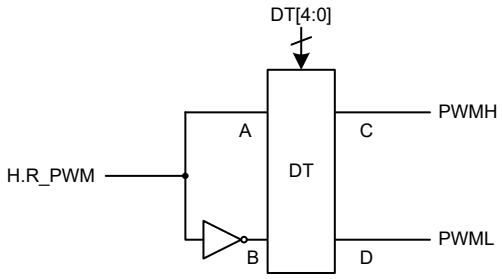
changes to count down until 0 at the next PWM clock trigger. After the counter reaches 0, it changes to count up again at the next PWM clock trigger. In the center-aligned mode, the PWM provides three interrupt signals, namely PWMDINT, PWMPINT and CROSSINT. When the PWM counter is the same as the PWMD value, the PWMDINT interrupt signal is triggered. When the PWM counter value is the same as the PWMP value, the PWMPINT interrupt signal is triggered. The CROSSINT interrupt signal is triggered when the PWM counter value is equal to 0. The center-aligned PWM timing diagram is shown as follows:

PWM Center-Aligned Mode

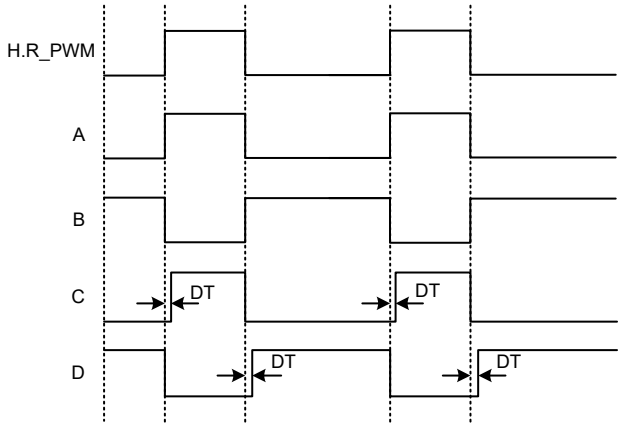


Dead Time Insert

The device provides a complementary output pair of signals which can be used as a PWM driver signal. The signal is sourced from the High Resolution PWM output signal, 8-bit PWM with DLL circuit. PWM output is an active high signal. The dead time generator is programmable using the DT[4:0] bits in the PWMC1 register, and a dead time will be inserted to prevent excessive DC currents. The dead time will be inserted whenever the rising edge of the dead time generator input signal occurs.



Complementary PWM Output with Dead Time Control



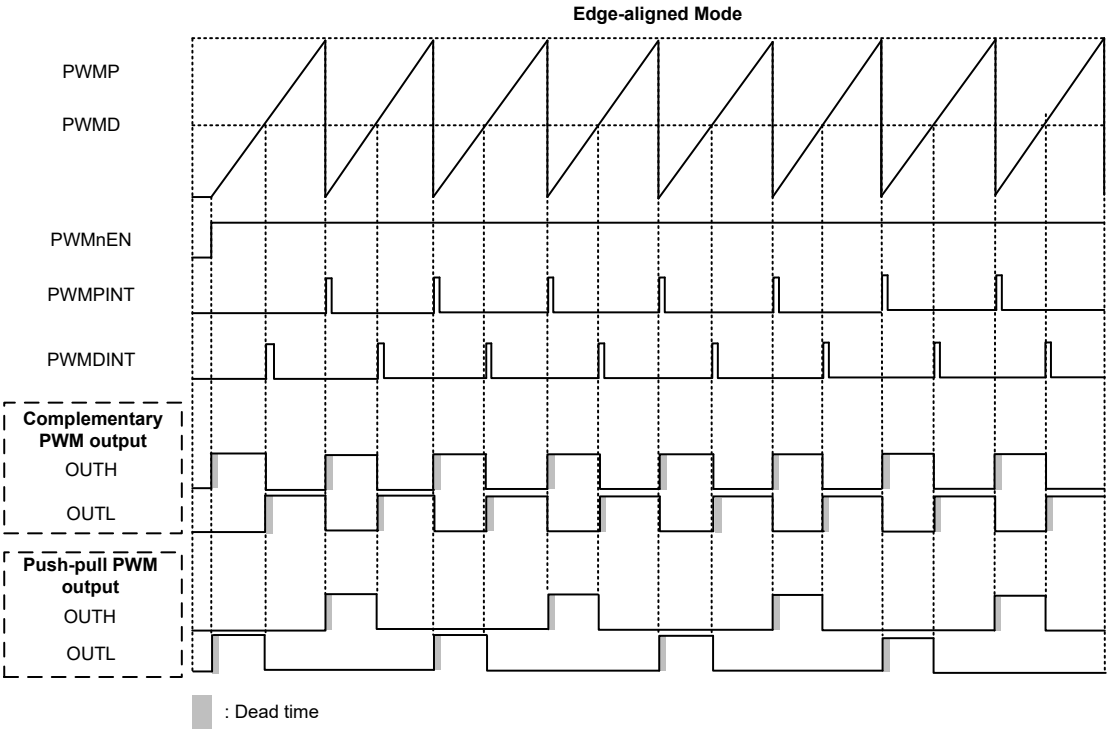
Note: C and D are the complementary PWM control with dead time circuitry's output signals.

