Half-Bridge Induction Cooker Flash MCU

HT45F0074A

Revision: V1.10 Date: March 15, 2024

www.holtek.com

Table of Contents

Features

CPU Features

- Operating Voltage
  - f_{SYS}=16MHz: 4.5V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 32MHz/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 to three instruction cycles
- Table read instructions
- 115 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 8K×16
- RAM Data Memory: 1024×8
- True EEPROM Memory: 512×8
- Watchdog Timer function
- Up to 20 bidirectional I/O lines
- Two external interrupt lines with digital debounce function, pin-shared with I/Os
- PCK external pin for peripherial clock output
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output or single pulse output functions
- Universal Serial Interface Module – USIM for SPI, I^2C or UART communication
- Dual Time-Base functions for generation of fixed time interrupt signals
- Cyclic Redundancy Check Unit – CRC
- Multiplier/Divider Unit – MDU
- 12-bit PWM generator and frequency jittering functions
  - A group of complementary PWM output
  - Programmable dead time control
  - 10-bit TMC for time measurement and phase signal capture measurement
  - Phase detection and protection mechanism
  - PWM auto-adjust and PWM auto-shutdown for error signal protection
  - Hardware and software frequency jittering

- 12-bit resolution A/D converter with internal reference voltage V_{BG}
  - 11 external analog input channels – AN0~AN10
  - Internal analog input channels for internal OPA output and reference voltage measurements
  - Supports 1-channel/2-channel automatic conversion mode
  - Various automatic conversion start trigger sources, programmable leading edge blanking and start delay time
- Nine groups of over-voltage protection circuit for phase detection, IGBT over-current protection, IGBT short-circuit protection, AC over-current protection, AC voltage zero-crossing detection
- Operational amplifier with programmable gain
- Low Voltage Reset function
- Low Voltage Detect function
- Package types: 16-pin NSOP, 20/24-pin SOP

General Description

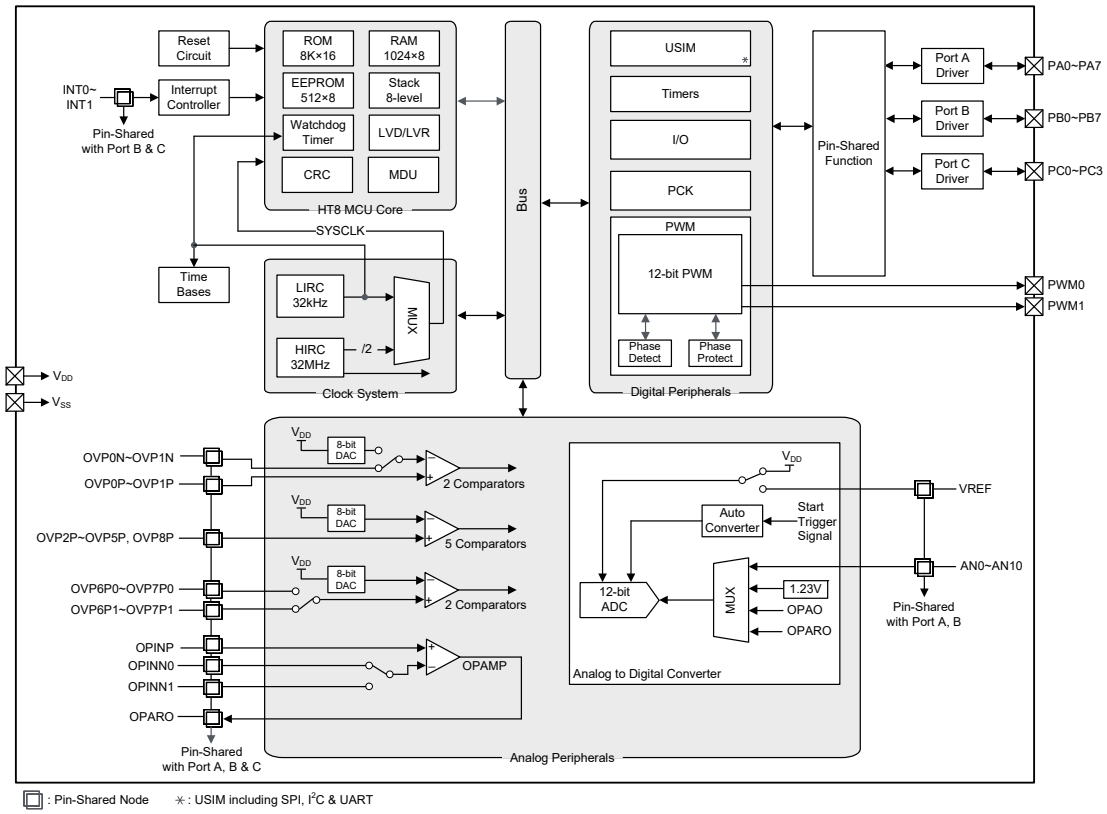
The HT45F0074A is a Flash Memory type 8-bit high performance RISC architecture microcontroller especially designed for half-bridge induction cooker applications. For power control requirements, the device provides a complete hardware protection mechanism which includes over-current protection as well as surge and phase protection. The device includes a fully integrated 12-bit A/D converter supporting a 2-channel automatic conversion function, which can be used for significant parameter measurements such as the induction cooker voltage and current values. The device is aimed at implementing the essential functions in half-bridge induction cooker applications and offers the benefits of a greatly reduced number of external components and smaller PCB areas.

The device provides an integrated PWM generator providing complementary outputs to implement the phase detection at different operating points. As the operating frequency affects the current value, this can achieve power control for half-bridge induction cookers with logic controls. The complementary PWM outputs have an adjustable frequency range of 7.8kHz~16MHz with an adjustable frequency unit of 1/32MHz, a 12-bit resolution Duty and a 12-bit programmable dead-time of 31.25ns~128μs. The complementary output period can be adjusted individually. These features combine to make the device especially suitable for half-bridge induction cooker applications. When the operating frequency is in the resonant capacitor region, the PWM period and duty will be adjusted automatically by the hardware according to user settings, pushing the operating frequency into the resonant inductor region.

The device also provides a set of operational amplifier and nine sets of over-voltage protection circuits. Here the operational amplifier is used to measure the resonant current for the power calculation. Two groups of OVP circuits are used for phase detection, one OVP circuit is used for AC over-current protection, two OVP circuits are used for IGBT over-current protection, two OVP circuits are used for IGBT short-circuit current protection, and the remaining two groups of OVP circuits are for control of the AC voltage zero-crossing detection (taking the peak voltage) and other circuit detection applications.

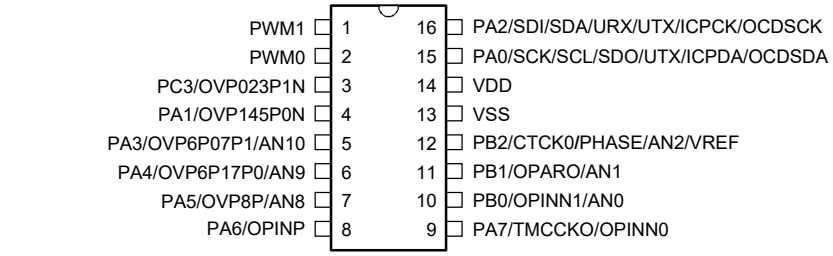
The device meets with all the mainstream specifications for half-bridge induction cookers. More details about application development information is provided in the Application Description section or visit Holtek website to find induction cooker Application Notes.

Block Diagram

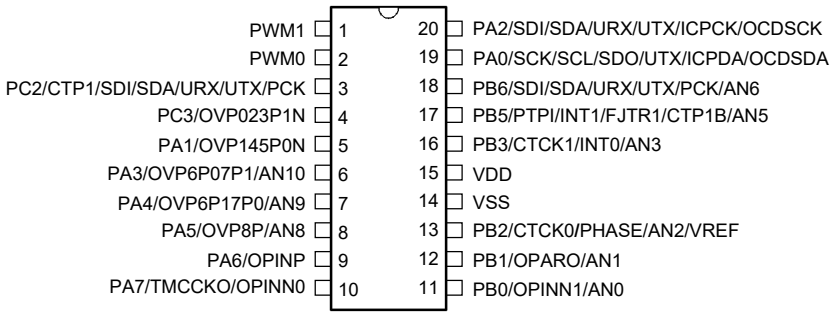


: Pin-Shared Node ✳ : USIM including SPI, I²C & UART

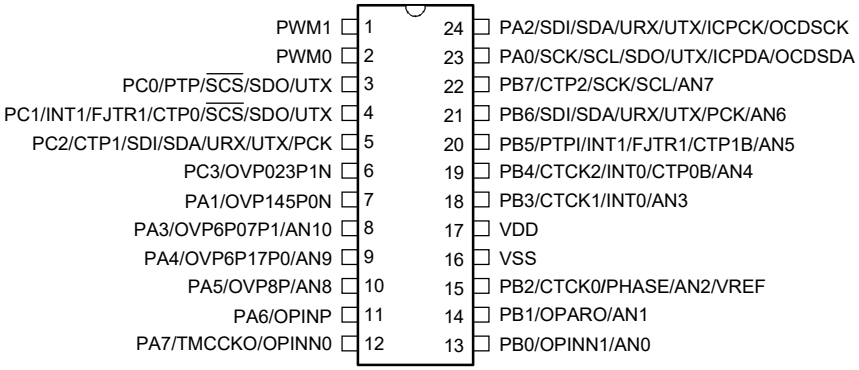
Pin Assignment



HT45F0074A/HT45V0074A
16 NSOP-A



HT45F0074A/HT45V0074A
20 SOP-A



HT45F0074A/HT45V0074A
24 SOP-A

Note: 1. If the pin-shared pin functions have multiple outputs simultaneously, the desired pin-shared function is determined by the corresponding software control bits.
2. For the unbounded pins, the line status should be properly configured to avoid unwanted current consumption resulting from floating input conditions. Refer to the "Standby Current Considerations" and "Input/Output Ports" sections.
3. Pin functions OCDSCK and OCDSDA which are pin-shared with PA2 and PA0 are only available in the OCDS EV device HT45V0074A.

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. Note that the pin description refers to the largest package size, as a result some pins may not exist on smaller package types.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/SCK/SCL/SDO/UTX/ ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	SCK	PAS0 IFS	ST	CMOS	SPI serial clock pin
	SCL	PAS0 IFS	ST	NMOS	I^2C clock line
	SDO	PAS0	—	CMOS	SPI serial data output pin
	UTX	PAS0	—	CMOS	UART transmitter pin
	ICPDA	—	ST	CMOS	ICP Address/Data pin
	OCDSDA	—	ST	CMOS	OCDS Address/Data pin, for EV chip only
PA1/OVP145P0N	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	OVP145P0N	PAS0	AN	—	OVP1, OVP4 and OVP5 non-inverting input pin; OVP0 inverting input pin

Pin Name	Function	OPT	I/T	O/T	Description
PA2/SDI/SDA/URX/UTX/ ICPCK/OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	SDI	PAS0 IFS	ST	—	SPI serial data input pin
	SDA	PAS0 IFS	ST	NMOS	I^2C data line
	URX/UTX	PAS0 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/OVP6P07P1/AN10	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	OVP6P07P1	PAS0	AN	—	OVP6 and OVP7 non-inverting input pin
	AN10	PAS0	AN	—	A/D converter external input channel 10
PA4/OVP6P17P0/AN9	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	OVP6P17P0	PAS1	AN	—	OVP6 and OVP7 non-inverting input pin
	AN9	PAS1	AN	—	A/D converter external input channel 9
PA5/OVP8P/AN8	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	OVP8P	PAS1	AN	—	OVP8 non-inverting input pin
	AN8	PAS1	AN	—	A/D converter external input channel 8
PA6/OPINP	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	OPINP	PAS1	AN	—	OPAMP non-inverting input pin
PA7/TMCCKO/OPINN0	PA7	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	TMCCKO	PAS1	—	CMOS	10-bit TMC TMCCKO clock signal output
	OPINN0	PAS1	AN	—	OPAMP inverting input pin 0
PB0/OPINN1/AN0	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	OPINN1	PBS0	AN	—	OPAMP inverting input pin 1
	AN0	PBS0	AN	—	A/D converter external input channel 0
PB1/OPARO/AN1	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	OPARO	PBS0	—	AN	OPAMP output pin
	AN1	PBS0	AN	—	A/D converter external input channel 1
PB2/CTCK0/PHASE/AN2/ VREF	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTCK0	PBS0	ST	—	CTM0 clock input pin
	PHASE	PBS0	—	CMOS	PHASE signal (phase detect circuit) output
	AN2	PBS0	AN	—	A/D converter external input channel 2
	VREF	PBS0	AN	—	A/D converter reference voltage input

Pin Name	Function	OPT	I/T	O/T	Description
PB3/CTCK1/INT0/AN3	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTCK1	PBS0	ST	—	CTM1 clock input pin
	INT0	PBS0 IFS	ST	—	External interrupt 0
	AN3	PBS0	AN	—	A/D converter external input channel 3
PB4/CTCK2/INT0/CTP0B/ AN4	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTCK2	PBS1	ST	—	CTM2 clock input pin
	INT0	PBS1 IFS	ST	—	External interrupt 0
	CTP0B	PBS1	—	CMOS	CTM0 inverted output pin
	AN4	PBS1	AN	—	A/D converter external input channel 4
PB5/PTPI/INT1/FJTR1/ CTP1B/AN5	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTPI	PBS1	ST	—	PTM capture input pin
	INT1	PBS1 IFS	ST	—	External interrupt 1
	FJTR1	PBS1 IFS	ST	—	FJTIMER input 1
	CTP1B	PBS1	—	CMOS	CTM1 inverted output pin
	AN5	PBS1	AN	—	A/D converter external input channel 5
PB6/SDI/SDA/URX/UTX/ PCK/AN6	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SDI	PBS1 IFS	ST	—	SPI serial data input pin
	SDA	PBS1 IFS	ST	NMOS	I^2C data line
	URX/UTX	PBS1 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication
	PCK	PBS1	—	CMOS	Peripheral clock output pin
	AN6	PBS1	AN	—	A/D converter external input channel 6
PB7/CTP2/SCK/SCL/AN7	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTP2	PBS1	—	CMOS	CTM2 output pin
	SCK	PBS1 IFS	ST	CMOS	SPI serial clock pin
	SCL	PBS1 IFS	ST	NMOS	I^2C clock line
	AN7	PBS1	AN	—	A/D converter external input channel 7
PC0/PTP/$\overline{SCS}$/SDO/UTX	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP	PCS0	—	CMOS	PTM output pin
	$\overline{SCS}$	PCS0 IFS	ST	CMOS	SPI slave select pin
	SDO	PCS0	—	CMOS	SPI serial data output pin
	UTX	PCS0	—	CMOS	UART transmitter pin

Pin Name	Function	OPT	I/T	O/T	Description
PC1/INT1/FJTR1/CTP0/ $\overline{SCS}$/SDO/UTX	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	INT1	PCS0 IFS	ST	—	External interrupt 1
	FJTR1	PCS0 IFS	ST	—	FJTIMER input 1
	CTP0	PCS0	—	CMOS	CTM0 output pin
	$\overline{SCS}$	PCS0 IFS	ST	CMOS	SPI slave select pin
	SDO	PCS0	—	CMOS	SPI serial data output pin
	UTX	PCS0	—	CMOS	UART transmitter pin
PC2/CTP1/SDI/SDA/URX/ UTX/PCK	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTP1	PCS0	—	CMOS	CTM1 output pin
	SDI	PCS0 IFS	ST	—	SPI serial data input pin
	SDA	PCS0 IFS	ST	NMOS	I^2C data line
	URX/UTX	PCS0 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication
	PCK	PCS0	—	CMOS	Peripheral clock output pin
PC3/OVP023P1N	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	OVP023P1N	PCS0	AN	—	OVP0, OVP2 and OVP3 non-inverting input pin. OVP1 inverting input pin
PWM0	PWM0	—	—	CMOS	PWM0 output pin
PWM1	PWM1	—	—	CMOS	PWM1 output pin
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply

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

Absolute Maximum Ratings

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

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

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

Operating Temperature ..... -40°C to 85°C

I_{OL} Total 80mA

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

Total Power Dissipation 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of this 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=25°C

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

Operating Current Characteristics

Ta=25°C

Symbol	Normal Operation	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode (LIRC)	5V	$f_{SYS}=f_{LIRC}$=32kHz	—	30	50	μA
	FAST Mode (HIRC)	5V	$f_{SYS}=f_{HIRC}$=16MHz	—	2.5	5.0	mA

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

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

Standby Current Characteristics

Ta=25°C

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB}	SLEEP Mode	5V	WDT on	—	3	5	μA
	IDLE0 Mode (LIRC)	5V	f_{SUB} on	—	5	10	μA
	IDLE1 Mode (HIRC)	5V	f_{SUB} on, f_{SYS}=16MHz	—	1.4	2.0	mA

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

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

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 5V voltage.

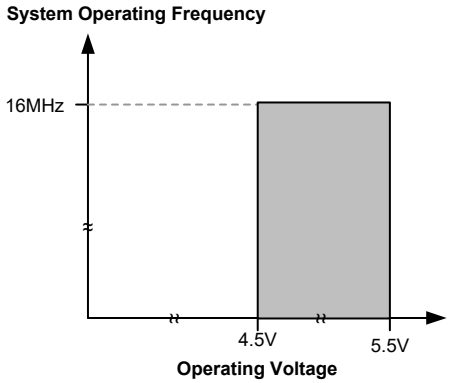
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	16MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	16	+1%	MHz
			-40°C~85°C	-2%	16	+2%	
		4.5V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C~85°C	-3%	16	+3%	
f_{PWM}	32MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	32	+1%	MHz
			-40°C~85°C	-2%	32	+2%	
		4.5V~5.5V	25°C	-2.5%	32	+2.5%	
			-40°C~85°C	-3%	32	+3%	

Note: 1. The 5V value for V_{DD} is provided as this is the voltage at which the HIRC frequency is trimmed by the writer.
2. The row below the 5V trim voltage row is provided to show the values for the specific V_{DD} range operating voltage.

Internal Low Speed Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	5V	25°C	-2%	32	+2%	kHz
		4.5V~5.5V	-40°C~85°C	-7%	32	+7%	

Operating Frequency Characteristic Curve



System Start Up Time Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
t_{SST}	System Start-up Time Wake-up from Conditions where f_{SYS} is Off	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{SYS}
		$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	System Start-up Time Wake-up from Conditions where f_{SYS} is On	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_{SYS}
		$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode	f_{HIRC} 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	14	16	18	ms
	System Reset Delay Time LVRC/WDTC Software Reset	—				
	System Reset Delay Time WDT Overflow Feset	—				
t_{SRESET}	Minimum Software Reset Width to Reset	—	45	90	120	μs

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

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

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

Input/Output Characteristics

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports or Input Pins	5V	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports or Input Pins	5V	—	3.5	—	5.0	V
		—		$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Ports	5V	$V_{OL}=0.1V_{DD}$	32	65	—	mA
	Sink Current for PWM0/PWM1	5V		32	65	—	mA
I_{OH}	Source Current for I/O Ports	5V	$V_{OH}=0.9V_{DD}$	-8	-16	—	mA
	Source Current for PWM0/PWM1	5V		-8	-16	—	mA
R_{PH}	Pull-High Resistance for I/O Ports (1)	5V	Ta=-40°C~85°C	10	30	50	kΩ
I_{LEAK}	Input Leakage Current	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{INT}	External Interrupt Minimum Pulse Width	—	—	0.3	—	—	μs
t_{EIRT}	External Interrupt Debounce Time	—	INTnDB[1:0]=00B(n=0~1)	—	0	—	t_H
			INTnDB[1:0]=01B(n=0~1)	7	—	8	
			INTnDB[1:0]=10B(n=0~1)	15	—	16	
			INTnDB[1:0]=11B(n=0~1)	23	—	24	
t_{FJT}	FJTR1 Pin Minimum Input Pulse Width	—	—	0.3	—	—	μs
t_{TCK}	CTCKn Pin Minimum Input Pulse Width	—	—	0.3	—	—	μs
t_{TPI}	PTM PTPI Pin Minimum Input Pulse Width	—	—	0.3	—	—	μs

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{TMCLK}	PTM Maximum Timer Clock Source Frequency	5V	—	—	—	1	f_{SYS}
t_{CPW}	PTM Minimum Capture Pulse Width	—	—	t_{CPW}(2)	—	—	μs

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

2. If PTCAPTS=0, then t_{CPW}=max(2×t_{TMCLK}, t_{TPI})
If PTCAPTS=1, then t_{CPW}=max(2×t_{TMCLK}, t_{TCK})
t_{TMCLK}=1/f_{TMCLK}
Ex1: If PTCAPTS=0, f_{TMCLK}=16MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.125μs, 0.3μs)=0.3μs
Ex2: If PTCAPTS=1, f_{TMCLK}=16MHz, t_{TCK}=0.3μs, then t_{CPW}=max(0.125μs, 0.3μs)=0.3μs
Ex3: If PTCAPTS=0, f_{TMCLK}=8MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.25μs, 0.3μs)=0.3μs
Ex4: If PTCAPTS=0, f_{TMCLK}=4MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.5μs, 0.3μs)=0.5μs

Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Flash Program Memory							
E_P	Cell Endurance	—	—	100K	—	—	E/W(2)
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
t_{ACTV}	ROM Activation Time – Wake-up from Power Down Mode(1)	—	—	32	—	64	μs
Data EEPROM Memory							
t_{EEER}	Erase Time	—	EWERTS=0	—	3.2	3.9	ms
		—	EWERTS=1	—	3.7	4.5	
t_{EEWR}	Write Time (Byte Mode)	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	
	Write Time (Page Mode)	—	EWERTS=0	—	2.2	2.7	
		—	EWERTS=1	—	3.0	3.6	
E_P	Cell Endurance	—	—	100K	—	—	E/W(2)
t_{RETD}	ROM Data Retention time	—	Ta=25°C	—	40	—	Year
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. “E/W” means Erase/Write times.

LVD/LVR Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 3.15V	-5%	3.15	+5%	V
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 3.3V	-5%	3.3	+5%	V
		—	LVD enable, voltage select 3.6V	-5%	3.6	+5%	
		—	LVD enable, voltage select 4.0V	-5%	4.0	+5%	
$I_{LVRLVDBG}$	Operating Current	5V	LVD enable, LVR enable, VBGEN=0	—	20	25	μA
			LVD enable, LVR enable, VBGEN=1	—	180	200	
t_{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	18	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	TLVR[1:0]=00B	120	240	480	μs
			TLVR[1:0]=01B	0.5	1.0	2.0	ms
			TLVR[1:0]=10B	1	2	4	
			TLVR[1:0]=11B	2	4	8	
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs

Internal Reference Voltage Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{BG}	Bandgap Reference Voltage	—	—	-5%	1.23	+5%	V

Note: The V_{BG} voltage is used as the A/D converter internal signal input.

A/D Converter Electrical Characteristics

Ta=25°C

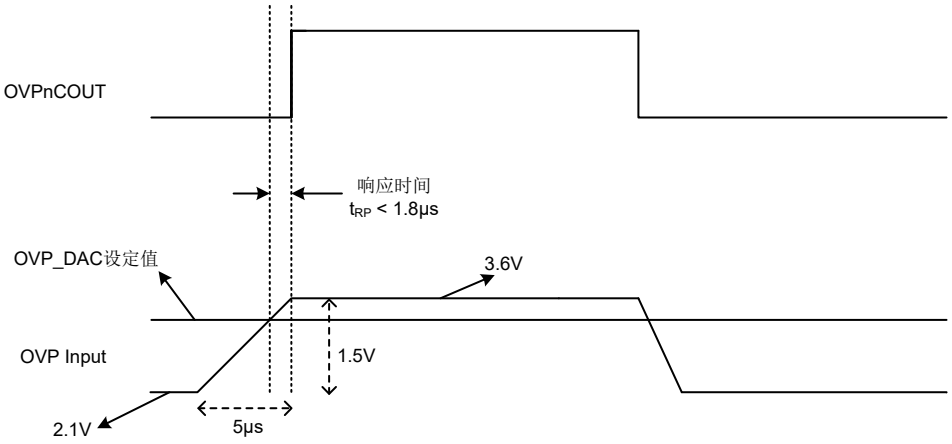
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	2	—	V_{DD}	V
N_R	Resolution	—	—	—	—	12	Bit
DNL	Differential Non-linearity	5V	V_{REF}=V_{DD}, t_{ADCK}=0.5μs	-3	—	+3	LSB
			V_{REF}=V_{DD}, t_{ADCK}=10μs				
INL	Integral Non-linearity	5V	V_{REF}=V_{DD}, t_{ADCK}=0.5μs	-4	—	+4	LSB
			V_{REF}=V_{DD}, t_{ADCK}=10μs				
I_{ADC}	Additional Current for A/D Converter Enable	5V	No load, t_{ADCK}=0.5μs	—	850	1000	μA
t_{ADCK}	Clock Period	—	4.5V≤V_{DD}≤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}

Over Voltage Protection Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OVP}	OVPn Operating Current	5V	OVPnEN=1 (n=2~8)	—	500	750	μA
			OVPnEN=1, DACnEN=0 (n=0,1)	—	100	140	μA
			OVPnEN=1, DACnEN=1 (n=0,1)	—	500	750	μA
V_{OS}	Input Offset Voltage	5V	With calibration	-2	—	2	mV
V_{HYS}	Hysteresis	5V	HYSn[1:0]=00B	0	0	5	mV
			HYSn[1:0]=01B	15	30	50	
			HYSn[1:0]=10B	30	60	90	
			HYSn[1:0]=11B	40	90	130	
V_{CM}	Common Mode Voltage Range	5V	—	V_{SS}	—	V_{DD}-1	V
R_O	R2R Output Resistance	5V	—	—	10	—	kΩ
DNL	Differential Non-linearity	5V	DAC V_{REF}=V_{DD}	—	—	±1	LSB
INL	Integral Non-linearity	5V	DAC V_{REF}=V_{DD}	—	—	±1.5	LSB
t_{RP}	OVP Response Time	5V	OVPnDA=10110011b OVPnDEB[2:0]=000 DAC V_{REF}=V_{DD} OVP Input=2.1V~3.6V (Note)	—	1.0	1.8	μs
		5V	With 60mV overdrive	—	—	1	

Note: OVP Response Time (t_{RP}) Verify Waveform (5V).



Operational Amplifier Electrical Characteristics

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OPA}	Additional Current for Operational Amplifier Enable	5V	No load	—	300	600	μA
V_{OS}	Input Offset Voltage	5V	Without calibration (OPOOF[5:0]=100000B)	-15	—	15	mV
			With calibration	-2	—	2	
V_{CM}	Common Mode Voltage Range	5V	—	V_{SS}	—	V_{DD}-1.4	V
V_{OR}	Maximum Output Voltage Range	5V	—	V_{SS}+0.1	—	V_{DD}-0.1	V
R_{OPAR1}	OPAR1 Resistance Value	5V	—	7.5	10	12.5	kΩ
SR	Slew Rate	5V	No load	0.6	1.8	—	V/μs
GBW	Gain Bandwidth	5V	R_{LOAD}=1MΩ, C_{LOAD}=100pF	—	2200	—	kHz
PSRR	Power Supply Rejection Ratio	5V	—	60	80	—	dB
CMRR	Common Mode Rejection Ratio	5V	—	60	80	—	dB
Ga	PGA Gain Accuracy(Note)	5V	Relative gain, Ta=-40°C~85°C	-5	—	5	%

Note: The PGA gain accuracy is guaranteed only when the PGA output voltage meets the V_{OR} specification.

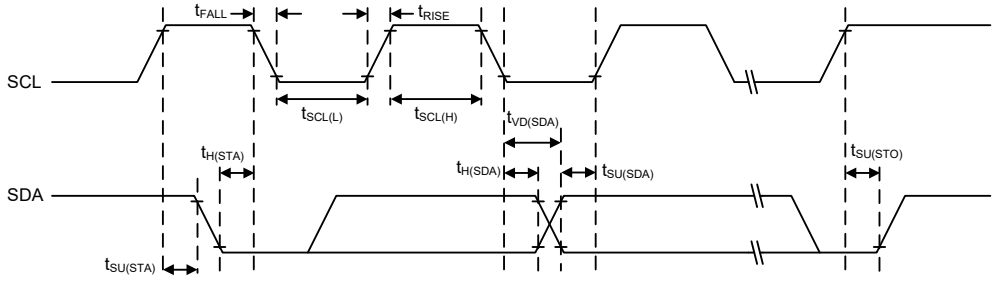
I²C Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{I2C}	I²C Standard Mode (100kHz) f_{SYS} Frequency (Note)	—	No clock debounce	2	—	—	MHz
			2 system clock debounce	4	—	—	
			4 system clock debounce	4	—	—	
	I²C Fast Mode (400kHz) f_{SYS} Frequency (Note)	—	No clock debounce	4	—	—	MHz
			2 system clock debounce	8	—	—	
			4 system clock debounce	8	—	—	
f_{SCL}	SCL Clock Frequency	5V	Standard mode	—	—	100	kHz
			Fast mode	—	—	400	
$t_{SCL(H)}$	SCL Clock High Time	5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
$t_{SCL(L)}$	SCL Clock Low Time	5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
t_{FALL}	SCL and SDA Fall Time	5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
t_{RISE}	SCL and SDA Rise Time	5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
$t_{SU(SDA)}$	SDA Data Setup Time	5V	Standard mode	0.25	—	—	μs
			Fast mode	0.1	—	—	
$t_{H(SDA)}$	SDA Data Hold Time	5V	—	0.1	—	—	μs
$t_{VD(SDA)}$	SDA Data Valid Time	5V	—	—	—	0.6	μs
$t_{SU(STA)}$	Start Condition Setup Time	5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	
$t_{H(STA)}$	Start Condition Hold Time	5V	Standard mode	4.0	—	—	μs
			Fast mode	0.6	—	—	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
$t_{SU(STO)}$	Stop Condition Setup Time	5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	

Note: Using the debounce function can make the transmission more stable and reduce the probability of communication failure due to interference.

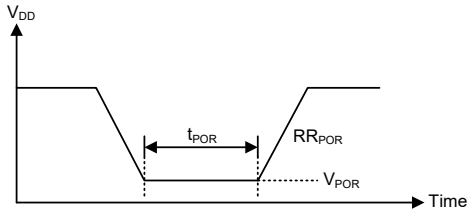


I²C Timing Diagram

Power-on Reset Characteristics

Ta=25°C

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

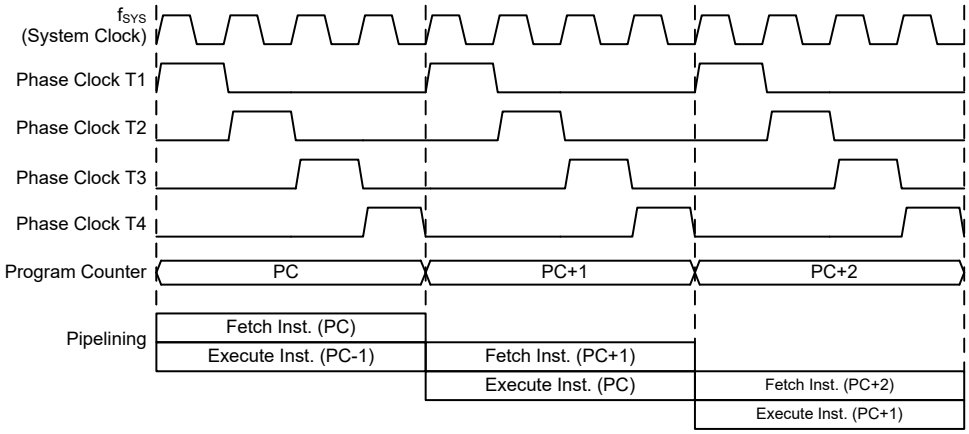


System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The range of the device take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one or two cycles for most of the standard or extended instructions respectively. The exceptions to this are 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 low-cost, high-volume production for controller applications.

Clocking and Pipelining

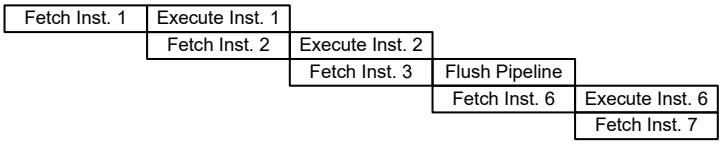
The main system clock, derived from either an HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the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



System Clocking and Pipelining

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

```
1          MOV A,[12H]
2          CALL DELAY
3          CPL [12H]
4          :
5          :
6  DELAY:  NOP
```



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as “JMP” or “CALL” that demands 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)
PC12~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.

Program Counter Read Registers

The Program Counter Read registers are a read only register pair for reading the program counter value which indicates the current program execution address. Read the low byte register first then the high byte register. Reading the low byte register, PCRL, will read the low byte data of the current program execution address, and place the high byte data of the program counter into the 8-bit PCRH buffer. Then reading the PCRH register will read the corresponding data from the 8-bit PCRH buffer.

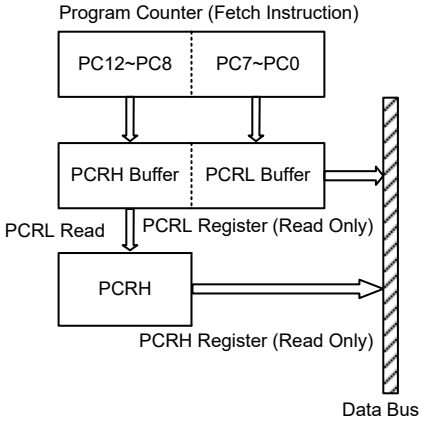
The following example shows how to read the current program execution address. When the current program execution address is 123H, the steps to execute the instructions are as follows:

(1) MOV A, PCRL → the ACC value is 23H, and the PCRH value is 01H;

MOV A, PCRH → the ACC value is 01H.

(2) LMOV A, PCRL → the ACC value is 23H, and the PCRH value is 01H;

LMOV A, PCRH → the ACC value is 01H.



- **PCRL 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**: Program Counter Read Low byte register bit 7 ~ bit 0

• **PCRH Register**

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

Bit 7~5 Unimplemented, read as "0"

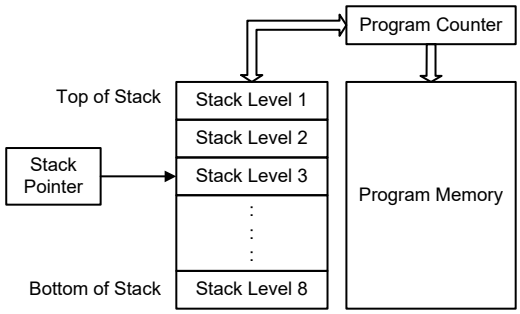
Bit 4~0 **D12~D8**: Program Counter Read High byte register bit 4 ~ bit 0

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 8 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, STKPTR, which is a read only register. 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	—	—	—	—	D2	D1	D0
R/W	R/W	—	—	—	—	R	R	R
POR	0	—	—	—	—	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 the hardware.

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **D2~D0**: Stack pointer register bit 2 ~ bit 0

The following examples show how the Stack Pointer and the overflow flag change after several CALL instructions or RET instructions are executed consecutively. By default, the STKPTR[2:0] and OSF values are all zero. The stack is 8 levels.

(1) When 9 consecutive CALL instructions are executed and no any RET instruction executed during this period, the changes of the STKPTR[2:0] and OSF bits are as follows:

CALL executed	0	1	2	3	4	5	6	7	8	9
STKPTR[2:0] value	0	1	2	3	4	5	6	7	0	1
OSF bit	0	0	0	0	0	0	0	0	0	1

(2) When the OSF is set high and it is not cleared, then the OSF bit will remain 1 no matter how many times the RET instruction is executed.

(3) When the stack is empty and then 8 consecutive RET instructions are executed, the changes of the STKPTR[2:0] and OSF bits are as follows:

RET executed	0	1	2	3	4	5	6	7	8
STKPTR[2:0] value	0	7	6	5	4	3	2	1	0
OSF bit	0	1	1	1	1	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,

  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:

  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,

  LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:

  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,

  LRRA, LRR, LRRCA, LRRC, LRLA, LRL, LRLCA, LRLC
- Increment and Decrement:

  INCA, INC, DECA, DEC,

  LINCA, LINC, LDECA, LDEC
- Branch decision:

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

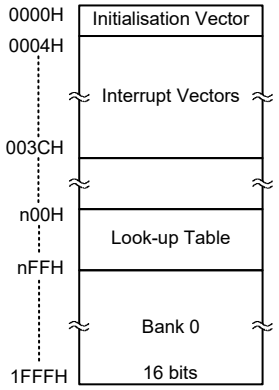
  LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

Flash Program Memory

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

Structure

The Program Memory has a capacity of 8K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupts 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 0000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

Look-up Table

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

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

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

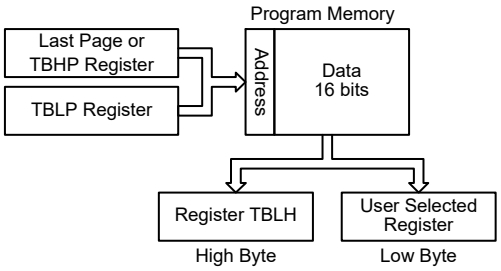


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "1F00H" which refers to the start address of the last page within the 8K words Program Memory. 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 "1F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the address specified by TBLP and TBHP if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

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

Table Read Program Example

```
tempreg1 db ?     ; temporary register #1
tempreg2 db ?     ; temporary register #2
:
mov a,06h         ; initialise low table pointer - note that this address is referenced
mov tblp,a        ; to the last page or the page that tbhp pointed
mov a,1fh         ; initialise high table pointer
mov tbhp,a        ; it is not necessary to set tbhp if executing tabrdl or ltabrdl
:
:
tabrd tempreg1    ; transfers value in table referenced by table pointer data at
                  ; program memory address "1F06H" 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 "1F05H" transferred to tempreg2 and TBLH
                  ; in this example the data "1AH" is transferred to tempreg1 and
                  ; data "0FH" to register tempreg2
:
org 1F00h         ; sets initial address of program memory last page
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

In Circuit Programming – ICP

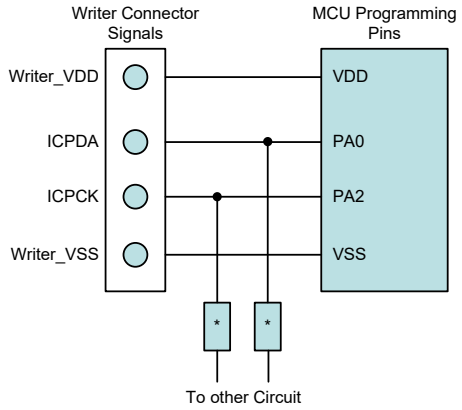
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides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

The Flash MCU to Writer programming pins correspondence table is as follows:

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming serial data/address
ICPCK	PA2	Programming clock
VDD	VDD	Power supply
VSS	VSS	Ground

The Program Memory can be programmed serially in-circuit using this 4-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply. 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 an EV chip named HT45V0074A which is used to emulate the real MCU device. The EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

Holtek e-Link Pins	EV Chip Pins	Pin Description
OCDSDA	OCDSDA	On-chip debug support data/address input/output
OCDSCK	OCDSCK	On-chip debug support clock input
VDD	VDD	Power supply
VSS	VSS	Ground

Data Memory

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

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

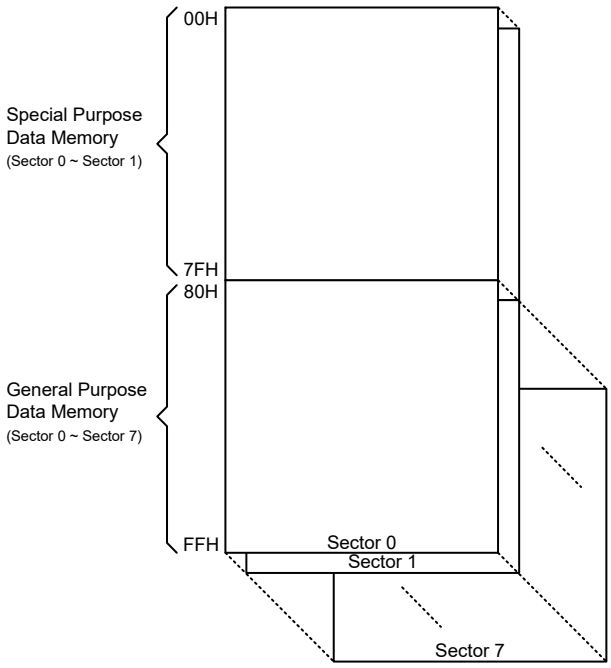
Structure

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

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

Special Purpose Data Memory	General Purpose Data Memory	
Located Sectors	Capacity	Sector: Address
0, 1	1024×8	0: 80H~FFH 1: 80H~FFH : : 7: 80H~FFH

Data Memory Summary



Data Memory Structure

Data Memory Addressing

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

Direct Addressing can be used in all sectors using the extended instructions which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address “m” in the extended instructions has 11 valid bits, the high byte indicates a sector and the low byte indicates a specific address within the sector.

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0	Sector 1
00H	IAR0	PAS0
01H	MP0	PAS1
02H	IAR1	PBS0
03H	MP1L	PBS1
04H	MP1H	PCS0
05H	ACC	
06H	PCL	IFS
07H	TBLP	LVRC
08H	TBLH	TLVRC
09H	TBHP	LVDC
0AH	STATUS	WDTC
0BH		INTEG
0CH	IAR2	INTDB
0DH	MP2L	PSCR
0EH	MP2H	TB0C
0FH	RSTFC	TB1C
10H	SCC	PCKC
11H	HIRCC	CTM0C0
12H		CTM0C1
13H	IECC	CTM0DL
14H	PA	CTM0DH
15H	PAC	CTM0AL
16H	PAPU	CTM0AH
17H	PAWU	CTM1C0
18H	PB	CTM1C1
19H	PBC	CTM1DL
1AH	PBPU	CTM1DH
1BH	PC	CTM1AL
1CH	PCC	CTM1AH
1DH	PCPU	CTM2C0
1EH	INTC0	CTM2C1
1FH	INTC1	CTM2DL
20H	INTC2	CTM2DH
21H	INTC3	CTM2AL
22H	MFI0	CTM2AH
23H	MFI1	
24H	MFI2	
25H		
26H	MFI4	
27H	MFI5	
28H	MFI6	
29H	MFI7	PTMC0
2AH	MFI8	PTMC1
2BH	SADC0	PTMDL
2CH	SADC1	PTMDH
2DH	SADC2	PTMAL
2EH	SADOL	PTMAH
2FH	SADOH	PTMRPL
30H	SADO1BL	PTMRPH
31H	SADO1BH	
32H	SADO2BL	
33H	SADO2BH	
34H	LEBC	
35H	ATAC1C	
36H	ATAC2C	
37H	ATADT	
38H	TCRL	
39H	TCRH	
3AH	TCRC	
3BH	CRCCR	
3CH	CRCIN	
3DH	CRCDL	
3EH	CRCDH	STKPTR
3FH	EEAL	

	Sector 0	Sector 1
40H	EEAH	EEC
41H	EED	FJTMC
42H	SIMC0	FJTMR1
43H	SIMC1/UUCR1	FJTMR2
44H	SIMD/UTXR_RXR	FJTMR3
45H	SIMA/SIMC2/UUCR2	FJTMR4
46H	UUCR3	FJTMD
47H	SIMTOC/UBRG	FJDT0AL
48H	UUSR	FJDT0AH
49H	PWMC0	FJDT0BL
4AH	PWMC1	FJDT0BH
4BH	PWMC2	FJDT1AL
4CH	ERSGC	FJDT1AH
4DH	PWMPC0	FJDT1BL
4EH	PWMPC1	FJDT1BH
4FH	PWMPC2	DTMINL
50H	PWMPL	DTMINH
51H	PWMPH	OPC0
52H	PWMDL	OPC1
53H	PWMDH	OPOCAL
54H	PWMRL	OVP0C0
55H	PWMRH	OVP0C1
56H	DT0L	OVP0C2
57H	DT0H	OVP0DA
58H	PLC	OVP1C0
59H	MDUWR0	OVP1C1
5AH	MDUWR1	OVP1C2
5BH	MDUWR2	OVP1DA
5CH	MDUWR3	OVP2C0
5DH	MDUWR4	OVP2C1
5EH	MDUWR5	OVP2C2
5FH	MDUWCTRL	OVP2DA
60H	DT1L	OVP3C0
61H	DT1H	OVP3C1
62H	PWMMAXPL	OVP3C2
63H	PWMMAXPH	OVP3DA
64H	PWMMINPL	OVP4C0
65H	PWMMINPH	OVP4C1
66H	DECPWMP	OVP4C2
67H	TMCC0	OVP4DA
68H	TMCC1	OVP5C0
69H	TMCDL	OVP5C1
6AH	TMCDH	OVP5C2
6BH	TMCCCRAL	OVP5DA
6CH	TMCCCRAH	OVP6C0
6DH	PPDSITA	OVP6C1
6EH		OVP6C2
6FH		OVP6DA
70H	FJC0	OVP7C0
71H	FJC1	OVP7C1
72H	FJF0	OVP7C2
73H	FJM1C0	OVP7DA
74H	FJM1C1	OVP8C0
75H	FJM1C2	OVP8C1
76H	FJM1C3	OVP8C2
77H	FJM0C0	OVP8DA
78H	FJM0C1	PCRL
79H	FJM0C2	PCRH
7AH	FJPAL	
7BH	FJPAH	
7CH	FJPBL	FJC2
7DH	FJPBH	
7EH	FJPHL	
7FH	FJPHH	

: Unused, read as 00H

: Reserved, cannot be changed

Special Purpose Data Memory

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, IAR2

The Indirect Addressing Registers, IAR0, IAR1 and IAR2, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0, IAR1 and IAR2 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0, MP1L/MP1H or MP2L/MP2H. Acting as a pair, IAR0 and MP0 can together access data from Sector 0 while the IAR1 register together with MP1L/MP1H register pair and IAR2 register together with MP2L/MP2H register pair can access data from any sector. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers directly will return a result of "00H" and writing to the registers directly will result in no operation.

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

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L and MP2H are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Sector 0, while MP1L&MP1H together with IAR1 and MP2L&MP2H together with IAR2 are used to access data from all sectors according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all sectors using the extended instruction which can address all available data memory space.

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

Indirect Addressing Program Example 1

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

Indirect Addressing Program Example 2