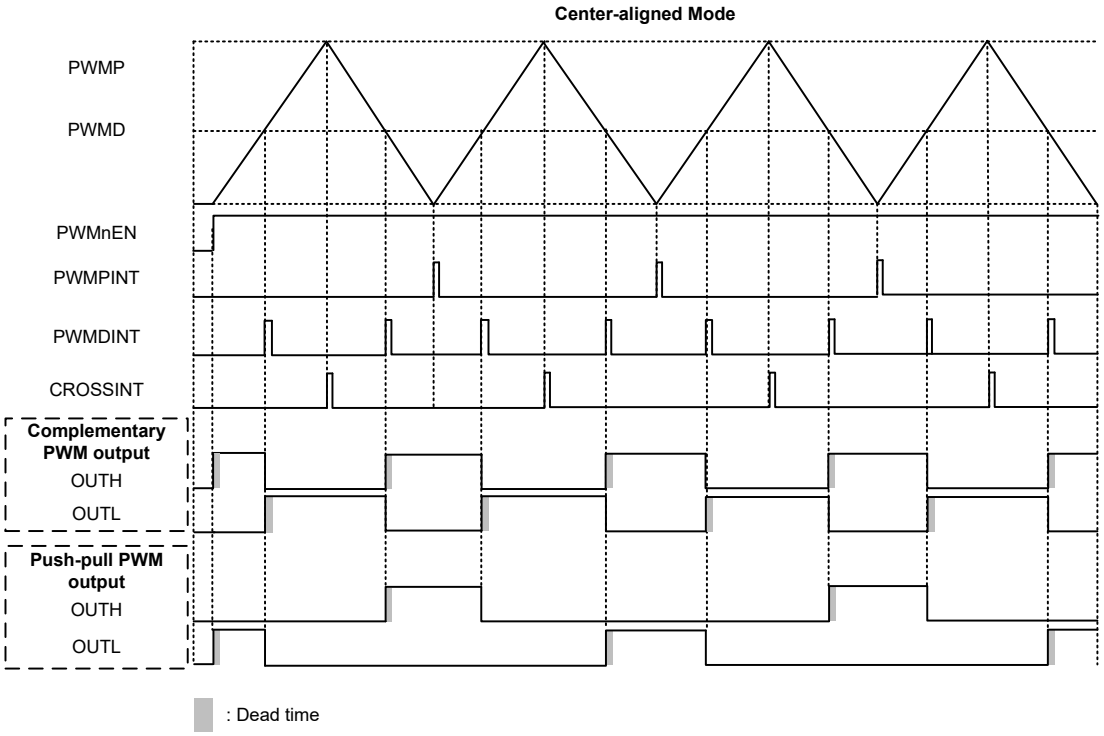
Dead Time Insertion Timing

Output Mode Selection

The PWM has two output modes, known as complementary output and push-pull output mode, which can be selected by the PWMOM bit.