```
data .section 'data'
adres1  db ?
adres2  db ?
adres3  db ?
adres4  db ?
block   db ?
code .section at 0 'code'
org 00h
start:
    mov a,04h              ; setup size of block
    mov block,a
    mov a,01h              ; setup the memory sector
    mov mp1h,a
    mov a, offset adres1   ; Accumulator loaded with first RAM address
    mov mp1l,a             ; setup memory pointer with first RAM address
loop:
    clr IAR1               ; clear the data at address defined by MP1L
    inc mp1l               ; increment memory pointer MP1L
    sdz block              ; check if last memory location has been cleared
    jmp loop
continue:
    :
```

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

Direct Addressing Program Example Using Extended Instructions

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

Note: here “m” is a data memory address located in any data memory sectors. For example, m=1F0H, it indicates address 0F0H in Sector 1.

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

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

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

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

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

- PDF is cleared by a system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction.
- TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.
- SC is the result of the "XOR" operation which is performed by the OV flag and the MSB of the current instruction operation result.

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

• STATUS Register

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

"x": Unknown

Bit 7 **SC**: The result of the "XOR" operation which is performed by the OV flag and the MSB of the instruction operation result

Bit 6 **CZ**: The operational result of different flags for different instructions
For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.
For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation Z flag.
For other instructions, the CZ flag will not be affected.

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

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

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

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

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

Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation
The "C" flag is also affected by a rotate through carry instruction.

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 512×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and write operations to the EEPROM can be carried out in either the byte mode or page mode determined by the mode selection bit, MODE, in the control register, EEC.

EEPROM Registers

Four registers control the overall operation of the internal EEPROM Data Memory. These are the address registers, EEAL and EEAH, the data register, EED and a single control register, EEC. As the EEAL, EEAH and EED registers are located in sector 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register, however, being located in sector 1, can only be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pair and Indirect Addressing Register, IAR1 or IAR2. Because the EEC control register is located at address 40H in sector 1, the Memory Pointer low byte register, MP1L or MP2L, must first be set to the value 40H and the Memory Pointer high byte register, MP1H or MP2H, set to the value, 01H, before any operations on the EEC register are executed.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEAL	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH	—	—	—	—	—	—	UNU51	EEAH0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD

EEPROM Register List

- **EEAL Register**

Bit	7	6	5	4	3	2	1	0
Name	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
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 **EEAL7~EEAL0**: Data EEPROM low byte address bit 7 ~ bit 0

- **EEAH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	UNU51	EEAH0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”
Bit 1 **UNU51**: Reserved, this bit must be fixed at “0”
Bit 0 **EEAH0**: Data EEPROM high byte address bit 0

• **EED Register**

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

Bit 7~0 **D7~D0**: Data EEPROM data bit 7 ~ bit 0

• **EEC Register**

Bit	7	6	5	4	3	2	1	0
Name	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD
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 **EWERTS**: Data EEPROM Erase time and Write time selection
0: Erase time is 3.2ms (t_{EEER}) / Write time is 2.2ms (t_{EEWR})
1: Erase time is 3.7ms (t_{EEER}) / Write time is 3.0ms (t_{EEWR})

Bit 6 **EREN**: Data EEPROM erase enable
0: Disable
1: Enable
This bit is used to enable Data EEPROM erase function and must be set high before Data EEPROM erase operations are carried out. This bit will be automatically reset to zero by hardware after the erase cycle has finished. Clearing this bit to zero will inhibit data EEPROM erase operations.

Bit 5 **ER**: Data EEPROM erase control
0: Erase cycle has finished
1: Activate an erase cycle
This is the Data EEPROM Erase Control Bit. When this bit is set high by the application program, an erase cycle will be activated. This bit will be automatically reset to zero by hardware after the erase cycle has finished. Setting this bit high will have no effect if the EREN has not first been set high.

Bit 4 **MODE**: Data EEPROM operation mode selection
0: Byte operation mode
1: Page operation mode
This is the EEPROM operation mode selection bit. When the bit is set high by the application program, the Page write, erase or read function will be selected. Otherwise, the byte write or read function will be selected. The EEPROM page buffer size is 16-byte.

Bit 3 **WREN**: Data EEPROM write enable
0: Disable
1: Enable
This is the Data EEPROM Write Enable Bit, which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations. Note that the WREN bit will automatically be cleared to zero after the write operation is finished.

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

Bit 1 **RDEN**: Data EEPROM read enable
0: Disable
1: Enable

This is the Data EEPROM Read Enable Bit, which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.

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

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

Read Operation from the EEPROM

Reading data from the EEPROM can be implemented by two modes for this device, byte read mode or page read mode, which is controlled by the EEPROM operation mode selection bit, MODE, in the EEC register.

Byte Read Mode

The EEPROM byte read operation can be executed when the mode selection bit, MODE, is cleared to zero. For a byte read operation the desired EEPROM address should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM byte read operation. Note that setting the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the read cycle terminates, the RD bit will automatically be cleared to zero and the EEPROM data can be read from the EED register. The data will remain in the EED register until another read or write operation is executed. The application program can poll the RD bit to determine when the data is valid for reading.

Page Read Mode

The EEPROM page read operation can be executed when the mode selection bit, MODE, is set high. The page size can be up to 16 bytes for the page read operation. For a page read operation the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM page read operation. Note that setting the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the current byte read cycle terminates, the RD bit will automatically be cleared to zero indicating that the EEPROM data can be read from the EED register and the current address will be incremented by one by hardware. The data which is stored in the next EEPROM address can continuously be read when the RD bit is again set high without reconfiguring the EEPROM address and RDEN control bit. The application program can poll the RD bit to determine when the data is valid for reading.

The EEPROM address higher 5 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page read operation mode the lower 4-bit address value will automatically be incremented by one. However, the higher 5-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not "roll over".

Page Erase Operation to the EEPROM

The EEPROM page erase operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page erase. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM erase enable control bit, namely EREN, is changed from "1" to "0", the internal page buffer will also be cleared. Note that when the EREN bit is changed from "0" to "1", the internal page buffer will not be cleared. The EEPROM address higher 5 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page erase operation mode the lower 4-bit address value will automatically be incremented by one after each dummy data byte is written into the EED register. However, the higher 5-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not "roll over".

For page erase operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers and the dummy data to be written should be placed in the EED register. The maximum data length for a page is 16 bytes. Note that the write operation to the EED register is used to tag address, it must be implemented to determine which addresses to be erased. When the page dummy data is completely written, then the EREN bit in the EEC register should be set high to enable erase operations and the ER bit must be immediately set high to initiate the EEPROM erase process. These two instructions must be executed in two consecutive instruction cycles to activate an erase operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing an erase operation and then set again after a valid erase activation procedure has completed.

As the EEPROM erase cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data have been erased from the EEPROM. Detecting when the erase cycle has finished can be implemented either by polling the ER bit in the EEC register or by using the EEPROM interrupt. When the erase cycle terminates, the ER bit will be automatically cleared to zero by the microcontroller, informing the user that the page data has been erased. The application program can therefore poll the ER bit to determine when the erase cycle has ended. After the erase operation is finished, the EREN bit will be cleared by hardware. The Data EEPROM erased page content will all be zero after a page erase operation.

Write Operation to the EEPROM

Writing data to the EEPROM can be implemented by two modes for this device, byte write mode or page write mode, which is controlled by the EEPROM operation mode selection bit, MODE, in the EEC register.

Byte Write Mode

The EEPROM byte write operation can be executed when the mode selection bit, MODE, is cleared to zero. For byte write operations the desired EEPROM address should first be placed in the EEAH and EEAL registers and the data to be written should be placed in the EED register. To write data to the EEPROM, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed 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 high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended. After the write operation is finished, the WREN bit will be cleared by hardware. Note that a byte erase operation will automatically be executed before a byte write operation is successfully activated.

Page Write Mode

Before a page write operation is executed, it is important to ensure that a relevant page erase operation has been successfully executed. The EEPROM page write operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page write. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM write enable control bit, namely WREN, is changed from "1" to "0", the internal page buffer will also be cleared. Note that when the WREN bit is changed from "0" to "1", the internal page buffer will not be cleared. A page write is initiated in the same way as a byte write initiation except that the EEPROM data can be written up to 16 bytes. The EEPROM address higher 5 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page write operation mode the lower 4-bit address value will automatically be incremented by one after each data byte is written into the EED register. However, the higher 5-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not "roll over". At this point any data write operations to the EED register will be invalid.

For page write operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers and the data to be written should be placed in the EED register. The maximum data length for a page is 16 bytes. Note that when a data byte is written into the EED register, then the data in the EED register will be loaded into the internal page buffer and the current address value will automatically be incremented by one. When the page data is completely written into the page buffer, then the WREN bit in the EEC register should be set high to enable write operations and the WR bit must be immediately set high to initiate the EEPROM write process. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended. After the write operation is finished, the WREN bit will be set low by hardware.

Write Protection

Protection against inadvertent write operation is provided in several ways. After the device is powered-on the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on the Memory Pointer high byte register, MP1H or MP2H, will be reset to zero, which means that Data Memory Sector 0 will be selected. As the EEPROM control register is located in Sector 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

EEPROM Interrupt

The EEPROM erase or write interrupt is generated when an EEPROM erase or write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. However as the EEPROM interrupt is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM erase or write cycle ends, the DEF request flag will be set. If the global and EEPROM interrupts are enabled and the stack is not full, a jump to the associated EEPROM interrupt vector will take place. When the interrupt is serviced, the EEPROM interrupt flag will be automatically reset. More details can be obtained in the Interrupt section.

Programming Considerations

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

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

Programming Examples

Reading a Data Byte from the EEPROM – Polling Method

```
MOV A, 040H             ; setup memory pointer low byte MP1L
MOV MP1L, A             ; MP1L points to EEC register
MOV A, 01H              ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4              ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H   ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L   ; user defined low byte address
MOV EEAL, A
SET IAR1.1              ; set RDEN bit, enable read operations
SET IAR1.0              ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0              ; check for read cycle end
JMP BACK
CLR IAR1                ; disable EEPROM read function
CLR MP1H
```

```
MOV A, EED              ; move read data to register
MOV READ_DATA, A
```

Reading a Data Page from the EEPROM – Polling Method

```
MOV A, 040H             ; setup memory pointer low byte MP1L
MOV MP1L, A             ; MP1L points to EEC register
MOV A, 01H              ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4              ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H   ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L   ; user defined low byte address
MOV EEAL, A
SET IAR1.1              ; set RDEN bit, enable read operations
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL READ
CALL READ
:
:
JMP PAGE_READ_FINISH
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
READ:
SET IAR1.0              ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0              ; check for read cycle end
JMP BACK
MOV A, EED              ; move read data to register
MOV READ_DATA, A
RET
:
PAGE_READ_FINISH:
CLR IAR1                ; disable EEPROM read function
CLR MP1H
```

Erasing a Data Page to the EEPROM − Polling Method

```
MOV A, 040H             ; setup memory pointer low byte MP1L
MOV MP1L, A             ; MP1L points to EEC register
MOV A, 01H              ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4              ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H   ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L   ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP Erase_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA      ; user defined data, erase mode don't care data value
MOV EED, A
RET
:
Erase_START:
CLR EMI
SET IAR1.6              ; set EREN bit, enable erase operations
SET IAR1.5              ; start Erase Cycle - set ER bit - executed immediately
                        ; after setting EREN bit
SET EMI
BACK:
```

```
SZ  IAR1.5              ; check for erase cycle end
JMP BACK
CLR MP1H
```

Writing a Data Byte to the EEPROM – Polling Method

```
MOV A, 040H             ; setup memory pointer low byte MP1L
MOV MP1L, A             ; MP1L points to EEC register
MOV A, 01H              ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4              ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H   ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L   ; user defined low byte address
MOV EEAL, A
MOV A, EEPROM_DATA      ; user defined data
MOV EED, A
CLR EMI
SET IAR1.3              ; set WREN bit, enable write operations
SET IAR1.2              ; start Write Cycle - set WR bit - executed immediately
                        ; after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2              ; check for write cycle end
JMP BACK
CLR MP1H
```

Writing a Data Page to the EEPROM – Polling Method