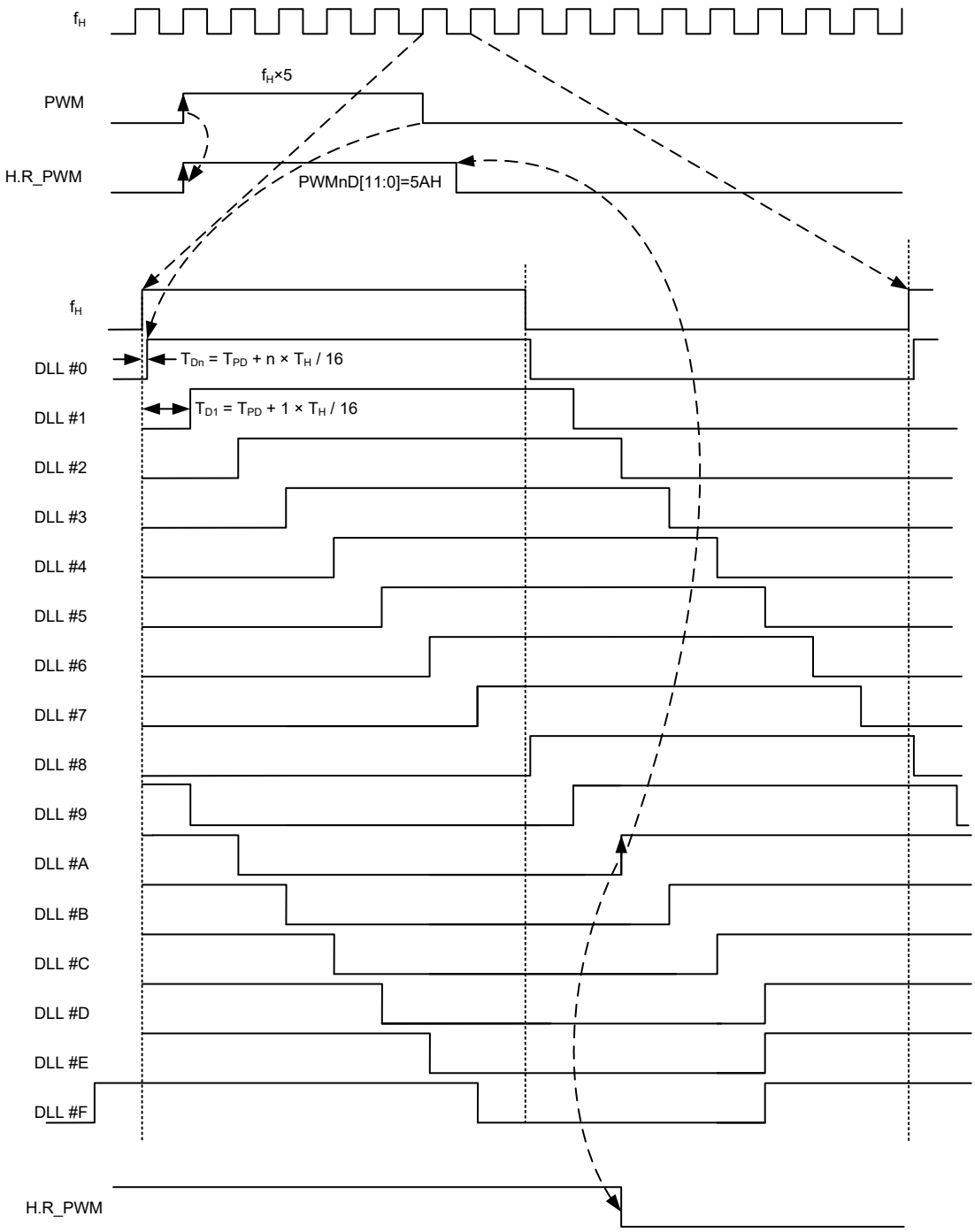
The complementary output means that the OUTL and OUTH output a PWM signal simultaneously, and the OUTL is the inverted signal of the OUTH, only in the dead time both are low at the same time. The push-pull output means that the OUTL and OUTH output a PWM signal alternately, and the alternating frequency is related to the PWM output frequency. The timing reference is shown as follows:





Delay Lock Loop

DLL is an abbreviation for Delay Lock Loop. The DLL can generate 16 phase outputs within one HIRC clock period. The 16 phase outputs are used to fine tune the PWM signal output. The PWM clock is f_{HIRC}, which means that the PWM output duty resolution is $1/f_{HIRC}$. The PWM signal passes through the DLL phase selection and PWM control which is set by the PWMD[3:0] bits in the DLL register circuit to output a fine-tuned PWM signal, H.R_PWM with the PWM duty resolution increased by 4 bits. (DLL #D/#E/#F output time is the same)



Losing Lock Protection

The device also provides the Losing lock protection circuit which can be enabled by the DLLLKEN bit.

If the MCU is disturbed, a losing lock error that the phase time generated by the DLL is 1.5 times of the normal phase time may occur. If the DLLLKEN bit is high, the losing lock circuit is enabled and then the phase time will return normal in 30μs seconds. Additionally the flag bit DLLLKF which is 0 in normal operation will be set high to notify users that a losing lock error occurred. If

the DLLLKEN bit is not set high, the Losing lock protection circuit is off. So after a losing lock condition occurs, the DLL phase time cannot return normal automatically and the DLLLKF flag is always 0.