```
MOV A, 040H             ; setup memory pointer low byte MP1L
MOV MP1L, A             ; MP1L points to EEC register
MOV A, 01H              ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4              ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H   ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L   ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP WRITE_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA      ; user defined data
MOV EED, A
RET
:
WRITE_START:
CLR EMI
SET IAR1.3              ; set WREN bit, enable write operations
SET IAR1.2              ; start Write Cycle - set WR bit - executed immediately
                        ; after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2              ; check for write cycle end
JMP BACK
CLR MP1H
```

Oscillators

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

Oscillator Overview

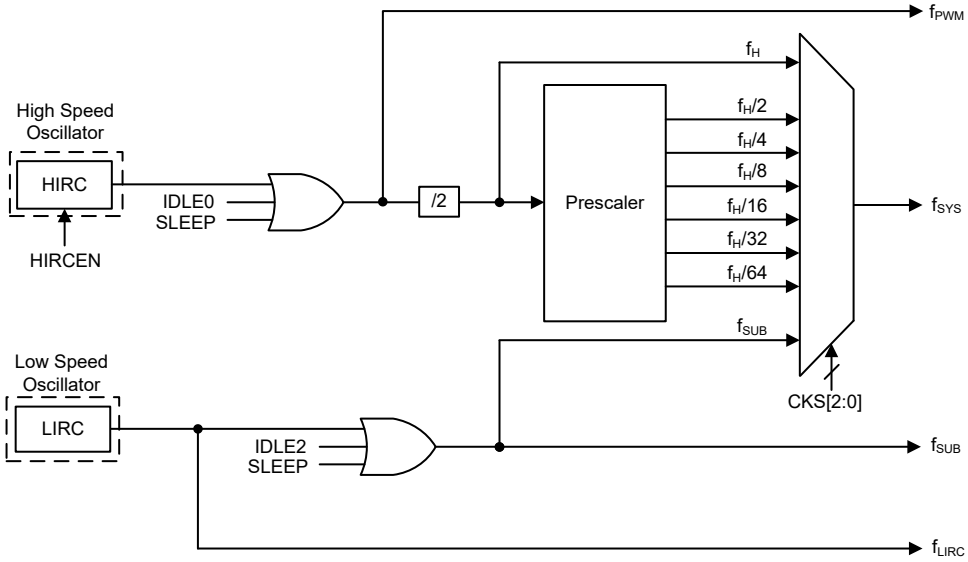
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. The fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options all can be selected through register programming. 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	Frequencies
Internal High Speed RC	HIRC	f_{PWM}=32MHz f_{HIRC}=16MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two oscillator sources, one high speed oscillator and one low speed oscillator. The high speed system clock f_H has a frequency of 16MHz and is sourced from the internal high speed RC oscillator, HIRC. The low speed system clock f_{SUB} is sourced from 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 high speed RC oscillator is a fully integrated oscillator requiring no external components. The HIRC oscillator provides a clock frequency of 32MHz, labeled as f_{PWM}, to the PWM circuits directly and a 16MHz clock, labeled as f_{HIRC}, is also provided by the HIRC oscillator and is used to be the high speed system clock f_H. 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 Oscillator is also a fully integrated RC oscillator with a typical frequency of 32kHz, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

The low speed oscillator, in addition to providing a system clock source, is also used to provide a clock source to the watchdog timer and the time base interrupt functions.

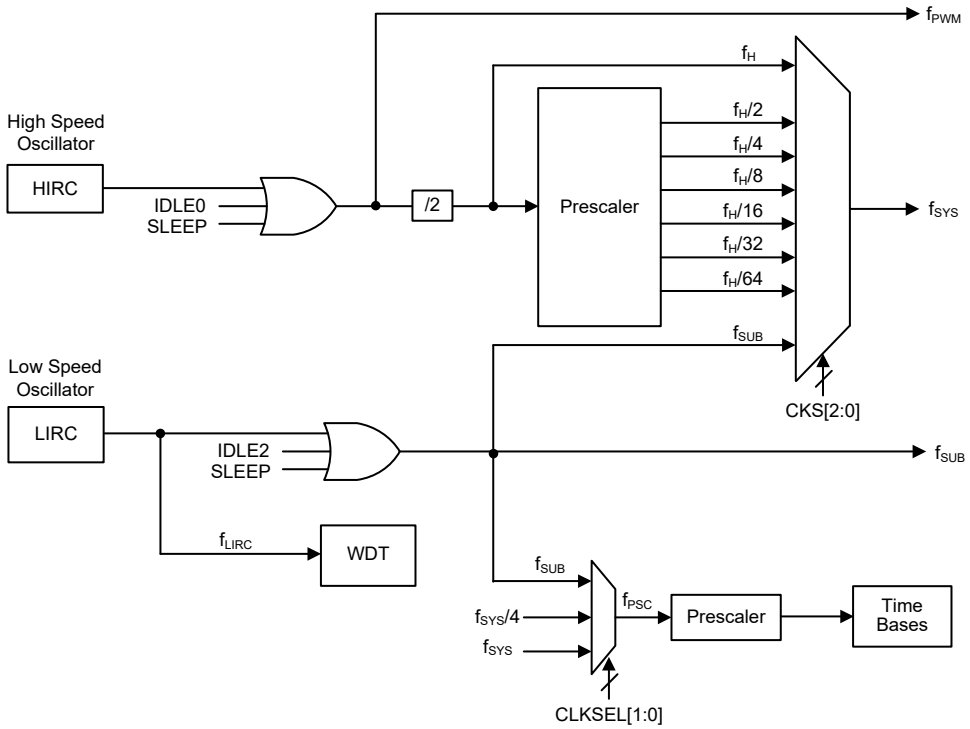
Operating Modes and System Clocks

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 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, while the low speed system clock source is sourced from the internal clock f_{SUB} which is sourced by 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 is switched to f_{SUB} from f_H, the high speed oscillator can be stop to conserve power by clearing the corresponding high speed oscillator enable control bit. However there are no f_H~$f_H/64$ or f_{PWM} clocks for peripheral circuits to use.

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(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 will be switched on since the WDT function is always enabled even 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 the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source from the HIRC high speed 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 bit both are low. In the SLEEP mode the CPU will be stopped, and the f_{SUB} clock to the peripheral function will also be stopped. However the f_{LIRC} clock still continues to operate since the WDT function is enabled.

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 if the system clock is provided by the low speed oscillator, it will continue to provide a clock source to keep some peripheral functions operational.

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 the system clock will continue to provide a clock source to keep some peripheral functions operational. The system clock can be provided by the high speed or low speed oscillator.

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 if the system clock is provided by the high speed oscillator, it will continue 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 HIRC oscillator configurations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
HIRCC	—	—	—	—	—	—	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	1	—	—	—	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 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 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. The LICR oscillator is controlled by this bit together with the WDT function enable control. If this bit is cleared to zero but the WDT function is enabled, the f_{LIRC} clock will also be enabled.

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	—	—	—	—	—	—	HIRCF	HIRCEN
R/W	—	—	—	—	—	—	R	R/W
POR	—	—	—	—	—	—	0	1

Bit 7~2 Unimplemented, read as "0"

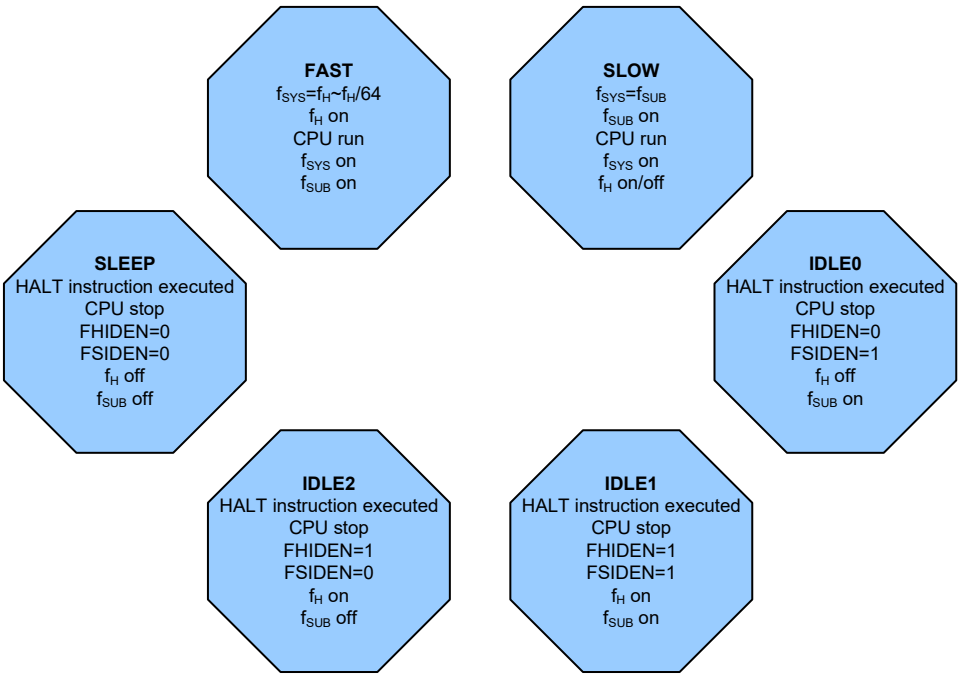
Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.It takes 16 clocks for the HIRC to be 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.

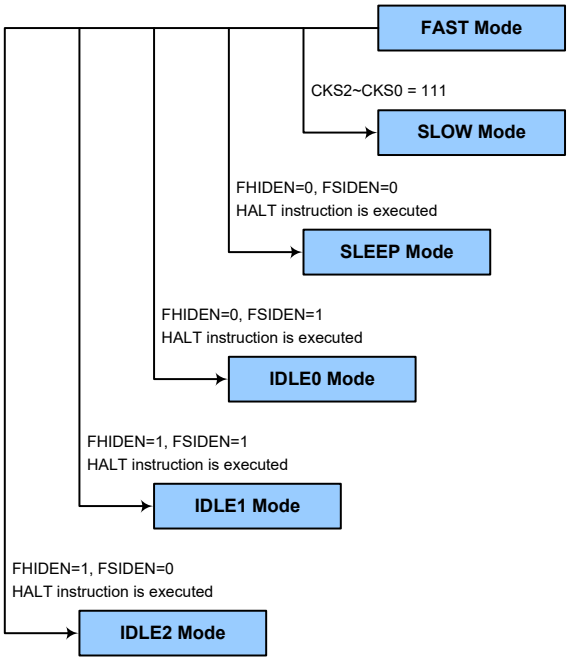
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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

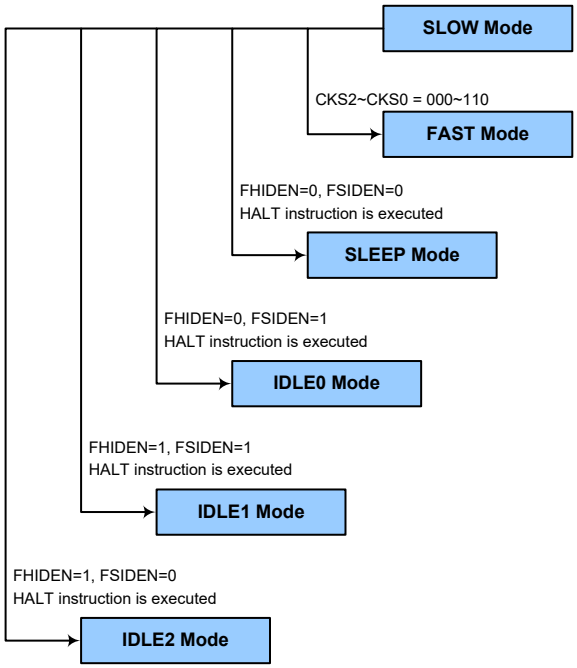
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



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 as the WDT function is enabled.

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 off and the f_{SUB} clock will be on and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The 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 as the WDT function is enabled.

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 as the WDT function is enabled.

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 and 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 as the WDT function is enabled.

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 Modes, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs.

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:

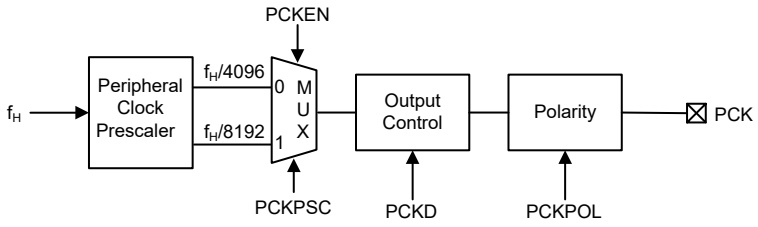
- An external falling edge on Port A
- A system interrupt
- A WDT overflow

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

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a 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.

Peripheral Clock Output

The clock output capability on the specific external output pin, PCK, allows the device to supply external hardwares with a clock signal synchronised to the microcontroller clock.



Peripheral Clock Output Block Diagram

Peripheral Clock Output Operation

The peripheral clock output pin PCK is pin-shared with other functions, the pin-shared function control register of the corresponding I/O pin must be properly configured to output the selected clock. The peripheral clock output function is controlled by the PCKC register. After the PCKEN bit has been set, writing a high value to the PCKD bit will enable the PCK output function, writing a zero value will disable the PCK output function and force the output into a low or high state by PCKPOL bit settings.

The clock source for the peripheral clock output can originate from a subdivided version of f_H. The division ratio value is determined by the PCKPSC bit in the PCKC register.

The PCK output truth table is shown as below.

PCKEN	PCKD	PCKPOL	PCK Output
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	PCK
1	1	1	PCK

Peripheral Clock Output Register

The peripheral clock output function is controlled by the PCKC register.

• **PCKC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PCKD	PCKPOL	PCKEN	—	—	—	PCKPSC
R/W	—	R/W	R/W	R/W	—	—	—	R/W
POR	—	0	0	0	—	—	—	0

Bit 7 Unimplemented, read as "0"

Bit 6 **PCKD**: PCK output control
0: Inactive
1: Active
This bit is used to control the PCK output active or inactive. If this bit is cleared to zero, the PCK output status is determined by the PCKPOL bit.

Bit 5 **PCKPOL**: PCK polarity control
0: Non-invert
1: Invert
When PCKD=0 or PCKEN=0, if this bit is low, PCK output is forced to be low; if this bit is high, the PCK output is forced to be high.

Bit 4 **PCKEN**: PCK function control
0: Disable
1: Enable

Bit 3~1 Unimplemented, read as "0"

Bit 0 **PCKPSC**: Peripheral clock prescaler selection
0: $f_{PCK}=f_H/4096$
1: $f_{PCK}=f_H/8192$

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 time-out period, the WDT enable operation as well as the MCU reset 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
01010/10101: Enable
Other values: Generates MCU reset
When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set high.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout 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. There are five bits, WE4~WE0, in the WDTC register to offer the enable control and reset control of the Watchdog Timer. The WDT function will be enabled when the WE4~WE0 bits are equal to 01010B or 10101B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after a delay time, t_{SRESET}. After power on these bits will have a value of 01010B.

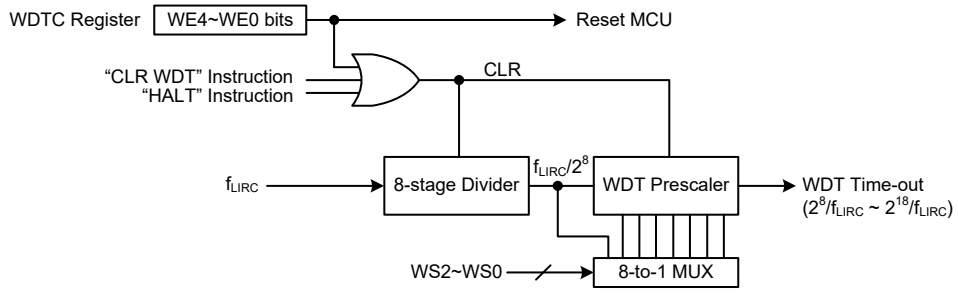
WE4~WE0 Bits	WDT Function
01010B or 10101B	Enable
Any other value	Reset MCU

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 WDTC software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bits, the second is using the Watchdog Timer software clear instruction, the third is via a HALT instruction.

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

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

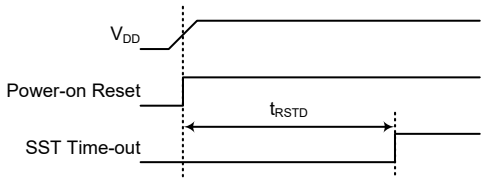
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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



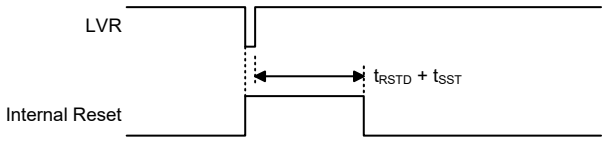
Power-on Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage, V_{DD}, and provides an MCU reset when the value falls below a certain predefined level.The LVR function is 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 a battery in the battery powered application, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will 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}. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The t_{LVR} value is selected by the TLVR1~TLVR0 bits in the TLVRC register.

The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01010101B.

Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.



Low Voltage Reset Timing Chart

Low Voltage Reset Registers

The LVRC and TLVRC registers are used to control the Low Voltage Reset function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
LVRC	LVS7	LVS6	LVS5	LVS4	LVS3	LVS2	LVS1	LVS0
TLVRC	—	—	—	—	—	—	TLVR1	TLVR0

Low Voltage Reset Register List

- **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
01010101/00110011/10011001/10101010: 3.15V
Any other value: Generates MCU reset – register is reset to POR value

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

Any register value, other than the four defined 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 content 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}

Refer to the LVD/LVR Electrical Characteristics for more details about t_{LVR}.

• **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"

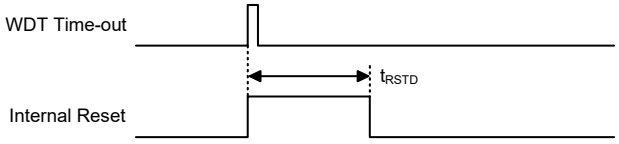
Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set high when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

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 LVRC 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

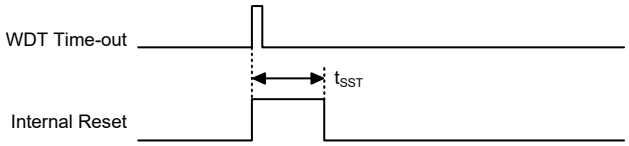
When the Watchdog time-out Reset during normal operation in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

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

"u" stands for unchanged

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

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

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

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	uuuu uuuu
ACC	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	---x xxxx	---u uuuu	---u uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu
SCC	001- --00	001- --00	uuu- --uu
HIRCC	---- --01	---- --01	---- --uu
IECC	0000 0000	0000 0000	uuuu uuuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
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
PC	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	uuuu 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
MFI0	-000 -000	-000 -000	-uuu -uuu
MFI1	0000 0000	0000 0000	uuuu uuuu
MFI2	--00 --00	--00 --00	--uu --uu
MFI4	-000 -000	-000 -000	-uuu -uuu
MFI5	0000 0000	0000 0000	uuuu uuuu
MFI6	0000 0000	0000 0000	uuuu uuuu
MFI7	0000 0000	0000 0000	uuuu uuuu
MFI8	-000 -000	-000 -000	-uuu -uuu
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	uuuu uuuu
SADC2	00-0 -000	00-0 -000	uu-u -uuu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
SADO1BL	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADO1BH	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADO2BL	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADO2BH	xxxx xxxx	xxxx xxxx	uuuu uuuu
LEBC	0--0 0000	0--0 0000	u--u uuuu
ATAC1C	--00 0000	--00 0000	--uu uuuu
ATAC2C	---0 0000	---0 0000	---u uuuu
ATADT	---- 0000	---- 0000	---- uuuu
TCRL	xxxx xxxx	xxxx xxxx	uuuu uuuu
TCRH	xxxx xxxx	xxxx xxxx	uuuu uuuu
TCRC	---0 --00	---0 --00	---u --uu
CRCCR	---- ---0	---- ---0	---- ---u
CRCIN	0000 0000	0000 0000	uuuu uuuu
CRCDL	0000 0000	0000 0000	uuuu uuuu
CRCDH	0000 0000	0000 0000	uuuu uuuu
EEAL	0000 0000	0000 0000	uuuu uuuu
EEAH	---- --00	---- --00	---- --uu
EED	0000 0000	0000 0000	uuuu uuuu
SIMC0	1110 0000	1110 0000	uuuu uuuu
SIMC1(UMD=0)	1000 0001	1000 0001	uuuu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
UUCR1*(UMD=1)	0000 00x0	0000 00x0	uuuu uuuu
SIMD/UTXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2/UUCR2	0000 0000	0000 0000	uuuu uuuu
UUCR3	---- ---0	---- ---0	---- ---u
SIMTOC(UMD=0)	0000 0000	0000 0000	uuuu uuuu
UBRG*(UMD=1)	xxxx xxxx	xxxx xxxx	uuuu uuuu
UUSR	0000 1011	0000 1011	uuuu uuuu
PWMC0	0000 0000	0000 0000	uuuu uuuu
PWMC1	0000 0000	0000 0000	uuuu uuuu
PWMC2	0000 0000	0000 0000	uuuu uuuu
ERSGC	0000 0000	0000 0000	uuuu uuuu
PWMPC0	0000 0000	0000 0000	uuuu uuuu
PWMPC1	0000 0000	0000 0000	uuuu uuuu
PWMPC2	0000 0000	0000 0000	uuuu uuuu
PWMPL	0000 0000	0000 0000	uuuu uuuu
PWMPH	---- 0000	---- 0000	---- uuuu
PWMDL	0000 0000	0000 0000	uuuu uuuu
PWMDH	---- 0000	---- 0000	---- uuuu
PWMRL	0000 0000	0000 0000	uuuu uuuu
PWMRH	---- 0000	---- 0000	---- uuuu
DT0L	0000 0000	0000 0000	uuuu uuuu
DT0H	---- 0000	---- 0000	---- uuuu
PLC	---- --00	---- --00	---- --uu
MDUWR0	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR1	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR2	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR3	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR4	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR5	xxxx xxxx	0000 0000	uuuu uuuu
MDUWCTRL	00-- ----	00-- ----	uu-- ----
DT1L	0000 0000	0000 0000	uuuu uuuu
DT1H	---- 0000	---- 0000	---- uuuu
PWMMAXPL	0000 0000	0000 0000	uuuu uuuu
PWMMAXPH	---- 0000	---- 0000	---- uuuu
PWMMINPL	0000 0000	0000 0000	uuuu uuuu
PWMMINPH	---- 0000	---- 0000	---- uuuu
DECPWMP	0000 0000	0000 0000	uuuu uuuu
TMCC0	-000 0000	-000 0000	-uuu uuuu
TMCC1	0000 ----	0000 ----	uuuu ----
TMCDL	0000 0000	0000 0000	uuuu uuuu
TMCDH	---- --00	---- --00	---- --uu
TMCCCRAL	0000 0000	0000 0000	uuuu uuuu
TMCCCRAH	---- --00	---- --00	---- --uu
PPDSITA	0000 0000	0000 0000	uuuu uuuu
FJC0	0000 0000	0000 0000	uuuu uuuu
FJC1	0000 --00	0000 --00	uuuu --uu
FJF0	0000 --00	0000 --00	uuuu --uu
FJM1C0	-000 -000	-000 -000	-uuu -uuu
FJM1C1	-000 0000	-000 0000	-uuu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
FJM1C2	0000 0000	0000 0000	uuuu uuuu
FJM1C3	0000 0000	0000 0000	uuuu uuuu
FJM0C0	-000 0000	-000 0000	-uuu uuuu
FJM0C1	0-00 0-00	0-00 0-00	u-uu u-uu
FJM0C2	0-00 0-00	0-00 0-00	u-uu u-uu
FJPAL	0000 0000	0000 0000	uuuu uuuu
FJPAH	---- 0000	---- 0000	---- uuuu
FJPBL	0000 0000	0000 0000	uuuu uuuu
FJPBH	---- 0000	---- 0000	---- uuuu
FJPHL	0000 0000	0000 0000	uuuu uuuu
FJPHH	---- 0000	---- 0000	---- uuuu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 0000	0000 0000	uuuu uuuu
PBS1	0000 0000	0000 0000	uuuu uuuu
PCS0	0000 0000	0000 0000	uuuu uuuu
IFS	-000 0000	-000 0000	-uuu uuuu
LVRC	0101 0101	0101 0101	uuuu uuuu
TLVRC	---- --01	---- --01	---- --uu
LVDC	--00 0000	--00 0000	--uu uuuu
WDTC	0101 0011	0101 0011	uuuu uuuu
INTEG	---- 0000	---- 0000	---- uuuu
INTDB	---- 0000	---- 0000	---- uuuu
PSCR	---- --00	---- --00	---- --uu
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
PCKC	-000 ---0	-000 ---0	-uuu ---u
CTM0C0	0000 0000	0000 0000	uuuu uuuu
CTM0C1	0000 0000	0000 0000	uuuu uuuu
CTM0DL	0000 0000	0000 0000	uuuu uuuu
CTM0DH	---- --00	---- --00	---- --uu
CTM0AL	0000 0000	0000 0000	uuuu uuuu
CTM0AH	---- --00	---- --00	---- --uu
CTM1C0	0000 0000	0000 0000	uuuu uuuu
CTM1C1	0000 0000	0000 0000	uuuu uuuu
CTM1DL	0000 0000	0000 0000	uuuu uuuu
CTM1DH	---- --00	---- --00	---- --uu
CTM1AL	0000 0000	0000 0000	uuuu uuuu
CTM1AH	---- --00	---- --00	---- --uu
CTM2C0	0000 0000	0000 0000	uuuu uuuu
CTM2C1	0000 0000	0000 0000	uuuu uuuu
CTM2DL	0000 0000	0000 0000	uuuu uuuu
CTM2DH	---- --00	---- --00	---- --uu
CTM2AL	0000 0000	0000 0000	uuuu uuuu
CTM2AH	---- --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

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PTMAL	0000 0000	0000 0000	uuuu uuuu
PTMAH	---- --00	---- --00	---- --uu
PTMRPL	0000 0000	0000 0000	uuuu uuuu
PTMRPH	---- --00	---- --00	---- --uu
STKPTR	0--- -000	0--- -000	u--- -000
EEC	0000 0000	0000 0000	uuuu uuuu
FJTMC	-000 0-00	-000 0-00	-uuu u-uu
FJTMR1	0000 0000	0000 0000	uuuu uuuu
FJTMR2	0000 0000	0000 0000	uuuu uuuu
FJTMR3	0000 0000	0000 0000	uuuu uuuu
FJTMR4	0000 0000	0000 0000	uuuu uuuu
FJTMD	0000 0000	0000 0000	uuuu uuuu
FJDT0AL	0000 0000	0000 0000	uuuu uuuu
FJDT0AH	---- 0000	---- 0000	---- uuuu
FJDT0BL	0000 0000	0000 0000	uuuu uuuu
FJDT0BH	---- 0000	---- 0000	---- uuuu
FJDT1AL	0000 0000	0000 0000	uuuu uuuu
FJDT1AH	---- 0000	---- 0000	---- uuuu
FJDT1BL	0000 0000	0000 0000	uuuu uuuu
FJDT1BH	---- 0000	---- 0000	---- uuuu
DTMINL	0000 0000	0000 0000	uuuu uuuu
DTMINH	---- 0000	---- 0000	---- uuuu
OPC0	---- 0000	---- 0000	---- uuuu
OPC1	0--0 0000	0--0 0000	u--u uuuu
OPOCAL	0010 0000	0010 0000	uuuu uuuu
OVP0C0	000- -000	000- -000	uuu- -uuu
OVP0C1	0001 0000	0001 0000	uuuu uuuu
OVP0C2	0--- 0000	0--- 0000	u--- uuuu
OVP0DA	0000 0000	0000 0000	uuuu uuuu
OVP1C0	000- -000	000- -000	uuu- -uuu
OVP1C1	0001 0000	0001 0000	uuuu uuuu
OVP1C2	0--- 0000	0--- 0000	u--- uuuu
OVP1DA	0000 0000	0000 0000	uuuu uuuu
OVP2C0	000- -000	000- -000	uuu- -uuu
OVP2C1	0001 0000	0001 0000	uuuu uuuu
OVP2C2	---- 00--	---- 00--	---- uu--
OVP2DA	0000 0000	0000 0000	uuuu uuuu
OVP3C0	000- -000	000- -000	uuu- -uuu
OVP3C1	0001 0000	0001 0000	uuuu uuuu
OVP3C2	---- 00--	---- 00--	---- uu--
OVP3DA	0000 0000	0000 0000	uuuu uuuu
OVP4C0	000- -000	000- -000	uuu- -uuu
OVP4C1	0001 0000	0001 0000	uuuu uuuu
OVP4C2	---- 00--	---- 00--	---- uu--
OVP4DA	0000 0000	0000 0000	uuuu uuuu
OVP5C0	000- -000	000- -000	uuu- -uuu
OVP5C1	0001 0000	0001 0000	uuuu uuuu
OVP5C2	---- 00--	---- 00--	---- uu--
OVP5DA	0000 0000	0000 0000	uuuu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
OVP6C0	000- -000	000- -000	uuu- -uuu
OVP6C1	0001 0000	0001 0000	uuuu uuuu
OVP6C2	---- 0000	---- 0000	---- uuuu
OVP6DA	0000 0000	0000 0000	uuuu uuuu
OVP7C0	000- -000	000- -000	uuu- -uuu
OVP7C1	0001 0000	0001 0000	uuuu uuuu
OVP7C2	---- 0000	---- 0000	---- uuuu
OVP7DA	0000 0000	0000 0000	uuuu uuuu
OVP8C0	000- -000	000- -000	uuu- -uuu
OVP8C1	0001 0000	0001 0000	uuuu uuuu
OVP8C2	---- 00--	---- 00--	---- uu--
OVP8DA	0000 0000	0000 0000	uuuu uuuu
PCRL	0000 0000	0000 0000	uuuu uuuu
PCRH	---0 0000	---0 0000	---u uuuu
FJC2	0000 -000	0000 -000	uuuu -uuu

Note: “u” stands for unchanged

“x” stands for unknown

“-” stands for unimplemented

“*”: The UUCR1 and SIMC1 registers share the same memory address while the UBRG and SIMTOC registers share the same memory address. The default value of the UUCR1 or UBRG register can be obtained when the UMD bit is set high by application program after a reset.

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	D7	D6	D5	D4	PC3	PC2	PC1	PC0
PCC	D7	D6	D5	D4	PCC3	PCC2	PCC1	PCC0
PCPU	D7	D6	D5	D4	PCPU3	PCPU2	PCPU1	PCPU0
IECC	IECS7	IECS6	IECS5	IECS4	IECS3	IECS2	IECS1	IECS0

I/O Logic Function Register List

Pull-high Resistors

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

Note that the pull-high resistor can be controlled by the relevant pull-high control registers 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 Px.n Pin pull-high function control

0: Disable

1: Enable

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

Note that the pull-high control bits denoted as "Dn" for port C 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 pin wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

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

• **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 Px.n Pin type selection
0: Output
1: Input

The PxCn bit is used to control the Px.n Pin type selection. Here the "x" is the Port name which can be A, B or C. However, the actual available bits for each I/O Port may be different.

Note that the port control bits denoted as "Dn" for port C should be cleared to 0 to set the corresponding pin as an output after power-on reset. This can prevent the device from consuming power due to input floating states for any unbounded pins.

Read PORT Function

The Read PORT function is used to manage the reading path of the output data from the data latch or I/O pin, which is specially designed for the IEC 60730 self-diagnostic test on the I/O function and A/D paths. There is a register, IECC, which is used to control the READ PORT function. When a specific data pattern, "11001010", is written into the IECC register, the internal signal named

IECM will be set high to enable the READ PORT function. If the READ PORT function is enabled, the value on the corresponding pins will be passed to the accumulator ACC when the read port instruction "mov acc, Px" is executed where the "x" stands for the corresponding I/O port name. Note that the READ PORT mode can only control the input path and will not affect the pin-shared function assignment and the current MCU operation. However, when the IECC register content is set to any other values rather than 11001010B, the IECM internal signal will be cleared to 0 to disable the READ PORT function and the reading path will be set from the latch. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function.

- **IECC Register**

Bit	7	6	5	4	3	2	1	0
Name	IECS7	IECS6	IECS5	IECS4	IECS3	IECS2	IECS1	IECS0
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

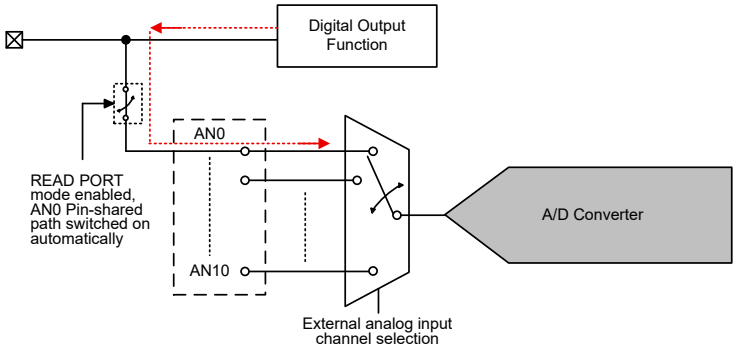
Bit 7~0 **IECS7~IECS0**: READ PORT function enable control
11001010: IECM is high, READ PORT function is enabled
Other values: IECM is low, READ PORT function is disabled

READ PORT Function	Disabled		Enabled	
Port Control Register Bit – PxC.n	1	0	1	0
I/O Function	Pin value	Data latch value	Pin value	
Digital Input Function				
Digital Output Function – except USIM	0			
USIM: SCK/SCL, SDI/SDA/URX/UTX	Pin value			
Analog Function	0			

Note: The value as shown shaded in grey in the above table is the content of the ACC register after "mov a, x" instruction is executed where "x" means the relevant port name.

The additional function of the READ PORT mode is to check the A/D path. When the READ PORT function is disabled, the A/D path from the external pin to the internal analog input will be switched off if the A/D input pin function is not selected by the corresponding selection bits. For the MCU with A/D converter channels, such as A/D AN10~AN0, the desired A/D channel can be switched on by properly configuring the external analog input channel selection bits in the A/D Control Register together with the corresponding analog input pin function is selected. However, the additional function of the READ PORT mode is to force the A/D path to be switched on. For example, when the AN0 is selected as the analog input channel as the READ PORT function is enabled, the AN0 analog input path will be switched on even if the AN0 analog input pin function is not selected. In this way, the AN0 analog input path can be examined by internally connecting the digital output on this shared pin with the AN0 analog input pin switch and then converting the corresponding digital data without any external analog input voltage connected.

Note that the A/D converter reference voltage should be equal to the I/O power supply voltage when examining the A/D path using the READ PORT function.



A/D Channel Input Path Internal Connection

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 Port x Output Function Selection registers, labeled as PxS0 and PxS1,where x is the different port name, and Input Function Selection register, labeled as IFS, are provided, therefore the desired function on multi-function pin-shared pins and a sutiable pin-out location of some function inputs can be configured.

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
IFS	—	IFS6	IFS5	IFS4	IFS3	IFS2	IFS1	IFS0

Pin-shared Function Selection Register List

• **PAS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PAS07~PAS06**: PA3 pin-shared function selection
00: PA3
01: OVP6P07P1
10: AN10
11: OVP6P07P1 & AN10

Bit 5~4 **PAS05~PAS04**: PA2 pin-shared function selection
00: PA2
01: SDI/SDA/URX/UTX
10: PA2
11: PA2

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00: PA1
01: OVP145P0N
10: Reserved
11: Reserved

Bit 1~0 **PAS01~PAS00**: PA0 pin-shared function selection
00: PA0
01: SDO/UTX
10: SCK/SCL
11: PA0

• **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
01: TMCCKO
10: OPINN0
11: Reserved

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

Bit 3~2 **PAS13~PAS12**: PA5 pin-shared function selection
00: PA5
01: OVP8P
10: AN8
11: OVP8P & AN8

Bit 1~0 **PAS11~PAS10**: PA4 pin-shared function selection
00: PA4
01: OVP6P17P0
10: AN9
11: OVP6P17P0 & AN9

• **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/CTCK1/INT0
01: AN3
10: PB3/CTCK1/INT0
11: PB3/CTCK1/INT0

Bit 5~4 **PBS05~PBS04**: PB2 pin-shared function selection
00: PB2/CTCK0
01: PHASE
10: AN2
11: VREF

Bit 3~2 **PBS03~PBS02**: PB1 pin-shared function selection
00: PB1
01: AN1
10: OPARO
11: OPARO & AN1

Bit 1~0 **PBS01~PBS00**: PB0 pin-shared function selection
00: PB0
01: OPINN1
10: AN0
11: PB0

• **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
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 **PBS17~PBS16**: PB7 pin-shared function selection
00: PB7
01: CTP2
10: SCK/SCL
11: AN7

Bit 5~4 **PBS15~PBS14**: PB6 pin-shared function selection
00: PB6
01: SDI/SDA/URX/UTX
10: PCK
11: AN6

Bit 3~2 **PBS13~PBS12**: PB5 pin-shared function selection
00: PB5/PTPI/INT1/FJTR1
01: CTP1B
10: AN5
11: PB5/PTPI/INT1/FJTR1

Bit 1~0 **PBS11~PBS10**: PB4 pin-shared function selection
00: PB4/CTCK2/INT0
01: CTP0B
10: AN4
11: PB4/CTCK2/INT0

• **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PCS07~PCS06**: PC3 pin-shared function selection
00: PC3
01: OVP023P1N
10: Reserved
11: Reserved

Bit 5~4 **PCS05~PCS04**: PC2 pin-shared function selection
00: PC2
01: CTP1
10: SDI/SDA/URX/UTX
11: PCK

Bit 3~2 **PCS03~PCS02**: PC1 pin-shared function selection
00: PC1/INT1/FJTR1
01: CTP0
10: $\overline{\text{SCS}}$
11: SDO/UTX

Bit 1~0 **PCS01~PCS00**: PC0 pin-shared function selection
00: PC0
01: PTP
10: $\overline{\text{SCS}}$
11: SDO/UTX

• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	IFS6	IFS5	IFS4	IFS3	IFS2	IFS1	IFS0
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 **IFS6**: FJTR1 input source pin selection
0: PB5
1: PC1

Bit 5 **IFS5**: INT1 input source pin selection
0: PB5
1: PC1

Bit 4 **IFS4**: INT0 input source pin selection
0: PB4
1: PB3

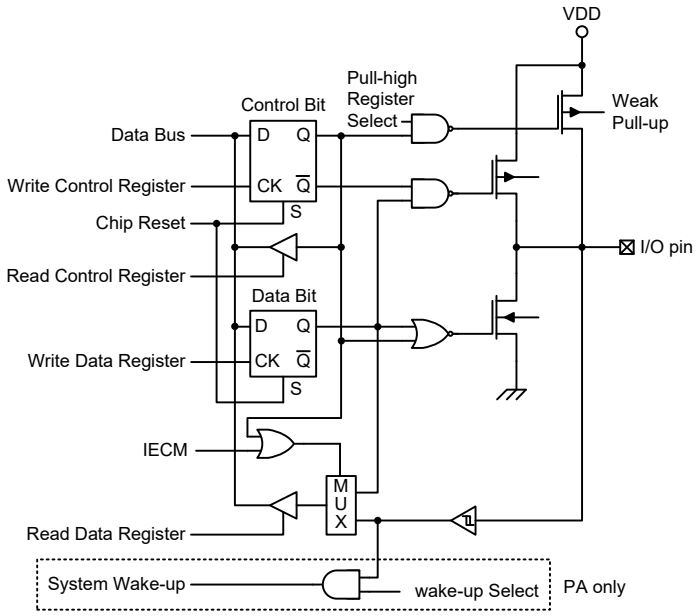
Bit 3~2 **IFS3~IFS2**: SDI/SDA/URX/UTX source pin selection
00: PA2
01: PA2
10: PC2
11: PB6

Bit 1 **IFS1**: SCK/SCL source pin selection
0: PB7
1: PA0

Bit 0 **IFS0**: $\overline{\text{SCS}}$ input source pin selection
0: PC0
1: PC1

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 diagram, it is supplied as a guide only to assist with the functional understanding of the logic function I/O pins. 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, PAC~PCC, are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers, PA~PC, are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

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

Timer Modules – TM

One of the most fundamental functions in any microcontroller 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 TM sections.

Introduction

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

TM 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 xTnCK2~xTnCK0 bits in the xTMn control registers, where “x” stands for C or P type TM and “n” stands for the specific TM serial number. 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. For the CTMn, the clock source also can be from the external CTCKn pin. The CTCKn pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

TM Interrupts

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

TM External Pins

The Compact TM has one input pin, with the label CTCKn. The CTCKn input pin is essentially a clock source for the CTMn and is selected using the CTnCK2~CTnCK0 bits in the CTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The CTCKn input pin can be chosen to have either a rising or falling active edge.

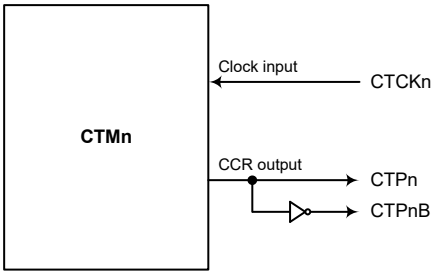
The Periodic TM also has an input pin, with the label of PTPI. The PTPI pin is the capture input pin whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTIO1~PTIO0 bits in the PTMC1 register.

The TMs each have one or two output pins, xTPn and xTPnB. The xTPnB is the inverted signal of the xTPn output. 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 xTPn or xTPnB output pin is also the pin where the TM generates the PWM output waveform.

As the TM input and output pins are pin-shared with other functions, the TM input or 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.

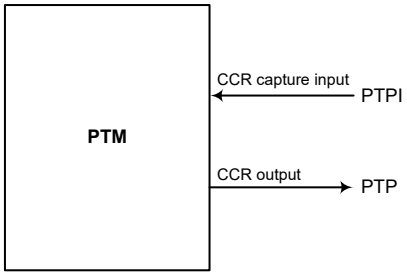
CTMn(n=0~2)		PTM	
Input	Output	Input	Output
CTCK0	CTP0, CTP0B	PTPI	PTP
CTCK1	CTP1, CTP1B		
CTCK2	CTP2		

TM External Pins



Note: CTPnB is not available for CTM2.

CTMn Function Pin Block Diagram (n=0~2)

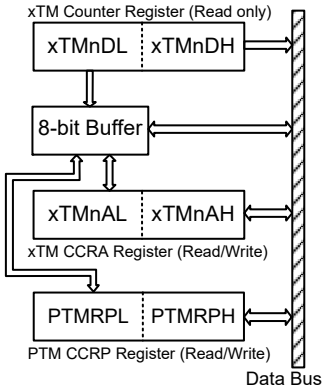


PTM Function Pin Block Diagram

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 using the following access procedures. Accessing the CCRA or CCRP low byte registers without following these access procedures will result in unpredictable values.

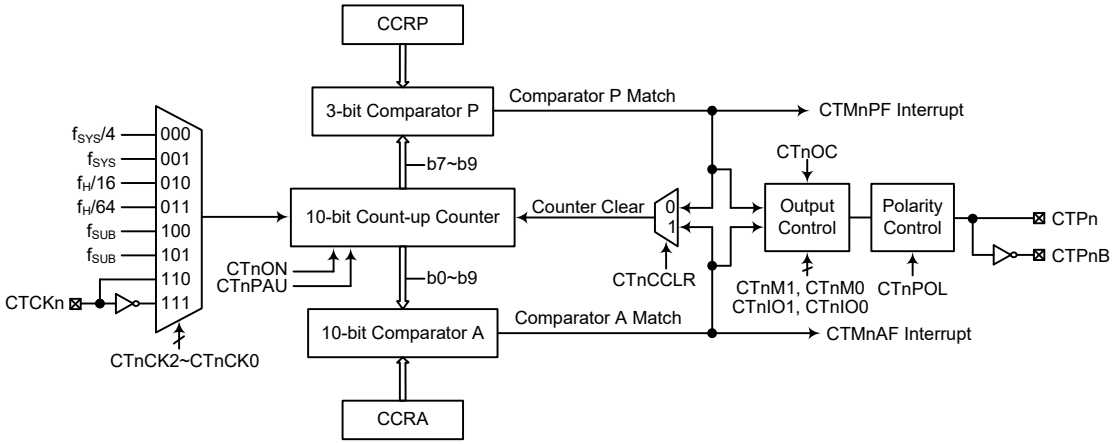


The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - Step 1. Write data to Low Byte xTMnAL or PTMRPL
    - Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte xTMnAH 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 xTMnDH, xTMnAH or PTMRPH
    - Here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - Step 2. Read data from the Low Byte xTMnDL, xTMnAL or PTMRPL
    - This step reads data from the 8-bit buffer.

Compact Type TM – CTM

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



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

2. For CTM2, there is no CTPnB external output pin.

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

Compact Type TM Operation

At 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 is 3-bit wide whose value is compared with the highest three bits in the counter while the CCRA is 10-bit wide and therefore compares 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 CTnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a CTMn interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control two output pins. 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 the three CCRP bits.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CTMnC0	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
CTMnC1	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
CTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnDH	—	—	—	—	—	—	D9	D8
CTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnAH	—	—	—	—	—	—	D9	D8

10-bit Compact Type TM Register List (n=0~2)

- **CTMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
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 **CTnPAU**: CTMn Counter Pause Control
0: Run
1: Pause

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

Bit 6~4 **CTnCK2~CTnCK0**: CTMn counter clock selection
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: CTCKn rising edge clock
111: CTCKn falling edge clock

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

Bit 3 **CTnON**: CTMn Counter On/Off Control
0: Off
1: On

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

If the CTMn is in the Compare Match Output Mode or the PWM Output Mode then the CTMn output pin will be reset to its initial condition, as specified by the CTnOC bit, when the CTnON bit changes from low to high.

Bit 2~0 **CTnRP2~CTnRP0**: CTMn CCRP 3-bit register, compared with the CTMn Counter bit 9 ~ bit 7 Comparator P Match Period
000: 1024 CTMn clocks
001~111: (1~7)×128 CTMn 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 CTnCCLR bit is set to zero. Setting the CTnCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest 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.

- **CTMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **CTnM1~CTnM0**: CTMn operating mode selection
00: Compare Match Output Mode
01: Undefined
10: PWM Output Mode
11: Timer/Counter Mode

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

Bit 5~4 **CTnIO1~CTnIO0**: CTMn external pin (CTPn) 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

These two bits are used to determine how the CTMn external pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the CTMn is running. These two bits have no effect if the CTMn is in the Timer/Counter Mode.

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

In the PWM Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the CTnIO1 and CTnIO0 bits only after the CTMn has been switched off. Unpredictable PWM outputs will occur if the CTnIO1 and CTnIO0 bits are changed when The CTMn is running.

Bit 3 **CTnOC**: CTPn Output control bit

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

PWM Output Mode
0: Active low
1: Active high

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

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

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

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

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

Bit 0 **CTnCCLR**: CTMn Counter clear condition selection
0: CTMn Comparatror P match
1: CTMn Comparatror A match

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

• CTMnDL Register

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

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

• CTMnDH 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**: CTMn Counter High Byte Register bit 1 ~ bit 0
CTMn 10-bit Counter bit 9 ~ bit 8

• **CTMnAL Register**

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

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

• **CTMnAH 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**: CTMn CCRA High Byte Register bit 1 ~ bit 0
CTMn 10-bit CCRA bit 9 ~ bit 8

Compact Type TM Operating Modes

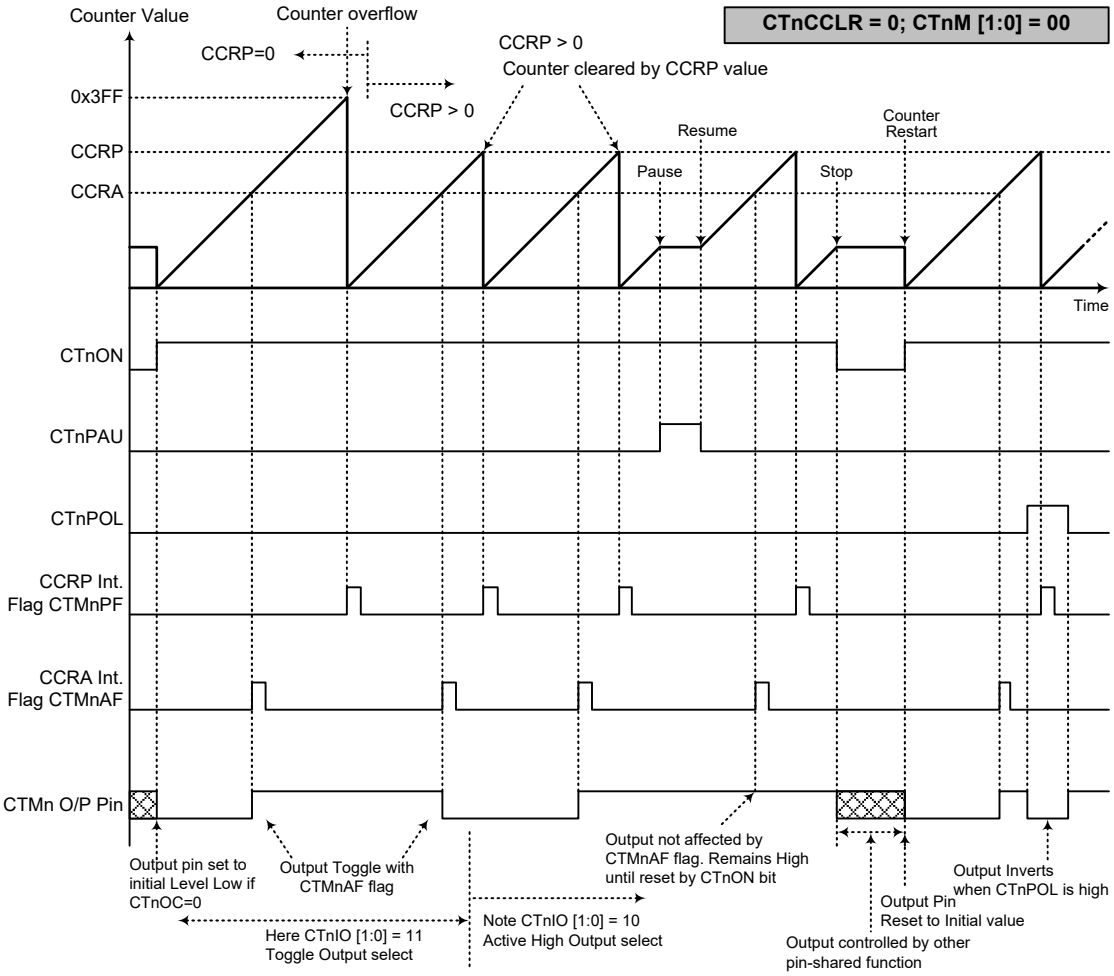
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

Compare Match Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match occurs from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMnAF and CTMnPF interrupt request flags for the Comparator A and Comparator P respectively, will both be generated.

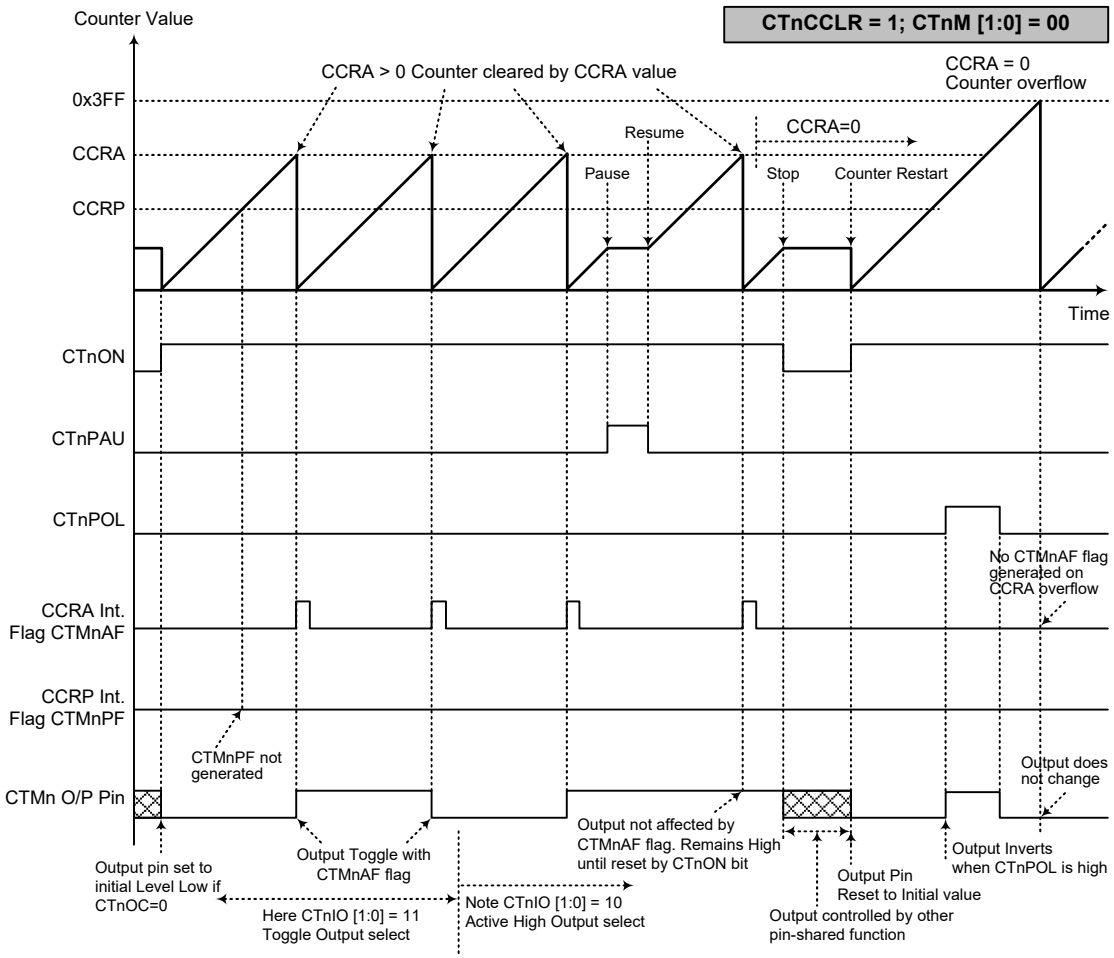
If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is high no CTMnPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the CTMnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the CTMn output pin will change state. The CTMn output pin condition however only changes state when a CTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTMn output pin. The way in which the CTMn output pin changes state are determined by the condition of the CTnIO1 and CTnIO0 bits in the CTMnC1 register. The CTMn output pin can be selected using the CTnIO1 and CTnIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the CTMn output pin, which is setup after the CTnON bit changes from low to high, is setup using the CTnOC bit. Note that if the CTnIO1 and CTnIO0 bits are zero then no pin change will take place.



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

Note: 1. With CTnCCLR=0, a Comparator P match will clear the counter
2. The CTMn output pin is controlled only by the CTMnAF flag
3. The output pin is reset to initial state by a CTnON bit rising edge



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

Note: 1. With CTnCCLR=1, a Comparator A match will clear the counter
2. The CTMn output pin is controlled only by the CTMnAF flag
3. The output pin is reset to initial state by a CTnON bit rising edge
4. The CTMnPF flag is not generated when CTnCCLR=1

Timer/Counter Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the CTMn output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the CTMn output pins are not used in this mode, the pins can be used as a general I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to 10 respectively. The PWM function within the CTMn is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the CTMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the CTnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTnDPX bit in the CTMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

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

• 10-bit CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=0

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

If f_{SYS}=16MHz, CTMn clock source is f_{SYS}/4, CCRP=4, CCRA=128,

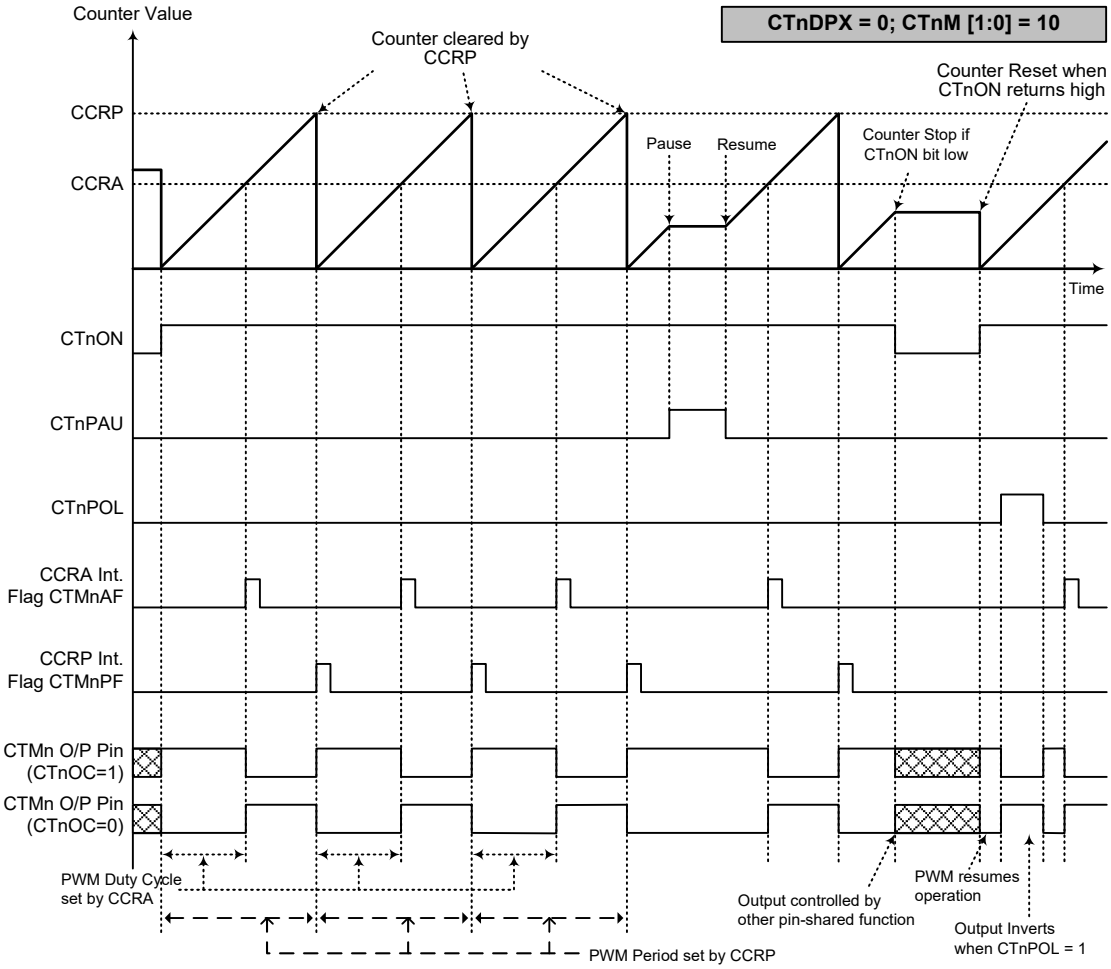
The CTMn PWM output frequency=(f_{SYS}/4)/(4×128)=f_{SYS}/2048=8kHz, duty=128/(4×128)=25%.

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 CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=1

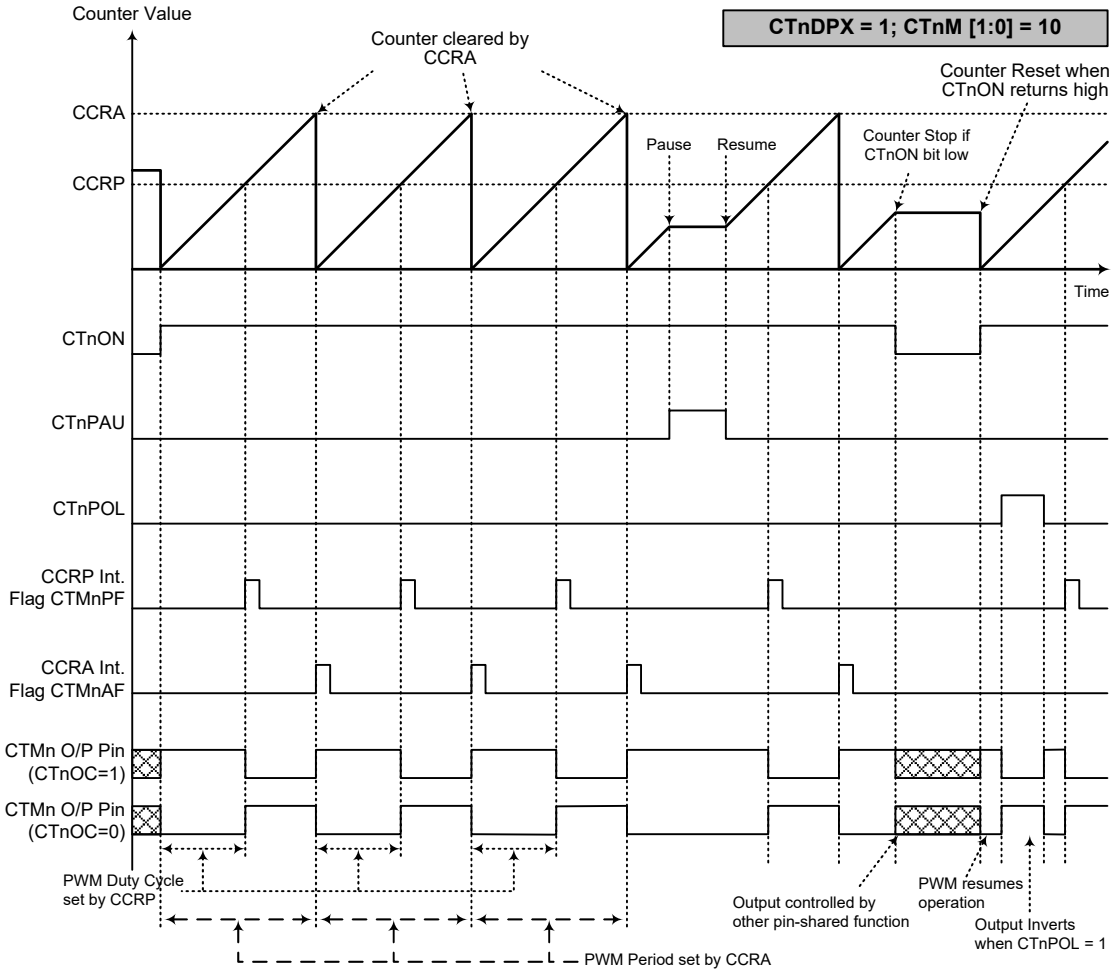
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

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



PWM Output Mode – CTnDPX=0 (n=0~2)

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

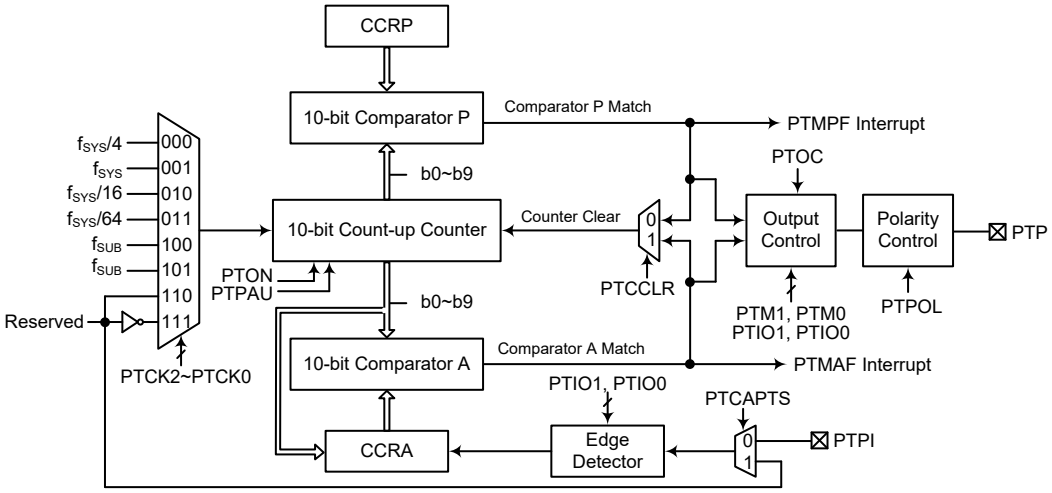


PWM Output Mode – CTnDPX=1(n=0~2)

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

Periodic Type TM – PTM

The Periodic Type TM contains five operating modes, which are Compare Match Output, Timer/Event Counter, Capture Input, Single Pulse Output and PWM Output modes.



Note: The PTM external pins are pin-shared with other functions, therefore before using the PTM function, ensure that the pin-shared function registers have been 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.

Periodic Type TM Block Diagram

Periodic TM Operation

The Periodic Type TM core is a 10-bit count-up counter which is driven by a user selectable internal 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 10-bit wide.

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 more than one output pin. All operating setup conditions are selected using relevant internal registers.

Periodic Type TM Register Description

Overall operation of the Periodic Type TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10-bit value, while two read/write register pairs exist to store the internal 10-bit CCRA value 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**: Select PTM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: Undefined, cannot be used
111: Undefined, cannot be used
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, clearing the bit disables the PTM. Clearing this bit to zero will stop the counter from counting and turn off the PTM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.
If the PTM is in the Compare Match Output Mode, PWM Output Mode or Single Pulse Output Mode then the PTM output pin will be reset to its initial condition, as specified by the PTOC bit, when the PTON bit changes from low to high.

Bit 2~0 Unimplemented, read as “0”

• **PTMC1 Register**

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

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

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

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

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

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

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

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. These two bits have no effect if the PTM is in the Timer/Counter Mode.

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

In the PWM Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the PTIO1 and PTIO0 bits only after the TM 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 bit

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

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

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

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

Bit 2 **PTPOL**: PTP Output polarity Control
0: Non-invert
1: Invert

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**: Select PTM Counter clear condition
0: PTM Comparator P match
1: PTM Comparator A match

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

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

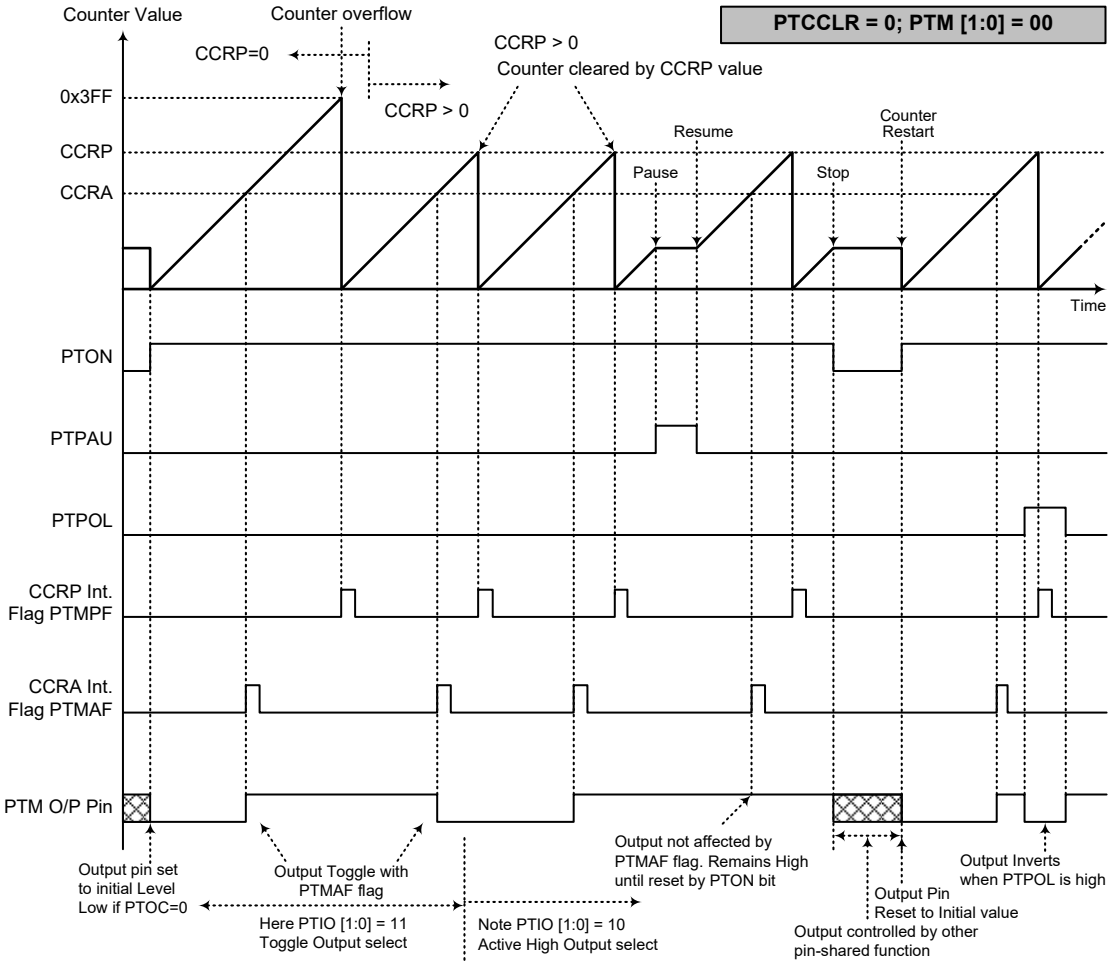
Compare Match Output Mode

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

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

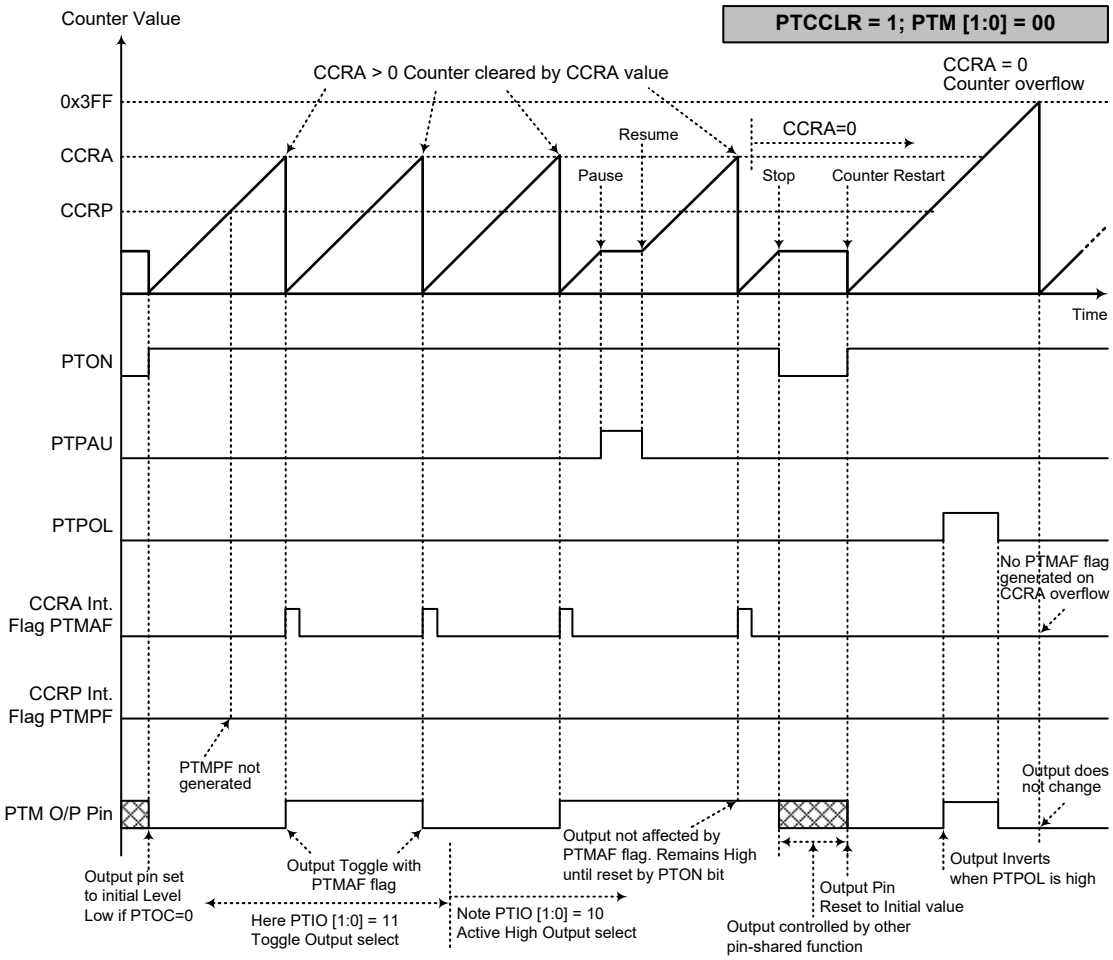
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the PTMAF interrupt request flag will not be generated.

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 TM 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 general I/O pin or other pin-shared function.

PWM Output Mode

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

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

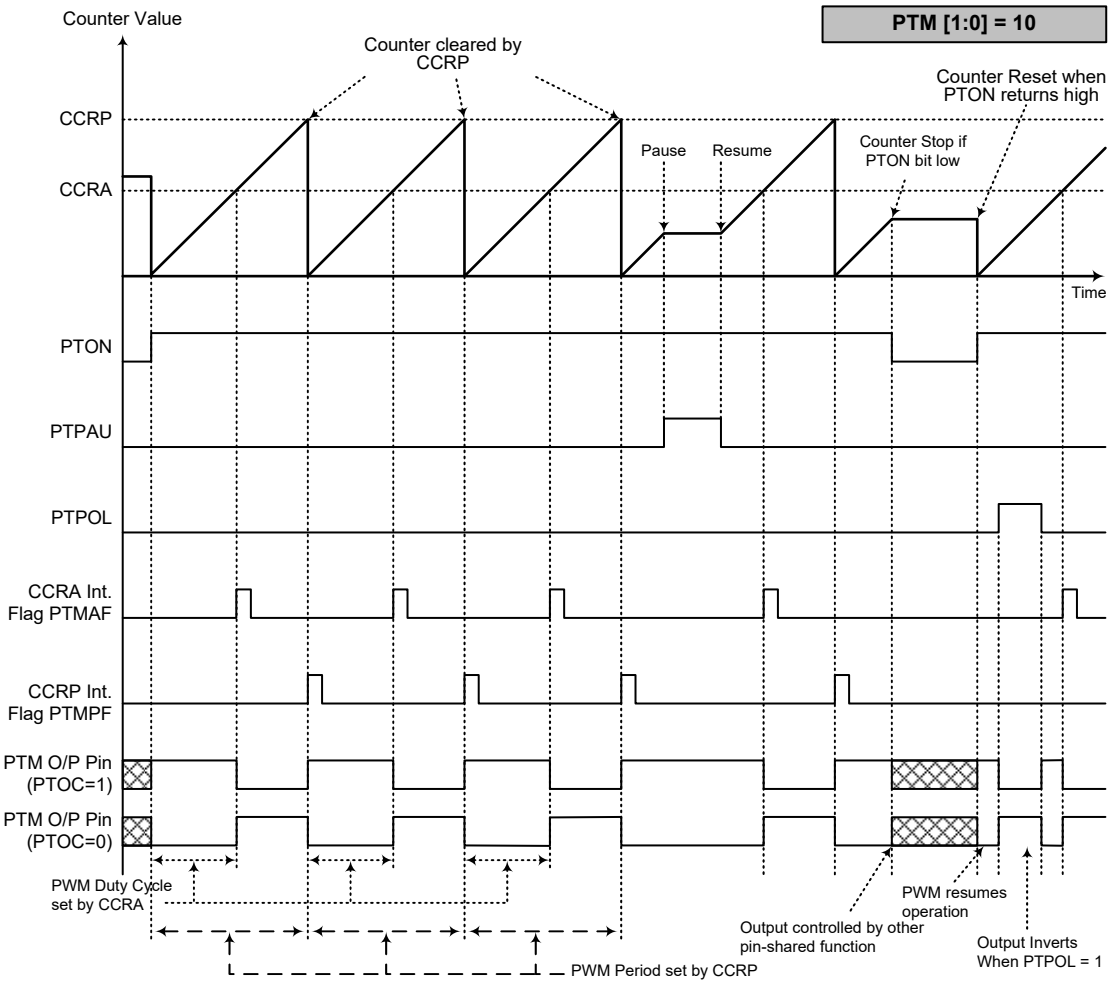
- **10-bit PTM, PWM Output Mode, Edge-aligned Mode**

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

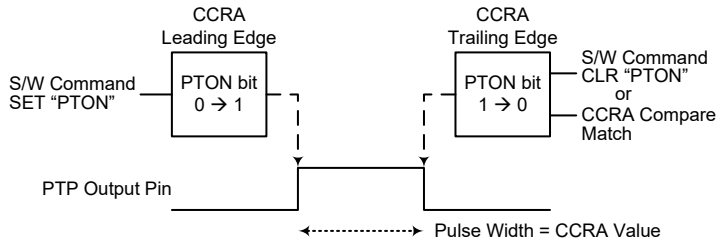
Note: 1. Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTIO[1:0]=00 or 01
4. The PTCCLR bit has no influence on PWM operation

Single Pulse Output Mode

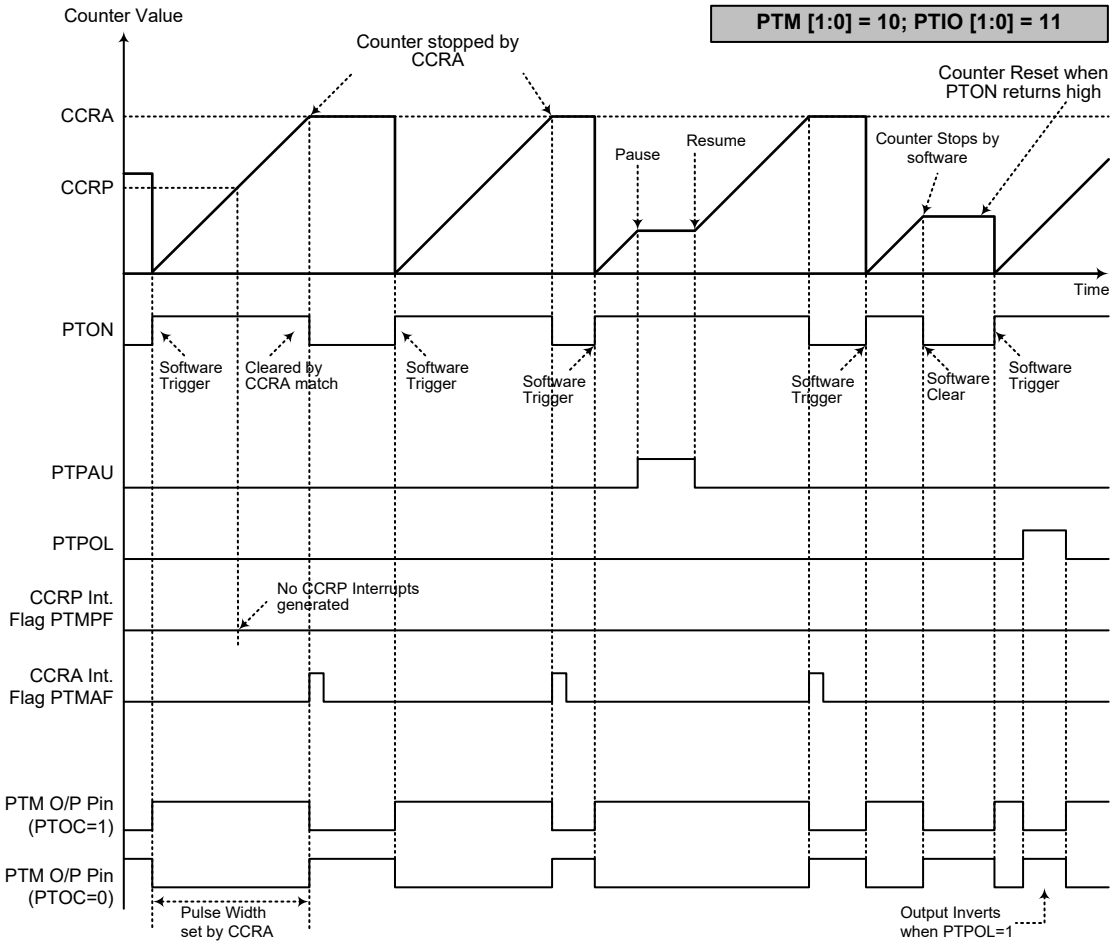
To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

The trigger for the pulse output leading edge is a low to high transition of the PTON bit, which can be implemented using the application program. 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 Mode CCRP is not used. The PTCCLR bit is not used in this Mode.



Single Pulse Generation



Single Pulse Output Mode

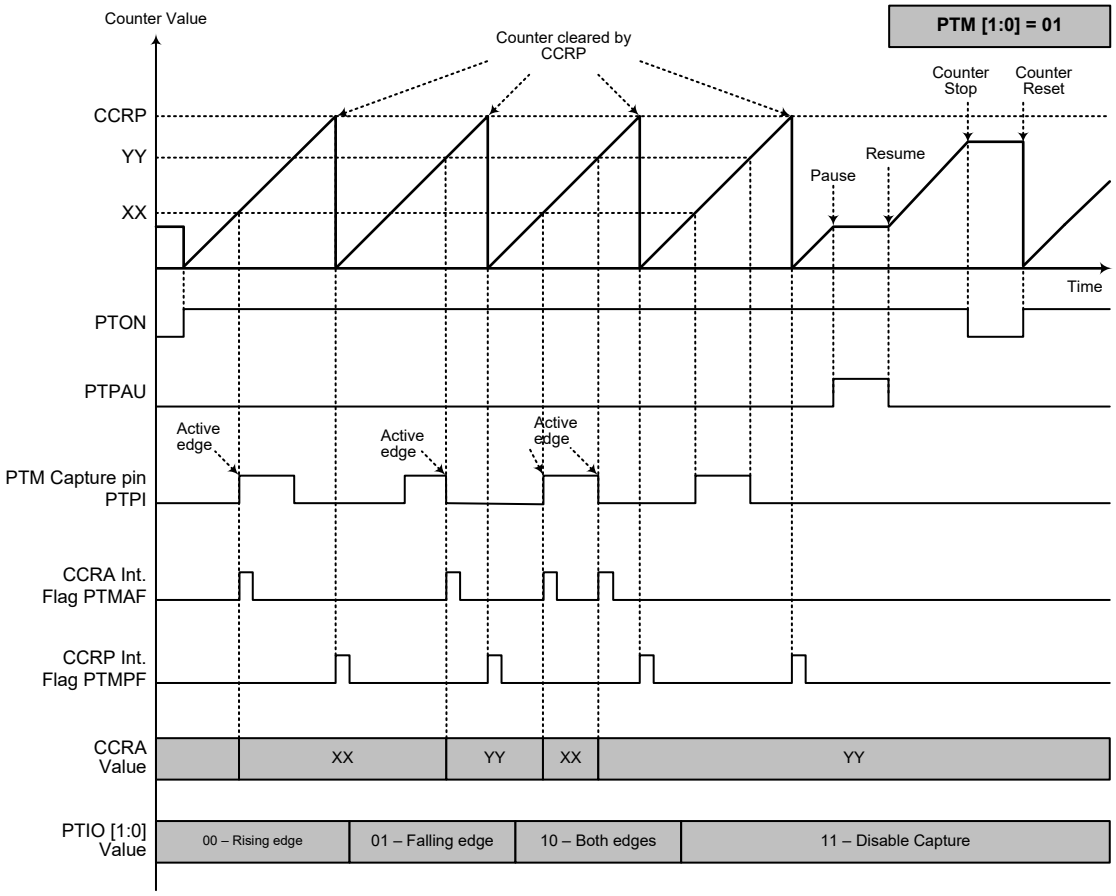
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse triggered by setting the PTON bit high
4. In the Single Pulse Mode, PTIO[1:0] must be set to “11” and cannot be changed

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

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 CCRA register by an active capture edge, the PTMAF flag will be set high after 0.5 timer clock period. The delay time from the active capture edge received to the action of latching counter value to CCRA register is less than 1.5 timer clock periods.



Capture Input Mode

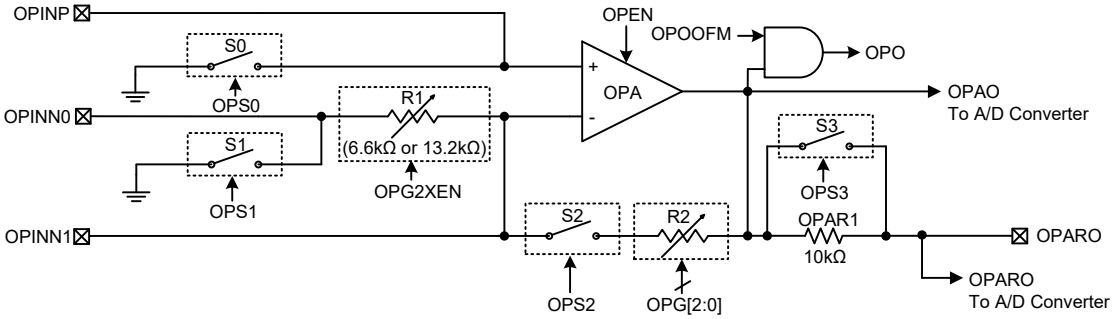
Note: 1. PTM[1:0]=01 and active edge set by the PTIO[1:0] bits
2. A PTM Capture input pin active edge transfers the counter value to CCRA
3. PTCCLR bit not used
4. No output function – PTOC and PTPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when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

Operational Amplifier – OPAMP

This device includes an operational amplifier function, OPAMP, which can produce an output potential that is multiple times larger than the potential difference between its input terminals. By integrating the OPAMP electronic circuitry into the microcontroller, the need for external components is reduced greatly.

Operational Amplifier Operation

The Operational Amplifier can be used for signal amplification according to specific user requirements by several internal registers. The gain is selectable by using the OPG2~OPG0 bits together with the OPG2XEN bit in the OPC0 register. The amplified output can be directly output on the OPARO pin, and also be internally connected to the A/D converter internal channel for measurements. The following block diagram illustrates the internal OPAMP function.



Operational Amplifier Circuit

OPAMP Register Description

The internal operational amplifier normal operation and input offset voltage calibration function are controlled by three registers. The OPC0 register is used to control the switches on or off. The OPC1 register is used to control the OPAMP function enable or disable and select the gain. The OPO bit in the OPC1 register together with the OPOCAL register are used in the offset calibration procedure.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OPC0	—	—	—	—	OPS3	OPS2	OPS1	OPS0
OPC1	OPEN	—	—	OPO	OPG2XEN	OPG2	OPG1	OPG0
OPOCAL	OPOOFM	OPORSP	OPOOF5	OPOOF4	OPOOF3	OPOOF2	OPOOF1	OPOOF0

OPAMP Register List

- **OPC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	OPS3	OPS2	OPS1	OPS0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **OPS3**: OPAMP switch S3 on/off control
0: Off
1: On

Bit 2 **OPS2**: OPAMP switch S2 on/off control
0: Off
1: On

Bit 1 **OPS1**: OPAMP switch S1 on/off control
0: Off
1: On

Bit 0 **OPS0**: OPAMP switch S0 on/off control
0: Off
1: On

• **OPC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	OPEN	—	—	OPO	OPG2XEN	OPG2	OPG1	OPG0
R/W	R/W	—	—	R	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 **OPEN**: OPAMP function enable/disable control
0: Disable
1: Enable

Bit 6~5 Unimplemented, read as "0"

Bit 4 **OPO**: OPAMP output status (positive logic)
When OPOOFM bit in the OPOCAL register is set to 1, OPO is defined as OPAMP output status, refer to Offset Calibration Procedure. This bit is read only.

Bit 3 **OPG2XEN**: R2/R1 ratio doubling enable control
0: Disable (R1=13.2kΩ)
1: Enable (R1=6.6kΩ)

Bit 2~0 **OPG2~OPG0**: R2/R1 ratio selection
000: R2/R1=2 (OPG2XEN=0); R2/R1=4 (OPG2XEN=1)
001: R2/R1=5 (OPG2XEN=0); R2/R1=10 (OPG2XEN=1)
010: R2/R1=6 (OPG2XEN=0); R2/R1=12 (OPG2XEN=1)
011: R2/R1=7 (OPG2XEN=0); R2/R1=14 (OPG2XEN=1)
100: R2/R1=20 (OPG2XEN=0); R2/R1=40 (OPG2XEN=1)
101: R2/R1=25 (OPG2XEN=0); R2/R1=50 (OPG2XEN=1)
110: R2/R1=30 (OPG2XEN=0); R2/R1=60 (OPG2XEN=1)
111: R2/R1=35 (OPG2XEN=0); R2/R1=70 (OPG2XEN=1)
Note that the gain selection using OPG2~OPG0 and OPG2XEN bits actually is implemented by the internal resistors, R1 and R2. Therefore when selecting the gain, it is necessary to ensure that the S1 switch is on or select the OPINN0 input together with the S2 switch being on. Only in this way can gain accuracy be guaranteed.

• **OPOCAL Register**

Bit	7	6	5	4	3	2	1	0
Name	OPOOFM	OPORSP	OPOOF5	OPOOF4	OPOOF3	OPOOF2	OPOOF1	OPOOF0
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 **OPOOFM**: OPAMP normal operation or input offset voltage calibration mode selection
0: Normal operation
1: Input offset voltage calibration mode

Bit 6 **OPORSP**: OPAMP input offset voltage calibration reference selection
0: Select inverting input as the reference input
1: Select non-inverting input as the reference input

Bit 5~0 **OPOOF5~OPOOF0**: OPAMP input offset voltage calibration control

Input Voltage Range

Together with different PGA operating mode, the input voltage on the OPAMP pins can be positive or negative for flexible applications. There are two operating mode examples as the following.

- For V_{IN}>0, the PGA operates in the non-inverting mode and the output voltage of the PGA is:

$$V_{OPGA}=(1+R_2/R_1)\times V_{IN}$$

- When the S0 and S2 switches are ON, S1 and S3 switches are OFF and the input node is OPINN0. For 0>V_{IN}>-0.2V, the PGA operates in the inverting mode, the output voltage of the PGA is:

$$V_{OPGA}=-(R_2/R_1)\times V_{IN}$$

Note: If V_{IN} is negative, it cannot be lower than -0.2V which may result in current leakage.

Offset Calibration Function

This OPAMP is equipped with an input offset calibration function. The calibrated data is stored in OPOOF[5:0] bits. OPOOFM is calibration mode control bit and OPORSP is used to specify which of the OPINP or OPINN1 is taken as the reference voltage input in the calibration mode. OPINP and OPINN1 are input pair of OPAMP and OPAO is OPAMP analog output voltage. The OPAMP digital output flag is OPO, which is used for OPAMP calibration mode.

Offset Calibration Procedure

Note that as the OPAMP input pins are pin-shared with other functions, these pins should be configured as OPAMP inputs first. The input offset calibration procedures are summarized as follows.

Step 1. Set OPOOFM=1 and OPORSP=1, the OPAMP is now under the offset calibration mode. To make sure V_{OS} as minimize as possible after calibration, the input reference voltage in calibration mode should be the same as the input DC operating voltage in the normal mode.

Step 2. Set OPOOF[5:0]=000000 and then read the OPO flag bit after a certain delay.

Step 3. Increase the OPOOF[5:0] value by 1 and then read the OPO flag bit after a certain delay.

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

If the OPO bit state has changed, record the OPOOF[5:0] value as V_{OS1} and then go to Step4.

Step 4. Set OPOOF[5:0]=111111 and then read the OPO flag bit after a certain delay.

Step 5. Decrease the OPOOF[5:0] value by 1 and then read the OPO flag bit after a certain delay.

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

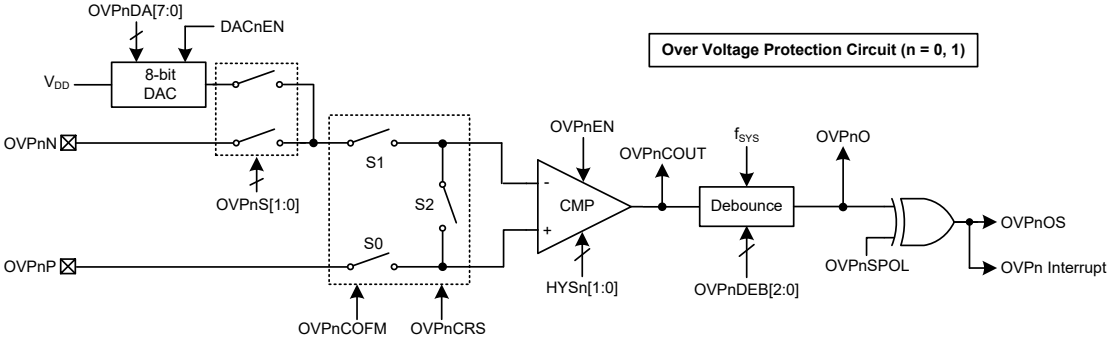
If the OPO bit state has changed, record the OPOOF[5:0] value as V_{OS2} and then go to Step 6.

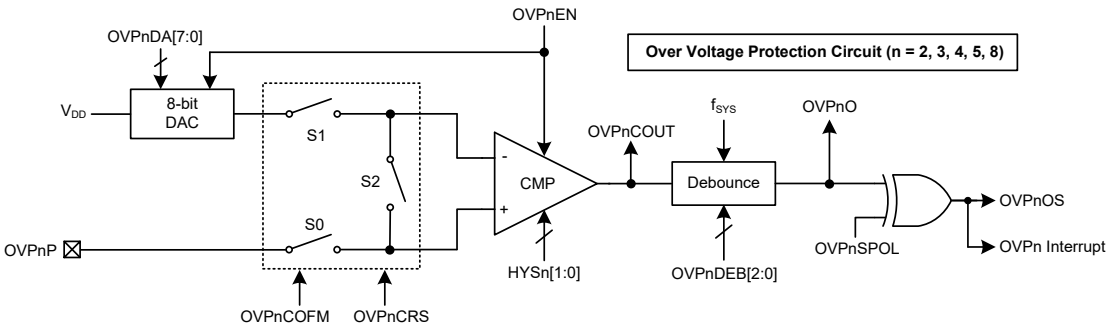
Step 6. Restore the Operational Amplifier input offset calibration value V_{OS} into the OPOOF[5:0] bits. The offset Calibration procedure is now finished.

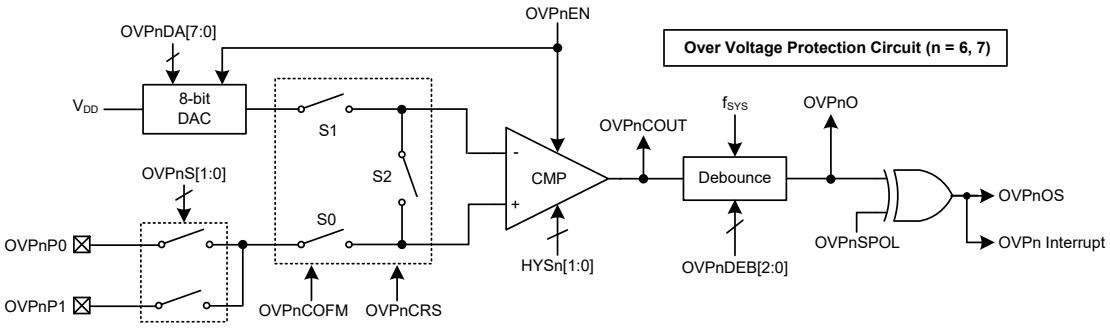
Where $V_{OS}=(V_{OS1}+V_{OS2})/2$. If $(V_{OS1}+V_{OS2})/2$ is not integral, discard the decimal.

Over Voltage Protection – OVP

The device includes nine groups of over voltage protection circuits, abbreviated as OVP0~OVP8. The OVP circuits provide protection mechanisms or generate output signals for different applications. To prevent the operating voltage from exceeding a specific level, the voltage on the OVPn input pin is compared with a reference voltage generated by an 8-bit DAC or an input voltage from the other OVPn input pin. When a preset over voltage event occurs, the OVPn will issue an output signal OVPnO to indicate the source voltage is over specifications. If the interrupt is enabled, an OVPn interrupt will be generated. The OVPnOS output can be set to be the same or opposite with the OVPnO signal by the OVPnSPOL control bit.







Note: 1. As the OVPn external pins are pin-shared with other functions, before using the OVPn function, ensure that the OVPn pin functions are selected using the corresponding Pin-shared Function Selection Registers.

2. The OVP1P, OVP4P, OVP5P and OVP0N pins shown in the figures above share the same external pin with the name of OVP145P0N in the device. The OVP0P, OVP2P, OVP3P and OVP1N pins in the figures above share the same external pin with the name of OVP023P1N in the device. The OVP6P0 and OVP7P1 pins shown in the figures above share the same external pin with the name of OVP6P07P1 in the device. The OVP6P1 and OVP7P0 pins shown in the figures above share the same external pin with the name of OVP6P17P0 in the device. Refer to the Pin Description chapter for more details.

3. The on/off control using the OVPnCOFM and OVPnCRS bits for the S0, S1 and S2 switches in the figures above is summarised below.

OVPnCOFM	OVPnCRS	S0	S1	S2
0	x	ON	ON	OFF
1	0	OFF	ON	ON
1	1	ON	OFF	ON

"x": Don't care

Over Voltage Protection Registers

Overall operation of the over voltage protection is controlled using several registers. As the control circuits of the OVP0~OVP8 have different structures, some register bits are different, such as the OVPnEN bit, the DACnEN bit and the OVPnS[1:0] bit selections. Refer to register definitions for more details.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OVPnC0	OVPnO	OVPnSPOL	OVPnEN	—	—	OVPnDEB2	OVPnDEB1	OVPnDEB0
OVPnC1	OVPnCOUT	OVPnCOFM	OVPnCRS	OVPnCOF4	OVPnCOF3	OVPnCOF2	OVPnCOF1	OVPnCOF0
OVPnC2	DACnEN	—	—	—	HYSn1	HYSn0	OVPnS1	OVPnS0
OVPnDA	D7	D6	D5	D4	D3	D2	D1	D0

OVPn Register List (n=0, 1)

Register Name	Bit							
	7	6	5	4	3	2	1	0
OVPnC0	OVPnO	OVPnSPOL	OVPnEN	—	—	OVPnDEB2	OVPnDEB1	OVPnDEB0
OVPnC1	OVPnCOUT	OVPnCOFM	OVPnCRS	OVPnCOF4	OVPnCOF3	OVPnCOF2	OVPnCOF1	OVPnCOF0
OVPnC2	—	—	—	—	HYSn1	HYSn0	—	—
OVPnDA	D7	D6	D5	D4	D3	D2	D1	D0

OVPn Register List (n=2, 3, 4, 5, 8)

Register Name	Bit							
	7	6	5	4	3	2	1	0
OVPnC0	OVPnO	OVPnSPOL	OVPnEN	—	—	OVPnDEB2	OVPnDEB1	OVPnDEB0
OVPnC1	OVPnCOUT	OVPnCOFM	OVPnCRS	OVPnCOF4	OVPnCOF3	OVPnCOF2	OVPnCOF1	OVPnCOF0
OVPnC2	—	—	—	—	HYSn1	HYSn0	OVPnS1	OVPnS0
OVPnDA	D7	D6	D5	D4	D3	D2	D1	D0

OVPn Register List (n=6, 7)

• OVPnC0 Register (n=0~8)

Bit	7	6	5	4	3	2	1	0
Name	OVPnO	OVPnSPOL	OVPnEN	—	—	OVPnDEB2	OVPnDEB1	OVPnDEB0
R/W	R	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7 **OVPnO**: OVPn comparator output bit after debounce
0: Positive input voltage < negative input voltage
1: Positive input voltage > negative input voltage

Bit 6 **OVPnSPOL**: OVPn debounced output signal polarity control
0: Non-invert
1: Invert

This bit determines the OVPnOS signal condition when the OVPnO bit changes state from low to high or high to low. Note that this bit is always effective even when OVPnEN bit is 0.

Bit 5 **OVPnEN**: OVPn function enable control bit
0: Disable
1: Enable

If the OVPnEN bit is cleared to 0, the over voltage protection function is disabled and no power will be consumed. For OVP2~OVP8, clearing OVPnEN bit results in that the comparator and D/A converter of the OVPn are switched off. For OVP0 or OVP1, when the OVPnEN bit is cleared to 0, only the comparator is switched off while the D/A converter on/off is controlled by the DACnEN bit.

When the OVPnEN bit is 0, the OVPnCOUT output is in a pull-low state. As the OVPnOS signal state is affected by OVPnSPOL bit settings, it will be in a pull-low state if the OVPnSPOL bit is 0 and in a high state if the OVPnSPOL bit is 1.

Bit 4~3 Unimplemented, read as "0"

Bit 2~0 **OVPnDEB2~OVPnDEB0**: OVPn comparator debounce time control bits
000: No debounce
001: (1~2)×t_{DEB}
010: (3~4)×t_{DEB}
011: (7~8)×t_{DEB}
100: (15~16)×t_{DEB}
101: (31~32)×t_{DEB}
110: (63~64)×t_{DEB}
111: (127~128)×t_{DEB}

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

• OVPnC1 Register (n=0~8)

Bit	7	6	5	4	3	2	1	0
Name	OVPnCOUT	OVPnCOFM	OVPnCRS	OVPnCOF4	OVPnCOF3	OVPnCOF2	OVPnCOF1	OVPnCOF0
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 **OVPnCOUT**: OVPn comparator output before debounce
0: Positive input voltage < negative input voltage
1: Positive input voltage > negative input voltage

When OVPnEN bit is 0 the OVPnCOUT output is in a pull-low state.

Bit 6 **OVPnCOFM**: OVPn comparator operation mode selection
0: Normal operation
1: Input offset voltage calibration mode

Bit 5 **OVPnCRS**: OVPn comparator input offset voltage calibration reference selection bit
0: Input reference voltage comes from negative input
1: Input reference voltage comes from positive input

Bit 4~0 **OVPnCOF4~OVPnCOF0**: OVPn comparator input offset voltage calibration control bits

This 5-bit field is used to perform the comparator input offset calibration operation and the value for the OVPn comparator input offset calibration can be restored into this bit field. More detailed information is described in the "Comparator Input Offset Cancellation" section.

• **OVPnC2 Register (n=0, 1)**

Bit	7	6	5	4	3	2	1	0
Name	DACnEN	—	—	—	HYSn1	HYSn0	OVPnS1	OVPnS0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **DACnEN**: OVPn D/A Converter enable control bit
0: Disable
1: Enable

Bit 6~4 Unimplemented, read as "0"

Bit 3~2 **HYSn1~HYSn0**: OVPn comparator hysteresis voltage window control
Refer to Over Voltage Protection Electrical Characteristics table for details.

Bit 1~0 **OVPnS1~OVPnS0**: OVPn negative input selection
00: No choose
01: OVPnN
10: DAC output
11: No choose

• **OVPnC2 Register (n=2, 3, 4, 5, 8)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HYSn1	HYSn0	—	—
R/W	—	—	—	—	R/W	R/W	—	—
POR	—	—	—	—	0	0	—	—

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HYSn1~HYSn0**: OVPn comparator hysteresis voltage window control
Refer to Over Voltage Protection Electrical Characteristics table for details.

Bit 1~0 Unimplemented, read as "0"

• **OVP6C2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HYS61	HYS60	OVP6S1	OVP6S0
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 **HYS61~HYS60**: OVP6 comparator hysteresis voltage window control
Refer to Over Voltage Protection Electrical Characteristics table for details.

Bit 1~0 **OVP6S1~OVP6S0**: OVP6 input selection
00: No choose
01: OVP6P07P1
10: OVP6P17P0
11: No choose

• **OVP7C2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HYS71	HYS70	OVP7S1	OVP7S0
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 **HYS71~HYS70**: OVP7 comparator hysteresis voltage window control
Refer to Over Voltage Protection Electrical Characteristics table for details.

Bit 1~0 **OVP7S1~OVP7S0**: OVP7 input selection
00: No choose
01: OVP6P17P0
10: OVP6P07P1
11: No choose

• **OVPnDA Register (n=0~8)**

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**: OVPn DAC output voltage control bits
DAC $V_{OUT}=(V_{DD}/256)\times OVPnDA[7:0]$

Comparator Input Offset Calibration

As the OVPn inputs are pin-shared with other pin functions, they should be configured as OVPn input pin functions first. It needs to be noted that before offset calibration, the hysteresis voltage should be zero by setting the HYSn[1:0] bits to 00. For comparator input offset calibration, the procedures are summarised as the following.

Step1: Set OVPnCOFM=1, OVPnCRS=0, the OVPn comparator is now under input offset calibration mode, S1 and S2 on.

Step2: Set the OVPnDA[7:0] bits. 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.

Step3: Set OVPnCOF[4:0]=00000 and then read the OVPnCOUT bit status after a certain delay.

Step4: Increase the OVPnCOF[4:0] value by 1 and then read the OVPnCOUT bit status after a certain delay.

If the OVPnCOUT state has not changed, repeat Step4 until the OVPnCOUT bit state has changed.

If the OVPnCOUT state has changed, record the OVPnCOF[4:0] value as V_{OS1} and then go to Step5.

Step5: Set OVPnCOF[4:0]=11111 and then read the OVPnCOUT bit status after a certain delay.

Step6: Decrease the OVPnCOF[4:0] value by 1 and then read the OVPnCOUT bit status after a certain delay.

If the OVPnCOUT state has not changed, repeat Step6 until the OVPnCOUT bit state has changed.

If the OVPnCOUT state has changed, record the OVPnCOF[4:0] value as V_{OS2} and then go to Step7.

Step7: Restore $V_{OS}=(V_{OS1}+V_{OS2})/2$ to the OVPnCOF[4:0] bits. The calibration is finished.

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

Note: For n=0 or 1, when using the DAC for the calibration, it is also necessary to set the DACnEN bit to 1 and the OVPnS[1:0] bits to 10.

Analog to Digital Converter – ADC

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

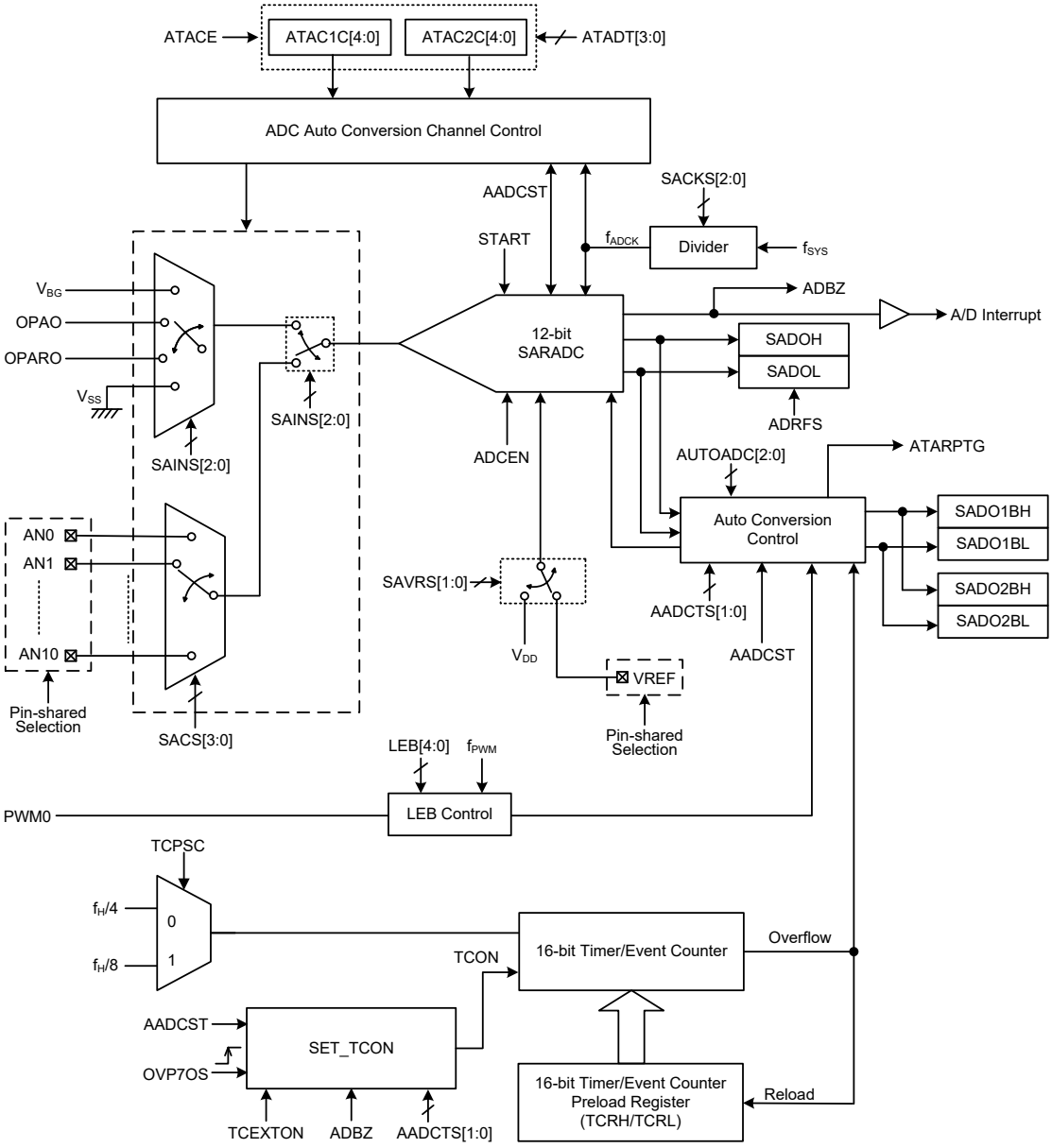
The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the internal Bandgap reference voltage, V_{BG}, the operational amplifier output, OPAO, the operational amplifier output, OPARO, or the A/D converter negative power supply, V_{SS}, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 and SACS3~SACS0 bits. Note that when the internal analog signal is selected to be converted, the external channel analog input will be automatically switched off. More detailed information about the A/D converter input signal is described in the "A/D Converter Control Registers" and "A/D Converter Input Signals" sections respectively.

External Input Channels	Internal Channel Signals
AN0~AN10	V_{BG}, OPAO, OPARO, V_{SS}

The A/D converter can be operated in manual trigger conversion mode, 2-channel automatic conversion mode, and 1-channel automatic conversion mode. The manual trigger conversion is to start an A/D conversion by controlling the START bit using the user program, which is the general A/D converter function. The automatic trigger conversion is to start a conversion automatically when receiving an active trigger signal. The trigger signal is selected by AADCTS1~AADCTS0 bits. The conversion number in a group of automatic conversion is defined by AUTOADC2~AUTOADC0 bits in the SADC2 register, which can be 1, 2, 4, 8 or 16.

AUTOADC[2:0] Bits	ATACE Bit	A/D Conversion Mode	Channel Input Select Bits
000	x	Manual trigger an A/D conversion	SAINS[2:0] & SACS[3:0]
001~111	0	1-channel automatic conversion	SAINS[2:0] & SACS[3:0]
001~111	1	2-channel automatic conversion	CH1: ATAC1C[4:0] CH2: ATAC2C[4:0]

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



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using several registers. A read only register pair exists to store the A/D converter data 12-bit value. Two register pairs, SADO1BH/SADO1BL and SADO2BH/SADO2BL, are used to store the cumulative automatic conversion result of the CH1 and CH2 in the automatic A/D conversion mode. The remaining registers are control registers which setup the operating and control functions of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADO1BH	D15	D14	D13	D12	D11	D10	D9	D8
SADO1BL	D7	D6	D5	D4	D3	D2	D1	D0
SADO2BH	D15	D14	D13	D12	D11	D10	D9	D8
SADO2BL	D7	D6	D5	D4	D3	D2	D1	D0
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
SADC2	AADCTS1	AADCTS0	—	ATARPTG	—	AUTOADC2	AUTOADC1	AUTOADC0
LEBC	AADCST	—	—	LEB4	LEB3	LEB2	LEB1	LEB0
ATAC1C	—	—	ATACE	ATAC1SS	ATAC1S3	ATAC1S2	ATAC1S1	ATAC1S0
ATAC2C	—	—	—	ATAC2SS	ATAC2S3	ATAC2S2	ATAC2S1	ATAC2S0
ATADT	—	—	—	—	ATADT3	ATADT2	ATADT1	ATADT0
TCRC	—	—	—	TCON	—	—	TCEXTON	TCPSC
TCRH	D15	D14	D13	D12	D11	D10	D9	D8
TCRL	D7	D6	D5	D4	D3	D2	D1	D0

A/D Converter Register List

A/D Converter Data Registers

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. Note that the A/D converter data register contents will remain unchanged if the A/D converter is disabled.

If the A/D converter operates in the automatic conversion mode (AUTOADC[2:0]≠000), the ADRFS bit will be fixed at 1 by the hardware. The conversion result data formart is: D[11:8]=SADOH[3:0]; D[7:0]=SADOL[7:0], unused bits in the SADOH register read as zero.

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

The device contains two pairs of registers which are the data buffer registers to store the CH1 and CH2 cumulative conversion result when the A/D converter operates in the automatic conversion mode.

Register	SADO1BH								SADO1BL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

A/D Automatic Conversion Data Buffer 1 Registers

Register	SADO2BH								SADO2BL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

A/D Automatic Conversion Data Buffer 2 Registers

A/D Converter Control Registers

To control the function and operation of the A/D converter, several control registers are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D converter operating mode, the A/D converter 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 SAINS2~SAINS0 bits in the SADC1 register and SACS3~SACS0 bits in the SADC0 register are used to select which analog signal from either the external or internal signals will be connected to the A/D converter when the A/D converter is in the manual trigger conversion mode or the 1-channel automatic conversion mode. If the A/D converter operates in the 2-channel automatic conversion mode, the CH1 and CH2 input signals are selected by the ATAC1C and ATAC2C registers respectively. The LEBC register is used to enable the automatic conversion and set the PWM0 trigger signal leading edge blanking time. The SADC2 register is used to select the automatic conversion start trigger signal and the conversion number. The ATAC1C, ATAC2C and ATADT registers can be used to enable the 2-channel automatic conversion mode, select the CH1 and CH2 signals and the time interval between two conversions.

The relevant pin-shared function selection bits determine which pins on I/O Ports are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. When the pin is selected to be an A/D converter 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 (for manually trigger conversion)
0→1→0: Start A/D conversion
Note: This bit is valid only when the AUTOADC[2:0] is “000”.
This bit is used to initiate an A/D conversion process in the manual trigger conversion mode. 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. The ADBZ flag is high to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to zero 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 converter internal function. This bit should be set high 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 converter data register pair, SADOH and SADOL, will remain unchanged.

Bit 4 **ADRFS**: A/D Converter data format control
0: ADC output data format → SADOH=D[11:4]; SADOL=D[3:0]
1: ADC output data format → SADOH=D[11:8]; SADOL=D[7:0]
This bit controls the format of the 12-bit converted A/D converter value in the two A/D converter data registers.
Note: This bit can be changed by software when the AUTOADC[2:0] is "000". When the AUTOADC[2:0] bits are not "000", the ADRFS bit is fixed at 1 by the hardware.

Bit 3~0 **SACS3~SACS0**: External analog input channel selection
0000: External AN0 input
0001: External AN1 input
0010: External AN2 input
0011: External AN3 input
0100: External AN4 input
0101: External AN5 input
0110: External AN6 input
0111: External AN7 input
1000: External AN8 input
1001: External AN9 input
1010: External AN10 input
1011~1111: Undefined, input floating
These bits are used to select which external analog input channel is to be converted in the manual trigger or automatic trigger 1-CH conversion mode.

• SADC1 Register

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
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 7~5 **SAINS2~SAINS0**: A/D converter input signal selection
000: External signal – External analog channel input, ANn
001: Internal signal – Internal Bandgap reference voltage, V_{BG}
010: Internal signal – Operational amplifier output, OPAO
011: Internal signal – Operational amplifier output, OPARO
100: Internal signal – A/D converter negative power supply, V_{SS}
101~111: External signal – External analog channel input, ANn
These bits are available in the manual trigger or automatic trigger 1-CH conversion mode.
When the internal analog signal is selected to be converted, the external channel input signal will automatically be switched off regardless of the SACS3~SACS0 bit value. It will prevent the external channel input from being connected together with the internal analog signal.

Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage selection
00: External VREF pin input
01: Internal power supply, V_{DD}
1x: External VREF pin input
These bits are used to select the A/D converter reference voltage. When the internal A/D converter power is selected as the reference voltage, the hardware will automatically disconnect the external VREF input.

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock (f_{ADCK}) 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	AADCTS1	AADCTS0	—	ATARPTG	—	AUTOADC2	AUTOADC1	AUTOADC0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

Bit 7~6 **AADCTS1~AADCTS0**: Automatic conversion start trigger source selection
00: PWM0 rising edge
01: Timer counter overflows – TCON is set high by an OVP7OS rising edge
10: Timer counter overflows – TCON is set high by AADCST changing from 0 to 1
11: Software bit trigger – change AADCST from 0 to 1 to start a conversion
Refer to the "Automatic Trigger Conversion Description" section for more details.

Bit 5 Unimplemented, read as "0"

Bit 4 **ATARPTG**: Repeat trigger flag in auto conversion mode
0: No trigger event occurred during an A/D conversion process
1: A trigger event occurred during an A/D conversion process
This bit is set by hardware and can be cleared by software and cannot be set high by software. When AADCST changes from 0 to 1, the hardware will clear this bit automatically. In auto conversion mode, if the A/D conversion is not completed and a trigger signal occurs, this bit will be set.

Bit 3 Unimplemented, read as "0"

Bit 2~0 **AUTOADC2~AUTOADC0**: Automatic conversion number selection
000: N=0, automatic conversion function is disabled
001: N=1
010: N=2
011: N=4
100: N=8
101~111: N=16
These three bits define the number of conversions, N, in a group of automatic conversions. The AUTOADC2~AUTOADC0 bits can also be used to disable the automatic conversion function.
If N≠0 and ATACE bit is 0, after the A/D converter performs N times of conversions, the cumulative converted result is stored in the SADO1BH and SADO1BL register pair and the ADF flag is set high. If N≠0 and ATACE bit is 1, after the A/D converter performs N times of conversions, the cumulative automatic conversion result of the CH1 is stored in the SADO1BH and SADO1BL register pair and the CH2 is stored in the SADO2BH and SADO2BL register pair and the ADF flag is set high.
Note: 1. When AUTOADC[2:0]≠000, the START bit in the SADC0 register is invalid. A group of automatic conversions is enabled by setting the AADCST bit high.

When the selected conversion start trigger signal occurs, an A/D conversion is started automatically.

2. In the automatic conversion mode, after an conversion, the A/D converter output data format is: D[11:8]=SADOH[3:0]; D[7:0]=SADOL[7:0], unused bits in SADOH are all zero.

• **LEBC Register**

Bit	7	6	5	4	3	2	1	0
Name	AADCST	—	—	LEB4	LEB3	LEB2	LEB1	LEB0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 **AADCST**: A/D automatic conversion control
0: Disable
1: Enable
This bit is set by the software and can be cleared by software or hardware. After finishing the specific number of A/D conversions which is defined by the AUTOADC [2:0] bits, this bit is cleared by the hardware. When the automatic conversion trigger source signal stops suddenly or abnormally, AADCST bit can be cleared using the software. For related applications, refer to the A/D Automatic Conversion Software Stop Descriptions section. When this bit changes from 0 to 1, the ATARPTG bit will be cleared to zero automatically.
Note that this bit is invalid when AUTOADC[2:0] is "000".

Bit 6~5 Unimplemented, read as "0"

Bit 4~0 **LEB4~LEB0**: Leading-edge blanking time control bits
LEB time=(LEB[4:0]+1)×16/f_{PWM}, f_{PWM}=32MHz
LEB[4:0]=0~31, providing a total of 32 levels of LEB to choose from
For example:
LEB[4:0]=0, LEB time=(0+1)×16×0.03125μs=0.5μs
LEB[4:0]=5, LEB time=(5+1)×16×0.03125μs=3μs
Note: the LEB[4:0] bits are valid only when the PWM0 rising edge is selected as the trigger source (AADCTS[1:0]=00).

• **ATAC1C Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	ATACE	ATAC1SS	ATAC1S3	ATAC1S2	ATAC1S1	ATAC1S0
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 **ATACE**: 2-channel automactic conversion mode enable control
0: Disable
1: Enable

Bit 4 **ATAC1SS**: CH1 input source selection (for 2-channel automatic conversion mode)
0: From internal signals
1: From external channel input, ANn
The ATAC1SS bit selection is available only when the 2-channel automatic conversion mode is enabled.

Bit 3~0 **ATAC1S3~ATAC1S0**: CH1 input signal selection (for 2-channel automatic conversion mode)
If ATAC1SS=0:
0000: V_{BG} reference voltage
0001: Operational amplifier output, OPAO
0010: Operational amplifier output, OPARO
0011: A/D converter negative power supply, V_{SS}
0100~1111: V_{BG} reference voltage

If ATAC1SS=1:
0000: External AN0 input
0001: External AN1 input
0010: External AN2 input
0011: External AN3 input
0100: External AN4 input
0101: External AN5 input
0110: External AN6 input
0111: External AN7 input
1000: External AN8 input
1001: External AN9 input
1010: External AN10 input
1011~1111: Undefined, input floating

The ATAC1S[3:0] bit selections are available only when the 2-channel automatic conversion mode is enabled.

• **ATAC2C Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	ATAC2SS	ATAC2S3	ATAC2S2	ATAC2S1	ATAC2S0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4 **ATAC2SS**: CH2 input source selection (for 2-channel automatic conversion mode)
0: From internal signals
1: From external channel input, ANn

The ATAC2SS bit selection is available only when the 2-channel automatic conversion mode is enabled.

Bit 3~0 **ATAC2S3~ATAC2S0**: CH2 input signal selection (for 2-channel automatic conversion mode)

If ATAC2SS=0
0000: V_{BG} reference voltage
0001: Operational amplifier output, OPAO
0010: Operational amplifier output, OPARO
0011: A/D converter negative power supply, V_{SS}
0100~1111: V_{BG} reference voltage

If ATAC2SS=1
0000: External AN0 input
0001: External AN1 input
0010: External AN2 input
0011: External AN3 input
0100: External AN4 input
0101: External AN5 input
0110: External AN6 input
0111: External AN7 input
1000: External AN8 input
1001: External AN9 input
1010: External AN10 input
1011~1111: Undefined, input floating

The ATAC2S[3:0] bit selections are available only when the 2-channel automatic conversion mode is enabled.

• **ATADT Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	ATADT3	ATADT2	ATADT1	ATADT0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **ATADT3~ATADT0**: Select time interval between two auto-triggered A/D conversions
0000: $1\times t_{ADCK}$
0001: $2\times t_{ADCK}$
0010: $3\times t_{ADCK}$
…
…
1110: $15\times t_{ADCK}$
1111: $16\times t_{ADCK}$
Note: $t_{ADCK}=1/f_{ADCK}$

Timer Counter Registers

A 16-bit timer is included in the A/D converter circuitry to carry out timing function and its overflow event can be used as the A/D automatic conversion start trigger signal. There are three registers related to the timer counter, including a control register, TCRC, and a pair of preload registers, TCRH and TCRL. The TCRC register defines the counter enabling method and selects its counting clock. When the TCON bit is set high, the timer will count from the initial value loaded by the preload register to the full count of FFFFH, at which point the timer overflows.

• **TCRC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	TCON	—	—	TCEXTON	TCPSC
R/W	—	—	—	R/W	—	—	R/W	R/W
POR	—	—	—	0	—	—	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4 **TCON**: Timer Counter enable control
0: Disable
1: Enable
This bit can only be cleared to zero by software but cannot be set by the software.
When AADCTS[1:0]=01: The TCON bit is set to 1 automatically at a rising edge of the OVP7OS signal and then the counter starts counting, and will stop counting when it is full and overflows, which will clear the TCON bit.
When AADCTS[1:0]=10: The TCON bit is set to 1 automatically when changing AADCST from 0 to 1 and then the counter starts counting. When the AADCST bit changes from 1 to 0, the TCON bit will be cleared to zero automatically.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **TCEXTON**: External rising edge trigger TCON being set enable control
0: Disable
1: Enable
When this bit is 0, OVP7OS signal rising edges have no effect on TCON bit. When this bit is 1 and the ADBZ bit is 0, the TCON bit will be set high when an OVP7OS rising edge is received.

Bit 0 **TCPSC**: Timer counter clock selection
0: $f_H/4$
1: $f_H/8$

• **TCRH 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	x	x	x	x	x	x	x	x

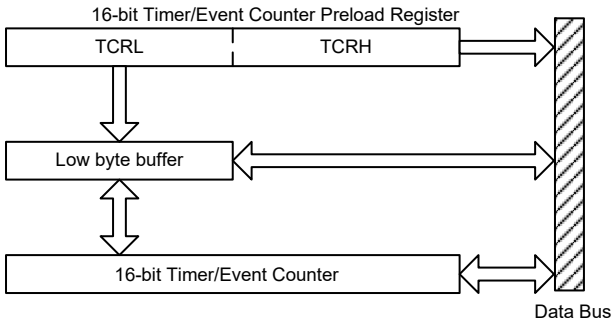
Bit 7~0 **D15~D8**: 16-bit Timer Counter preload register high byte

• **TCRL 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

Bit 7~0 **D7~D0**: 16-bit Timer Counter preload register low byte

Note that writing to the TCRL register will only write the data into an internal lower byte buffer while writing to the TCRH register will transfer the high byte data and the contents of the lower byte buffer into the TCRH and TCRL registers respectively. Therefore the timer counter preload register is changed by each write operation to the TCRH register. Reading from the TCRH register will latch the contents of the TCRH and TCRL counters to the destination and the lower byte buffer respectively, while reading from TCRL will read the contents of the lower byte buffer.



Reading or Writing to TCRH & TCRL

A/D Converter Reference Voltage

The actual reference voltage supply to the A/D converter can be supplied from the positive power supply, V_{DD} or an external reference source supplied on pin VREF. The desired selection is made using the SAVRS1~SAVRS0 bits in the SADC1 register. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage input 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 V_{DD} 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 signal values must not be allowed to exceed the value of the selected A/D Converter reference voltage.

SAVRS[1:0]	Reference Source	Description
00, 1x	VREF pin	External A/D converter reference pin VREF
01	V_{DD}	Internal power supply voltage

A/D Converter Reference Voltage Selection

A/D Converter Input Signals

All of the external A/D converter 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 PxS0 and PxS1 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 functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D Converter input as when the relevant A/D converter input function selection bits enable an A/D converter input, the status of the port control register will be overridden.