Auto-adjust Circuit

In order to increase the Push-pull Boost response speed, the device provides an auto-adjust circuit together with the PWM generator. The following summarises the steps to implement the auto-adjust function.

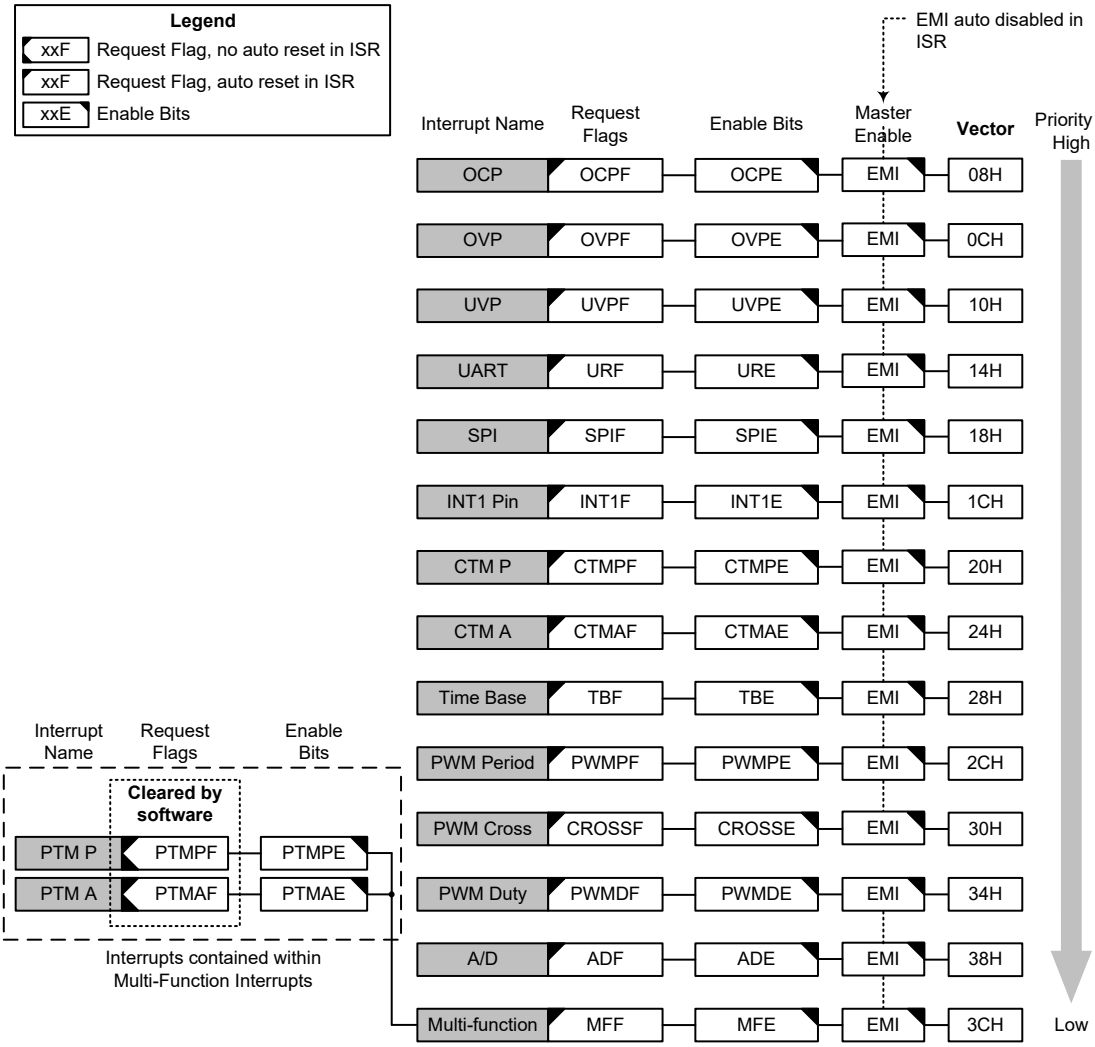
- Step 1. Clear the ADJEN bit to zero to turn off the auto-adjust circuit for initialization:
  - Set the Maximum/Minimum Duty by programming the 12-bit ADJMAXH/ADJMAXL and ADJMINH/ADJMINL registers.