As this 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. For manual trigger A/D conversion mode and automatic trigger 1-channel A/D conversion mode, the SAINS2~SAINS0 bits in the SADC1 register are used to determine whether the analog signal to be converted comes from an 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 SAINS2~SAINS0 bits are set to “000”, “101”~“111” the external channel input will be selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected. For the automatic trigger 2-channel A/D conversion mode, the input signals of the CH1 and CH2 are selected by the ATAC1C and ATAC2C registers. The ATACnSS bit is used to determine whether the analog signal to be converted comes from the external channel input or internal analog signal. The ATACnS3~ATACnS0 bits are used to determine which external channel input or which internal signal is selected to be converted.

If the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off. It will prevent the external channel input from being connected together with the internal analog signal.

SAINS[2:0]	SACS[3:0]	Input Signals	Description
000, 101~111	0000~1010	AN0~AN10	External channel input, ANn
	1011~1111	—	Floating
001	xxxx	V_{BG}	Internal reference voltage, V_{BG}
010	xxxx	OPAO	Internal OPAMP output, OPAO
011	xxxx	OPARO	Internal OPAMP output, OPARO
100	xxxx	V_{SS}	A/D converter negative power supply, V_{SS}

A/D Converter Input Signal Selection – Manual or 1-CH Auto Mode

ATAC1SS/ ATAC2SS	ATAC1S [3:0]/ ATAC2S [3:0]	CH1/CH2 Input Signals	Description
0	0000, 0100~1111	V_{BG}	Internal reference voltage, V_{BG}
	0001	OPAO	Internal OPAMP output, OPAO
	0010	OPARO	Internal OPAMP output, OPARO
	0011	V_{SS}	A/D converter negative power supply, V_{SS}
1	0000~1010	AN0~AN10	External channel input, ANn
	1011~1111	—	Floating

A/D Converter Input Signal Selection – 2-CH Auto Mode

A/D Converter Clock Source and Power

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 conversion clock source is determined by the system clock f_{SYS}, and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D conversion clock source speed that can be selected. As the recommended value of permissible A/D conversion clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken when selecting the clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the SACKS2~SACKS0 bits should not be set to "000", "110" or "111". Doing so will give A/D conversion clock periods that are less than the permissible minimum A/D conversion clock period or greater than the permissible maximum A/D conversion 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 Conversion 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*
16MHz	62.5ns*	125ns*	250ns*	500ns	1μs	2μs	4μs	8μs

A/D Conversion Clock Period Examples

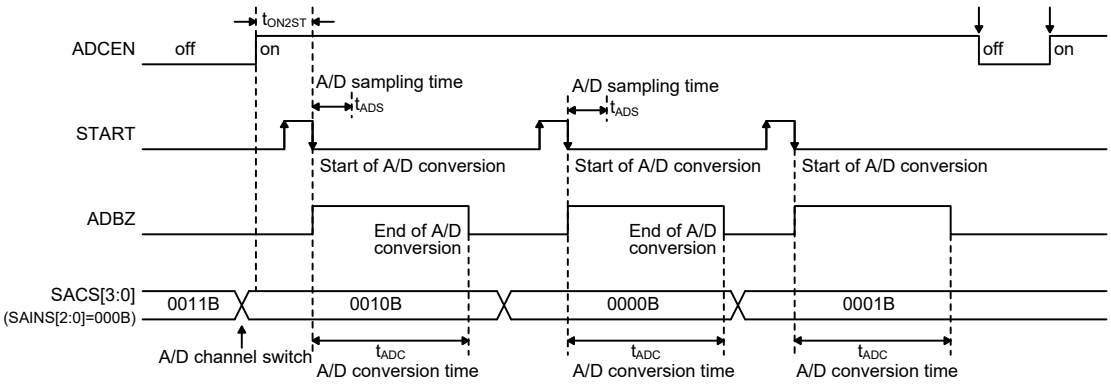
Controlling the power on/off function of the A/D conversion circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D conversion internal circuitry a certain delay, as indicated in the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

A/D 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 conversion clock cycles and the data conversion takes 12 A/D converter clock cycles. Therefore a total of 16 A/D conversion clock periods for an 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 analog to digital conversion process and its associated timing. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is $16t_{ADCK}$ clock cycles where t_{ADCK} is equal to the A/D conversion clock period.



A/D Conversion Timing – AUTOADC[2:0]=000

Manual Trigger A/D Conversion

This is general A/D converter operating mode (AUTOADC[2:0]=000). The start of each A/D conversion is triggered manually. the A/D converter will perform conversions on the channel specified by the SACS[3:0] bits in the SADC0 register together with the SAINS[2:0] bits in the SADC1 register.

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

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process has finished processing or not. This bit will be automatically set to "1" by the microcontroller after an A/D conversion has been successfully initiated. When an A/D conversion is complete, the ADBZ will be cleared to "0". In addition, the corresponding A/D converter interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D converter internal interrupt signal will direct the program flow to the associated A/D converter 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 following summarises the individual steps that should be executed in order to implement a manual trigger 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 "1".

- Step 3

  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS2~SAINS0 bits.

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

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

- Step 4

  If the external channel input is 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 configuring the relevant pin-shared function control bits. Then go to Step 6.

- Step 5

  If the SAINS2~SAINS0 bits are set to 001~011, 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 reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register.

- Step 7

  Select the A/D converter output data format by configuring the ADRFS bit in the SADC0 register.

- Step 8

  If the A/D converter interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt control bit, ADE, must all be set high.

- Step 9

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

- Step 10

  If an A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process has completed, the ADBZ flag will go low after which the output data can be read from the 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.

Automatic Trigger 1-Channel A/D Conversion

In the 1-channel automatic A/D conversion mode (ATACE=0, AUTOADC[2:0]≠000), the A/D converter will perform conversions on the channel specified by the SACS[3:0] bits in the SADC0 register together with the SAINS[2:0] bits in the SADC1 register. After configuring basic A/D converter parameters, users should select the start trigger signal and setup the number of automatic conversions. The LEB time should be configured if the trigger signal is selected to be the PWM0 rising edge and the timer counter related parameters should be configured if the trigger condition is that the timer counter overflows.

When the AADCST bit is changed from 0 to 1, the automatic conversion is enabled. When an active trigger signal occurs, it may activate the LEB timing or the timer counter to start counting or start an A/D conversion immediately. The converted data will be stored in the SADO1BH and SADO1BL registers from the SADOH and SADOL registers. After the whole group of automatic conversions is completed, the value in SADO1B register is the cumulative conversion result of these conversions.

The following summarises the individual steps that should be executed in order to implement an automatic trigger 1-channel 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 “1”.

- Step 3

  Enable the 1-channel automatic conversion function by clearing the ATACE bit in the ATAC1C register to “0”.

- Step 4

  Select which signal is to be connected to the internal A/D converter in 1-channel automatic conversion mode by correctly configuring the SAINS2~SAINS0 bits.

  Select the external channel input to be converted, go to Step 5.

  Select the internal analog signal to be converted, go to Step 6.

- Step 5

  If the SAINS2~SAINS0 bits are programmed to select the external channel input, 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 configuring the relevant pin-shared function control bits. Then go to Step 7

- Step 6

  If the SAINS2~SAINS0 bits are set to 001, 010 or 011, 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 7.

- Step 7

  Select the number of the automatic A/D conversions by setting the AUTOADC[2:0] bits in the SADC2 register.

- Step 8

  Select the A/D automatic conversion trigger signal by setting the AADCTS[1:0] bits in the SADC2 register.

  If PWM0 rising edge is selected(00B), go to Step 9.

  If OVP7OS rising edge activated Timer overflow event is selected(01B), go to Step 10.

  If AADCST rising edge activated Timer overflow event is selected(10B), go to Step 11.

  If AADCST rising edge immediately trigger is selected(11B), go to Step 13.

- Step 9

  Set the leading edge blanking time by programming the LEB[4:0] bits in the LEBC register, then go to Step 14.

- Step 10

  Setup whether the OVP7OS signal rising edge is used to activate the timer counter using the TCEXTON bit in the TCRC register, then go to Step 11.

- Step 11

  Setup the timer counter clock source using the TCPSC bit in the TCRC register, then go to Step 12.

- Step 12

  Configure the timer counter preload value using the TCRH and TCRL registers, then go to Step 14.

- Step 13

  Setup the interval time between two A/D conversions by configuring the ATADT[3:0] bits in the ATADT register, then go to Step 14.

- Step 14

  Select the required reference voltage using the SAVRS[1:0] bits in the SADC1 register.

- Step 15

  If the A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt control bit, ADE, must all be set high.

- Step 16

  Start the A/D conversion process by setting the AADCST bit in the LEBC register from low to high.

Note: After specified times of A/D automatic conversions are complete, the AADCST bit will be cleared by the hardware and the A/D converter interrupt flag ADF bit will be set high.

Automatic Trigger 2-Channel A/D Conversion

In the 2-channel automatic A/D conversion mode (ATACE=1, AUTOADC[2:0]≠000), the A/D converter CH1 or CH2 input signal is selected by using the ATACnSS and ATACnS[3:0] bits respectively. After the basic A/D converter parameters are set, then set the ATACE bit to enable the 2-channel conversion mode, select the number of automatic conversions, the trigger signal in automatic mode and the interval time between two conversions. The LEB time should be configured if the trigger signal is selected to be the PWM0 rising edge and the timer counter related parameters should be configured if the trigger condition is that the timer counter overflows.

When the AADCST bit is changed from 0 to 1, the automatic conversion is enabled. When an active trigger signal occurs, it may activate the LEB timing or the timer counter to start counting or start an A/D conversion immediately. The A/D converter performs the CH1 conversion first. When the interval time has elapsed, the A/D converter will perform the CH2 conversion. The CH1 converted data will be stored in the SADO1BH and SADO1BL registers from the SADOH and SADOL registers and the CH2 converted data will be stored in the SADO2BH and SADO2BL registers from the SADOH and SADOL registers. After the specified times of A/D automatic conversions, the 2-channel automatic conversion is completed. The value in the SADO1B register is the cumulative result of CH1 conversions and the value in the in the SADO2B register is the cumulative result of CH2 conversions.

When AADCTS[1:0]=11, the delay time between two conversions is determined by the ATADT[3:0] setting.

The following summarises the individual steps that should be executed in order to implement an automatic trigger 2-channel 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 “1”.

- Step 3

  Enable the 2-channel automatic conversion function by setting the ATACE bit in the ATAC1C register to “1”.

- Step 4

  Select whether an external or internal signal is to be connected to the Automatic Trigger ADC input CH1 and Automatic Trigger ADC input CH2 by correctly configuring the ATACnSS bit in the ATACnC register respectively.

  Select the external channel input to be converted (ATACnSS=1), go to Step5.

  Select the internal analog signal to be converted (ATACnSS=0), go to Step6.

- Step 5

  The desired external channel input can be selected by configuring the ATACnS[3:0] bits. As the ANn pin is pin-shared with other functions, the corresponding pin should be configured as an A/D input function by configuring the relevant pin-shared function control bits. Then go to Step7.

- Step 6

  The desired internal analog signal can be selected by configuring the ATACnS[3:0] bits. Then go to Step7.

- Step 7

  Select the number of the automatic A/D conversions by setting the AUTOADC[2:0] bits in the SADC2 register.

- Step 8

  Select the A/D automatic conversion trigger signal by setting the AADCTS[1:0] bits in the SADC2 register.

  If PWM0 rising edge is selected(00B), go to Step 9.

  If OVP7OS rising edge activated Timer overflow event is selected(01B), go to Step 10.

  If AADCST rising edge activated Timer overflow event is selected(10B), go to Step 11.

  If AADCST rising edge immediately trigger is selected(11B), go to Step 13.

- Step 9

  Set the leading edge blanking time by programming the LEB[4:0] bits in the LEBC register, go to Step 13.

- Step 10

  Setup whether the OVP7OS signal rising edge is used to activate the timer counter using the TCEXTON bit in the TCRC register, go to Step 11.

- Step 11

  Setup the timer counter clock source using the TCPSC bit in the TCRC register, go to Step 12.

- Step 12

  Configure the timer counter preload value using the TCRH and TCRL registers, go to Step 13.

- Step 13

  Setup the interval time between two A/D conversions by configuring the ATADT[3:0] bits in the ATADT register, then go to step 14.

- Step 14

  Select the required reference voltage using the SAVRS[1:0] bits in the SADC1 register.

- Step 15

  If the A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt control bit, ADE, must all be set high.

- Step 16

  Start the A/D conversion process by setting the AADCST bit in the LEBC register from low to high.

  Note: After specified times of A/D automatic conversions are complete, the AADCST bit will be cleared by the hardware and the A/D converter interrupt flag ADF bit will be set high.

Automatic Trigger Conversion Description

An automatic conversion can be initiated by a PWM0 rising edge with a LEB time, an overflow event of the 16-bit timer counter which is started by an OVP7OS signal rising edge or by setting the AADCST bit, or initiated directly by setting the AADCST bit from low to high. The following section describes how the A/D converter is started by these triggers and complete a group of automatic conversions.

AADCTS[1:0]=00B

When AADCTS[1:0] bits are 00 and the AUTOADC[2:0] bits are not equal to 000, setting the AADCST bit high will start the A/D automatic synchronization delay conversion. After setting the AADCST bit high, each PWM0 rising edge will start the leading edge blanking circuit. After the LEB time configured by LED4~LED0 bits, an A/D conversion will start. After N conversions have been completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set. Here the number N is defined by the AUTOADC[2:0] bits which can be 1, 2, 4, 8 or 16. If the PWM0 output stops abnormally, the AADCST bit can be cleared to zero by the software.

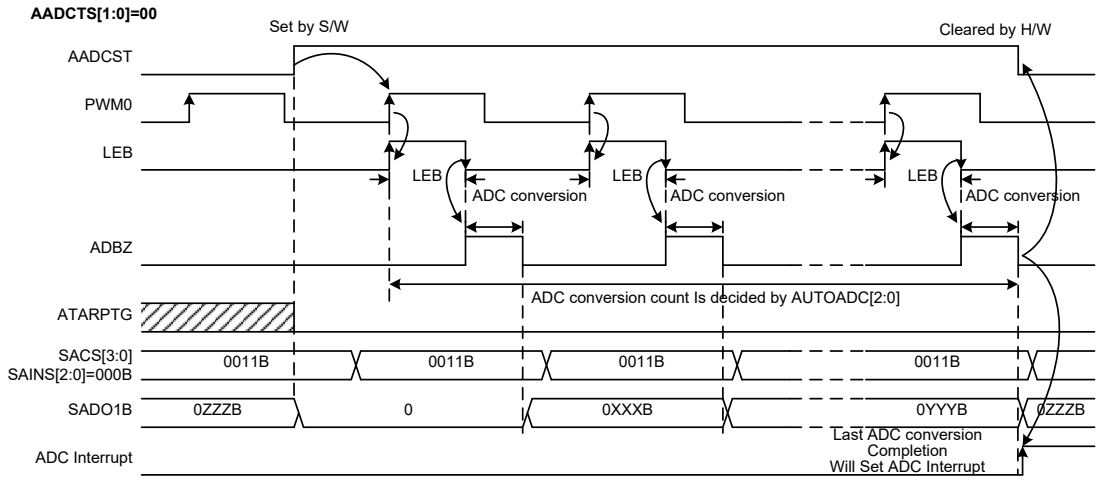
The AADCST is unable to provide a repeatable trigger. Once the AADCST bit is set high, the circuit cannot be triggered until the specified number N of automatic conversions are completed. In program design it should be noted that after the AADCST bit is cleared, the ADF flag should also be cleared using the application progrm.

Note that during the automatic conversion process, if the AADCST bit is cleared to 0 when the ADBZ bit is 1, then the automatic conversion circuit will not be reset to its initial state immediately until the current A/D conversion is complete and the ADBZ bit changes to 0. If the AADCST bit is cleared to 0 when the ADBZ bit is 0 which means the circuit is counting for the LEB time, the counting will stop immediately and the automatic conversion circuit will be reset. After the reset is complete, the AADCST bit can be set high again to enable another group of automatic conversions. If the specified conversions have been completed when the AADCST bit is cleared, the ADC interrupt can still be triggered.

1-Channel Automatic Conversion

When AADCST changes from 0 to 1, the ATARPTG bit will be cleared to 0 automatically. Then when a PWM0 rising edge trigger signal arrives, the LEB circuit will be triggered and the delay timing starts. When the LEB time has elapsed, the A/D converter starts a conversion. After the specified number of automatic conversions is completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set.

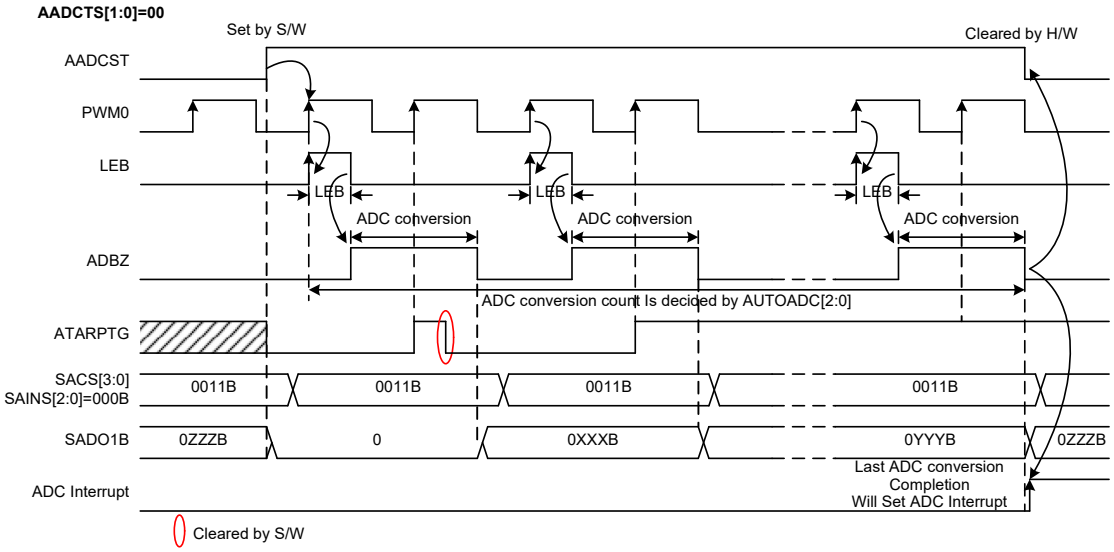
In the following example, after a PWM0 rising edge signal arrives, there is enough time for the A/D converter to convert the data before the next PWM0 rising edge arrives.



1-CH Automatic Conversion Timing – 1 (AADCTS[1:0]=00)

Note: 1. AUTOADC[2:0]≠000
2. XXXB is the first A/D converted data
3. YYYB is the accumulated A/D converted result
4. ZZZB is the final accumulated result of the N automatic conversions

When an A/D conversion is in progress, if there occurs a PWM0 rising edge, the ATARPTG bit will be set to 1. The A/D converter continues to perform the current conversion, ignoring this trigger signal. After finishing it, the A/D converter will not start the next conversion until a new PWM0 rising edge occurs. After the specified times of automatic conversions are completed, the AADCST bit will be cleared by the hardware. The following shows a 1-channel automatic conversion timing diagram, where a PWM0 rising edge arrives before the previous conversion has not been completed.



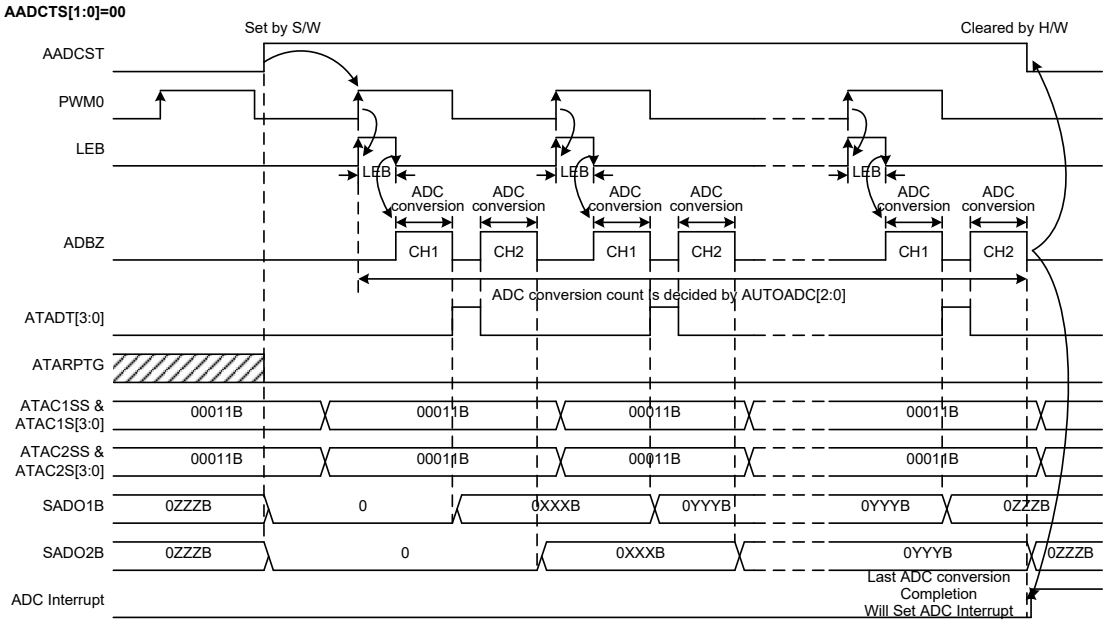
1-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=00)

Note: 1. AUTOADC[2:0]≠000
2. XXXB is the first A/D converted data
3. YYYB is the accumulated A/D converted result
4. ZZZB is the final accumulated result of the N automatic conversions

2-Channel Automatic Conversion

When AADCST changes from 0 to 1, the ATARPTG bit will be cleared to 0 automatically. When a PWM0 rising edge trigger signal arrives, the LEB circuit will be triggered and the delay timing starts. When the LEB time has elpased, the ADC starts to perform a conversion on the CH1. After completion of the CH1 conversion, a interval time which is defined by the ATADT[3:0] bits is inserted. Then the conversion on CH2 starts. When the CH2 conversion is completed, the A/D converter will wait for the next PWM0 rising edge to start the next cycle of CH1 conversion and CH2 conversion. After N conversion cycles have completed, the AADCST bit will be cleared by the hardware and the ADC interrupt flag ADF will be set.

In the following example, after a PWM0 rising edge signal arrives, there is enough time for the A/D converter to convert the data of the two channels before the next PWM0 rising edge arrives.



2-CH Automatic Conversion Timing – 1(AADCTS[1:0]=00)

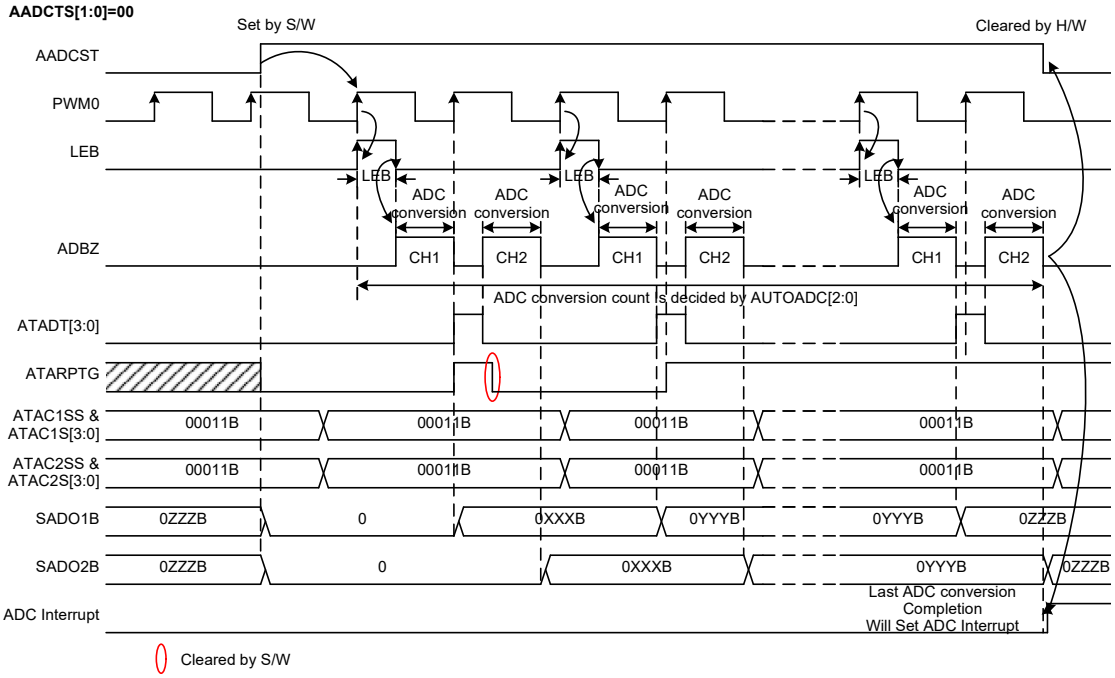
Note: 1. AUTOADC[2:0]≠000

2. XXXB is the first A/D converted data

3. YYYB is the accumulated A/D converted result

4. ZZZB is the final accumulated data of the N automatic conversion results

The following figure shows another example of the 2-channel automatic conversion. In this example, after a PWM0 rising edge signal arrives, there is no enough time for the ADC to complete the conversions on the two channels before the next PWM0 rising edge arrives. Then the ATARPTG bit will be set. The A/D converter continues to perform the current conversion, ignoring this trigger signal. After completing the CH1 and CH2 conversions, the A/D converter will not start the next cycle of conversions until a new PWM0 rising edge occurs.



2-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=00)

Note: 1. AUTOADC[2:0]≠000
2. XXXB is the first A/D converted data
3. YYYB is the accumulated A/D converted result
4. ZZZB is the final accumulated result of the N automatic conversions

AADCTS[1:0]=01B

When AADCTS[1:0] bits are 01 and the AUTOADC[2:0] bits are not equal to 000, setting the AADCST bit high will start the A/D automatic synchronization delay conversion. After setting the AADCST bit high, a rising edge of the OVP7OS signal will set the TCON bit to 1 to enable the timer counter to start counting. The timer will count from the initial value loaded by the preload register, TCR[15:0], to the full count of FFFFH, at which point the timer overflows. TCON bit is cleared to 0 automatically and an A/D conversion starts. The timer then waits for the next OVP7OS signal rising edge. After N conversions have been completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set. Here the number N is defined by the AUTOADC[2:0] bits which can be 1, 2, 4, 8 or 16. If the trigger source signal is abnormal and no rising edge is generated, the AADCST bit can be cleared using the software.

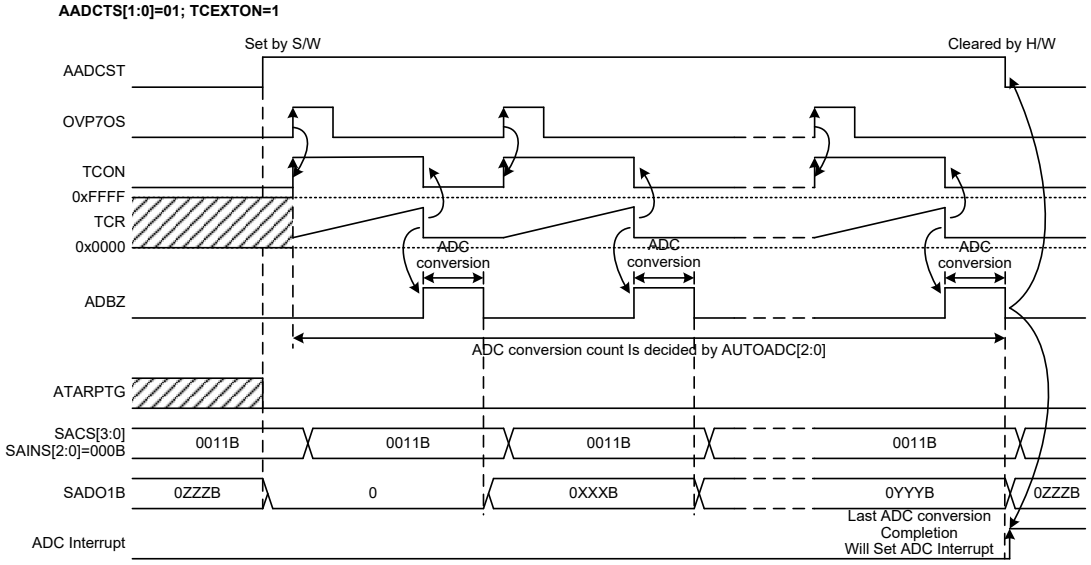
The AADCST is unable to provide a repeatable trigger. Once the AADCST bit is set high, the circuit cannot be triggered until the specified number N of automatic conversions are completed. In program design it should be noted that after the AADCST bit is cleared, the ADF flag should also be cleared using the application progrm.

Note that during the automatic conversion process, and the AADCST bit is cleared to 0, if the ADBZ bit is 1, wait for the ADBZ bit changes to 0, then switch to manual trigger conversion mode (AUTOADC[2:0]=000), and enable the A/D conversion one time (by setting the START bit from low to high and then low again). When the conversion is completed, switch back to automatic conversion mode, and then the automatic conversion circuit will be reset to its initial state. After the reset is completed, the AADCST bit can be set high again to enable another group of automatic conversions. If the specified conversions have been completed when the AADCST bit is cleared, the ADC interrupt can still be triggered.

Note: if AADCST is 0 or AADCST=1 & TCEXTON=0, an OVP7OS signal rising edge will not set the TCON bit to 1.

1-Channel Automatic Conversion

When AADCST changes from 0 to1, the ATARPTG bit will be cleared to 0 automatically. When an OVP7OS rising edge signal arrives, the timer counter will be enabled to start counting. When the timer counter overflows, the TCON bit will be cleared to zero and the A/D converter starts the conversion. After the specified times of automatic conversions are completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set.



1-CH Automatic Conversion Timing – 1 (AADCTS[1:0]=01)

Note: 1. AUTOADC[2:0]≠000

2. XXXB is the first A/D converted data

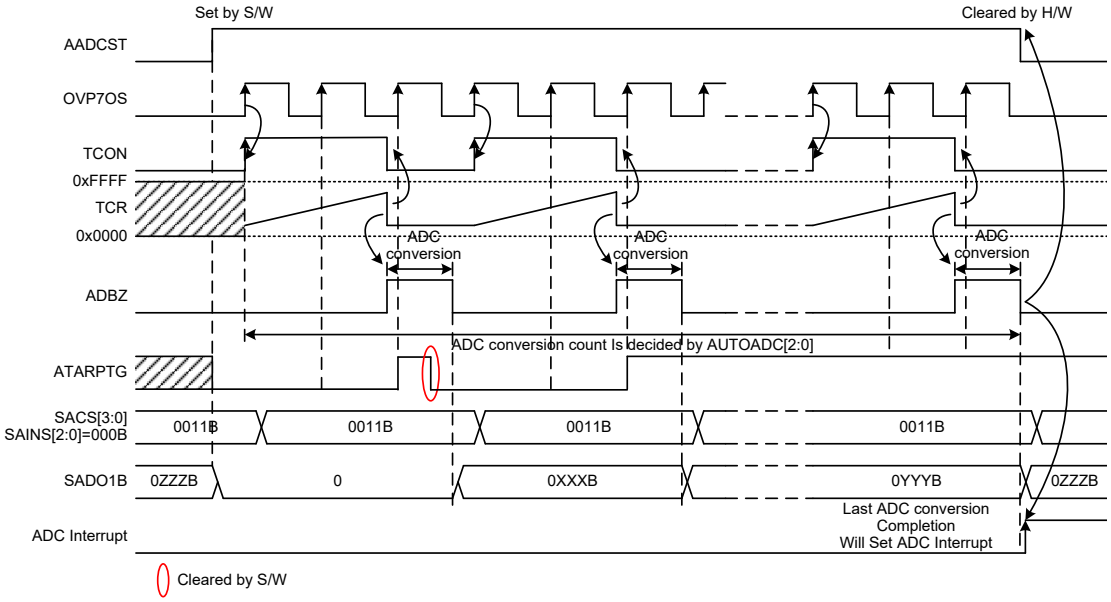
3. YYYB is the accumulated A/D converted result

4. ZZZB is the final accumulated result of the N automatic conversions

In the following example, after AADCST changes from 0 to 1, an OVP7OS rising edge will trigger the timer counter to start counting. After it overflows, the TCON bit will be cleared to 0 and an A/D conversion starts. There is no enough time for the A/D converter to complete the conversion before the next OVP7OS rising edge arrives. Then the ATARPTG bit will be set. The A/D converter circuitry continues to perform the current conversion, ignoring this OVP7OS rising edge trigger signal. After finishing the conversion, the A/D converter will not start the next conversion until a new OVP7OS rising edge started timer counter overflows.

Note: When the timer counter is counting-up and has not reached the maximum value, if an OVP7OS rising edge occurs, the TCON and ATARPTG bits will not be affected.

AADCTS[1:0]=01; TCEXTON=1



1-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=01)

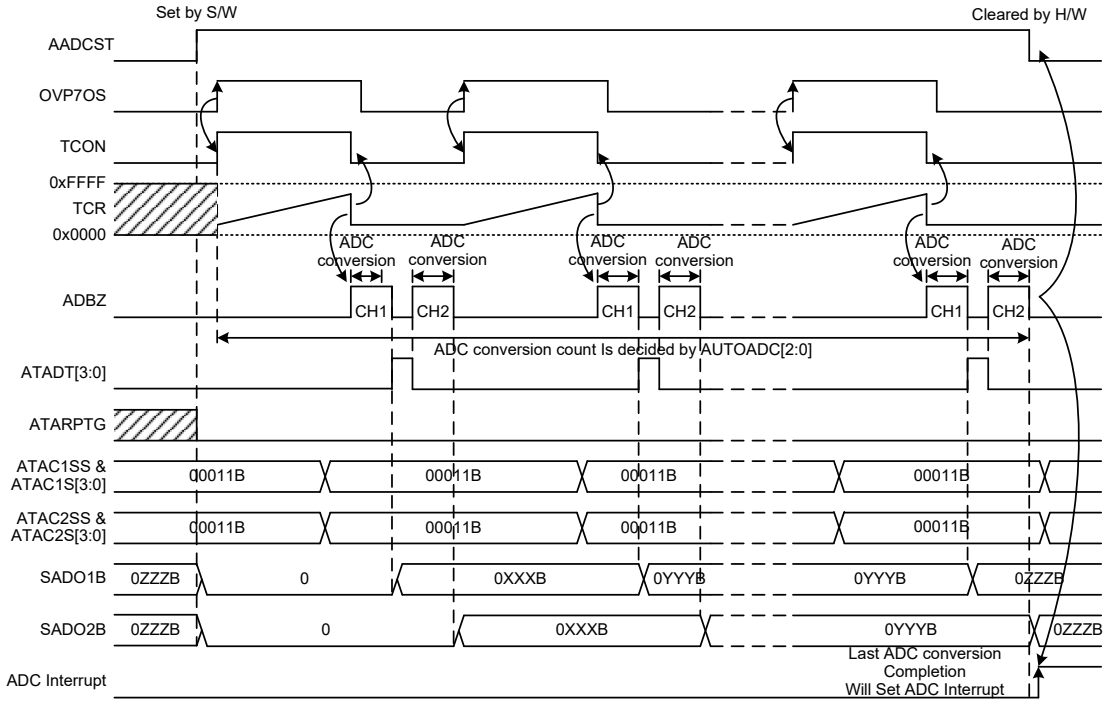
Note: 1. AUTOADC[2:0]≠000
2. XXXB is the first A/D converted data
3. YYYB is the accumulated A/D converted result
4. ZZZB is the final accumulated result of the N automatic conversions

2-Channel Automatic Conversion

After changing the AADCST bit from low to high, the ATARPTG bit will be cleared to 0 automatically. A rising edge of the OVP7OS signal will set the TCON bit to 1 to enable the timer counter to start counting up. Once the timer counter overflows, the TCON bit will be cleared and an A/D conversion on CH1 will start. After completion of the CH1 conversion, a interval time which is defined by the ATADT[3:0] bits is inserted. Then an A/D conversion on CH2 starts. After CH2 conversion has completed, an additional OVP7OS rising edge will be required. When the number of conversions on CH1 and CH2 reaches the set value N, the hardware will clear the AADCST bit automatically and the ADC interrupt flag ADF will be set. Here the number N is defined by the AUTOADC[2:0] bits which can be 1, 2, 4, 8 or 16.

The following shows an example of the 2-channel automatic conversion. In this example after AADCST changes from 0 to 1, the timer counter can be started by an OVP7OS rising edge and will count up. After it overlows, an A/D conversion will start. The A/D converter has enough time to complete the conversion on CH1 and CH2 before the next OVP7OS rising edge arrives and then converts the next data until all configured conversions are completed.

AADCTS[1:0]=01; TCEXTON=1



2-CH Automatic Conversion Timing – 1 (AADCTS[1:0]=01)

Note: 1. AUTOADC[2:0]≠000

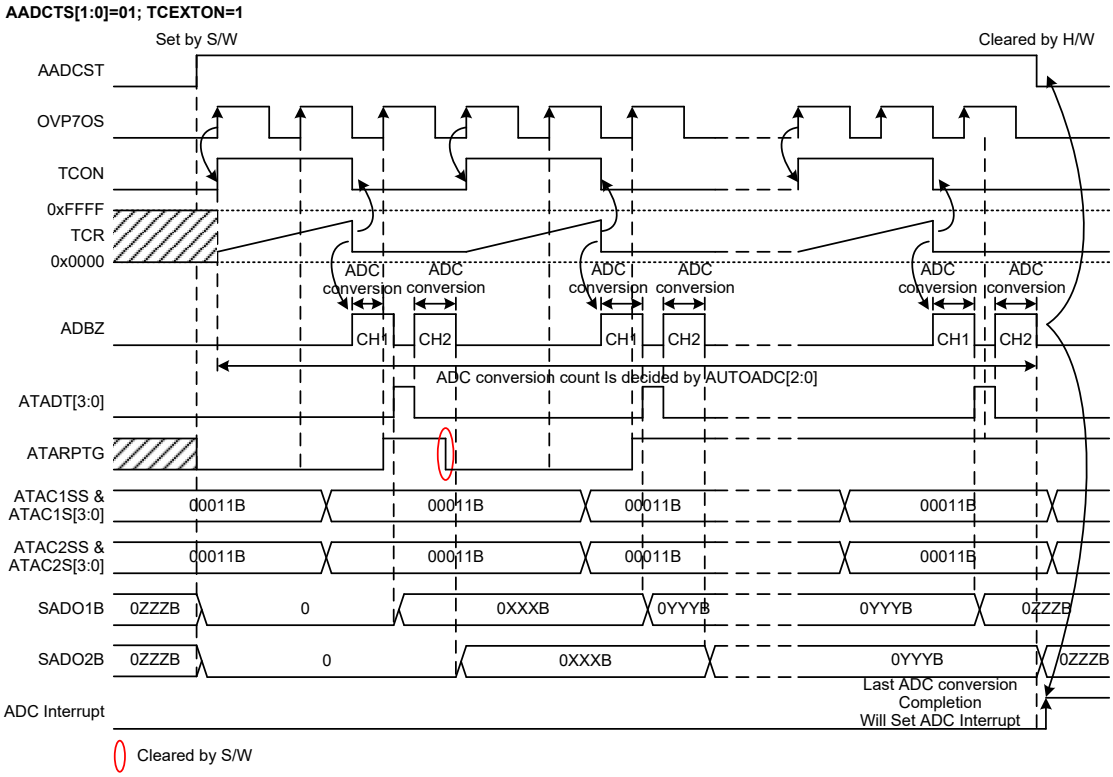
2. XXXB is the first A/D converted data

3. YYYB is the accumulated A/D converted result

4. ZZZB is the final accumulated result of the N automatic conversions

In the following timing diagram, after AADCST changes from 0 to 1, an OVP7OS rising edge will trigger the timer counter to start counting. After it overflows, the TCON bit will be cleared to 0 and an A/D conversion is started. However there is no enough time for the A/D converter to complete the conversions on CH1 and CH2 before the next OVP7OS rising edge arrives, then the ATARPTG bit will be set. The A/D converter circuitry continues to perform the current conversion, ignoring this OVP7OS rising edge trigger signal. After completing the conversion, the A/D converter will not start the next conversion until a new coming OVP7OS rising edge started timer counter overflows, and then convert the next data until all configured conversions are completed.

Note: When the timer counter is counting-up and has not reached the maximum value, if an OVP7OS rising edge occurs, the TCON and ATARPTG bits will not be affected.



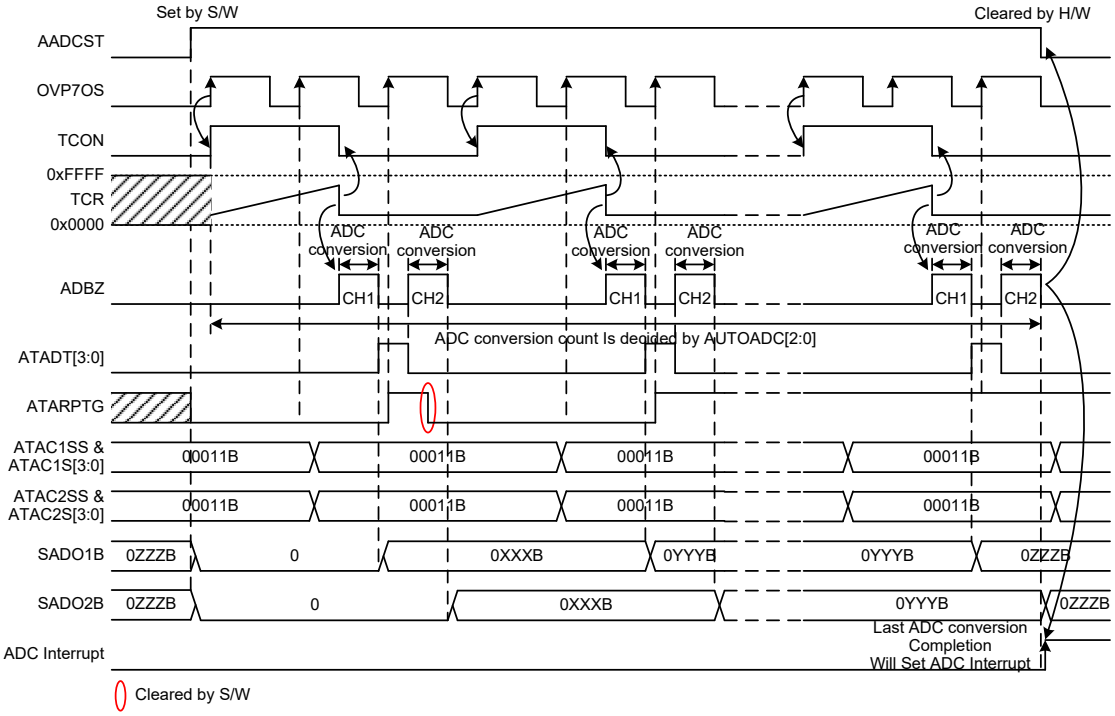
2-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=01)

Note: 1. AUTOADC[2:0]≠000
2. XXXB is the first A/D converted data
3. YYYB is the accumulated A/D converted result
4. ZZZB is the final accumulated result of the N automatic conversions

In the following timing diagram, after AADCST changes from 0 to 1, an OVP7OS rising edge will set TCON bit and the timer counter starts counting up. When it overflows, the TCON bit will be cleared to 0 automatically and an A/D conversion starts. If an OVP7OS rising edge occurs during the ATADT[3:0] period inserted between CH1 and CH2 conversions, the ATARPTG bit will be set. The A/D converter circuitry continues to perform the current conversion, ignoring this OVP7OS rising edge signal. After finishing the conversion, the A/D converter will not start the next conversion until a new coming OVP7OS rising edge started timer counter overflows, and then convert the next data until all configured conversions are completed.

Note: When the timer counter is counting-up and has not reached the maximum value, if an OVP7OS rising edge occurs, the TCON and ATARPTG bits will not be affected.

AADCTS[1:0]=01; TCEXTON=1



2-CH Automatic Conversion Timing – 3 (AADCTS[1:0]=01)

Note: 1. AUTOADC[2:0]≠000
2. XXXB is the first A/D converted data
3. YYYB is the accumulated A/D converted result
4. ZZZB is the final accumulated result of the N automatic conversions

AADCTS[1:0]=10B

When AADCTS[1:0] bits are 10 and the AUTOADC[2:0] bits are not equal to 000, setting the AADCST bit high will start the A/D automatic synchronization delay conversion. When setting the AADCST bit high, the TCON bit will also be set to 1 by the hardware to enable the timer counter to start counting. The timer counter will count from the initial value loaded by the preload register, TCR[15:0], to the full count of FFFFH, at which point the timer counter overflows, and an A/D conversion is started. The timer counter counts from the reload value again and when it overflows, the next conversion will be triggered. After N conversions have been completed, the TCON and AADCST bits will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set. Here the number N is defined by the AUTOADC[2:0] bits which can be 1, 2, 4, 8 or 16. If the timer counter is abnormal or TCON bit is cleared, the AADCST bit can be cleared using the software.

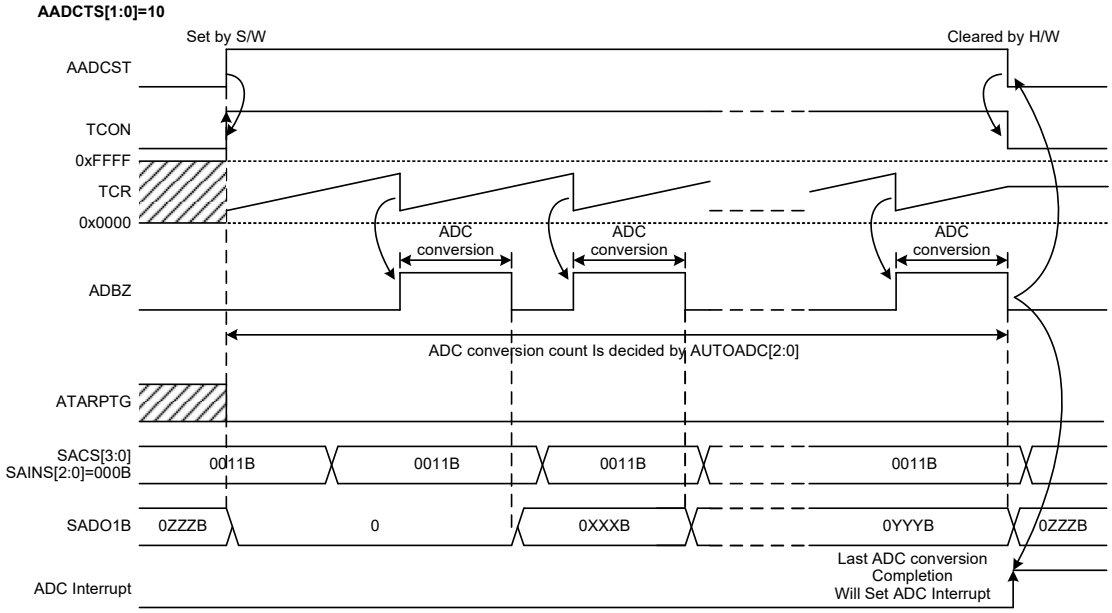
The AADCST is unable to provide a repeatable trigger. Once the AADCST bit is set high, the circuit cannot be triggered until the specified number N of automatic conversions are completed. In program design it should be noted that after the AADCST bit is cleared, the ADF flag should also be cleared using the application progrm.

Note that during the automatic conversion process, and the AADCST bit is cleared to 0, if the ADBZ bit is 1, wait for the ADBZ bit changes to 0, then switch to manual trigger conversion mode (AUTOADC[2:0]=000), and enable the A/D conversion one time (by setting the START bit from low to high and then low again). When the conversion is completed, switch back to automatic conversion mode, and then the automatic conversion circuit will be reset to its initial state. After the reset is completed, the AADCST bit can be set high again to enable another group of automatic conversions. If the specified conversions have been completed when the AADCST bit is cleared, the ADC interrupt can still be triggered.

1-Channel Automatic Conversion

When AADCST changes from 0 to1, the ATARPTG bit will be cleared to 0 automatically and the TCON bit is set to 1 to enable the timer counter to start counting up. When the timer counter overflows, the A/D converter starts a conversion and the timer counter restarts counting from the preload value. After the specified times of automatic conversions are completed, the AADCST and TCON bits will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set.

In the following example, after an A/D conversion is started by the timer counter overflow event, there is enough time for the ADC to complete a data conversion before the timer counter overflows again.



1-CH Automatic Conversion Timing – 1 (AADCTS[1:0]=10)

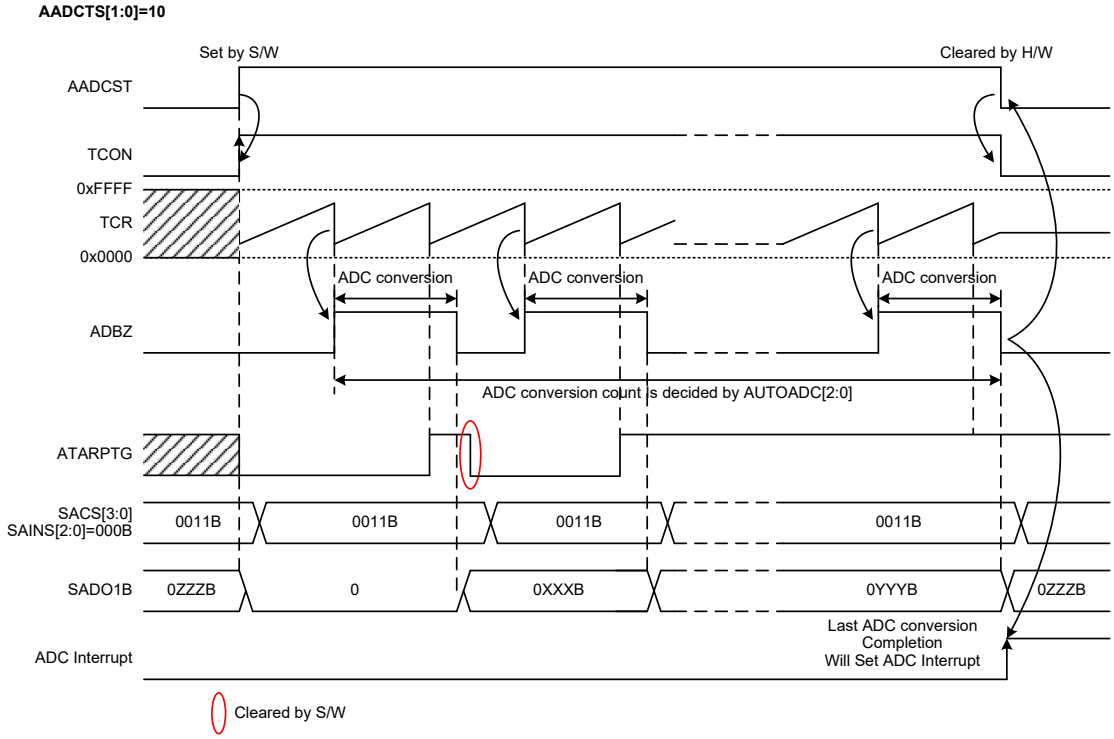
Note: 1. AUTOADC[2:0]≠000

2. XXXB is the first A/D converted data

3. YYYB is the accumulated A/D converted result

4. ZZZB is the final accumulated result of the N automatic conversions

In the following timing diagram, after an A/D conversion is started by the timer counter overflow event, there is no enough time for the ADC to complete the conversion before the timer counter overflows again. Then the ATARPTG bit will be set. The A/D converter circuitry continues to perform the current conversion, ignoring this overflow event. After completing the conversion, the A/D converter will not start the next conversion until a new timer counter overflow occurs, and then convert the next data until all configured conversions are completed.



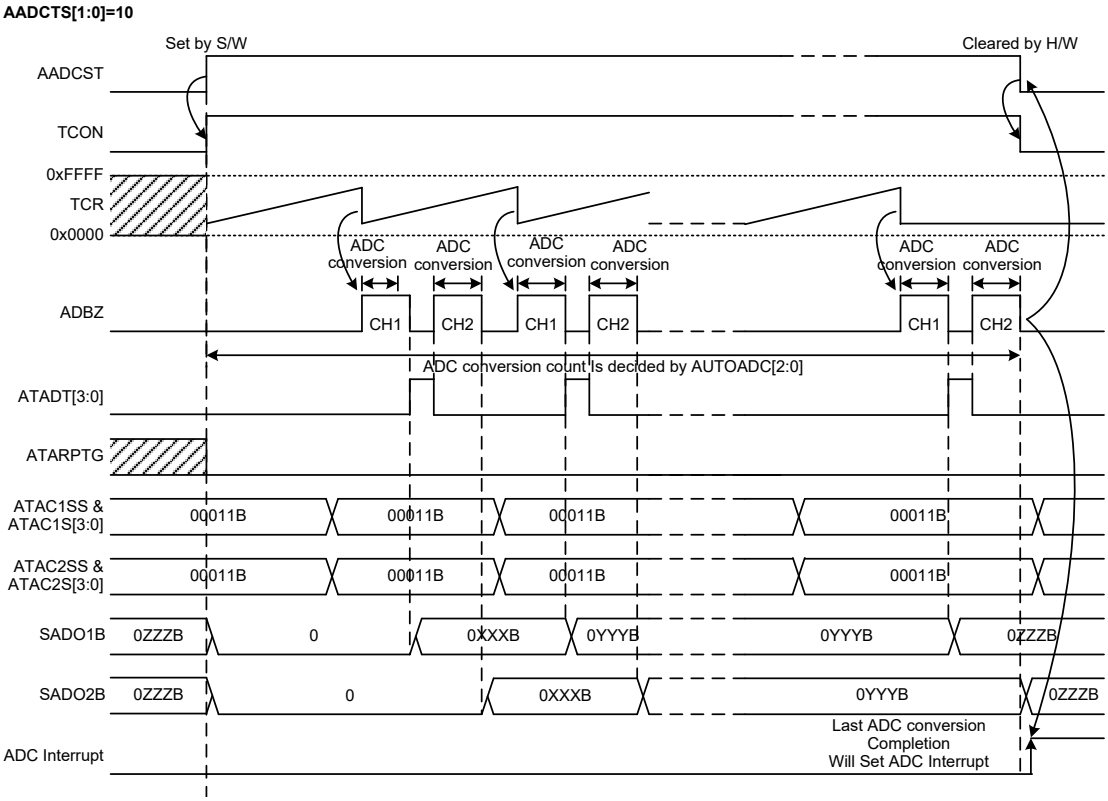
1-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=10)

Note: 1. AUTOADC[2:0]≠000
2. XXXB is the first A/D converted data
3. YYYB is the accumulated A/D converted result
4. ZZZB is the final accumulated result of the N automatic conversions

2-Channel Automatic Conversion

When setting the AADCST bit high, the ATARPTG bit will be cleared to 0 automatically and the TCON bit will also be set to 1 by the hardware to enable the timer counter to start counting. The timer counter will count from the initial value loaded by the preload register, TCR[15:0], to the full count of FFFFH, at which point the timer overflows and restarts counting up from the reload value. At the same time an A/D conversion on CH1 is started. After completion of the CH1 conversion, an interval time which is defined by the ATADT[3:0] bits is required. Then an A/D conversion on CH2 starts. After this CH2 conversion has completed, the circuit will wait for the next timer counter overflow event to start the next cycle of conversions. When the number of conversion cycles on CH1 and CH2 reaches the set value N, the hardware will clear the AADCST and TCON bits automatically and the ADC interrupt flag ADF will be set. Here the number N is defined by the AUTOADC[2:0] bits which can be 1, 2, 4, 8 or 16.

In the following timing diagram, after an A/D conversion is started by the timer counter overflow event, the A/D converter has enough time to complete the conversion on CH1 and CH2 before the timer counter overflows again, and then convert the next data until all configured conversions are completed.



2-CH Automatic Conversion Timing – 1 (AADCTS[1:0]=10)

Note: 1. AUTOADC[2:0]≠000

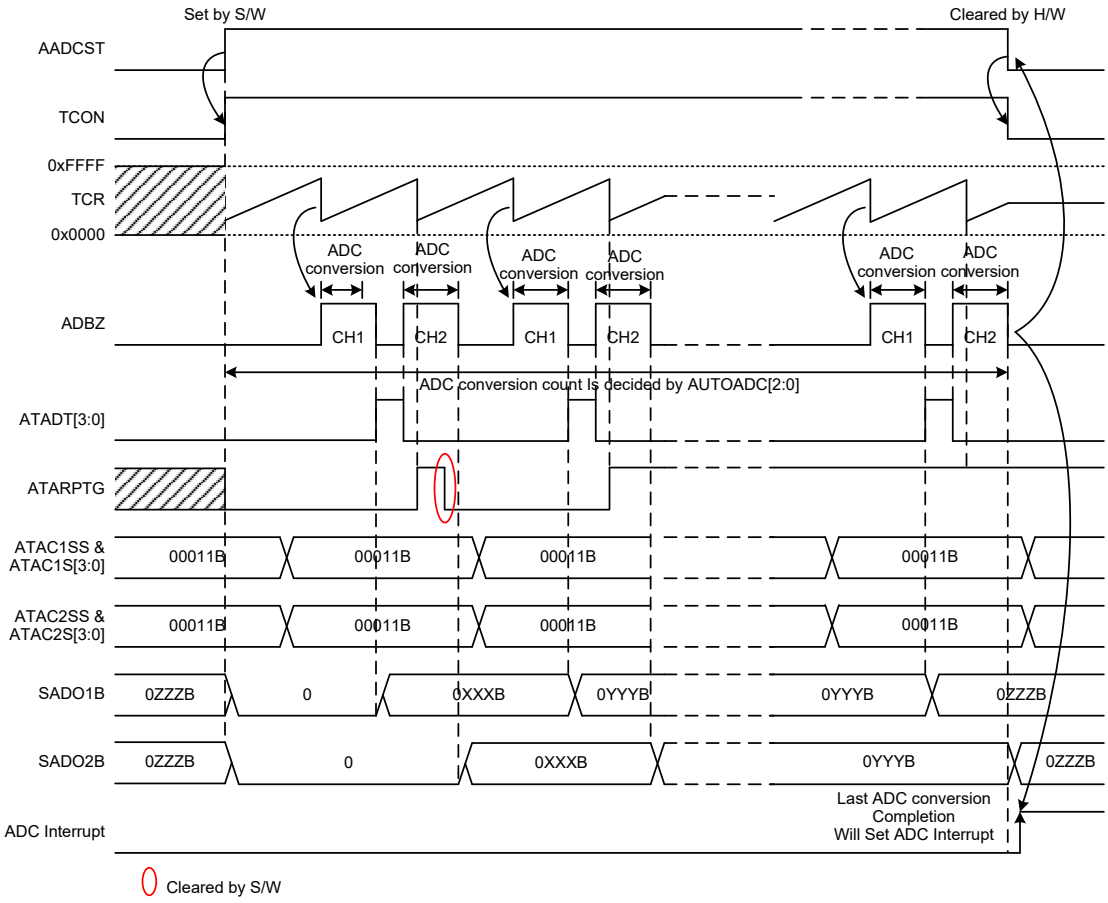
2. XXXB is the first A/D converted data

3. YYYB is the accumulated A/D converted result

4. ZZZB is the final accumulated result of the N automatic conversions

In the following timing diagram, after an A/D conversion is started by the timer counter overflow event, there is no enough time for the A/D converter to complete the conversions on CH1 and CH2 before the timer counter overflows again, then the ATARPTG bit will be set. The A/D converter circuitry continues to perform the current conversion, ignoring this counter overflow event. After finishing the conversion, the A/D converter will not start the next conversion until a new timer counter overflow event occurs, and then convert the next data until all configured conversions are complete.

AADCTS[1:0]=10



2-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=10)

Note: 1. AUTOADC[2:0]≠000
2. XXXB is the first A/D converted data
3. YYYB is the accumulated A/D converted result
4. ZZZB is the final accumulated result of the N automatic conversions

AADCTS[1:0]=11B

When AADCTS[1:0] bits are 11 and the AUTOADC[2:0] bits are not equal to 000, the A/D converter operates in automatic conversion mode and can start the conversion synchronously on setting AADCST high. When setting the AADCST bit high, an A/D conversion is started. when the conversion is complete, after a period of time which is defined by the ATADT[3:0] bits, the next conversion starts. After N conversions have been completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set. Here the number N is defined by the AUTOADC[2:0] bits which can be 1, 2, 4, 8 or 16.

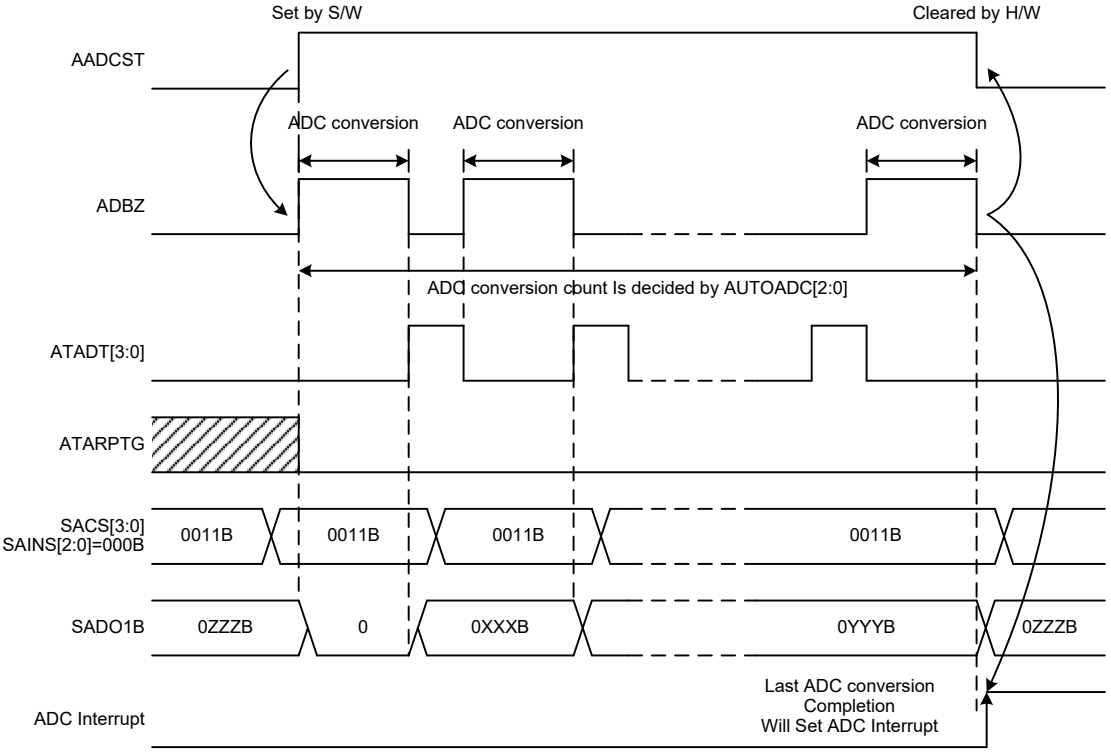
The AADCST is unable to provide a repeatable trigger. Once the AADCST bit is set high, the circuit cannot be triggered until the specified number N of automatic conversions are completed. In program design it should be noted that after the AADCST bit is cleared, the ADF flag should also be cleared using the application progrm.

Note that during the automatic conversion process, if the AADCST bit is cleared to 0 when the ADBZ bit is 1, then the automatic conversion circuit will not be reset to its initial state immediately until the current A/D conversion is complete and the ADBZ bit changes to 0. If the AADCST bit is cleared to 0 when the ADBZ bit is 0, the automatic conversion circuit will be reset immediately. After the reset is complete, the AADCST bit can be set high again to enable another group of automatic conversions. If the specified conversions have been completed when the AADCST bit is cleared, the ADC interrupt can still be triggered.

1-Channel Automatic Conversion

When AADCST changes from 0 to 1, the ATARPTG bit will be cleared to 0 automatically and an A/D automatic conversion starts. After the specified times of automatic conversions are completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set.

AADCTS[1:0]=11



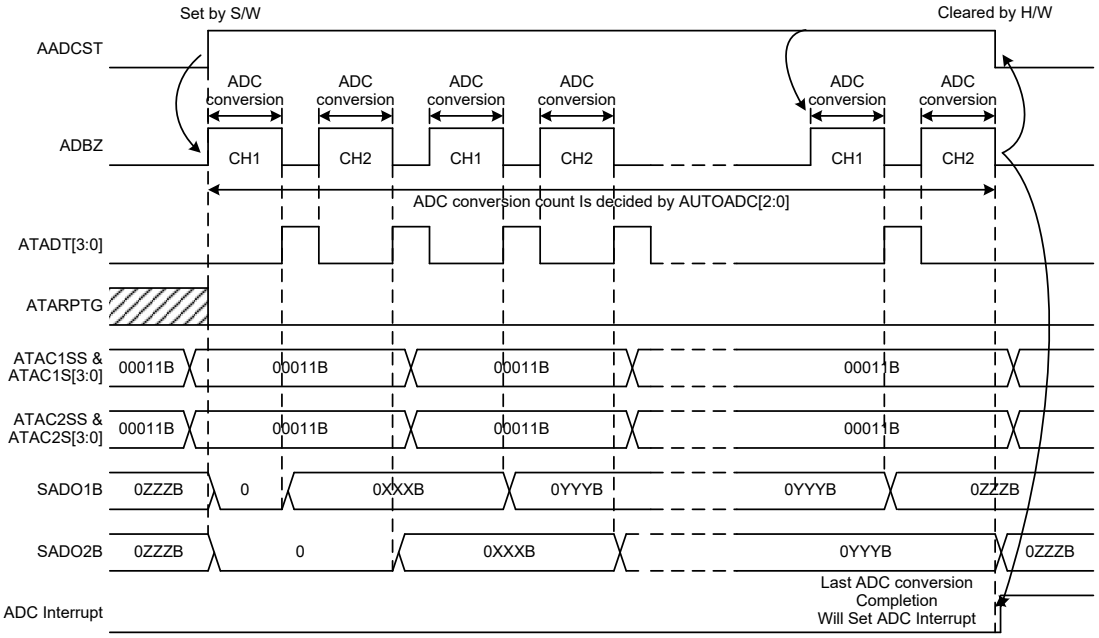
1-CH Automatic Conversion Timing (AADCTS[1:0]=11)

Note: 1. AUTOADC[2:0]≠000
2. XXXB is the first A/D converted data
3. YYYB is the accumulated A/D converted result
4. ZZZB is the final accumulated result of the n automatic conversions

2-Channel Automatic Conversion

When setting the AADCST bit from 0 to 1, the ATARPTG bit will be cleared to 0 automatically and an A/D conversion on CH1 is started. After completion of the CH1 conversion, an interval time which is defined by the ATADT[3:0] bits is required. Then an A/D conversion on CH2 starts. After this CH2 conversion is completed and an interval time defined by the ATADT[3:0] bits has passed, the next cycle of conversions on CH1 and CH2 starts again. When the number of conversion cycles on CH1 and CH2 reaches the set value N, the hardware will clear the AADCST bit automatically and the ADC interrupt flag ADF will be set. Here the number N is defined by the AUTOADC[2:0] bits which can be 1, 2, 4, 8 or 16.

AADCTS[1:0]=11



2-CH Automatic Conversion Timing (AADCTS[1:0]=11)

Note: 1. AUTOADC[2:0]≠000
2. XXXB is the first A/D converted data
3. YYYB is the accumulated A/D converted result
4. ZZZB is the final accumulated result of the n automatic conversions

A/D Automatic Conversion Software Stop Descriptions

When AADCTS[1:0]=00&11, the automatic conversion is processing, if the AADCST bit is cleared to 0 when the ADBZ bit is 1, then the automatic conversion circuit will not be reset to its initial state immediately until the current A/D conversion is completed and the ADBZ bit changes to 0. After the reset is completed, the AADCST bit can be set high again to enable another group of automatic conversions.

When AADCTS[1:0]=01&10, the automatic conversion is processing, and the AADCST bit is cleared to 0, if the ADBZ bit is 1, wait for the ADBZ bit changes to 0, then switch to manual trigger conversion mode (AUTOADC[2:0]=000), and enable the A/D conversion one time (by setting the START bit from low to high and then low again). When the conversion is completed, switch back to automatic conversion mode, and then the automatic conversion circuit will be reset to its initial state. After the reset is completed, the AADCST bit can be set high again to enable another group of automatic conversions. If the specified conversions have been completed when the AADCST bit is cleared, the ADC interrupt can still be triggered.

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D conversion internal circuitry can be switched off to reduce power consumption, by clearing the ADCEN bit in the SADC0 register. When this happens, the internal A/D conversion circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, as this may lead to some increase in power consumption.

A/D Conversion Function

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

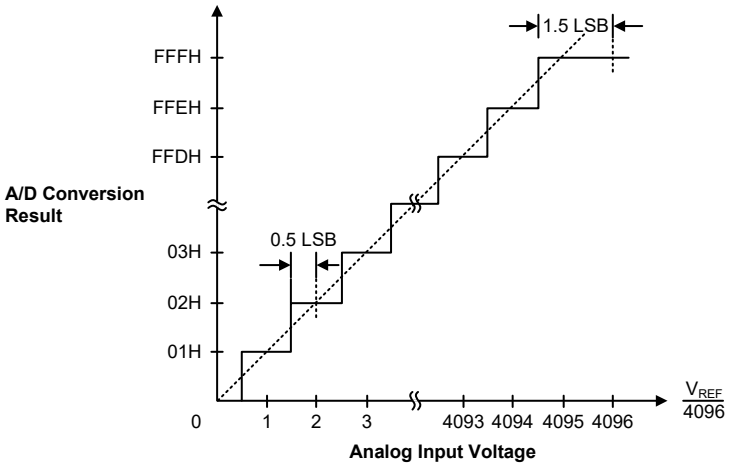
$$1\ LSB = V_{REF} \div 4096$$

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

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

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

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



Ideal A/D Conversion Function

A/D Converter Programming Examples

The following two programming examples illustrate how to setup and implement a Manual trigger start 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 converter interrupt is used to determine when the conversion is complete.

Example 1: using an ADBZ polling method to detect the end of conversion