    Note: DT[4:0]+1<Min<PWMD[11:0]<Max<PWMP×16-DT[4:0]-1
  - Configure the ADJDT[5:0] bits to set the delay time after every trigger.
  - Set the ADJV bit in the ADJC register to select the duty adjust action (increase or decrease).
  - Select the duty adjusting step (0~15) when OUVP occurs by setting the ADJS register.
  - Set the OVP and UVP limited Voltage. Note: UVP<Target Voltage<OVP
- Step 2. PWM Start:
  - PWM initialization, including PWMP and PWMDH/PWMDL initialization
  - Set the PWMON bit high to enable the PWM generator.
  - Adjust the Duty after reading the OUVP output, make sure the output voltage equal to the target voltage.
- Step 3. Set the ADJEN bit high to enable the auto-adjust circuit, then the OVPL/UVPL level can trigger the auto-adjustment function.
- Step 4. If an UVP/OVP interrupt occurs, the firmware can determine whether a firmware intervention is required.
  - When ADJEN=1 & ADJSW=0, Duty Buffer=PWMDH+PWMDL, that is, the duty buffer is written from the PWMDH+PWMDL registers by firmware.
  - When ADJEN=0, the auto-adjust is turned off.
  - When PWMON=1, if the duty buffer is written, the PWMDH and PWMDL data will be updated after the PWM counter is cleared to zero.

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 INT1 pin, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, SPI, UART and the A/D converter, etc.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector.



Interrupt Structure

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the

accompanying table. The registers fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INT1 Pin	INT1E	INT1F	—
Over Circuit Protection	OCPE	OCPF	—
Over Voltage Protection	OVPE	OVPF	—
Under Voltage Protection	UVPE	UVPF	—
UART	URE	URF	—
SPI	SPIE	SPIF	—
Time Base	TBE	TBF	—
PWM	PWMPE	PWMPF	—
	CROSSE	CROSSF	—
	PWMDE	PWMDF	—
A/D Converter	ADE	ADF	—
Multi-function	MFE	MFF	—
CTM	CTMPE	CTMPF	—
	CTMAE	CTMAF	—
PTM	PTMPE	PTMPF	—
	PTMAE	PTMAF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	D1	D0
INTC0	—	OVPF	OCPF	D4	OVPE	OCPE	D1	EMI
INTC1	INT1F	SPIF	URF	UVPF	INT1E	SPIE	URE	UVPE
INTC2	PWMPF	TBF	CTMAF	CTMPF	PWMPE	TBE	CTMAE	CTMPE
INTC3	MFF	ADF	PWMDF	CROSSF	MFE	ADE	PWMDE	CROSSE
MFI	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE

Interrupt Register List

- **INTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	D1	D0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 1~0 **D1~D0**: Reserved bits, should remain unchanged after power-on reset

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	OVPF	OCPF	D4	OVPE	OCPE	D1	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 **OVPF**: OVP interrupt request flag
0: No request
1: Interrupt request

Bit 5 **OCPF**: OCP interrupt request flag
0: No request
1: Interrupt request

Bit 4 **D4**: Reserved bit, should remain unchanged after power-on reset

Bit 3 **OVPE**: OVP interrupt control
0: Disable
1: Enable

Bit 2 **OCPE**: OCP interrupt control
0: Disable
1: Enable

Bit 1 **D1**: Reserved bit, should remain unchanged after power-on reset

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

• **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	INT1F	SPIF	URF	UVPF	INT1E	SPIE	URE	UVPE
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 **INT1F**: INT1 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **SPIF**: SPI interrupt request flag
0: No request
1: Interrupt request

Bit 5 **URF**: UART transfer interrupt request flag
0: No request
1: Interrupt request

Bit 4 **UVPF**: UVP interrupt request flag
0: No request
1: Interrupt request

Bit 3 **INT1E**: INT1 interrupt control
0: Disable
1: Enable

Bit 2 **SPIE**: SPI interrupt control
0: Disable
1: Enable

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

Bit 0 **UVPE**: UVP interrupt control
0: Disable
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	PWMPF	TBF	CTMAF	CTMPF	PWMPE	TBE	CTMAE	CTMPE
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 **TBF**: Time Base interrupt request flag
0: No request
1: Interrupt request

Bit 5 **CTMAF**: CTM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTMPF**: CTM Comparator P match interrupt request flag
0: No request
1: Interrupt request

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

Bit 2 **TBE**: Time Base interrupt control
0: Disable
1: Enable

Bit 1 **CTMAE**: CTM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **CTMPE**: CTM Comparator P match interrupt control
0: Disable
1: Enable

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	MFF	ADF	PWMDF	CROSSF	MFE	ADE	PWMDE	CROSSE
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 **MFF**: Multi-function interrupt request flag
0: No request
1: Interrupt request

Bit 6 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 5 **PWMDF**: PWM duty match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CROSSF**: PWM CROSS interrupt request flag
0: No request
1: Interrupt request

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

Bit 2 **ADE**: A/D Converter interrupt control
0: Disable
1: Enable

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

Bit 0 **CROSSE**: PWM CROSS interrupt control
0: Disable
1: Enable

• **MFI Register**

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

Bit 7~6 Unimplemented, read as "0"

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

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

Bit 3~2 Unimplemented, read as "0"

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

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

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.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the

interrupt will not be immediately serviced, the request flag will still be recorded.

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

External Interrupts

The external interrupts are controlled by signal transitions on the pin INT1. An external interrupt request will take place when the external interrupt request flag, INT1F, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to their respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register 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 pins, a subroutine call to the corresponding external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

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

Over Current Protection Interrupt

The OCP interrupt is controlled by detecting the OCP input current. An OCP interrupt request will take place when the OCP interrupt request flag, OCPF, is set, which occurs when an over current condition is detected. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and OCP interrupt enable bit, OCPE, must first be set. When the interrupt is enabled, the stack is not full and an over current condition is detected, a subroutine call to the OCP interrupt vector, will take place. When the interrupt is serviced, the OCP interrupt flag, OCPF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Over Voltage Protection Interrupt

The OVP interrupt is controlled by detecting the OVP input voltage. An OVP interrupt request will take place when the OVP interrupt request flag, OVPF, is set, which occurs when the over voltage protection circuit detects an over voltage condition. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and OVP interrupt enable bit, OVPE, must first be set. When the interrupt is enabled, the stack is not full and an over voltage condition is detected, a subroutine call to the OVP interrupt vector, will take place. When the interrupt is serviced, the OVP interrupt flag, OVPF, will be automatically cleared. The EMI bit will also be

automatically cleared to disable other interrupts.

Under Voltage Protection Interrupt

The UVP interrupt is controlled by detecting the UVP input voltage. An UVP interrupt request will take place when the UVP interrupt request flag, UVPF, is set, which occurs when the under voltage protection circuit detects an under voltage condition. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and UVP interrupt enable bit, UVPE, must first be set. When the interrupt is enabled, the stack is not full and an under voltage condition is detected, a subroutine call to the UVP interrupt vector, will take place. When the interrupt is serviced, the UVP interrupt flag, UVPF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

UART Interrupt

The UART Interrupt is controlled by several UART transfer conditions. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and UART Interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of the conditions described above occurs, a subroutine call to the corresponding UART Interrupt vector, will take place. When the interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

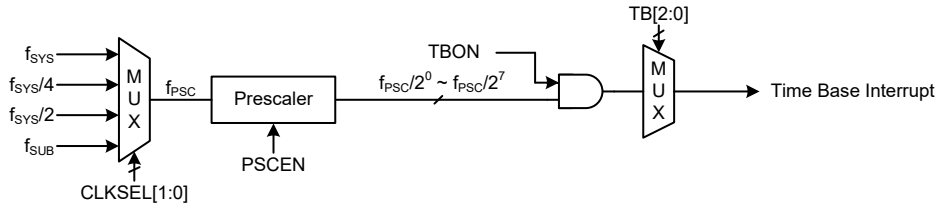
SPI Interrupt

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

Time Base Interrupt

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

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, f_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$, $f_{SYS}/2$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL1~CLKSEL0 bits in the PSCR register.



Time Base Interrupt

- **PSCR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PSCEN	CLKSEL1	CLKSEL0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **PSCEN**: Prescaler clock enable control
0: Disable
1: Enable
The PSCEN bit is the Prescaler clock enable or disable control bit. When the Prescaler clock is disabled, it can reduce extra power consumption.

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source selection
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/2$
11: f_{SUB}

- **TBC Register**

Bit	7	6	5	4	3	2	1	0
Name	TBON	—	—	—	—	TB2	TB1	TB0
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TBON**: Time Base enable control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB2~TB0**: Time Base Time-out Period selection
000: $2^0/f_{PSC}$
001: $2^1/f_{PSC}$
010: $2^2/f_{PSC}$
011: $2^3/f_{PSC}$
100: $2^4/f_{PSC}$
101: $2^5/f_{PSC}$
110: $2^6/f_{PSC}$
111: $2^7/f_{PSC}$