```
mov  a,00h            ; disable A/D automatic conversion function
mov  SADC2,a
clr  ADE              ; disable ADC interrupt
mov  a,0Bh            ; select fSYS/8 as A/D clock and A/D input signal comes from
                      ; external channel
mov  SADC1,a          ; select VDD as A/D reference voltage source
mov  a,02h            ; setup PBS0 to configure pin AN0
mov  PBS0,a
mov  a,20h            ; enable A/D converter and select AN0 external channel input
mov  SADC0,a
:
start_conversion:
clr  START            ; high pulse on start bit to initiate conversion
set  START            ; reset A/D converter
clr  START            ; start A/D conversion
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
```

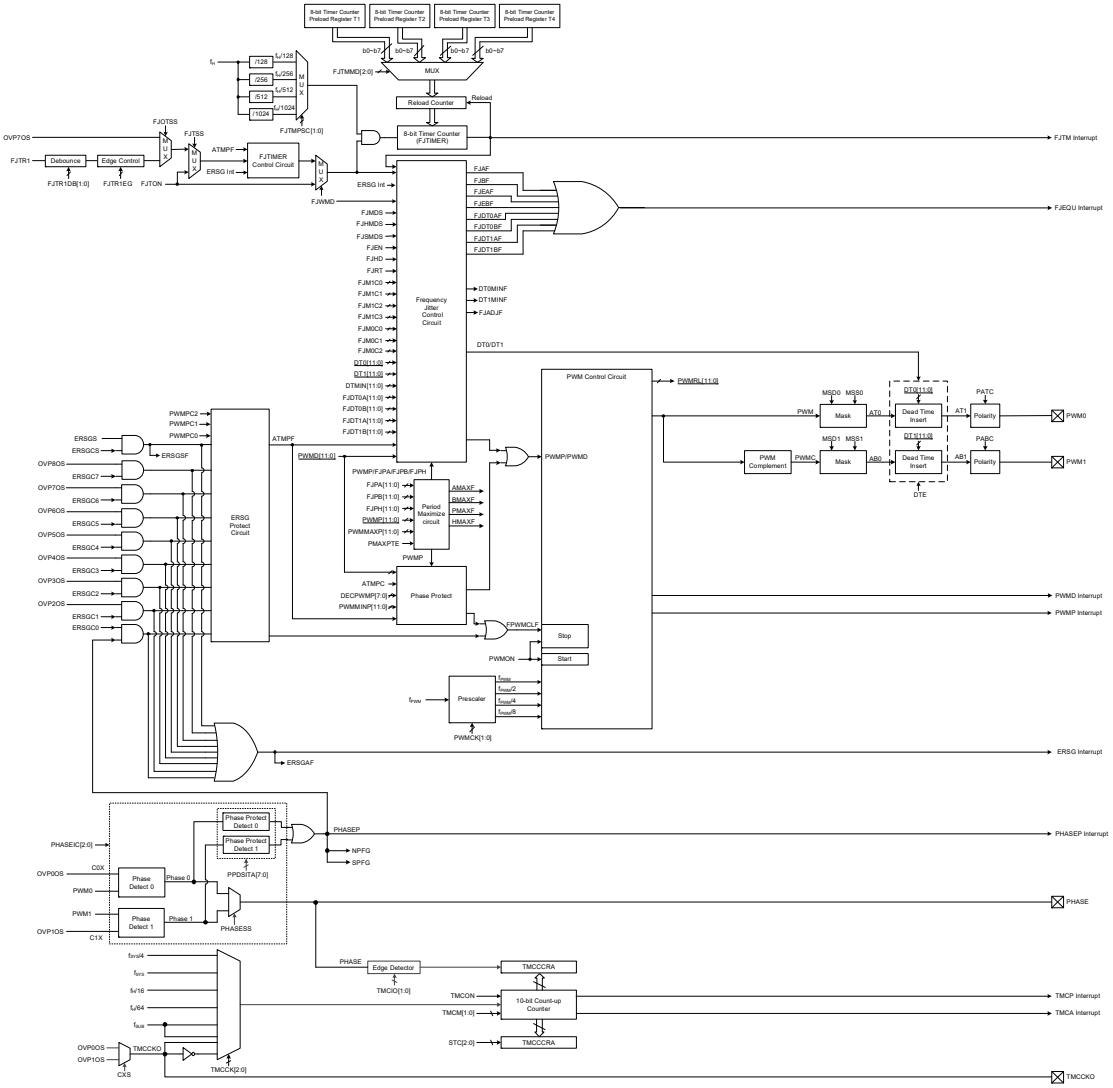
Example 2: using the interrupt method to detect the end of conversion

```
mov  a,00h            ; disable A/D automatic conversion function
mov  SADC2,a
clr  ADE              ; disable ADC interrupt
mov  a,0Bh            ; select fSYS/8 as A/D clock and A/D input signal comes from external
channel
mov  SADC1,a          ; select VDD as A/D reference voltage source
mov  a,02h            ; setup PBS0 to configure pin AN0
mov  PBS0,a
mov  a,20h            ; enable A/D converter and select AN0 external channel input
mov  SADC0,a
:
Start_conversion:
clr  START            ; high pulse on START bit to initiate conversion
set  START            ; reset A/D converter
clr  START            ; start A/D conversion
clr  ADF              ; clear ADC interrupt request flag
set  ADE              ; enable ADC interrupt
set  EMI              ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov  acc_stack,a      ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a   ; save STATUS to user defined memory
:
:
mov  a,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
```

High Resolution PWM with Dead-Time

The device contains a multi feature fully integrated PWM Generator which has complementary outputs for maximum application flexibility. The device also provides one group of complementary output pins. A multiple function protection mechanism is also provided. When an over current or phase error event occurs, the PWM switching can be shut off immediately. A 10-bit timer is provided for counter/timer and input capture operations. The PWM ciurcuit also contains a frequency jitter function which forces the original fixed operating frequency to experience periodic changes to reduce EMI interference.



Note: It should be note that the OVPnOS signal is sourced from the OVPn output "OVPnCOUT" non-inverting or inverting debounce signal, which can be selected by the software.

PWM Generator / Protection Circuit / Frequency Jitter Function Block Diagram

PWM Register Description

Overall PWM operation, phase protection, phase detection protection, 10-bit TMC, error signal protection, frequency jitter and other operations are controlled by a series of registers as listed in the following table.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PWMC0	MSS1	MSS0	PWMCK1	PWMCK0	UNU63	ATMPC	DTE	PWMON
PWMC1	FPWMCLF	ATMPF	PHASESS	CXS	PMAXPTE	ATMNPC	MSD1	MSD0
PWMC2	NPFG	PHASEIC2	PHASEIC1	PHASEIC0	SPFG	D2	ERSGS	UNU70
PLC	—	—	—	—	—	—	PABC	PATC
PWMPC0	PWMPC07	PWMPC06	PWMPC05	PWMPC04	PWMPC03	PWMPC02	PWMPC01	PWMPC00
PWMPC1	PWMPC17	PWMPC16	PWMPC15	PWMPC14	PWMPC13	PWMPC12	PWMPC11	PWMPC10
PWMPC2	ERSGAF	—	ERSGSF	ERSGCS	—	—	PWMPC21	PWMPC20
ERSGC	ERSGC7	ERSGC6	ERSGC5	ERSGC4	ERSGC3	ERSGC2	ERSGC1	ERSGC0
PWMPL	D7	D6	D5	D4	D3	D2	D1	D0
PWMPH	—	—	—	—	D11	D10	D9	D8
PWMDL	D7	D6	D5	D4	D3	D2	D1	D0
PWMDH	—	—	—	—	D11	D10	D9	D8
PWMRL	D7	D6	D5	D4	D3	D2	D1	D0
PWMRH	—	—	—	—	D11	D10	D9	D8
DT0L	D7	D6	D5	D4	D3	D2	D1	D0
DT0H	—	—	—	—	D11	D10	D9	D8
DT1L	D7	D6	D5	D4	D3	D2	D1	D0
DT1H	—	—	—	—	D11	D10	D9	D8
PWMMAXPL	D7	D6	D5	D4	D3	D2	D1	D0
PWMMAXPH	—	—	—	—	D11	D10	D9	D8
PWMMINPL	D7	D6	D5	D4	D3	D2	D1	D0
PWMMINPH	—	—	—	—	D11	D10	D9	D8
DECPWMP	D7	D6	D5	D4	D3	D2	D1	D0
TMCC0	—	TMCCK2	TMCCK1	TMCCK0	TMCON	STC2	STC1	STC0
TMCC1	TMCM1	TMCM0	TMCIO1	TMCIO0	—	—	—	—
TMCDL	D7	D6	D5	D4	D3	D2	D1	D0
TMCDH	—	—	—	—	—	—	D9	D8
TMCCCRAL	D7	D6	D5	D4	D3	D2	D1	D0
TMCCCRAH	—	—	—	—	—	—	D9	D8
PPDSITA	D7	D6	D5	D4	D3	D2	D1	D0
FJTMC	—	FJTMMD2	FJTMMD1	FJTMMD0	FJTON	—	FJTMPSC1	FJTMPSC0
FJTMR1	D7	D6	D5	D4	D3	D2	D1	D0
FJTMR2	D7	D6	D5	D4	D3	D2	D1	D0
FJTMR3	D7	D6	D5	D4	D3	D2	D1	D0
FJTMR4	D7	D6	D5	D4	D3	D2	D1	D0
FJTMD	D7	D6	D5	D4	D3	D2	D1	D0
FJPAL	D7	D6	D5	D4	D3	D2	D1	D0
FJPAH	—	—	—	—	D11	D10	D9	D8
FJPBL	D7	D6	D5	D4	D3	D2	D1	D0
FJPBH	—	—	—	—	D11	D10	D9	D8
FJPHL	D7	D6	D5	D4	D3	D2	D1	D0
FJPHH	—	—	—	—	D11	D10	D9	D8
DTMINL	D7	D6	D5	D4	D3	D2	D1	D0
DTMINH	—	—	—	—	D11	D10	D9	D8
FJDT0AL	D7	D6	D5	D4	D3	D2	D1	D0

Register Name	Bit							
	7	6	5	4	3	2	1	0
FJDT0AH	—	—	—	—	D11	D10	D9	D8
FJDT0BL	D7	D6	D5	D4	D3	D2	D1	D0
FJDT0BH	—	—	—	—	D11	D10	D9	D8
FJDT1AL	D7	D6	D5	D4	D3	D2	D1	D0
FJDT1AH	—	—	—	—	D11	D10	D9	D8
FJDT1BL	D7	D6	D5	D4	D3	D2	D1	D0
FJDT1BH	—	—	—	—	D11	D10	D9	D8
FJC0	FJWMD	FJEN	FJMDS	FJSMDS	FJHMDS	FJTSS	FJOTSS	FJADJF
FJC1	FJEAF	FJEBF	FJAF	FJBF	—	—	FJHD	FJRT
FJC2	FJDT0AF	FJDT0BF	FJDT1AF	FJDT1BF	—	FJTR1EG	FJTR1DB1	FJTR1DB0
FJF0	AMAXF	BMAXF	HMAXF	PMAXF	—	—	DT1MINF	DT0MINF
FJM1C0	—	FJCNT2	FJCNT1	FJCNT0	—	FJSA2	FJSA1	FJSA0
FJM1C1	—	FJTIMES2	FJTIMES1	FJTIMES0	FJACNT1	FJACNT0	FJASA1	FJASA0
FJM1C2	FJDT0EN	FJDT0ASA1	FJDT0ASA0	FJDT0CNT1	FJDT0CNT0	FJDT0SA2	FJDT0SA1	FJDT0SA0
FJM1C3	FJDT1EN	FJDT1ASA1	FJDT1ASA0	FJDT1CNT1	FJDT1CNT0	FJDT1SA2	FJDT1SA1	FJDT1SA0
FJM0C0	—	CFJCNT2	CFJCNT1	CFJCNT0	CFJS	CFJSA2	CFJSA1	CFJSA0
FJM0C1	CFJDT0EN	—	CFJDT0CNT1	CFJDT0CNT0	CFJDT0S	—	CFJDT0SA1	CFJDT0SA0
FJM0C2	CFJDT1EN	—	CFJDT1CNT1	CFJDT1CNT0	CFJDT1S	—	CFJDT1SA1	CFJDT1SA0

Register List

- **PWMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	MSS1	MSS0	PWMCK1	PWMCK0	UNU63	ATMPC	DTE	PWMON
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 **MSS1**: PWM1 output signal selection bit
0: Output PWMC signal (PWM signal)
1: Output MSD1 (software signal)
Note that if one of PWM0 and PWM1 is selected to output PWM signal, the other cannot be set to output the software signal. Otherwise it will be forced to output the PWM signal with the corresponding MSS1 or MSS0 bit being changed to 0 by the hardware.

Bit 6 **MSS0**: PWM0 output signal selection bit
0: Output PWM signal (PWM signal)
1: Output MSD0 (software signal)
Note that if one of PWM0 and PWM1 is selected to output PWM signal, the other cannot be set to output the software signal. Otherwise it will be forced to output the PWM signal with the corresponding MSS1 or MSS0 bit being changed to 0 by the hardware.

Bit 5~4 **PWMCK1~PWMCK0**: PWM counter clock source f_{PWMPSC} select
00: f_{PWM}
01: $f_{PWM}/2$
10: $f_{PWM}/4$
11: $f_{PWM}/8$

Bit 3 **UNU63**: Reserved, this bit must be fixed at "0"

Bit 2 **ATMPC**: PWM auto-adjust stop control
0: No operation
1: Stop auto-adjust
Note: 1. When PWMPCn[m+1:m] is equal to 00 or 01, setting the ATMPC bit to 1 has no effects, this bit can not be set to 1.

2. When PWMPCn[m+1:m] is equal to 10 or 11, if the corresponding signal disappears, setting the ATMPC bit to 1, the PWM auto-adjust will stop at the next period.
3. After the PWM auto-adjust stops, this bit will be cleared to 0 by the hardware.
4. Refer to the "ATMPC Setting Notes" section for details.

Bit 1 **DTE**: Dead-time insertion enable control
0: Disable
1: Enable
When both DTE and PWMON bits are high, the dead-time insertion function will be enabled.

Bit 0 **PWMON**: PWM circuit on/off control
0: Off
1: On
This bit controls the on/off of the overall PWM circuit. Setting the bit high enables the counter to run, clearing the bit disables the PWM. Clearing this bit to zero will stop the counter from counting and turn off the PWM which will reduce its power consumption. 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.
When PWMON=0 and MSS1 or MSS0=0, the PWM0 and PWM1 outputs will be floating.
The PWM outputs in different bit conditions are shown in the following table:

Bit Settings			PWM Outputs	
PWMON	MSS1	MSS0	PWM1	PWM0
0	0	0	Floating	Floating
0	0	1	Floating	Floating
0	1	0	Floating	Floating
0	1	1	MSD1	MSD0
1	0	0	PWMC	PWM
1	0	1	PWMC	PWM
1	1	0	PWMC	PWM
1	1	1	MSD1	MSD0

"x": Don't care

- **PWMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	FPWMCLF	ATMPF	PHASESS	CXS	PMAXPTE	ATMNPC	MSD1	MSD0
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 **FPWMCLF**: PWM auto shutdown flag
0: No PWM auto-shutdown occurred
1: PWM auto-shutdown occurred
This bit can only be cleared and not set by the application program.

Bit 6 **ATMPF**: PWM period auto-adjust flag
0: PWM period auto-adjust not started
1: PWM period auto-adjust started
This bit can only be cleared and not set by the application program. When PWMON changes from 0 to 1, the ATMPF will be cleared to zero by hardware.

Bit 5 **PHASESS**: Select the phase signal to be measured
0: Phase0
1: Phase1

Bit 4 **CXS**: Select the TMCCKO signal source
0: C0X (OVP0OS) signal
1: C1X (OVP1OS) signal

OVP0OS and OVP1OS are the over voltage protection circuit 0 and over voltage protection circuit 1 output signals.

Bit 3 **PMAXPTE**: PWMP maximum value limit control
0: No limit on the register written values
1: Maximum written value=PWMMAXP

Note that when the PMAXPTE bit is "0", the data written into the PWMP/FJPA/FJPB/FJPH will not be limited by the PWMMAXP value. When the PMAXPTE bit is "1", the data written into the PWMP/FJPA/FJPB/FJPH will be limited within the PWMMAXP value. If the value to write is greater than PWMMAXP, the data cannot be written into the PWMP/FJPA/FJPB/FJPH. But if a value greater than the PWMMAXP has been written into the PWMP/FJPA/FJPB/FJPH before changing the PMAXPTE from low to high and then no write action to the PWMP/FJPA/FJPB/FJPH occurs when the bit is 1, the PWMP/FJPA/FJPB/FJPH value will not be changed by the hardware.

The following table gives an example about writing data into PWMP:

Step	PMAXPTE	PWMMAXP	PWMP (To Write)	PWMP (Actual Result)
1	0	1000	500	500
2	0	1000	1024	1024
3	1	1000	—	1024
4	1	1000	1010	1024
5	1	1000	980	980
6	1	1000	1010	980

Bit 2 **ATMNPC**: Auto-adjust stops at ERSG signal being 0 (for ERSG1 and ERSG3 signals only)
0: No operation
1: Auo-adjust stops when the ERSG signal changed from 1 to 0

Note that when ERSG1 or ERSG3 signal is high, the corresponding PWMPCn[m+1:m] bits are 11, and the ATMNPC bit is 1, if the ERSG1 or ERSG3 signal changes from 1 to 0, the auto-adjust will stop in the same way as setting the ATMPC bit to 1.

Bit 1 **MSD1**: PWM1 Mask Data bit
0: Logic low
1: Logic high

Note that when both the MSS0 and MSS1 bits are 1, the PWM0 and PWM1 may output the corresponding MSD0 and MSD1 software signals simultaneously. However it should be noted that the PWM0 and PWM1 cannot output 1 simultaneously, otherwise they will be forced to output 0 simultaneously.

Bit 0 **MSD0**: PWM0 Mask Data bit
0: Logic low
1: Logic high

Note that when both the MSS0 and MSS1 bits are 1, the PWM0 and PWM1 may output the corresponding MSD0 and MSD1 software signals simultaneously. However it should be noted that the PWM0 and PWM1 cannot output 1 simultaneously, otherwise they will be forced to output 0 simultaneously.

- **PWMC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	NPFG	PHASEIC2	PHASEIC1	PHASEIC0	SPFG	D2	ERSGS	UNU70
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 **NPFG**: No phase detected flag
0: A phase width has been detected
1: No phase width detected

This bit can only be cleared and not set by the application program.

Bit 6~4 **PHASEIC2~PHAEIC0**: Phase detect & phase protect detect enable control
000: After PWMON bit changes to 1, start phase detection immediately
001: After PWMON bit changes to 1, wait for 1 PWM period before starting phase detection
010: After PWMON bit changes to 1, wait for 2 PWM periods before starting phase detection
011: After PWMON bit changes to 1, wait for 3 PWM periods before starting phase detection
100: After PWMON bit changes to 1, wait for 4 PWM periods before starting phase detection
101: After PWMON bit changes to 1, wait for 5 PWM periods before starting phase detection
110: After PWMON bit changes to 1, wait for 6 PWM periods before starting phase detection
111: After PWMON bit changes to 1, wait for 7 PWM periods before starting phase detection
When the PWMON bit changes from 0 to 1, the counter starts counting the PWM periods. When the preset value defined in these bits is reached, the counter stops and the phase detection starts. When PWMON changes from 1 to 0, the counter value will be cleared to 0.

Bit 3 **SPFG**: Small phase flag
0: Phase width > the value definded in PPDSITA
1: Phase width < the value definded in PPDSITA
This bit can only be cleared and not set by the application program.

Bit 2 Reserved, this bit must be fixed at "0"

Bit 1 **ERSGS**: Error signal software generation bit
0: Logic low
1: Logic high
Note that this bit is valid only when ERSGCS bit is 1. Setting the ERSGS bit to 1 can trigger the corresponding ERSG protection circuit which will adjust the PWM period or turn off the PWM according to PWMPC2[1:0] configurations.

Bit 0 **UNU70**: Reserved, this bit must be fixed at "0"

• ERSGC Register

Bit	7	6	5	4	3	2	1	0
Name	ERSGC7	ERSGC6	ERSGC5	ERSGC4	ERSGC3	ERSGC2	ERSGC1	ERSGC0
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 **ERSGC7**: Error Signal 7 (ERSG7 signal) to trigger ERSG protection circuit control
0: Disable
1: Enable
The ERSG7 is connected to the OVP8 output signal (OVP8OS).

Bit 6 **ERSGC6**: Error Signal 6 (ERSG6 signal) to trigger ERSG protection circuit control
0: Disable
1: Enable
The ERSG6 is connected to the OVP7 output signal (OVP7OS).

Bit 5 **ERSGC5**: Error Signal 5 (ERSG5 signal) to trigger ERSG protection circuit control
0: Disable
1: Enable
The ERSG5 is connected to the OVP6 output signal (OVP6OS).

Bit 4 **ERSGC4**: Error Signal 4 (ERSG4 signal) to trigger ERSG protection circuit control
0: Disable
1: Enable
The ERSG4 is connected to the OVP5 output signal (OVP5OS).

Bit 3 **ERSGC3**: Error Signal 3 (ERSG3 signal) to trigger ERSG protection circuit control
0: Disable
1: Enable
The ERSG3 is connected to the OVP4 output signal (OVP4OS).

Bit 2 **ERSGC2**: Error Signal 2 (ERSG2 signal) to trigger ERSG protection circuit control
0: Disable
1: Enable
The ERSG2 is connected to the OVP3 output signal (OVP3OS).

Bit1 **ERSGC1**: Error Signal 1 (ERSG1 signal) to trigger ERSG protection circuit control
0: Disable
1: Enable
The ERSG1 is connected to the OVP2 output signal (OVP2OS).

Bit 0 **ERSGC0**: Error Signal 0 (ERSG0 signal) to trigger ERSG protection circuit control
0: Disable
1: Enable
The ERSG0 is connected to the phase detect protect circuit output signal (PHASEP).

- **PWMPC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PWMPC07	PWMPC06	PWMPC05	PWMPC04	PWMPC03	PWMPC02	PWMPC01	PWMPC00
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 **PWMPC07~PWMPC06**: ERSG3 (OVP4OS) PWM protection control bits
00: PWM shutdown immediately
01: PWM shutdown after the current PWM period is completed
10: PWM is adjusted per PWM period until PWMP<PWMMINP, then shutdown
11: PWM is adjusted per PWM period until PWMP<PWMMINP, no shutdown
Refer to the following ERSG Protection Circuit section for more details.

Bit 5~4 **PWMPC05~PWMPC04**: ERSG2 (OVP3OS) PWM protection control bits
00: PWM shutdown immediately
01: PWM shutdown after the current PWM period is completed
10: PWM is adjusted per PWM period until PWMP<PWMMINP, then shutdown
11: PWM is adjusted per PWM period until PWMP<PWMMINP, no shutdown
Refer to the following ERSG Protection Circuit section for more details.

Bit 3~2 **PWMPC03~PWMPC02**: ERSG1 (OVP2OS) PWM protection control bits
00: PWM shutdown immediately
01: PWM shutdown after the current PWM period is completed
10: PWM is adjusted per PWM period until PWMP<PWMMINP, then shutdown
11: PWM is adjusted per PWM period until PWMP<PWMMINP, no shutdown
Refer to the following ERSG Protection Circuit section for more details.

Bit 1~0 **PWMPC01~PWMPC00**: ERSG0 (PHASEP) PWM protection control bits
00: PWM shutdown immediately
01: PWM shutdown after the current PWM period is completed
10: PWM is adjusted per PWM period until PWMP<PWMMINP, then shutdown
11: PWM is adjusted per PWM period until PWMP<PWMMINP, no shutdown
Refer to the following ERSG Protection Circuit section for more details.

• **PWMPC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PWMPC17	PWMPC16	PWMPC15	PWMPC14	PWMPC13	PWMPC12	PWMPC11	PWMPC10
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 **PWMPC17~PWMPC16**: ERSG7 (OVP8OS) PWM protection control bits
00: PWM shutdown immediately
01: PWM shutdown after the current PWM period is completed
10: PWM is adjusted per PWM period until PWMP<PWMMINP, then shutdown
11: PWM is adjusted per PWM period until PWMP<PWMMINP, no shutdown
Refer to the following ERSG Protection Circuit section for more details.

Bit 5~4 **PWMPC15~PWMPC14**: ERSG6 (OVP7OS) PWM protection control bits
00: PWM shutdown immediately
01: PWM shutdown after the current PWM period is completed
10: PWM is adjusted per PWM period until PWMP<PWMMINP, then shutdown
11: PWM is adjusted per PWM period until PWMP<PWMMINP, no shutdown
Refer to the following ERSG Protection Circuit section for more details.

Bit 3~2 **PWMPC13~PWMPC12**: ERSG5 (OVP6OS) PWM protection control bits
00: PWM shutdown immediately
01: PWM shutdown after the current PWM period is completed
10: PWM is adjusted per PWM period until PWMP<PWMMINP, then shutdown
11: PWM is adjusted per PWM period until PWMP<PWMMINP, no shutdown
Refer to the following ERSG Protection Circuit section for more details.

Bit 1~0 **PWMPC11~PWMPC10**: ERSG4 (OVP5OS) PWM protection control bits
00: PWM shutdown immediately
01: PWM shutdown after the current PWM period is completed
10: PWM is adjusted per PWM period until PWMP<PWMMINP, then shutdown
11: PWM is adjusted per PWM period until PWMP<PWMMINP, no shutdown
Refer to the following ERSG Protection Circuit section for more details.

• **PWMPC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	ERSGAF	—	ERSGSF	ERSGCS	—	—	PWMPC21	PWMPC20
R/W	R	—	R/W	R/W	—	—	R/W	R/W
POR	0	—	0	0	—	—	0	0

Bit 7 **ERSGAF**: Protection occurred flag
0: No protection occurred
1: Protection occurred
Note: 1. the ERSGAF bit is the result of the OR operation following the operation that the ERSG[7:1], ERSGS and PHASEP INT are respectively ANDed with their corresponding enable bits. When any input signal that the enable bit is 1 changes from 0 to 1, the ERSGAF bit will be set high and then will be cleared by the hardware when the protection signal disappears.
2. When ERSGAF=1 (protection occurred), the PWMPCn[m+1:m] bits cannot be changed by software (n=0~2; m=0, 2, 4, 6).

Bit 6 Unimplemented, read as "0"

Bit 5 **ERSGSF**: ERSGS software bit enabled ERSG protection circuit flag
0: No ERSGS software bit triggered ERSG protection occured
1: ERSGS software bit triggered ERSG protection occured
This bit can only be cleared and not set by the application program.

Bit 4 **ERSGCS**: Using ERSGS software bit to trigger ERSG protection circuit control
0: Disable
1: Enable

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **PWMPC21~PWMPC20**: ERSGS (software generation bit) ERSG protection control bits
00: PWM shutdown immediately
01: PWM shutdown after the current PWM period is completed
10: PWM is adjusted per PWM period until PWMP<PWMMINP, then shutdown
11: PWM is adjusted per PWM period until PWMP<PWMMINP, no shutdown
Refer to the following ERSG Protection Circuit section for more details.

• PLC Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1 **PABC**: PWM1 output polarity control
0: Non-invert
1: Invert

Bit 0 **PATC**: PWM0 output polarity control
0: Non-invert
1: Invert

• PWMPL 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**: 12-bit PWM period low byte buffer register
PWM 12-bit PWMP buffer register bit 7 ~ bit 0

• PWMPH Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **D11~D8**: 12-bit PWM period high byte buffer register
PWM 12-bit PWMP buffer register bit 11 ~ bit 8
PWM period=f_{PWMPSC}/(PWMP[11:0]+1), f_{PWMPSC} is the PWM counter clock, determined by PWMCK1~PWMCK0 bits.

• PWMDL 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**: 12-bit PWM duty low byte buffer register
PWM 12-bit PWMD buffer register bit 7 ~ bit 0

• **PWMDH Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **D11~D8**: 12-bit PWM duty high byte buffer register
PWM 12-bit PWMD buffer register bit 11 ~ bit 8
PWM duty = f_{PWM}/(PWMD[11:0]+1)
The PWMD value should meet the condition: 1≤PWMD≤(PWMP-1)

• **PWMRL Register**

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

Bit 7~0 **D7~D0**: 12-bit PWM counter low byte register
PWM 12-bit counter bit 7 ~ bit 0

• **PWMRH Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **D11~D8**: 12-bit PWM counter high byte register
PWM 12-bit counter bit 11 ~ bit 8

• **DT0L 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**: 12-bit PWM0 dead-time low byte buffer register
PWM0 dead-time buffer register bit 7 ~ bit 0

• **DT0H Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **D11~D8**: 12-bit PWM0 dead-time high byte buffer register
PWM0 dead time buffer register bit 11 ~ bit 8
PWM0 Dead-Time=(DT0[11:0]+1)/f_{DT}, where f_{DT}=f_{PWM}
If DT0 is set greater than PWMD, the maximum counting value is PWMD.

• **DT1L 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**: 12-bit PWM1 dead-time low byte buffer register
PWM1 dead-time buffer register bit 7 ~ bit 0

• **DT1H Register**

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

Bit 7~4 Unimplemented, read as "0"
Bit 3~0 **D11~D8**: 12-bit PWM1 dead-time high byte buffer register
PWM1 dead-time buffer register bit 11 ~ bit 8
PWM1 Dead-Time=(DT1[11:0]+1)/f_{DT}, where $f_{DT}=f_{PWM}$
If DT1 is set greater than (PWMP-PWMD), the maximum counting value is (PWMP-PWMD).

• **PWMMAXPL 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**: 12-bit PWM maximum period low byte register
PWM 12-bit PWMMAXP bit 7 ~ bit 0

• **PWMMAXPH Register**

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

Bit 7~4 Unimplemented, read as "0"
Bit 3~0 **D11~D8**: 12-bit PWM maximum period high byte register
PWM 12-bit PWMMAXP bit 11 ~ bit 8

• **PWMMINPL 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**: 12-bit PWM minimum period low byte register
PWM 12-bit PWMMINP bit 7 ~ bit 0

• **PWMMINPH Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **D11~D8**: 12-bit PWM minimum period high byte register
PWM 12-bit PWMMINP bit 11 ~ bit 8

• **DECPWMP 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 Decrease PWM period register
8-bit DECPWMP Register bit 7 ~ bit 0

• **TMCC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TMCCK2	TMCCK1	TMCCK0	TMCON	STC2	STC1	STC0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~4 **TMCCK2~TMCCK0**: Select TMC counter clock source
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: TMCCKO rising edge
111: TMCCKO falling edge
These three bits are used to select the clock source for the Timer counter. The TMCCKO clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the Operating Modes and System Clocks section.

Bit 3 **TMCON**: TMC counter on/off control
0: Off
1: On
This bit controls the overall on/off function of the Timer. Setting the bit high enables the counter to run, clearing the bit disables the Timer. Clearing this bit to zero will stop the counter from counting and turn off the Timer which will reduce its power consumption. When the bit changes state from low to high the internal counter and the TMCCCRA value will be reset to zero, however when the bit changes from high to low, the internal counter and TMCCCRA will retain its residual value until the bit returns high again.

Bit 2~0 **STC2~STC0**: Select a period of time
000: $f_{PWM}/8192$
001: $f_{PWM}/16384$
010: $f_{PWM}/32768$
011: $f_{PWM}/65536$
100: $f_{PWM}/131072$

101: f_{PWM}/262144
110: f_{PWM}/524288
111: f_{PWM}/1048576

As f_{PWM} is 32MHz, these three bits provide eight timer period selections of around 0.25ms, 0.5ms, 1ms, 2ms, 4ms, 8ms, 16ms and 32ms. When the selected time has elapsed, the value of the 10-bit counter will be written into the TMCCCRA register and the TMCAF flag will be set.

• TMCC1 Register

Bit	7	6	5	4	3	2	1	0
Name	TMCM1	TMCM0	TMCIO1	TMCIO0	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

Bit 7~6 **TMCM1~TMCM0**: Select TMC operating mode
00/01: Capture Input Mode
10/11: Timer/Counter Mode
These two bits setup the 10-bit TMC operating mode, Capture Input Mode or Timer/Counter Mode. Refer to the following 10-bit TMC Counter section for more details about these two modes.

Bit 5~4 **TMCIO1~TMCIO0**: Select TMC function for Capture Input Mode
00: Input capture at rising edge of PHASE
01: Input capture at falling edge of PHASE
10: Input capture at falling/rising edge of PHASE
11: Input capture disabled
These two bits are invaild if the TMC is in the Timer/Counter Mode.

Bit 3~0 Unimplemented, read as "0"

• TMCDL Register

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

Bit 7~0 **D7~D0**: 10-bit TMC counter low byte register
TMC 10-bit counter bit 7 ~ bit 0

• TMCDH Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **D9~D8**: 10-bit TMC counter high byte register
TMC 10-bit counter bit 9 ~ bit 8

• TMCCCRAL 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**: TMCCCRA low byte register bit 7 ~ bit 0
10-bit TMCCCRA bit 7 ~ bit 0

• **TMCCCRAH 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**: TMCCCRA high byte register bit 1 ~ bit 0
10-bit TMCCCRA bit 9 ~ bit 8

• **PPDSITA 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**: Phase Detect Protect Sita register bit 7 ~ bit 0
$\Delta time=(PPDSITA[7:0]+1)/f_{PWM}$

• **FJTMC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	FJTMMD2	FJTMMD1	FJTMMD0	FJTON	—	FJTMPSC1	FJTMPSC0
R/W	—	R	R	R	R/W	—	R/W	R/W
POR	—	0	0	0	0	—	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~4 **FJTMMD2~FJTMMD0**: FJTIMER working section indication bits
000: T1 section – t0~t1
001: T2 section – t1~t2
010: T3 section – t2~t3
011: T4 section – t3~t4
1xx: T5 section – t4~t0

Bit 3 **FJTON**: FJTIMER counter enable control
0: Disable
1: Enable
When PWMON is 0 or PWMON is 1 with FJWMD being 0, the FJTIMER can be used as a general timer and the reload register uses the FJTMR1. When FJTON changes from 0 to 1, the FJTMR1 value will be reloaded to the FJTIMER. Clearing the FJTON bit can turn off the FJTIMER counter and the divider clock, thus reducing additional power consumption.

Bit 2 Unimplemented, read as "0"

Bit 1~0 **FJTMPSC1~FJTMPSC0**: FJTIMER counter clock selection – $f_{FJTIMER}$
00: $f_H/128$
01: $f_H/256$
10: $f_H/512$
11: $f_H/1024$

• **FJTMR1 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**: FJTIMER T1 preload register byte
T1 Interval = $(256\text{-FJTMR1}[7:0])\times f_{FJTIMER}$

• **FJTMR2 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**: FJTIMER T2 preload register byte
T2 Interval = (256-FJTMR2[7:0])×$f_{FJTIMER}$

• **FJTMR3 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**: FJTIMER T3 preload register byte
T3 Interval = (256-FJTMR3[7:0])×$f_{FJTIMER}$

• **FJTMR4 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**: FJTIMER T4 preload register byte
T4 Interval = (256-FJTMR4[7:0])×$f_{FJTIMER}$

• **FJTMD 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**: FJTIMER counter register byte

• **FJPAL 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**: 12-bit PWM period approach Taget A low byte register
12-bit FJPA bit 7 ~ bit 0

• **FJPAH Register**

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

Bit 7~4 Unimplemented, read as "0"
Bit 3~0 **D11~D8**: 12-bit PWM period approach Taget A high byte register
12-bit FJPA bit 11 ~ bit 8

• **FJPBL 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**: 12-bit PWM period approach Taget B low byte register
12-bit FJPB bit 7 ~ bit 0

• **FJPBH Register**

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

Bit 7~4 Unimplemented, read as "0"
Bit 3~0 **D11~D8**: 12-bit PWM period approach Taget B high byte register
12-bit FJPB bit 11 ~ bit 8

• **FJPHL 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**: PWM period modification low byte register during frequency jittering process
12-bit FJPH bit 7 ~ bit 0

• **FJPHH Register**

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

Bit 7~4 Unimplemented, read as "0"
Bit 3~0 **D11~D8**: PWM period modification high byte register during frequency jittering process
12-bit FJPH bit 11 ~ bit 8
The data in the FJPH registers can be used by the hardware to modify the PWMP and PWMD values during frequency jittering process.

• **DTMINL 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**: 12-bit dead-time minimum definition low byte register
12-bit PWM DTMIN bit 7 ~ bit 0

• **DTMINH Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **D11~D8**: 12-bit dead-time minimum definition high byte register
12-bit PWM DTMIN bit 11 ~ bit 8
Minimum Dead Time = (DTMIN[11:0]+1) f_{DT}, where $f_{DT}=f_{PWM}$

• **FJDT0AL 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**: 12-bit PWM approach Taget-A dead-time 0 low byte register
12-bit FJDT0A bit 7 ~ bit 0

• **FJDT0AH Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **D11~D8**: 12-bit PWM approach Taget-A dead-time 0 high byte register
12-bit FJDT0A bit 11 ~ bit 8
FJDT0A dead time = (FJDT0A[11:0]+1)/f_{DT}, where $f_{DT}=f_{PWM}$

• **FJDT0BL 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**: 12-bit PWM approach Taget-B dead-time 0 low byte register
12-bit FJDT0B bit 7 ~ bit 0

• **FJDT0BH Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **D11~D8**: 12-bit PWM approach Taget-B dead-time 0 high byte register
12-bit FJDT0B bit 11 ~ bit 8
FJDT0B dead time=(FJDT0B[11:0]+1)/f_{DT}, where $f_{DT}=f_{PWM}$.

• **FJDT1AL 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**: 12-bit PWM approach Taget-A dead-time 1 low byte register
12-bit FJDT1A bit 7 ~ bit 0

• **FJDT1AH Register**

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

Bit 7~4 Unimplemented, read as "0"
Bit 3~0 **D11~D8**: 12-bit PWM approach Taget-A dead-time 1 high byte register
12-bit FJDT1A bit 11 ~ bit 8
FJDT1A dead time=(FJDT1A[11:0]+1)/f_{DT}, where $f_{DT}=f_{PWM}$.

• **FJDT1BL 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**: 12-bit PWM approach Taget-B dead-time 1 low byte register
12-bit FJDT1B bit 7 ~ bit 0

• **FJDT1BH Register**

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

Bit 7~4 Unimplemented, read as "0"
Bit 3~0 **D11~D8**: 12-bit PWM approach Taget-B dead-time 1 high byte register
12-bit FJDT1B bit 11 ~ bit 8
FJDT1B dead time=(FJDT1B[11:0]+1)/f_{DT}, where $f_{DT}=f_{PWM}$.

• **FJC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	FJWMD	FJEN	FJMDS	FJSMDS	FJHMDS	FJTSS	FJOTSS	FJADJF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
POR	0	0	0	0	0	0	0	0

Bit 7 **FJWMD**: Frequency jitter operation mode selection
0: S/W frequency jitter mode (FJTIMER unused)
1: H/W frequency jitter mode (FJTIMER to be used together)
When ATMPF changes from 0 to 1, this bit will be cleared to 0 by hardware.
Bit 6 **FJEN**: Frequency jitter function enable control
0: Disable
1: Enable

Note: 1. Only when FJWMD bit is 0, this bit can be changed by the software.
2. If an ERSG interrupt occurs, the FJEN bit will be cleared to zero by the hardware automatically.
3. If PWMON=0 or ATMPF=1, the frequency jitter function cannot be started (FJEN=0).

Bit 5 **FJMDS**: Mode selection for S/W or H/W frequency jittering T1 and T4 sections
0: Count mode – fixed number of counting times
1: Approach mode – decrease or increase to approach the taget

Bit 4 **FJSMDS**: Approach register selection for S/W frequency jittering
0: FJPA/FJDT0A/FJDT1A
1: FJPB/FJDT0B/FJDT1B
Note that this bit is available only when FJMDS is 1.

Bit 3 **FJHMDS**: H/W frequency jittering working section selection
0: T1~T4 sections
1: T2~T3 sections

Bit 2 **FJTSS**: Trigger signal selection for starting H/W frequency jittering function
0: External signal (OVP7OS or FJTR1) trigger
1: FJTON bit
If external signal is selected, the FJOTSS bit will determine which signal to use.

Bit 1 **FJOTSS**: External signal selecton for starting H/W frequency jittering function
0: OVP7OS
1: FJTR1
Note that this bit is available only when FJTSS is 0.

Bit 0 **FJADJF**: Frequency jitter function start flag
0: Not started
1: Start
Note that when this bit is 1 and FJHD is 0, the contents of the registers including PWMP, PWMD, FJM1C0 @ FJWMD=1, DT0 @ FJDT0EN=1/CFJDT0EN=1, FJM1C2 (FJDT0SA[2:0] bits) @ FJDT0EN=1, DT1 @ FJDT1EN=1/CFJDT1EN=1, and FJM1C3 (FJDT1SA[2:0] bits) @ FJDT1EN=1, cannot be modified by the software.

- **FJC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	FJEAF	FJEBF	FJAF	FJBF	—	—	FJHD	FJRT
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7 **FJEAF**: PWMP approaching FJPA complete flag (ends due to the maximum value limit)
0: Unfinished
1: Finished
This bit can only be cleared and not set by the application program. If this bit is 1, then it will be cleared to zero automatically by hardware when the FJEN bit changes from 0 to 1 in S/W frequency jittering mode or the FJTON bit changes from 0 to 1 in H/W frequency jittering mode. This bit can also be cleared to zero by hardware when FJMDS=1 & FJWMD=1 & FJHMDS=0 and FJTMMD[2:0] changes from 010 to 011.

Bit 6 **FJEBF**: PWMP approaching FJPB complete flag (ends due to the maximum value limit)
0: Unfinished
1: Finished
This bit can only be cleared and not set by the application program. If this bit is 1, then it will be cleared to zero automatically by hardware when the FJEN bit changes from 0 to 1 in S/W frequency jittering mode or the FJTON bit changes from 0 to 1 in H/W frequency jittering mode. This bit can also be cleared to zero by hardware when FJMDS=1 & FJWMD=1 & FJHMDS=0 and FJTMMD[2:0] changes from 001 to 010.

Bit 5 **FJAF**: PWMP approaching FJPA complete flag
0: Unfinished
1: Finished
This bit can only be cleared and not set by the application program. If this bit is 1, then it will be cleared to zero automatically by hardware when the FJEN bit changes from 0 to 1 in S/W frequency jittering mode or the FJTON bit changes from 0 to 1 in H/W frequency jittering mode. This bit can also be cleared to zero by hardware when FJMDS=1 & FJWMD=1 & FJHMDS=0 and FJTMMD[2:0] changes from 010 to 011.

Bit 4 **FJBF**: PWMP approaching FJPB complete flag
0: Unfinished
1: Finished
This bit can only be cleared and not set by the application program. If this bit is 1, then it will be cleared to zero automatically by hardware when the FJEN bit changes from 0 to 1 in S/W frequency jittering mode or the FJTON bit changes from 0 to 1 in H/W frequency jittering mode. This bit can also be cleared to zero by hardware when FJMDS=1 & FJWMD=1 & FJHMDS=0 and FJTMMD[2:0] changes from 001 to 010.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **FJHD**: Change PWMP/PWMD values using FJPH by sofeware in frequency jittering process
1→0: Change PWMP/PWMD values
After the required data has been written into the FJPH registers by software, when FJHD bit changes from 1 to 0, the PWMP and PWMD values will be changed by hardware based on the FJPH register data. The PWMD value is equal to the FJPH value divided by 2 with decimal omitted unconditionally.

Bit 0 **FJRT**: Restart counting for frequency jitter variables
0: No operation
1: Clear the counter
When this bit is 1, then changing the FJHD bit from 1 to 0 correctly will let the internal counter be cleared and restart counting, and the FJRT bit is cleared to zero by hardware.

• FJC2 Register

Bit	7	6	5	4	3	2	1	0
Name	FJDT0AF	FJDT0BF	FJDT1AF	FJDT1BF	—	FJTR1EG	FJTR1DB1	FJTR1DB0
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 **FJDT0AF**: DT0 approaching FJDT0A complete flag
0: Unfinished
1: Finished
This bit can only be cleared and not set by the application program. If this bit is 1, then it will be cleared to zero by hardware when the FJEN bit changes from 0 to 1 in S/W frequency jittering mode or the FJTON bit changes from 0 to 1 in H/W frequency jittering mode.

Bit 6 **FJDT0BF**: DT0 approaching FJDT0B complete flag
0: Unfinished
1: Finished
This bit can only be cleared and not set by the application program. If this bit is 1, then it will be cleared to zero by hardware when the FJEN bit changes from 0 to 1 in S/W frequency jittering mode or the FJTON bit changes from 0 to 1 in H/W frequency jittering mode.

Bit 5 **FJDT1AF**: DT1 approaching FJDT1A complete flag
0: Unfinished
1: Finished
This bit can only be cleared and not set by the application program. If this bit is 1, then it will be cleared to zero by hardware when the FJEN bit changes from 0 to 1 in S/W frequency jittering mode or the FJTON bit changes from 0 to 1 in H/W frequency jittering mode.

Bit 4 **FJDT1BF**: DT1 approaching FJDT1B complete flag
0: Unfinished
1: Finished
This bit can only be cleared and not set by the application program. If this bit is 1, then it will be cleared to zero by hardware when the FJEN bit changes from 0 to 1 in S/W frequency jittering mode or the FJTON bit changes from 0 to 1 in H/W frequency jittering mode.

Bit 3 Unimplemented, read as "0"

Bit 2 **FJTR1EG**: FJTR1 signal edge selection for triggering FJTIMER
0: Rising edge
1: Falling edge

Bit 1~0 **FJTR1DB1~FJTR1DB0**: FJTR1 pin input debounce time selection
00: Bypass, no debounce
01: (7~8)×t_H
10: (15~16)×t_H
11: (23~24)×t_H

• FJF0 Register

Bit	7	6	5	4	3	2	1	0
Name	AMAXF	BMAXF	HMAXF	PMAXF	—	—	DT1MINF	DT0MINF
R/W	R	R	R	R	—	—	R	R
POR	0	0	0	0	—	—	0	0

Bit 7 **AMAXF**: FJPA write invalid flag
0: Correct wirte
1: Invalid write (FJPA > PWMMAXP)
Note: when PMAXPTE=0, any data can be written into FJPA, regardless of its value and the AMAXF bit remains 0. When PMAXPTE=1, writing a data greater than PWMMAXP into the FJPA will be failed, the AMAXF bit being 1 while writing a data equal to or lower than PWMMAXP can be successful, the AMAXF bit being 0.

Bit 6 **BMAXF**: FJPB write invalid flag
0: Correct wirte
1: Invalid write (FJPB > PWMMAXP)
Note: when PMAXPTE=0, any data can be written into FJPB, regardless of its value and the BMAXF bit remains 0. When PMAXPTE=1, writing a data greater than PWMMAXP into the FJPB will be failed, the BMAXF bit being 1 while writing a data equal to or lower than PWMMAXP can be successful, the BMAXF bit being 0.

Bit 5 **HMAXF**: FJPH write invalid flag
0: Correct wirte
1: Invalid write (FJPH > PWMMAXP)
Note: when PMAXPTE=0, any data can be written into FJPH, regardless of its value and the HMAXF bit remains 0. When PMAXPTE=1, writing a data greater than PWMMAXP into the FJPH will be failed, the HMAXF bit being 1 while writing a data equal to or lower than PWMMAXP can be successful, the HMAXF bit being 0.

Bit 4 **PMAXF**: PWMP write invalid flag
0: Correct wirte
1: Invalid write (PWMP > PWMMAXP)
Note: when PMAXPTE=0, any data can be written into PWMP, regardless of its value and the PMAXF bit remains 0. When PMAXPTE=1, writing a data greater than PWMMAXP into the PWMP will be failed, the PMAXF bit being 1 while writing a data equal to or lower than PWMMAXP can be successful, the PMAXF bit being 0.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **DT1MINF**: DT1 write invalid flag
0: Correct wirte
1: Invalid write (DT1 < DTMIN)
Note: 1. When using software to change DT1 or DTMIN content, the judgment will be triggered. If a data lower than DTMIN is being written into DT1, the DT1MINF flag will be set and the write operation failed. However a data greater than DT1 can be written into DTMIN.
2. When using hardware to change DT1 content, if a data lower than DTMIN is being written into DT1, the DT1MINF flag will be set and the write operation failed.

Bit 0 **DT0MINF**: DT0 write invalid flag
0: Correct wirte
1: Invalid write (DT0 < DTMIN)
Note: 1. When using software to change DT0 or DTMIN content, the judgment will be triggered. If a data lower than DTMIN is being written into DT0, the DT0MINF flag will be set and the write operation failed. However a data greater than DT0 can be written into DTMIN.
2. When using hardware to change DT0 content, if a data lower than DTMIN is being written into DT0, the DT0MINF flag will be set and the write operation failed.

• **FJM1C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	FJCNT2	FJCNT1	FJCNT0	—	FJSA2	FJSA1	FJSA0
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~4 **FJCNT2~FJCNT0**: Select PWM trigger number in frequency jittering approach mode
000: 1
001: 2
010: 3
011: 4
100: 5
101: 6
110: 7
111: 8

Bit 3 Unimplemented, read as "0"

Bit 2~0 **FJSA2~FJSA0**: Select approach value in frequency jittering approach mode
000: Period±2, Duty±1
001: Period±4, Duty±2
010: Period±6, Duty±3
011: Period±8, Duty±4
100: Period±10, Duty±5
101: Period±12, Duty±6
110: Period±14, Duty±7
111: Period±16, Duty±8

• **FJM1C1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	FJTIMES2	FJTIMES1	FJTIMES0	FJACNT1	FJACNT0	FJASA1	FJASA0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~4 **FJTIMES2~FJTIMES0**: Select adjustment number in frequency jittering approach mode
000: 1
001: 2

010: 3
011: 4
100: 5
101: 6
110: 7
111: 8

Note that one time of adjustment here includes changing the FJCNT[2:0], FJSA[2:0], FJDT1SA[2:0] and FJDT0SA[2:0] values.

Bit 3~2 **FJACNT1~FJACNT0**: Trigger number change during T2&T3 sections in frequency jittering approach mode
00: Unchanged
01: Unchanged
10: +1 in T2 section; -1 inT3 section
11: -1 in T2 section; +1 in T3 section

Note: 1. When FJCNT[2:0] is increased to the maximum value of 111 or reduced to the minimum value of 000, it will remain at the maximum or mimimum value, ignoring the FJACNT[1:0] settings.
2. When PWM has reached the approached target value, FJCNT[2:0] remains unchanged.

Bit 1~0 **FJASA1~FJASA0**: Approach value change during T2&T3 sections in frequency jittering approach mode
00: Unchanged
01: Unchanged
10: +1 in T2 section; -1 inT3 section
11: -1 in T2 section; +1 in T3 section