PWM Interrupts

The High Resolution PWM circuit has three interrupts, a PWM period match interrupt, a PWM duty match interrupt and a PWM CROSS interrupt. A PWM interrupt request will take place when any of the PWM interrupt request flags, PWMPF, PWMDF or CROSSF, are set, which occurs when the PWM period or duty matches or the PWM counter is cleared to zero. To allow the program to branch to their respectively interrupt vector addresses, the global interrupt enable bit, EMI, and the corresponding PWM interrupt enable bit, PWMPE, PWMDE or CROSSE, must first be set. When

the interrupt is enabled, the stack is not full and the PWM period or duty matches or the PWM counter is cleared to zero, a subroutine call to the respective PWM interrupt vector will take place. When the interrupt is serviced, the relevant PWM interrupt request flags will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

A/D Converter Interrupt

The A/D Converter interrupt is controlled by the termination of an A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter Interrupt vector will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Multi-function Interrupts

Within the device there is a Multi-function interrupt. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts.

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

However, it must be noted that, although the Multi-function Interrupt 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.

TM Interrupts

The Compact and Periodic Type TMs each have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. The Compact Type TM interrupts are individual interrupts and the Periodic Type TM interrupts are contained within the Multi-function Interrupt. For all of the TM types there are two interrupt request flags and two enable control bits.

A CTM interrupt request will take place when any of the CTM request flags are set, a situation which occurs when a CTM 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 and respective CTM Interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and a CTM comparator match situation occurs, a subroutine call to the relevant CTM Interrupt vector locations, will take place. When the interrupt is serviced, the CTM interrupt request flags will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

A PTM interrupt request will take place when any of the PTM request flags together with the associated Multi-function interrupt request flag are set, a situation which occurs when a PTM 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 PTM Interrupt enable

bit, and relevant Multi-function Interrupt enable bit, MFE, must first be set. When the interrupt is enabled, the stack is not full and a PTM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector locations, will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the related MFF flag will be automatically cleared. As the PTM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

Interrupt Wake-up Function

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

Programming Considerations

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

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

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

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

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

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

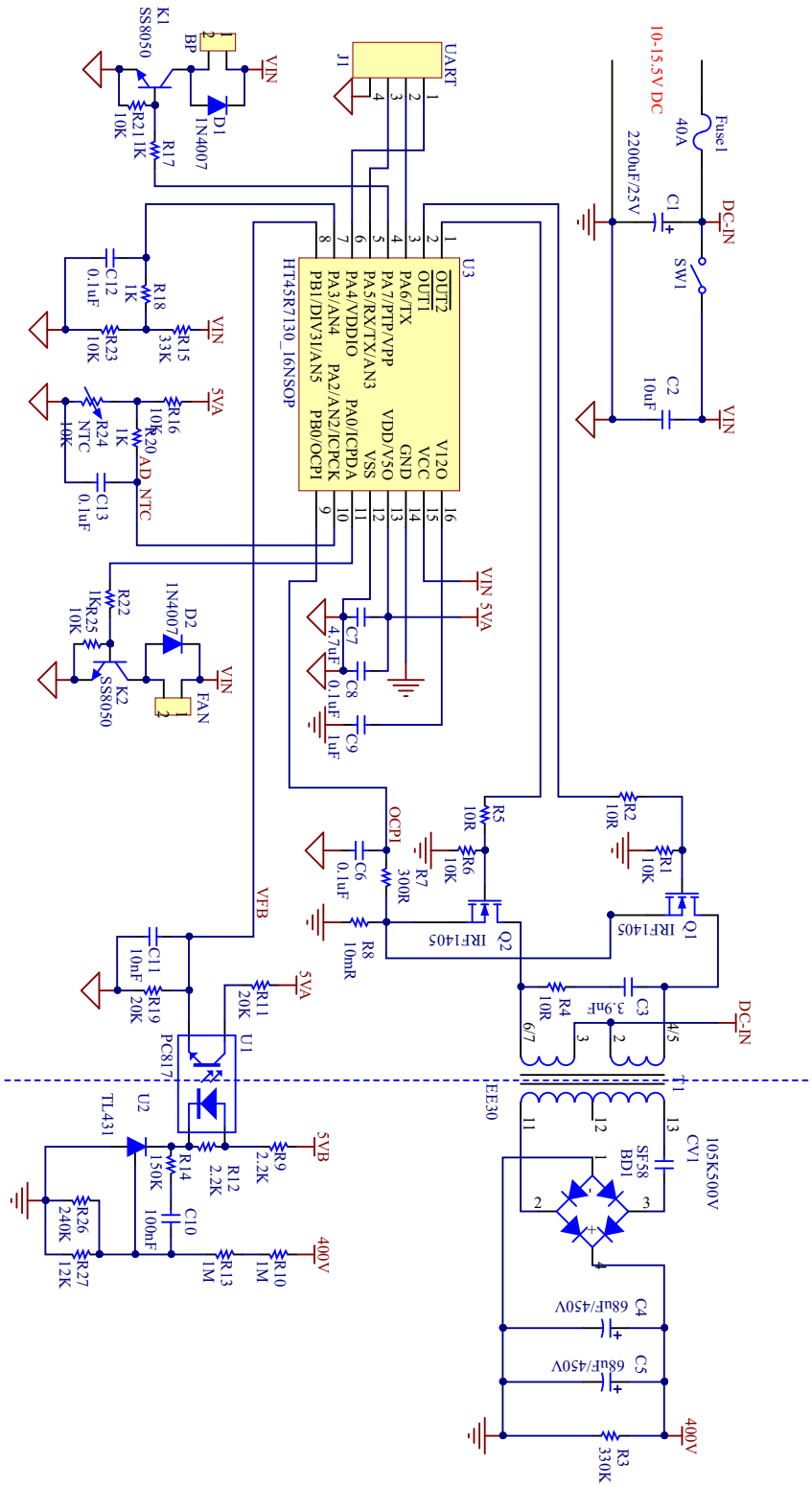
Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Option	
1	HIRC Frequency Selection – f_{HIRC}: 8MHz, 12MHz or 16MHz

Note: When the HIRC has been configured at a frequency shown in this table, the HIRC1 and HIRC0 bits should also be setup to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

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

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

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC ″AND″ x

Affected flag(s) Z

ANDM A,[m]	Logical AND ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″AND″ [m]
Affected flag(s)	Z

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None

CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None

CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None

CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	[m] ← $\overline{[m]}$
Affected flag(s)	Z

CPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC ← $\overline{[m]}$
Affected flag(s)	Z

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

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

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

Affected flag(s): C

DEC [m] Decrement Data Memory

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

Operation: [m] ← [m] − 1

Affected flag(s): Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation: ACC ← [m] − 1

Affected flag(s): Z

HALT Enter power down mode

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

Operation: TO ← 0
PDF ← 1

Affected flag(s): TO, PDF

INC [m] Increment Data Memory

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

Operation: [m] ← [m] + 1

Affected flag(s): Z

INCA [m] Increment Data Memory with result in ACC

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

Operation: ACC ← [m] + 1

Affected flag(s): Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None

RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None

RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None

RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← [m].7
Affected flag(s)	None

RLA [m]	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← [m].7
Affected flag(s)	None

RLC [m]	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← C C ← [m].7
Affected flag(s)	C

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

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

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

Affected flag(s): C

RR [m] Rotate Data Memory right

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

Operation: $ACC \leftarrow ACC - [m] - \overline{C}$

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

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

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

Operation $[m] \leftarrow ACC - [m] - \overline{C}$

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

SDZ [m] Skip if decrement Data Memory is 0

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

Operation $[m] \leftarrow [m] - 1$
Skip if [m]=0

Affected flag(s) None

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

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

Operation $ACC \leftarrow [m] - 1$
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation $[m].i \leftarrow 1$

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation $[m] \leftarrow [m] + 1$
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

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

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

Affected flag(s) None

TABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

XORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″XOR″ [m]
Affected flag(s)	Z

XOR A,x	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ x
Affected flag(s)	Z

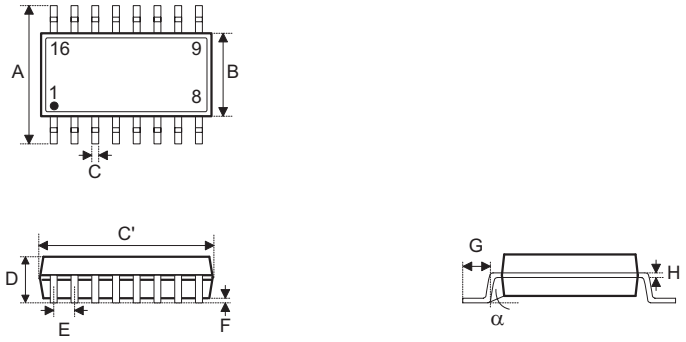
Package Information

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

16-pin NSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.012	—	0.020
C’		0.390 BSC	
D	—	—	0.069
E		0.050 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		6.00 BSC	
B		3.90 BSC	
C	0.31	—	0.51
C’		9.90 BSC	
D	—	—	1.75
E		1.27 BSC	
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

Copyright© 2026 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

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