Note: 1. When FJSA[2:0] is increased to the maximum value of 111 or reduced to the minimum value of 000, it will remain at the maximum or mimimum value, ignoring the FJASA[1:0] settings.
2. When PWM has reached the approached target value, FJSA[2:0] remains unchanged.

• FJM1C2 Register

Bit	7	6	5	4	3	2	1	0
Name	FJDT0EN	FJDT0ASA1	FJDT0ASA0	FJDT0CNT1	FJDT0CNT0	FJDT0SA2	FJDT0SA1	FJDT0SA0
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 **FJDT0EN**: DT0 modification enable control in frequency jittering approach mode
0: Disable
1: Enable

Bit 6~5 **FJDT0ASA1~FJDT0ASA0**: DT0 approach value change during T2&T3 sections in frequency jittering approach mode
00: Unchanged
01: Unchanged
10: +1 in T2 section; -1 inT3 section
11: -1 in T2 section; +1 in T3 section

Note that when FJDT0SA[2:0] is increased to the maximum value of 111 or reduced to the minimum value of 000, it will remain at the maximum or mimimum value, ignoring the FJDT0ASA[1:0] settings.

Bit 4~3 **FJDT0CNT1~FJDT0CNT0**: Select the number of trigger times for modifying DT0 in frequency jittering approach mode
00: 4
01: 8
10: 12
11: 16

Bit 2~0 **FJDT0SA2~FJDT0SA0**: Select DT0 change value in frequency jittering approach mode
000: 1
001: 2
010: 3
011: 4
100: 5
101: 6
110: 7
111: 8

• **FJM1C3 Register**

Bit	7	6	5	4	3	2	1	0
Name	FJDT1EN	FJDT1ASA1	FJDT1ASA0	FJDT1CNT1	FJDT1CNT0	FJDT1SA2	FJDT1SA1	FJDT1SA0
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 **FJDT1EN**: DT1 modification enable control in frequency jittering approach mode
0: Disable
1: Enable

Bit 6~5 **FJDT1ASA~FJDT1ASA0**: DT1 approach value change during T2&T3 sections in frequency jittering approach mode
00: Unchanged
01: Unchanged
10: +1 in T2 section; -1 inT3 section
11: -1 in T2 section; +1 in T3 section
Note that when FJDT1SA[2:0] is increased to the maximum value of 111 or reduced to the minimum value of 000, it will remain at the maximum or mimimum value, ignoring the FJDT1ASA[1:0] settings.

Bit 4~3 **FJDT1CNT~FJDT1CNT0**: Select the number of trigger times for modifying DT1 in frequency jittering approach mode
00: 4
01: 8
10: 12
11: 16

Bit 2~0 **FJDT1SA2~FJDT1SA0**: Select DT1 change value in frequency jittering approach mode
000: 1
001: 2
010: 3
011: 4
100: 5
101: 6
110: 7
111: 8

• **FJM0C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	CFJCNT2	CFJCNT1	CFJCNT0	CFJS	CFJSA2	CFJSA1	CFJSA0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as “0”

Bit 6~4 **CFJCNT2~CFJCNT0**: Select the number of increament/decreament times in frequency jittering count mode
000: 1
001: 2
010: 3

011: 4
100: 5
101: 6
110: 7
111: 8

Bit 3 **CFJS**: Select the Period and Duty change direction in frequency jittering count mode
0: Increase
1: Decrease

Bit 2~0 **CFJSA2~CFJSA0**: Select Period and Duty change value in frequency jittering count mode
000: Period±2, Duty±1
001: Period±4, Duty±2
010: Period±6, Duty±3
011: Period±8, Duty±4
100: Period±10, Duty±5
101: Period±12, Duty±6
110: Period±14, Duty±7
111: Period±16, Duty±8

• **FJM0C1 Register**

Bit	7	6	5	4	3	2	1	0
Name	CFJDT0EN	—	CFJDT0CNT1	CFJDT0CNT0	CFJDT0S	—	CFJDT0SA1	CFJDT0SA0
R/W	R/W	—	R/W	R/W	R/W	—	R/W	R/W
POR	0	—	0	0	0	—	0	0

Bit 7 **CFJDT0EN**: DT0 modification enable control in frequency jittering count mode
0: Disable
1: Enable

Bit 6 Unimplemented, read as "0"

Bit 5~4 **CFJDT0CNT1~CFJDT0CNT0**: Select the number of trigger times for modifying DT0 in frequency jittering count mode
00: 1
01: 2
10: 3
11: 4

Bit 3 **CFJDT0S**: Select the DT0 change direction in frequency jittering count mode
0: Increase
1: Decrease

Bit 2 Unimplemented, read as "0"

Bit 1~0 **CFJDT0SA1~CFJDT0SA0**: Select DT0 change value in frequency jittering count mode
00: ±1
01: ±2
10: ±3
11: ±4

• **FJM0C2 Register**

Bit	7	6	5	4	3	2	1	0
Name	CFJDT1EN	—	CFJDT1CNT1	CFJDT1CNT0	CFJDT1S	—	CFJDT1SA1	CFJDT1SA0
R/W	R/W	—	R/W	R/W	R/W	—	R/W	R/W
POR	0	—	0	0	0	—	0	0

Bit 7 **CFJDT1EN**: DT1 modification enable control in frequency jittering count mode
0: Disable
1: Enable

Bit 6 Unimplemented, read as "0"

Bit 5~4 **CFJDT1CNT1~CFJDT1CNT0**: Select the number of trigger times for modifying DT1 in frequency jittering count mode
00: 1
01: 2
10: 3
11: 4

Bit 3 **CFJDT1S**: Select the DT1 change direction in frequency jittering count mode
0: Increase
1: Decrease

Bit 2 Unimplemented, read as "0"

Bit 1~0 **CFJDT1SA1~CFJDT1SA0**: Select DT1 change value in frequency jittering count mode
00: ±1
01: ±2
10: ±3
11: ±4

Functional Description

The high resolution PWM generator consists of one 12-bit counter circuit to generate the complementary PWM outputs. 12-bit programmable dead time insertion and output polarity control circuits are included for the PWM0 and PWM1 outputs. A protection mechanism is also provided to turn off the PWM generator immediately or adjust the PWM period when an error signal occurs.

Aimed at induction cooker applications, zero current switch signal outputs, cookware detection and phase detection and width measurement, phase protection can also be implemented by using the device. A frequency jitter function is also included in the device. It can be implemented in a software or hardware way that force the PWM frequency and duty to experience periodic changes. The following will give descriptions of the circuit functions and how to use them.

PWM Generator Circuit

A 12-bit PWM counter together with a complement output circuit can be used to generate complementary PWM outputs. As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. Both of the PWMP and PWMD registers are used to generate the PWM waveform, the 12-bit PWMP is used to control the PWM waveform frequency while the 12-bit PWMD is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the PWMP and PWMD registers.

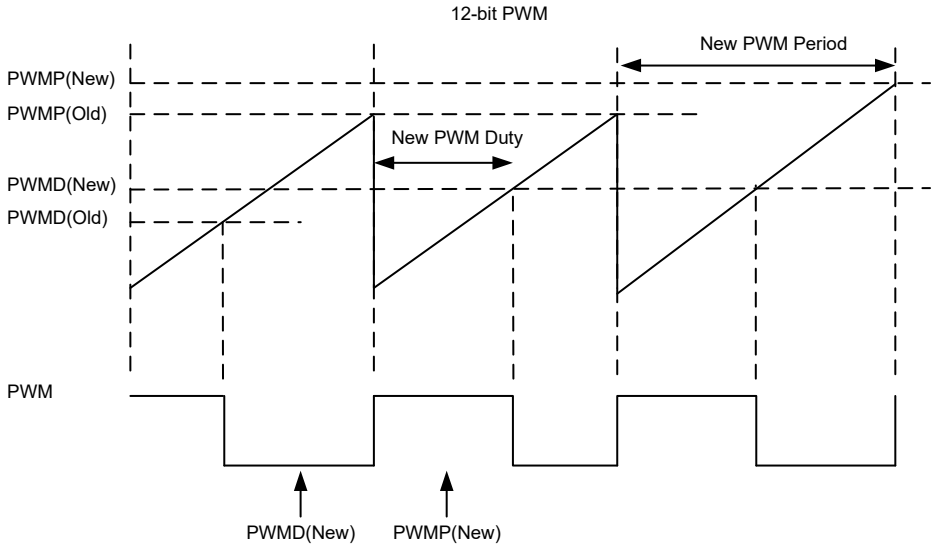
The following is an example of the period and duty register settings.

If the PWM counter clock $f_{PWMPSC}=f_{PWM}$=32MHz, PWMP=511, PWMD=255,

The PWM period=f_{PWMPSC}/(PWMP[11:0]+1)=32MHz/512=62.5kHz, PWM duty=(255+1)/512=50%

Note: In applications, the PWMD value should meet the condition: 1≤PWMD≤(PWMP-1)

An interrupt flag, one for each of the PWMD and PWMP, will be generated when either a PWMD or PWMP compare match occurs.



When writing new data into the PWMP, PWMD, DT0 and DT1 buffer registers using software, these register data will be updated by hardware when the PWMR counter value equals to zero.

Dead Time Insertion Function

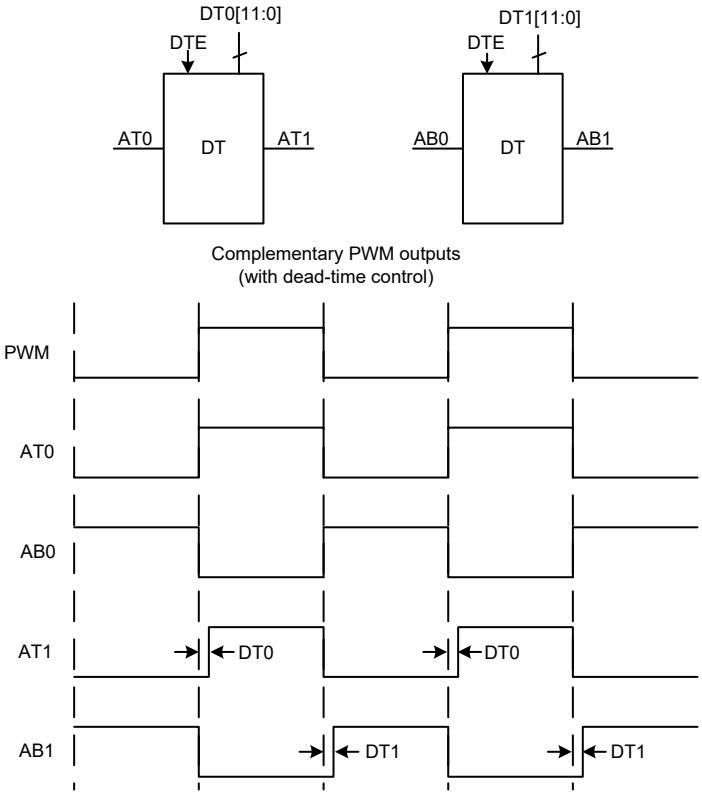
During the transition of the external driving transistors, there may be a short period when both the high and low sides are simultaneously on which will result in a momentary short-circuit situation. To avoid this situation, a 12-bit dead time insertion function is integrated in the device for PWM0 and PWM1 output. The dead time ranges from 0μs~128μs, definded in the DT0 or DT1 register pairs using software configuration.

When the dead time insertion function is enabled, the PWM0~PWM1 output actions are as follows:

- Each PWM0 rising edge is delayed with dead time 0 and then output.
- Each PWM1 rising edge is delayed with dead time 1 and then output.

Here, dead time 0 = (DT0[11:0]+1)/f_{PWM}; dead time 1 = (DT1[11:0]+1)/f_{PWM}

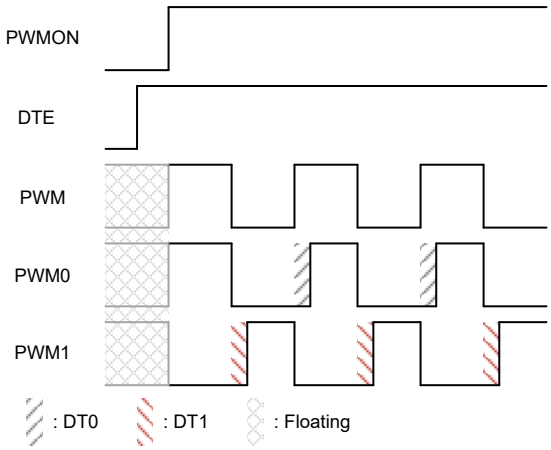
The dead time insertion function can be used in half-bridge induction cooker control applications and enabled by setting the DTE bit in PWMC0 register.



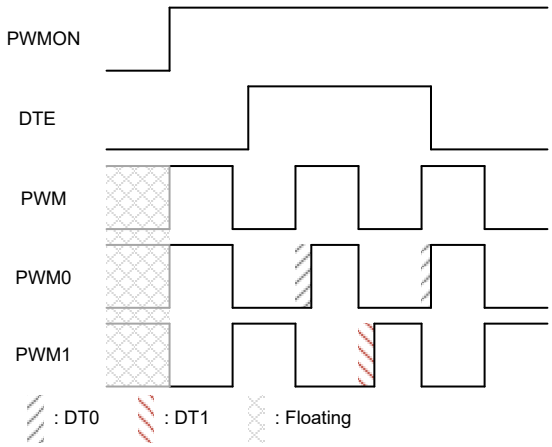
Dead-time Insertion Performed for Complementary Outputs

DTE setting notes:

- When DTE=1, the PWMON bit is set to 1, then the first rising edge of the PWM0 will not be delayed with the dead time but the first rising edge of the PWM1 will be.

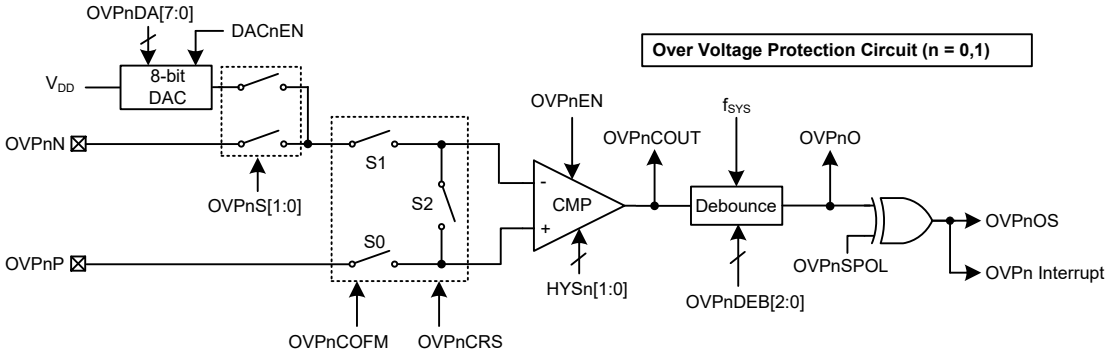


- When PWMON=1, the DTE bit is set from 0 to 1, the dead time circuit will be started immediately, and the dead time will be inserted at the next rising edge of the PWM0 or PWM1. When the DTE bit changes from 1 to 0, the dead time circuit will be turned off immediately even if the counting for DT0 or DT1 is not finished.



Resonant Current to Digital Signal

If two signals of the resonant current are connected to the OVP0 and OVP1 positive and negative inputs through a current sensor respectively, then the OVP0 and OVP1 will output a current digital signal, which are named OVP0OS and OVP1OS signals respectively. The OVP0 and OVP1 negative inputs also can be a 0V voltage from the internal 8-bit D/A converter. The OVP0OS and OVP1OS signals, which will be processed by passing the debounce circuit with a programmable debounce time, can be used to implement cookware detection.

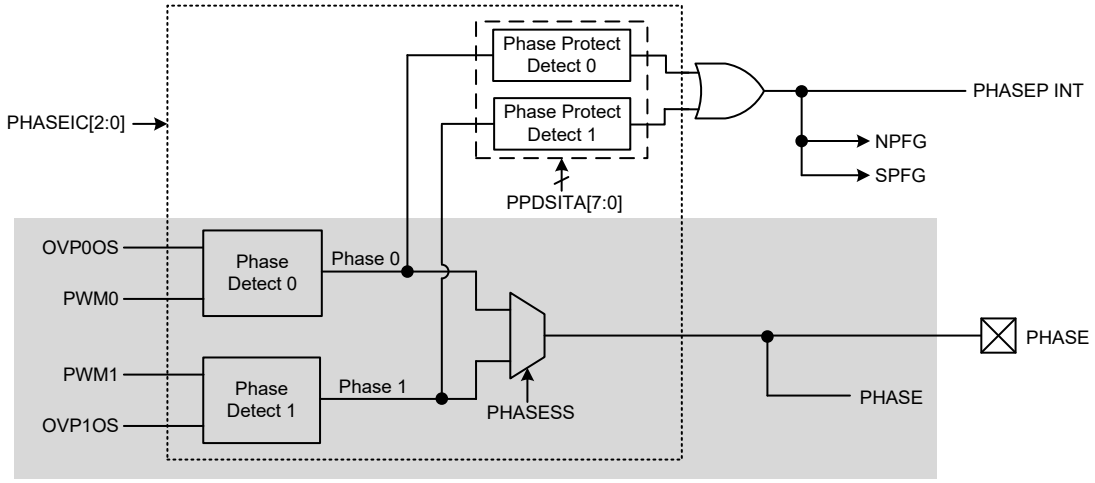


OVP0/OVP1 Block Diagram

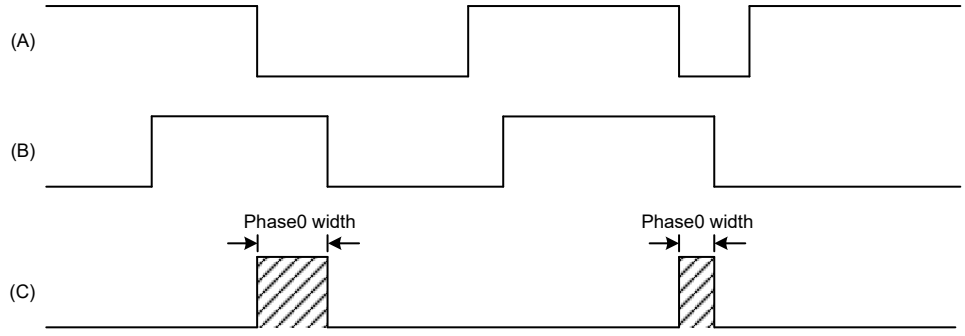
Note: Refer to the "Over Voltage Protection" section for details.

Phase Detect Circuits and Protection Mechanism

Two groups of phase detect circuits, Phase Detect 0 and Phase Detect 1 are provided. The Phase Detect 0 is to compare the PWM0 and OVP0OS signal phases and will generate an output signal, labled as Phase0, while the Phase Detect 1 is to compare the PWM1 and OVP1OS signal phases and will generate an output signal, labled as Phase1. The phase detect circuit structure is shown in gray shadow in the accompanying figure.



The figure below takes the Phase Detect 0 as an example to illustrate the circuit function. If the OVP0OS signal (current signal) phase lags behind the PWM signal (voltage signal), a Phase 0 output will be generated from the Phase detect 0 circuit and the Phase 0 width can be used to calculate the phase difference between the current and voltage signals, thus adjusting the induction cooker output power. If there is no phase difference between the OVP0OS signal (current signal) and the PWM signal (voltage signal), the Phase 0 signal will remain at a constant zero level. In this case, the induction cooker provides the maximum output power. If the voltage signal phase lags behind the current signal, the phase 0 output width cannot be detected and then the protection circuit will be triggered. The same operating principle applies to Phase Detect 1 circuit.



(A): PWM0 signal

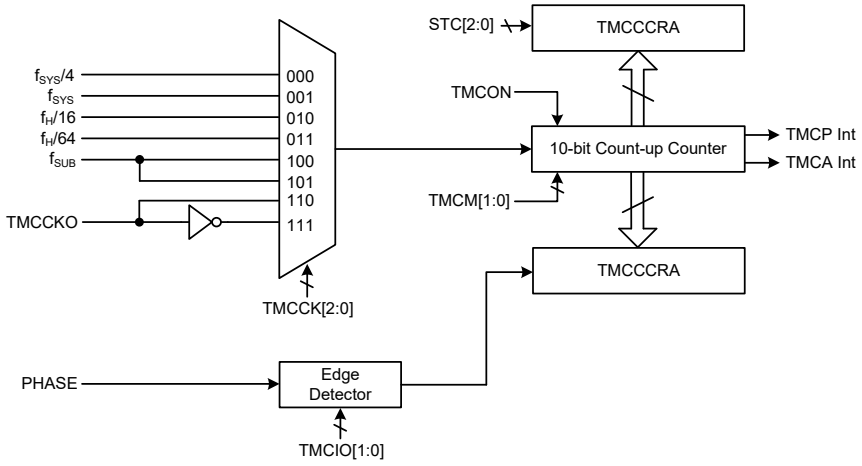
(B): OVP0OS signal

(C): Phase0 signal

Phase Detect 0 Circuit Timing Diagram

10-bit TMC Counter

The TMC is a 10-bit count-up counter. The clock source which drives the counter can be a ratio of the system clock f_{SYS} or the internal high speed clock f_H, the f_{SUB} low speed clock or the internal TMCCKO signal. The TMC can operate in one of two operating modes which are the Capture Input Mode and Timer/Counter Mode. The only way of changing the value of the 10-bit counter and the TMCCCRA registers using the application program, is to clear the counter by changing the TMCON bit from low to high.



10-bit TMC Block Diagram

Timer/Counter Mode

To select this mode, bits TMCM1 and TMCM0 in the TMCC1 register should be set to "10" or "11". The clock source can be selected using the TMCCK2~TMCCK0 bits in the TMCC0 register. In the Timer/Counter Mode, the counter counts up continuously from 0 to 1023. The counter will overflow when its reaches its maximum value 1023 at which point the TMCP interrupt flag will be set. When the time which is setup by the STC[2:0] bits has elapsed, the 10-bit counter present value will be latched into the TMCCCRA register and the TMCA interrupt flag will be set and the TMCON bit will be cleared to zero automatically.

The TMCCKO allows an internal signal to be a clock source to drive the TMC or for event counting. Select the TMCCKO rising or falling signal as the TMC clock and configure the waiting time using the STC[2:0] bits. Then set the TMCON bit high to enable the timer. When the time has elapsed, the TMCON bit will be cleared. The counter value will be latched into the TMCCCRA register and the TMCA interrupt flag will be generated.

Capture Input Mode

To select this mode, bits TMCM1 and TMCM0 in the TMCC1 register should be set to "00" or "01". This mode enables internal signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. In the Capture Input Mode, the input signal (PHASE signal) active edge can be either a rising edge, a falling edge or both rising and falling edges.

The PHASE signal can be the capture input signal; the active edge transition type is selected using the TMCIO1 and TMCIO0 bits in the TMCC1 register. The counter is started when the TMCON bit changes from low to high which is initiated using the application program. When a required edge transition appears on the PHASE signal the present value of the 10-bit counter will be latched into the TMCCCRA register and the TMCA interrupt flag is set. Irrespective of what events occur on the PHASE signal, the counter will continue to free run until the TMCON bit changes from high to low.

Cookware Detection and Phase Width Measurement Circuit

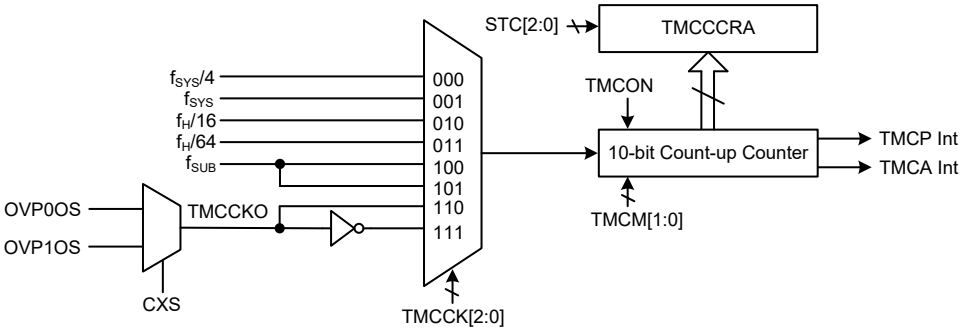
The OVP0OS and OVP1OS signals can be used to detect whether a cookware is present upon the heating coil or not. The OVP0OS or OVP1OS which is selected as the TMCCKO signal can be connected to the 10-bit TMC for counting.The counter value then can be used to dertermine the presence or absence of cookware. If the OVP0OS/OVP1OS (current signal) phase lags behind the corresponding PWM signal (voltage signal), the phase width of Phase 0/Phase 1 can be used for

cookware detection and phase width measurement by connecting the phase signal to the 10-bit TMC as its capture input source. Then the phase difference between the current and voltage signals can be worked out and then used to adjust the output power of the induction cooker.

Cookware Detection Circuit

The OVP0OS or OVP1OS signal can be connected to the 10-bit TMC TMCCKO input. Based on the counted pulse number, the presence or absence of cookware on the heating coil can be determined. The following summarises the steps that should be executed in order to implement cookware detection.

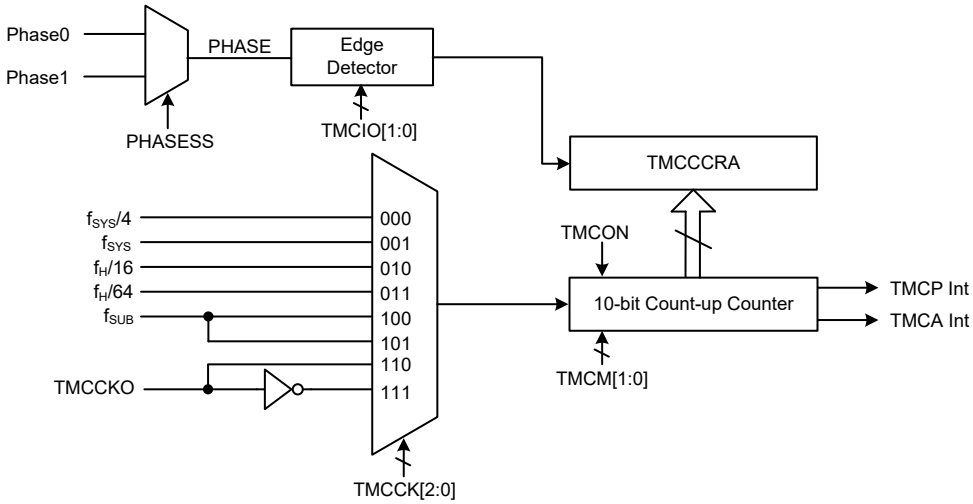
1. Select the internal TMCCKO signal to be used as the TMC event counter signal source. Here the TMCCKO signal can be sourced from the OVP0OS or OVP1OS which is selected using the CXS bit in the PWMC1 register.
2. Select the waiting time using the STC2~STC0 bits in the TMCC0 register.
3. Set the TMCON bit high to enable the counter.
4. When the waiting time has elapsed, the TMCON bit will be cleared to zero automatically. The present counter value will be latched into the TMCCCRA register and the TMCA interrupt flag will be generated.
5. According to the counter value, implement the cookware detection judgment.



Phase Width Measurement Circuit

When the OVP0OS/OVP1OS signal (current signal) lags behind the corresponding PWM (voltage signal), then a phase difference signal will be detected and can be used to calculate the phase width between the current and the voltage signals. The 10-bit TMC can be configured in Capture Input Mode for phase width measurement. The larger the phase width is, the lower the power of the half-bridge induction cooker outputs. Similarly, the smaller the phase width is, the higher the output power is. When the phase width is zero, the output power will be at its maximum and the frequency will be at the resonant frequency.

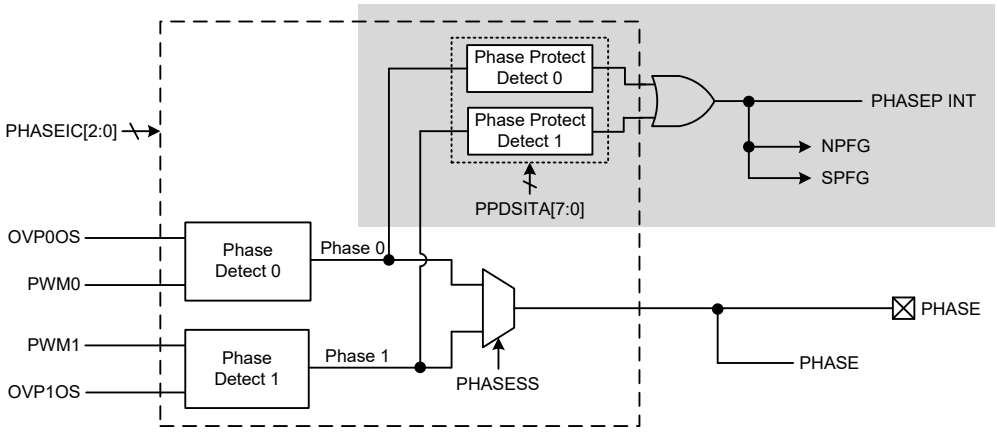
Generally, the induction cooker operates at a security frequency which is slightly higher than the resonant frequency. So a required maximum frequency value can be stored into the PWMMAXP register using the F/W to limit the output power of the induction cooker. If setting the PMAXPTE bit high to enable the PWMP maximum value limit function, then the value to be written into the PWMP register should be equal to or lower than the PWMMAXP register value. Otherwise, the data cannot be written into the PWMP register, which will also result in the PMAXF bit being set high by the hardware. In this way, the half-bridge output power will not exceed the maximum value.



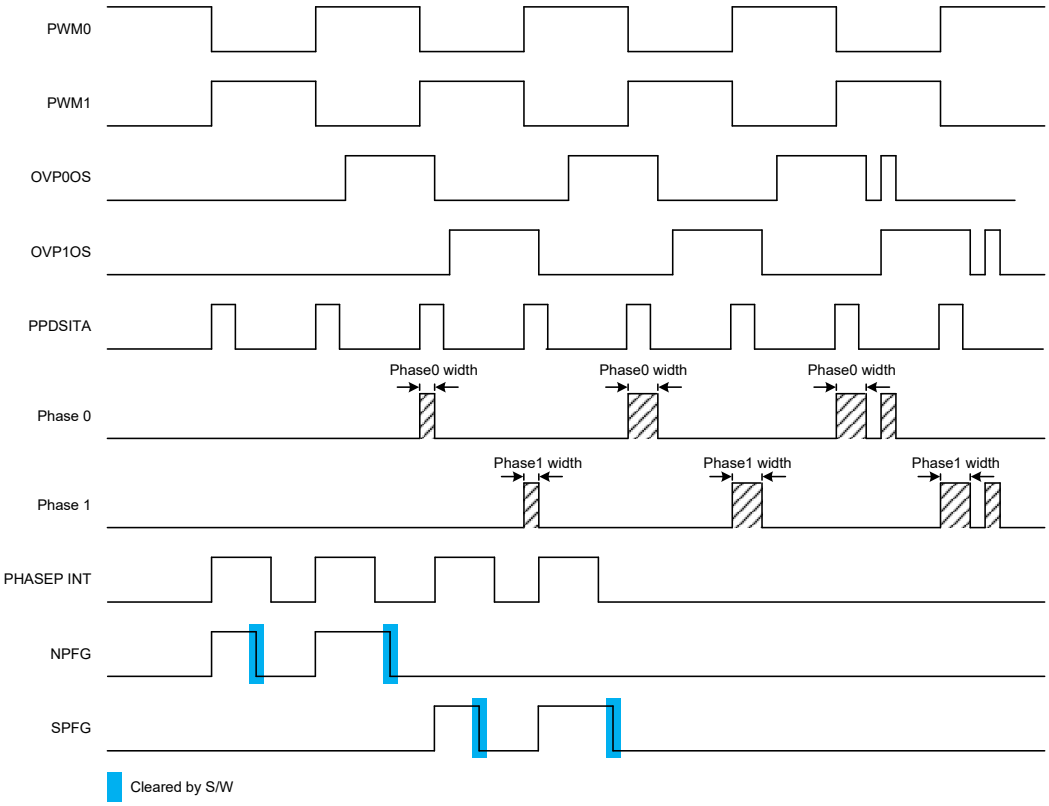
Note: Phase 0 and Phase 1 signals in the figure above are respectively the Phase Detect 0 and Phase Detect 1 circuit outputs.

Phase Protect Detect Circuit

The phase protect detection 0 and phase protect detection 1 are provided to implement Phase 0 and Phase 1 signal width judgement. If the Phase 0 or Phase 1 width is smaller than the setup value in PPDSITA register, the SPFG flag will be set high by the hardware. If the phase width is 0, the NPFG flag will be set by the hardware. Either condition can trigger the PHASEP interrupt, which means that phase protection will be activated. If the ERSGC0 bit in the ERSGC register is set to 1, then the corresponding ERSG protection circuit will be activated. Note that the SPFG and NPFG bit can only be set by the hardware and should be cleared by the software and cannot be set by the software. The phase protect detect circuit structure is shown in gray shadow in the accompanying figure.



Phase Protect Detect Circuit Diagram



Phase Protect Detect Circuit Timing Diagram

Phase Protect Detect 0 Action Description

During the detection period, if no phase width or a small phase width is detected, then protection operations will be activated.

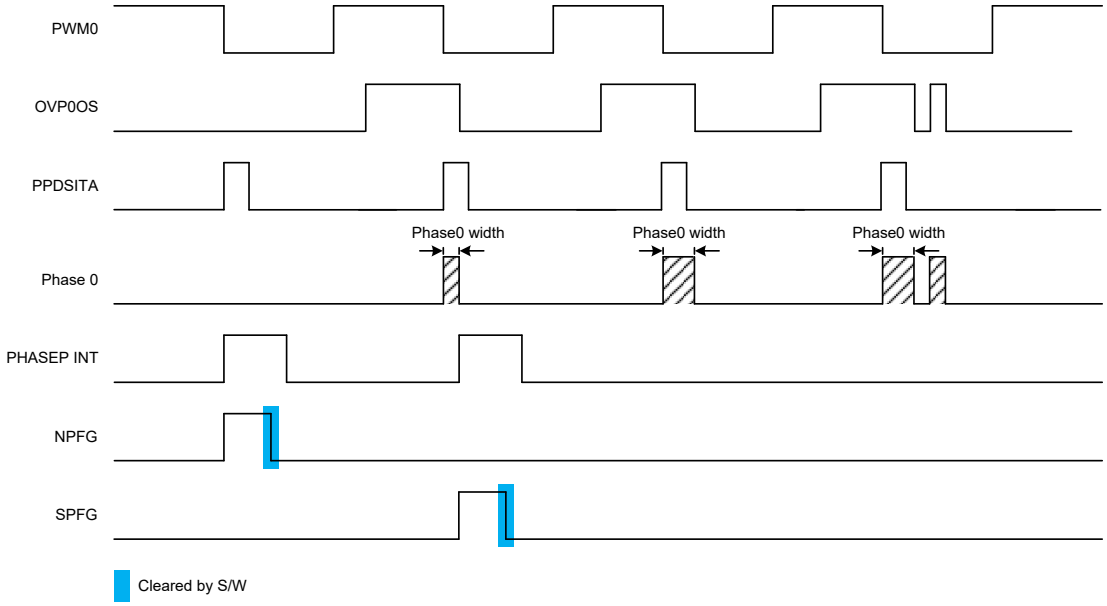
- When no phase is detected during the detection period:

  When using the PWM0 as the trigger signal, if the OVP0OS signal is 0, the PHASEP INT signal will be set high and the NPFG flag will be set to 1. If the OVP0OS signal is 1, the PHASEP INT signal and the NPFG flag will not be set.

- When a small phase is detected during the detection period:

  Only the first phase width duing a detection period will be measured. If the phase width is less than the value definded in PPDSITA register, the PHASEP INT signal will be set high and the SPFG flag will be set to 1. Otherwise the PHASEP INT signal and the SPFG flag remain low.

Note: 1. The phase protect detect circuit only detects the first phase signal during each detection period and other phase signals in the remaining time will be ignored.
2. After the SPFG and NPFG flag bits are set to 1 by the hardware, they should be cleared by the software, otherwise they will remain at 1.



Phase Protect Detection 0 Timing Diagram

Phase Protect Detect 1 Action Description

During the detection period, if no phase width or a small phase width is detected, then protection operations will be activated.

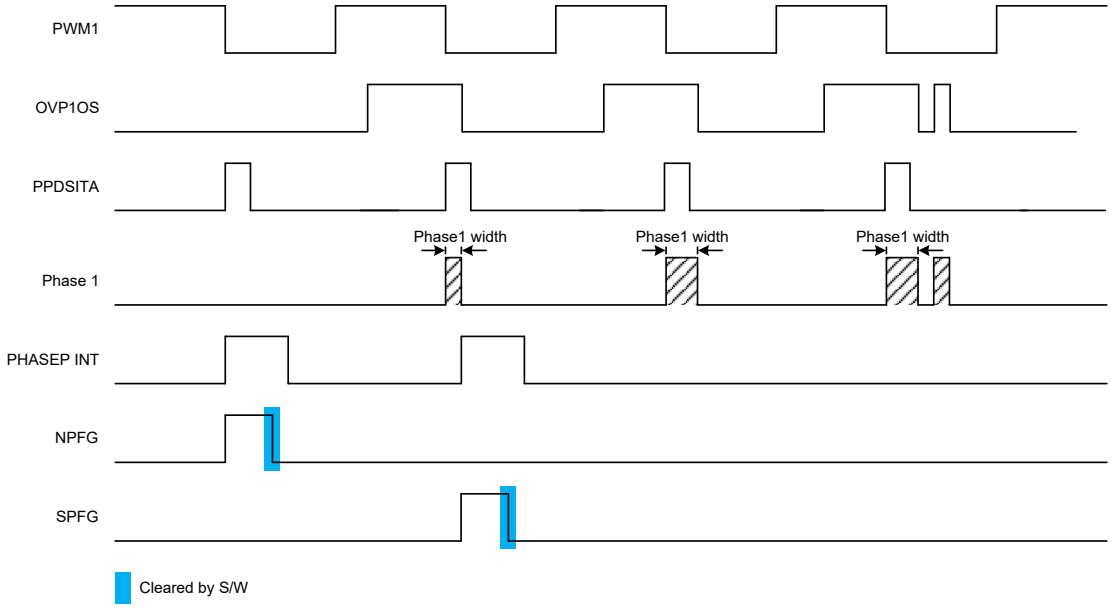
- When no phase is detected during the detection period:

  When using the PWM1 as the trigger signal, if the OVP1OS signal is 0, the PHASEP INT signal will be set high and the NPFG flag will be set to 1. If the OVP1OS signal is 1, the PHASEP INT signal and the NPFG flag will not be set.

- When a small phase is detected during the detection period:

  Only the first phase width duing a detection period will be measured. If the phase width is less than the value definded in PPDSITA registers, the PHASEP INT signal will be set high and the SPFG flag will be set to 1. Otherwise the PHASEP INT signal and the SPFG flag remain low.

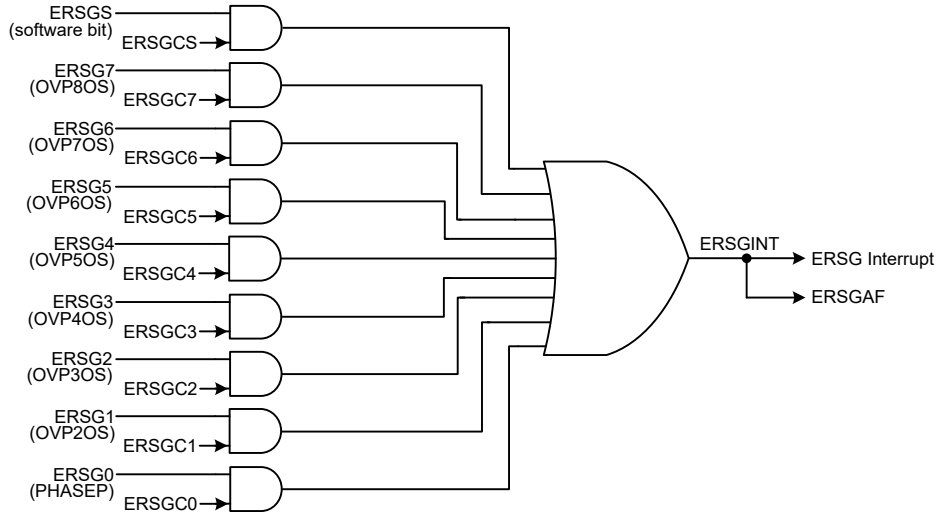
Note: 1. The phase protect detection circuit only detects the first phase signal during each detection period and other phase signals in the remaining time will be ignored.
2. After the SPFG and NPFG flag bits are set to 1 by hardware, they should be cleared by software, otherwise they will remain at 1.



Phase Protect Detection 1 Timing Diagram

ERSG Protection Circuit

The ERSG protection circuit has totally nine signal source inputs to trigger to start the protection action. These signals include ERSG0 (the phase protect detect circuit output – PHASEP signal), ERSG1~ERSG7 (the OVP circuit output – OVP2OS~OVP8OS) and ERSGS (software trigger bit). When the signal is "High", the ERSG interrupt will be triggered if the corresponding ERSGC[7:0] bit in the ERSGC register or ERSGCS bit in the PWMPC2 register is set.



ERSG Interrupt Structure

For each error signal, the ERSG protection circuit provides four protect actions which can be selected using PWMPCn[m+1:m] bits in PWMPC0~PWMPC2 registers. The protect action is implemented using two circuits that one is PWM shutdown circuit and the other is PWM auto-adjust circuit. However if more than one ERSG trigger signal are input to the ERSG protection circuit but their corresponding protect action configruations are different, the priority levels are determined by the corresponding PWMPCn[m+1:m] bit value, 00>01>10>11.

The following gives an example to introduce how the ERSG protection circuit functions, where the ERSG2 PWM protection control bits PWMPC0[5:4] are set to 11 and the ERSG4 PWM protection control bits PWMPC1[1:0] are set to 00. If an ERSG2 active trigger signal occurs first, the ERSG protection circuit will take actions according to the "11" mode. Subsequently if an ERSG4 active trigger signal occurs, the ERSG protection circuit will switch to the "00" mode immediately, as the "00" mode has priority over the "11" mode.

ERSG Protect Action Descriptions

For each ERSG trigger signal, the ERSG protection circuit provides four responding actions which are selected by the PWMPCn[m+1:m] bits, where n=0~2 and m=0, 2, 4, 6.

For PWMPCn[m+1:m]=00B: The PWMP auto-adjust will not be implemented. The hardware will immediately turn off the PWM circuit automatically and set the MSS0 and MSS1 bits to 1, forcing the PWM0 and PWM1 to output the corresponding MSD0 or MSD1 logic level, and clear the PWMON bit and set the FPWMCLF bit to 1. A complete PWM shutdown circuit is executed.

For PWMPCn[m+1:m]=01B: The PWMP auto-adjust will not be implemented. After the current PWM period output is finished, the PWM shutdown circuit will be executed. The hardware will turn off the PWM circuit automatically and set the MSS0 and MSS1 bits to 1, forcing the PWM0 and PWM1 to output the corresponding MSD0 or MSD1 logic level, and clear the PWMON bit and set the FPWMCLF bit to 1.

For PWMPCn[m+1:m]=10B: The PWMP will be adjusted once in each PWM period with ATMPF bit being set high by the hardware. If (PWMP-DECPWMP)≥PWMMINP, the value of (PWMP-DECPWMP) will be written into the PWMP register, thus implementing PWMP automatical adjustment. Repeat the operation following this rule in each PWM period until (PWMP-DECPWMP)<PWMMINP, at this time the value of (PWMP-DECPWMP) will not be written into the PWMP registers and the PWM auto-adjust operation is finished. Note that if the ATMPC bit is set high during the auto-adjust stage, the PWM auto-adjust operation will be stopped. When the PWM auto-adjust operation is finished and the PWM output cycle corresponding to the last adjustment is completed, the PWM shutdown circuit will be executed. The hardware will turn off the PWM circuit automatically and set the MSS0 and MSS1 bits to 1, forcing the PWM0 and PWM1 to output the corresponding MSD0 or MSD1 logic level, and clear the PWMON bit and set the FPWMCLF bit to 1.

For PWMPCn[m+1:m]=11B and ATMNPC=0: The PWMP will be adjusted once in each PWM period with ATMPF bit being set high by the hardware. If (PWMP-DECPWMP)≥PWMMINP, the value of (PWMP-DECPWMP) will be written into the PWMP register, thus implementing PWMP automatical adjustment. Repeat the operation following this rule in each PWM period until (PWMP-DECPWMP)<PWMMINP, at this time the value of (PWMP-DECPWMP) will not be written into the PWMP registers and the PWM auto-adjust operation is finished. Note that if the ATMPC bit is set high during the auto-adjust stage, the PWM auto-adjust operation will be stopped. After the PWM auto-adjust operation is finished, the PWMON and FPWMCLF bits remain unchanged.

For PWMPCn[m+1:m]=11B and ATMNPC=1 (for ERSG1 and ERSG3 signals only): The PWMP will be adjusted once in each PWM period with ATMPF bit being set high by the hardware. If (PWMP-DECPWMP)≥PWMMINP, the value of (PWMP-DECPWMP) will be written into the PWMP register, thus implementing PWMP automatical adjustment. Repeat the operation following this rule in each PWM period until (PWMP-DECPWMP)<PWMMINP, at this time the value of (PWMP-DECPWMP) will not be written into the PWMP registers and the PWM auto-adjust operation is finished. During the auto-adjust procedure, if the ERSG signal that triggers the protect action changes to 0, or the ATMPC bit is set high halfway, the PWM auto-adjust operation will also be stopped. After the PWM auto-adjust operation is finished, the PWMON and FPWMCLF bits remain unchanged.

Note: 1. Before the ERSG protection circuit is in operation, if changing the PWMPCn[m+1:m] bit value, the new setting will take effect immediately. However once the ERSG protection circuit starts operating, if changing the PWMPCn[m+1:m] bit value, the new setting cannot take effect until the current whole ERSG protection action is finished. The new setup action will be executed at the next ERSG trgger signal.

2. When PWMPCn[m+1:m] is 00 or 01, an ERSG interrupt trigger singal will not enable PWM auto-adjust circuit. When PWMPCn[m+1:m] is 10 or 11, an ERSG interrupt trigger singal will enable the PWM auto-adjust circuit. The PWMP and PWMD values will be adjusted once during each PWM period. If (PWMP-DECPWMP)≥PWMMINP, the hardware will store the (PWMP-DECPWMP) value into the PWMP registers. The (PWMD-(DECPWMP/2) value will be stored into the PWMD registers. The hardware repeats the operation following this rule in each PWM period until (PWMP-DECPWMP)<PWMMINP, at this time the value of (PWMP-DECPWMP) will not be written into the PWMP registers and the PWM auto-adjust operation is finished. Refer to the accompanying table for an example which shows the related bit settings and PWM adjustment procedure.

PWM Period	ERSG INT	PWMPCn[m+1:m]	PWMMINP	DECPWMP	ATMPF	PWMP	PWMD
1	0	x	400	30	0	499	249
2	0	x	400	30	0	499	249
3	1	00 or 01	400	30	0	499	249
4	1	10 or 11	400	30	1	469	234
5	1	10 or 11	400	30	1	439	219
6	1	10 or 11	400	30	1	409	204
7	1	10 or 11	400	30	1	409	204

"x": Don't care

An example with ERSG1 signal=1, ATMNPC=1 and PWMPC0[3:2]=1.

PWM Period	ERSG1 INT	PWMMINP	DECPWMP	ATMPF	PWMP	PWMD
1	0	400	30	0	499	249
2	0	400	30	0	499	249
3	1	400	30	1	469	234
4	1	400	30	1	439	219
5	0	400	30	1	439	219
6	0	400	30	1	439	219
7	0	400	30	1	439	219

Note: 1. If DECPWMP/2 is not integral, discard the decimal.

2. During the PWMP auto-adjust procedure, the hardware will restrict the software from changing the PWMP, PWMMINP, DECPWMP and PWMD register values.

Frequency Jitter Function

The frequency jitter function can be implemented in five operating modes, including S/W-Count Mode, S/W-Approach Mode, H/W-Count&Approach Mode, H/W-Full Approach Mode, and H/W-Partial Approach Mode. The following table shows how to configure related control bits to select the required mode and whether the FJTIMER counter will be used or not in each mode.

Mode	Name	Related Bit Settings			FJTIMER
		FJWMD	FJMDS	FJHMDS	
1	S/W-Count Mode	0	0	x	No
2	S/W-Approach Mode	0	1	x	No
3	H/W-Count&Approach Mode	1	0	0	Yes
4	H/W-Full Approach Mode	1	1	0	Yes
5	H/W-Partial Approach Mode	1	x	1	Yes

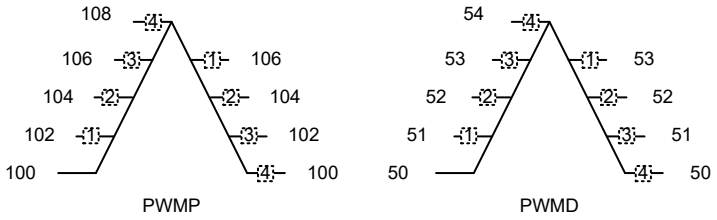
"x": Don't care

As can be seen in the table above, the frequency jittering operation can be executed in two modes, count mode and approach mode. The following will introduce the differences between them.

Count Mode

In the count mode, how to change the PWMP, PWMD, DT0 and DT1 is controlled by FJM0C0~FJM0C2 registers. The FJM0C0 register is used to control the way of changing PWMP and PWMD. The CFJCNT[2:0] bits determine the number of stepping times, the CFJS bit determines whether the PWMP/PWMD value will be added or subtracted from the value set by the CFJSA[2:0] bits. Here it should be noted that when the number defined by the CFJCNT[2:0] bits has reached, the PWMP and PWMD will decrease or increase in the opposite direction.

For example, if CFJCNT[2:0]=011(four times), CFJS=0 & CFJSA[2:0]=000(Period+2/Duty+1), PWMP=100, PWMD=50，the PWMD and PWMP changes as below:



How DT0 and DT1 change is controlled by the FJM0C1 and FJM0C2 register respectively. The CFJDTnEN bit is used to decide whether the hardware will automatically adjust the DTn. The CFJDTnCNT[1:0] bits define how many PWM periods to trigger before adjusting DTn once. The CFJDTnS bit determines whether the DTn value will be added or subtracted from the value set by the CFJDTnSA[1:0] bits. The following provides an example.

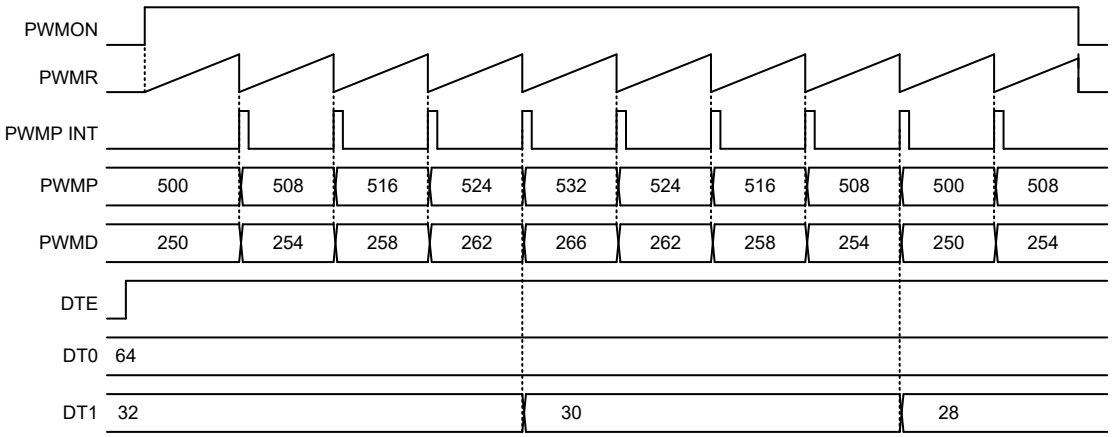
- The related register setup

Register	Data	Description
PWMP	500	
PWMD	250	
DT0	64	
DT1	32	
FJM0C0	33H	Stepping times=4; Period +8; Duty +4
FJM0C1	13H	Disable; DT0 +4 for every 2 triggers
FJM0C2	B9H	Enable; DT1 -2 for every 4 triggers

- The result

Number	Registers				Internal Counters			
	PWMP	PWMD	DT0	DT1	PWMP	PWMD	DT0	DT1
0	500	250	64	32	500	250	64	32
1	508	254	64	32	500	250	64	32
2	516	258	64	32	508	254	64	32
3	524	262	64	32	516	258	64	32
4	532	266	64	30	524	262	64	32
1	524	262	64	30	532	266	64	30
2	516	258	64	30	524	262	64	30
3	508	254	64	30	516	258	64	30
4	500	250	64	28	508	254	64	30
1	508	254	64	28	500	250	64	28

• The waveform



Approach Mode

In the approach mode, how to change the PWMP, PWMD, DT0 and DT1 is controlled by FJM1C0~FJM1C3 registers. The approaching target values of PWMP and PWMD are determined by FJPA and FJPB registers. The way of changing PWMP and PWMD is mainly controlled by FJM1C0 register. The FJCNT[2:0] bits determine the number of trigger times. The FJSA[2:0] bits define the amount that the PWMP and PWMD value will plus or minus. Furthermore the FJCNT[2:0] and FJSA[2:0] values also can be controlled by the FJM1C1 register to ±1 or remain unchanged after certain times of adjustments during the the frequency jittering process.

The approaching target values of DT0 and DT1 are determined by corresponding FJDTnA or FJDTnB registers. The ways of changing the DT0 and DT1 are controlled respectively by the FJM1C2 and FJM1C3 registers. The FJDTnEN bit is used to control whether the DTn will be modified by the hardware during frequency jittering approach mode. The FJDTnCNT[1:0] bits define how many PWM periods to trigger before adjusting DTn once. The change amount is defined by the FJDTnSA[2:0] bits and can also be controlled by the FJDTnASA[1:0] bits to ±1 or remain unchanged during the hardware adjusting process. The following provides an example to show how to approach the FJPA/FJDT0A/FJDT1A.

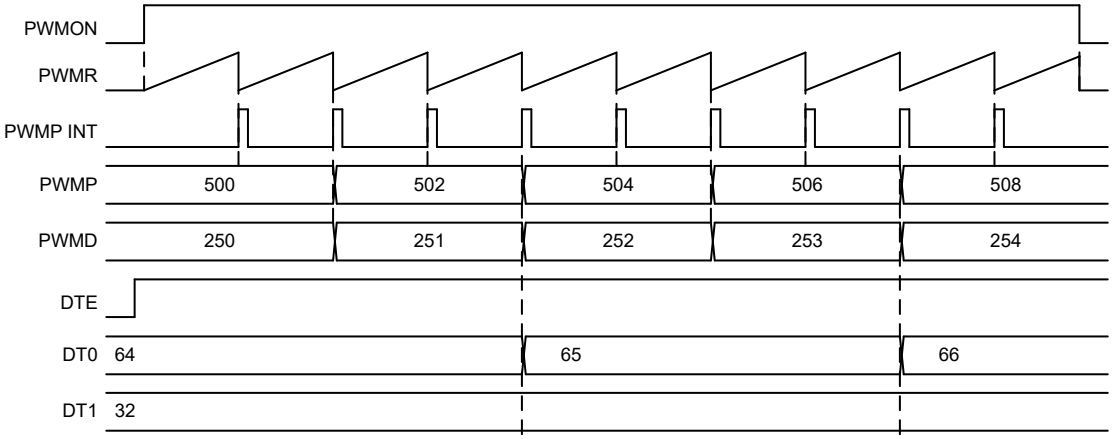
• The related register setup

Register	Data	Description
PWMP	500	
PWMD	250	
DT0	64	
DT1	32	
FJPA	510	
FJDT0A	68	
FJDT1A	30	
FJM1C0	10H	After 2 triggers, Period +2, Duty +1
FJM1C1	10H	After 2 approaches, trigger number unchanged, change value unchanged
FJM1C2	80H	DT0 modify enabled; DT0 ±1 for every 4 triggers
FJM1C3	0CH	DT1 modify disabled; DT1 ±5 for every 8 triggers

- The result

Number	Registers							Internal Counters			
	PWMP	PWMD	DT0	DT1	FJPA	FJDT0A	FJDT1A	PWMP	PWMD	DT0	DT1
0	500	250	64	32	510	68	30	500	250	64	32
1	500	250	64	32				500	250	64	32
2	502	251	64	32				500	250	64	32
3	502	251	64	32				502	251	64	32
4	504	252	65	32				502	251	64	32
5	504	252	65	32				504	252	65	32
6	506	253	65	32				504	252	65	32
7	506	253	65	32				506	253	65	32
8	508	254	66	32				506	253	65	32
9	508	254	66	32				508	254	66	32

- The waveform



The operating principles of the count mode and approach mode have been elementarily introduced above. The following will explain how to set up related registers in these five modes and the peripheral functions to be used together to implement frequency jittering. The hardware and software implementation methods will be introduced individually.

S/W Frequency Jittering

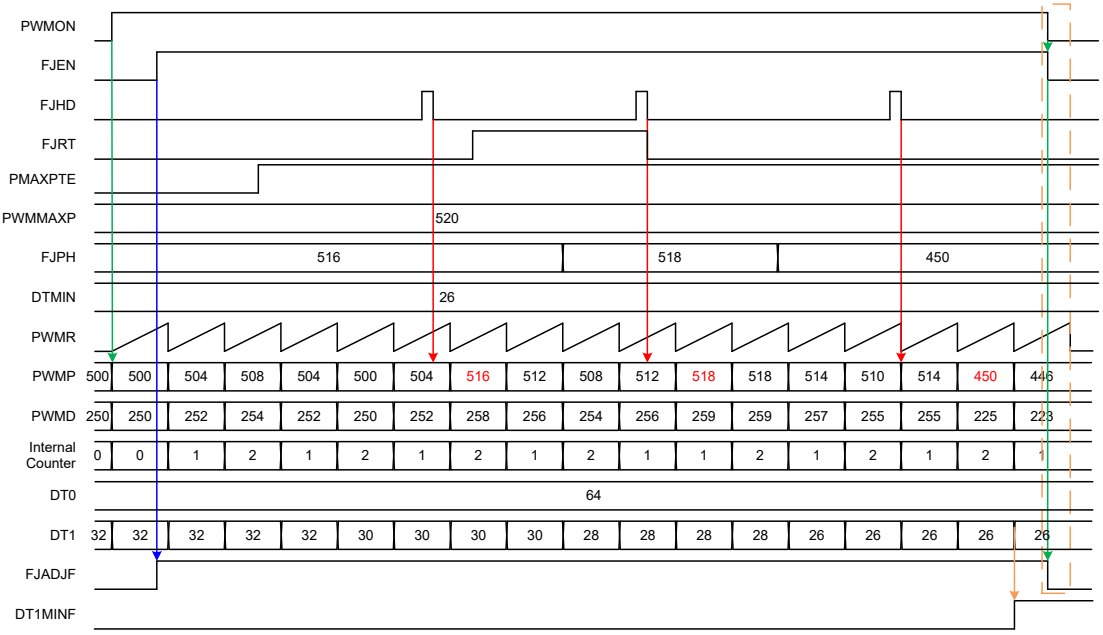
When using software frequency jitter function, after setting the FJEN bit to 1 by users, the PWM period will be adjusted according to the frequency jitter related registers settings. Note that when an ERSG interrupt occurs or ERSG auto-adjust function is enabled(ATMPF=1), if FJEN bit is 1, the hardware will clear the FJEN bit while if the FJEN bit is changed from 0 to 1 by the software, then the hardware will ignore this trigger and clear the FJEN bit.

Mode	Name	Setup Bits			Adjustment Control Register	
		FJWMD	FJMDS	FJSMDS	FJM0Cn(n=0~2)	FJM1Cn(n=0~3)
1	S/W-Count Mode	0	0	X	Y	X
2-1	S/W-Approach Mode(A)	0	1	0	X	Y
2-2	S/W-Approach Mode(B)	0	1	1	X	Y

- Mode 1: S/W - Count Mode

  The register setup:

Register	Data	Description
PWMP	500	
PWMD	250	
DT0	64	
DT1	32	
FJM0C0	11H	Stepping times = 2; Period +4; Duty +2
FJM0C1	13H	Disable; DT0 +4 for every 2 triggers
FJM0C2	B9H	Enable; DT1 -2 for every 4 triggers

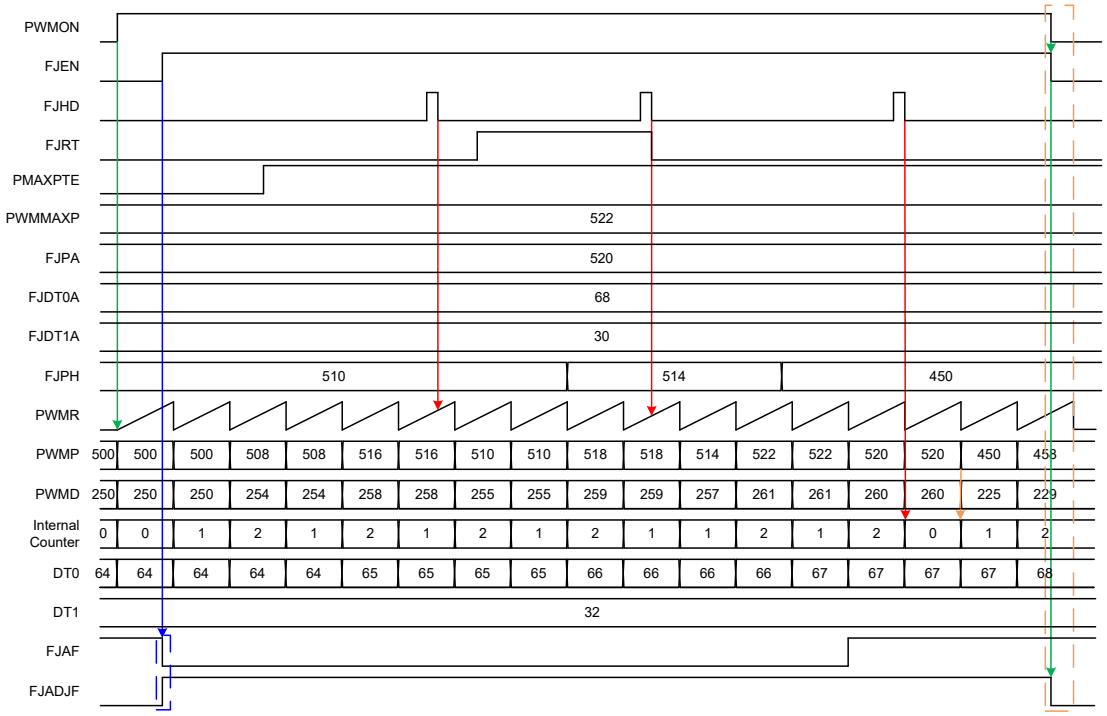


- Mode 2-1: S/W-Approach Mode(A)

  The following give two examples to show the register settings and the corresponding timing diagram in the S/W-Approach Mode(A).

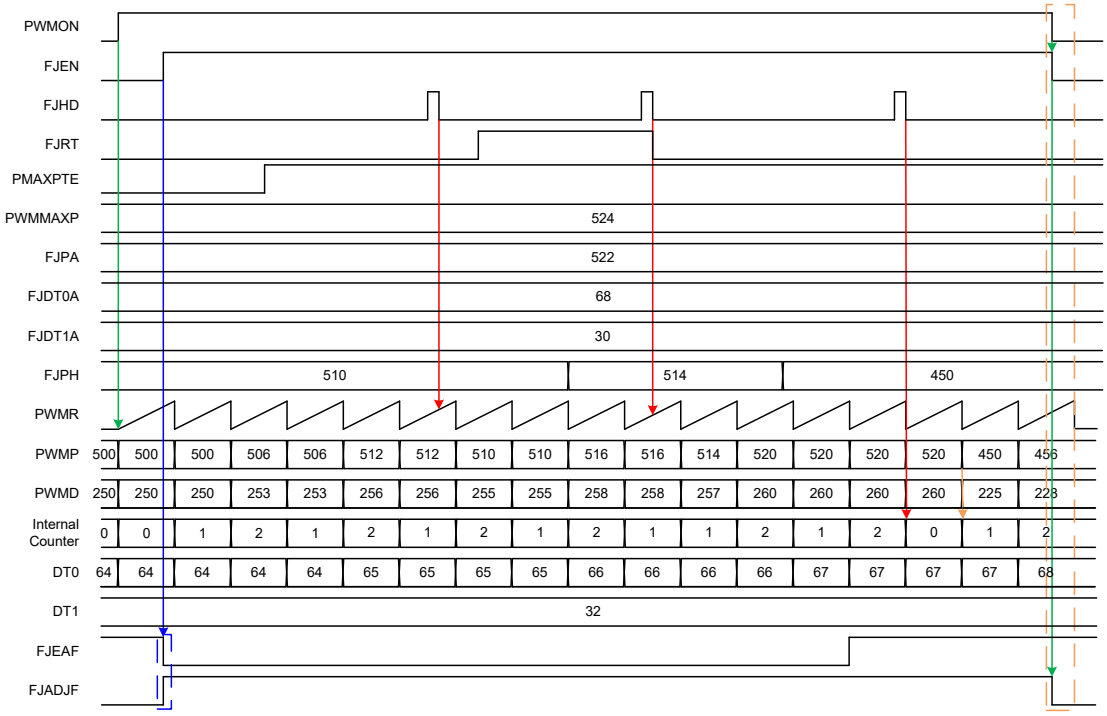
  - Example 1

Register	Data	Description
PWMP	500	
PWMD	250	
DT0	64	
DT1	32	
FJPA	520	
FJDT0A	68	
FJDT1A	30	
FJM1C0	15H	After 2 triggers, Period +8, Duty +4
FJM1C1	10H	After 2 approaches, trigger number unchanged, change value unchanged
FJM1C2	80H	Enable, DT0 ±1 for every 4 triggers
FJM1C3	0CH	Disable, DT1 ±5 for every 8 triggers



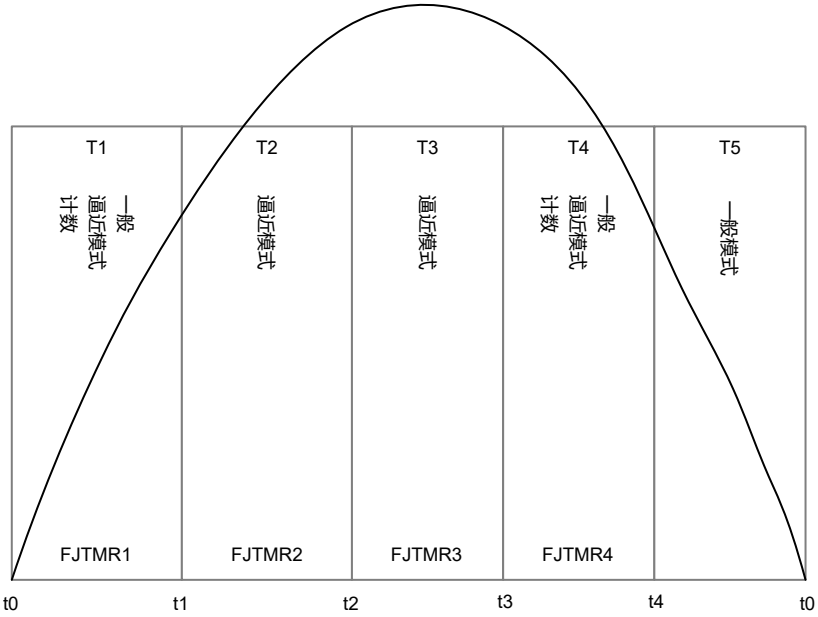
- Example 2

Register	Data	Description
PWMP	500	
PWMD	250	
DT0	64	
DT1	32	
FJPA	522	
FJDT0A	68	
FJDT1A	30	
FJM1C0	14H	After 2 triggers, Period +6, Duty +3
FJM1C1	10H	After 2 approaches, trigger number unchanged, change value unchanged
FJM1C2	80H	Enable, DT0 ±1 for every 4 triggers
FJM1C3	0CH	Disable, DT1 ±5 for every 8 triggers



H/W Frequency Jittering

In the hardware auto-adjust mode, the FJTIMER which providing four timing sections, T1~T4, will be used and the changes of PWM periods will be different in these four timing sections. The FJTIMER can start counting on an external signal rising edge or by the software setting FJTON bit to 1, which is selected by FJTSS bit. Here the mentioned external signal can be OVP7OS or FJTR1 signal and is selected using the FJOTSS bit. The PWM period change has different actions in the FJTIMER T1~T4 sections, as shown in the figure below. The first and the fourth timing intervals will be explained together while the second and the third timing intervals will be explained together.



- T1(t0~t1, FJTMR1)/T4(t3~t4, FJTMR4): There are three operating modes. The actions are described as follows.
  - Count Mode: Use FJM0C0~FJM0C2 to control
  - Approach Mode: Use FJM1C0~FJM1C3 to control. Note that FJM1C1 register and FJDTnASA[1:0] bits in FJM1C2/FJM1C3 registers are invalid in T1 and T4 timing intervals. It means that FJCNT[2:0], FJSA[2:0], FJDTnSA[2:0] bit value will not be changed. In T1 timing section the adjustment aim is FJPB/FJDT0B/FJDT1B, and in T4 the adjustment aim is FJPA/FJDT0A/FJDT1A.
  - Unchange Mode (General Mode): PWMP and PWMD are not controlled by FJM0C0~FJM0C2 or FJM1C0~FJM1C3 register settings, being operating without automatical adjustments and can be modified using the software.
- T2(t1~t2, FJTMR2)/T3(t2~t3, FJTMR3): There is only one operating mode which is the approach mode, using FJM1C0~FJM1C3 to control. In T2 and T3 the FJCNT[2:0], FJSA[2:0] and FJDTnSA[2:0] values will be adjusted according to the settings of the FJM1C1 and FJDTnASA[1:0] bits in FJM1C2/FJM1C3 registers. In T2, the adjustment aim is FJPA/FJDT0A/FJDT1A, and in T3 the adjustment aim is FJPB/FJDT0B/FJDT1B.

- T5 (t4~t0, no timing): PWMP and PWMD are not controlled by FJM0C0~FJM0C2 or FJM1C0~FJM1C3 register settings, being operating without automatical adjustments and can be modified using the software.

Mode	Name	Setup Bits			Adjustment Control Register	
		FJWMD	FJMDS	FJHMDS	FJM0Cn (n=0~2)	FJM1Cn (n=0~3)
3	H/W-Count&Approach Mode	1	0	0	Y	Y
4	H/W-Full Approach Mode	1	1	0	X	Y
5	H/W-Partial Approach Mode	1	X	1	X	Y

The setup process is show as follows:

Step1: Wirte initial values into PWMP, PWMD, DT0, DT1, PWMMAXP and PWMMINP registers.

Step2: Wirte initial values into FJPA, FJPB, FJDT0A, FJDT0B, FJDT1A, FJDT1B and DTMIN registers.

Step3: Setup PWM & phase protection related registers: PWMC0, PWMC1, PWMC2, PLC and PPDSITA related settings.

Step4: Setup ERSG protection related registers: PWMPC0, PWMPC1, PWMPC2, ERSGC, DECPWMP.

Step5: Setup FJTIMER related registers: FJTMC, FJTMR1~FJTMR4.

Step6: Setup frequency jittering count mode related registers: FJM0C0~FJM0C2.

Step7: Setup frequency jittering approach mode related registers: FJM1C0~FJM1C3.

Step8: Setup other PWM related registers. As this chapter is explaining the approach mode, other registers are not listed here.

Step9: If the interrupt is used, the interrupt control registers must be correctly configured to ensure the approach match interrupt function is active. The master interrupt control bit, EMI, the approach match interrupt control bit, FJEQUE, and its corresponding multi-function interrupt control bit must all be set to 1.

Step10: Setup frequency jitter function related registers: FJC0. Note that the FJWMD bit cannot be set to 1 until completing all the settings mentioned above. Otherwise, when FJTSS is 0, once an external trigger signal occurs the frequency jitter function will be activated, which may result in unexpected conditions.

Step11: When |PWMP-FJPA|≤FJSA or |PWMP-FJPB|≤FJSA, the hardware will write the FJPA or FJPB data into the PWMP and set FJAF or FJBF bit to 1. If PMAXPTE is 1, when PWMP+FJSA>PWMMAXP, the PWMP will not plus FJSA and the FJEAF or FJEBF bit will be set to 1 by the hardware. Therefore users can use software polling the FJAF or FJBF, FJEAF or FJEBF bit to determine whether the PWMP is equal to the FJPA or FJPB or approaches the maximum value.

Step12: Reading the FJTMMD[2:0] bits can obtain the current FJTIMER operating timing section.

Note: 1. If the method of software polling the FJAF/FJBF/FJEAF/FJEBF is used, the interrupt enable step above can be omitted.

2. After entering the interrupt vector, it is necessary to read the FJAF/FJBF/FJEAF/FJEBF bit again to determine if the value remains at 1, ensuring the PWMP value has reached the setup value actually.

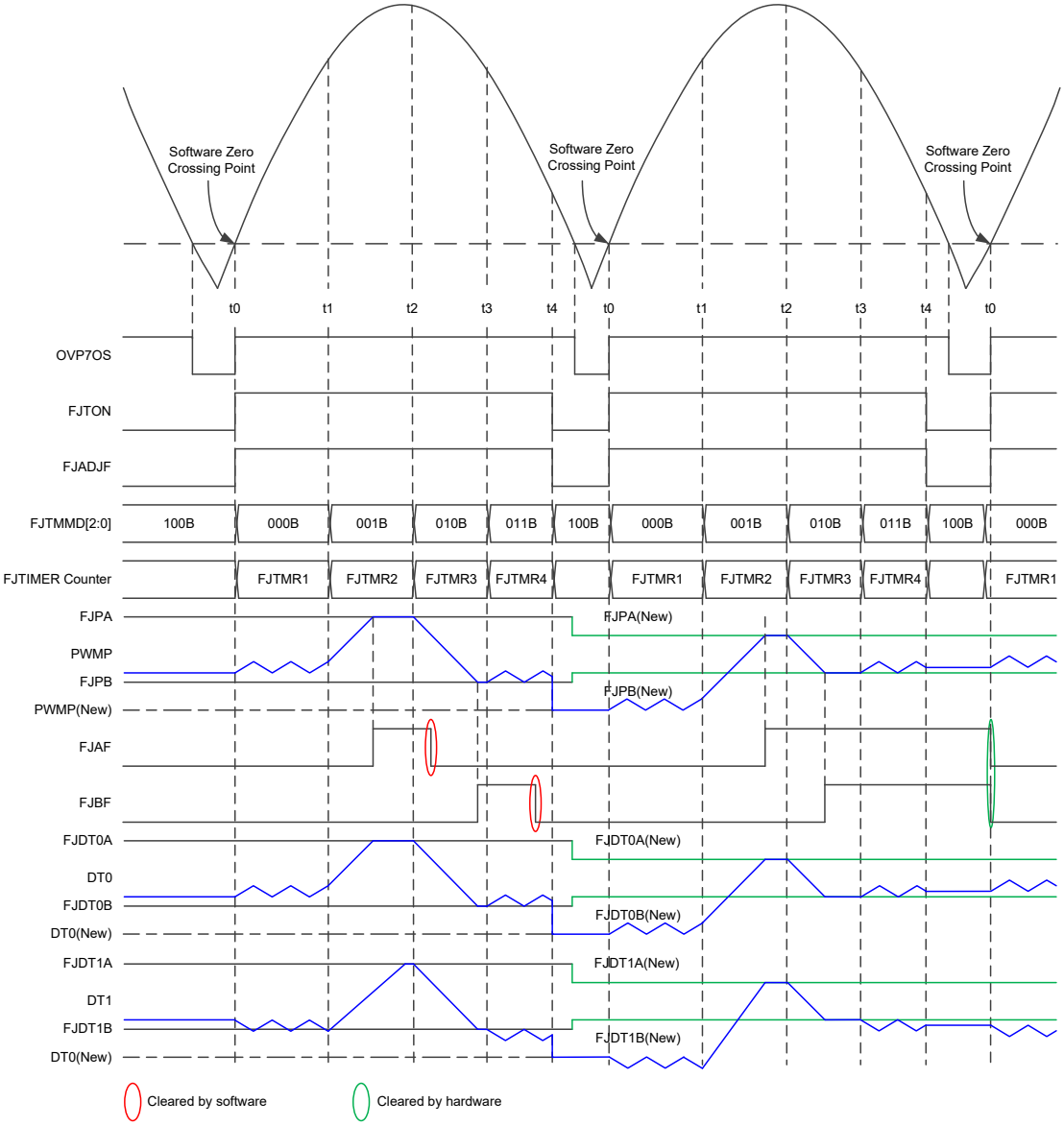
3. When |PWMP-FJPA|>FJSA or |PWMP-FJPB|>FJSA, the PWMP will be written with (PWMP-FJSA).

   a. If PWMD±FJSA>0, the PWMD will be written with PWMD±FJSA.

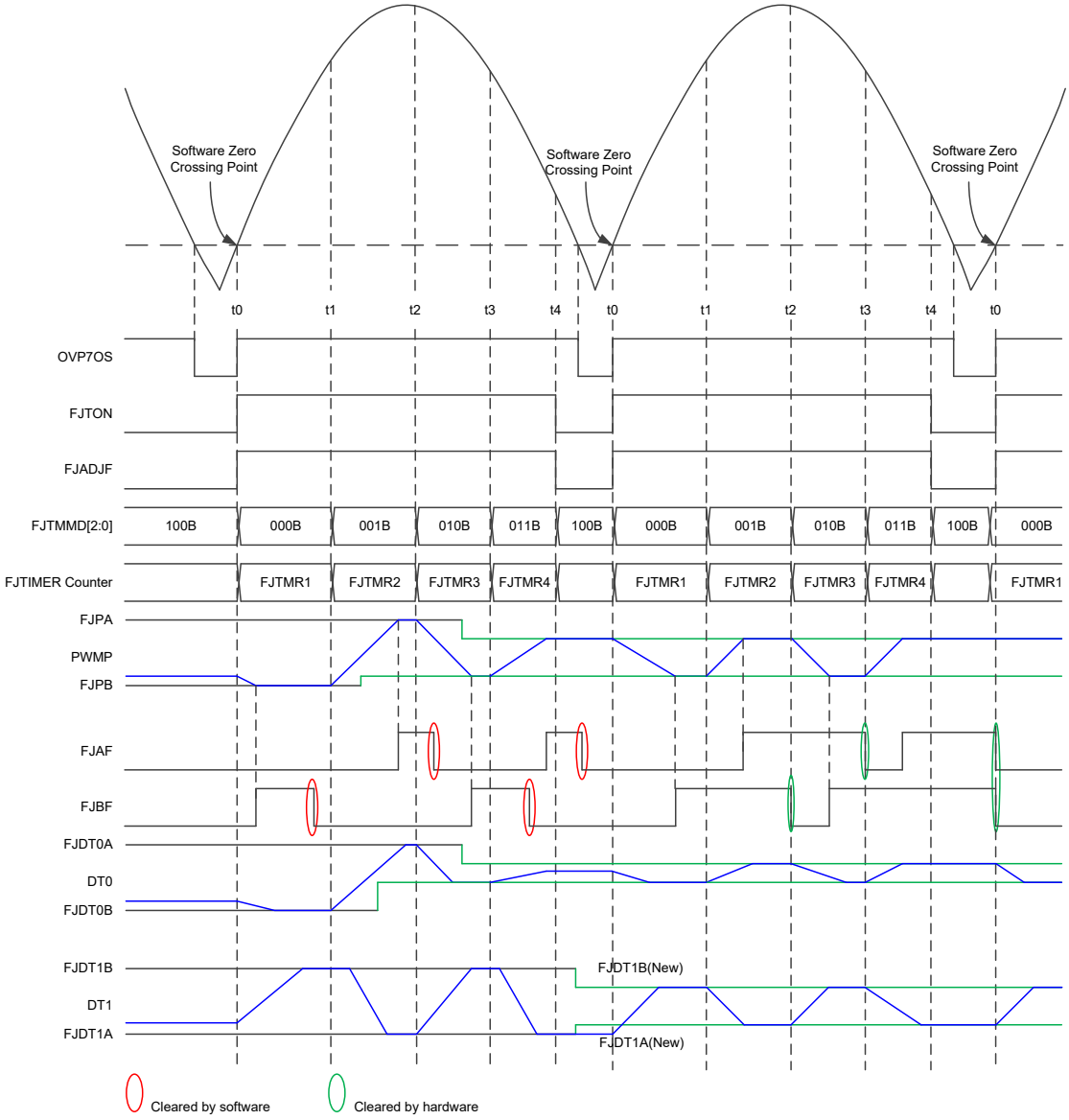
   b. If PWMD±FJSA<0, the PWMD will not change.

4. When |PWMP-FJPA|≤FJSA or |PWMP-FJPB|≤FJSA, the PWMP will be written with FJPA or FJPB.
   a. If PWMD±FJSA>0, the PWMD will be written with PWMD±(PWMP-FJPA)/2 or PWMD±(PWMP-FJPB)/2, discard the decimal.
   b. If PWMD±FJSA<0, the PWMD will not change.

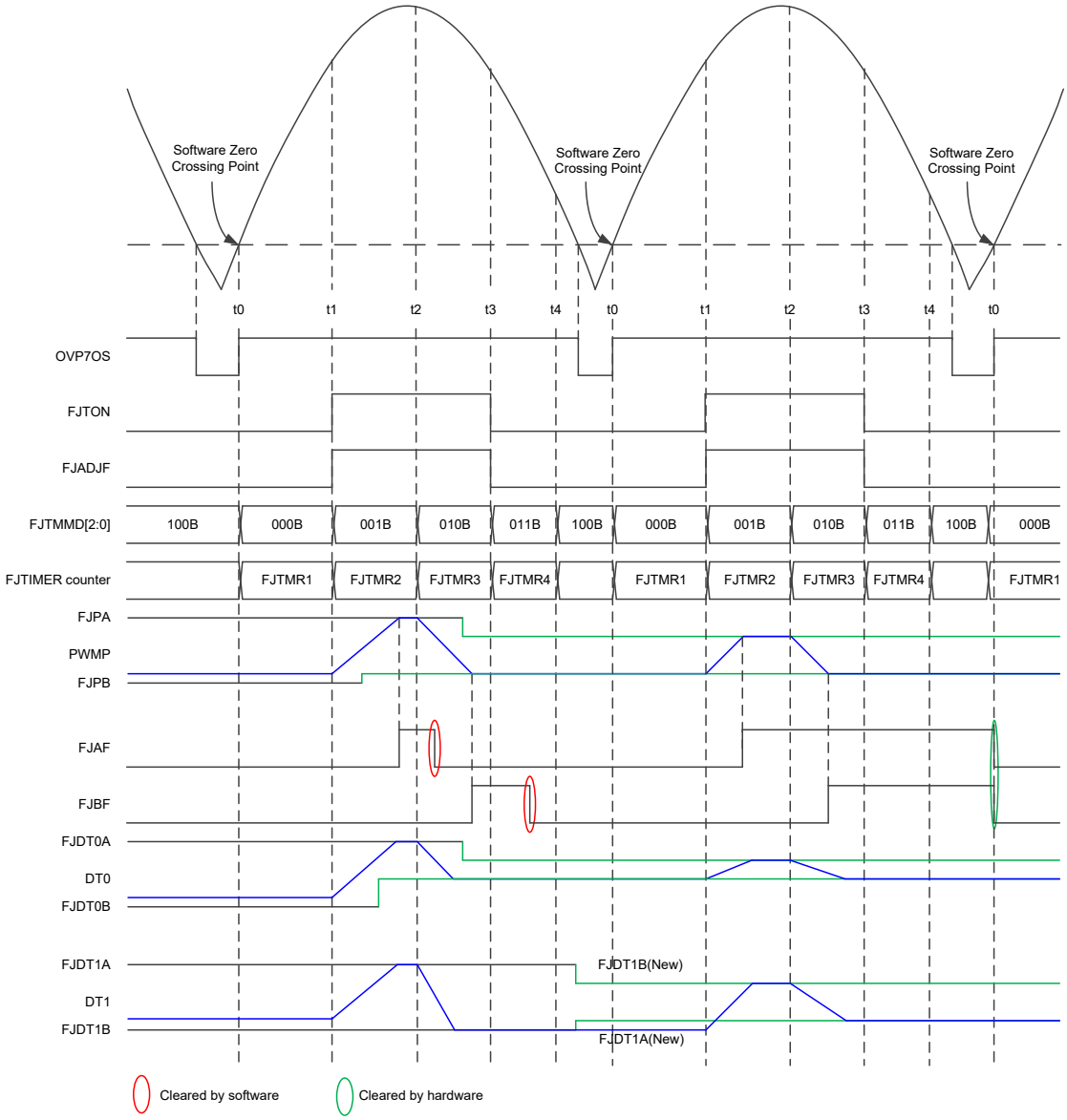
- Mode 3: H/W-Count&Approach Mode



- Mode 4: H/W- Full Approach Mode



- Mode 5: H/W- Partial Approach Mode



Startup Frequency Jitter Function

The ways to start frequency jitter function are different in differnet operating modes. For software modes, it is to use the F/W to set the FJEN bit to 1 to start the frequency jitter function. For hardware modes, one way is to use F/W to set the FJTON bit to 1 and the other way is to use an external signal rising edge to start the frequency jitter function. However, whether in software or hardware modes, the following points need to be noted.

1. When PWMON=0, whether FJEN is 1 or 0, the PWMP and the counter for frequency jittering will not be changed. This is because the counter counting up is driven by each PWMP overflow or changing PWMON from 0 to 1.
2. When FJEN=1, if an ERSG interrupt occurs, the FJEN bit will be cleared to zero by the hardware, and the PWM signal will be controlled by the ERSG protection circuit.
3. When ATMPF=1, if the sofeware sets the FJEN bit from 0 to 1 (S/W modes) or FJTON from 0 to 1, or the selected external trigger signal is received(H/W modes), these events will all be ignored and their "setting high" signals will also be cleared by the hardware.

Turnoff Frequency Jitter Function

The frequency jitter function can be disabled by software clearing PWMON or FJEN bit. In addition, when an ERSG interrupt occurs, the frequency jittering operation will stop immediately, and the FJTON bit (H/W modes) and FJEN bits will be cleared to 0 by the hardware.

PWMON	FJEN	ERSG INT	PWMPCn [m+1:m]	ATMPF	FPWMCLF	PWM Auto-Adjust	PWM Function	Frequency Jitter Function
0	x	x	x	x	x	N	Disable	Disable
1	0	0	x	x	x	N	Enable	Disable
1	0	↑	00	0	1	N	Enable → Disable	Disable
			01	0	1	N	Enable → Disable	Disable
			10	1	0 → 1	Y(Protect)	Enable → Disable	Disable
			11	1	0	Y(Protect)	Enable	Disable
1	1	0	x	x	x	Y(Protect)	Enable	Enable
1	1	↑	00	0	1	N	Enable → Disable	Enable → Disable
			01	0	1	N	Enable → Disable	Enable → Disable
			10	1	0 → 1	Y(Protect)	Enable → Disable	Enable → Disable
			11	1	0	Y(Protect)	Enable	Enable → Disable

Start Trigger Signal Selection

There are three ways, which are selected by FJTSS and FJOTSS bits, to start the H/W frequency jittering function. They are to use external signals or use the software to set FJTON bit from 0 to 1 to start the H/W frequency jittering function.

FJTSS	FJOTSS	Signal Description
0	0	OVP7OS
0	1	FJTR1
1	x	FJTON bit control

FJTIMER Operating Status

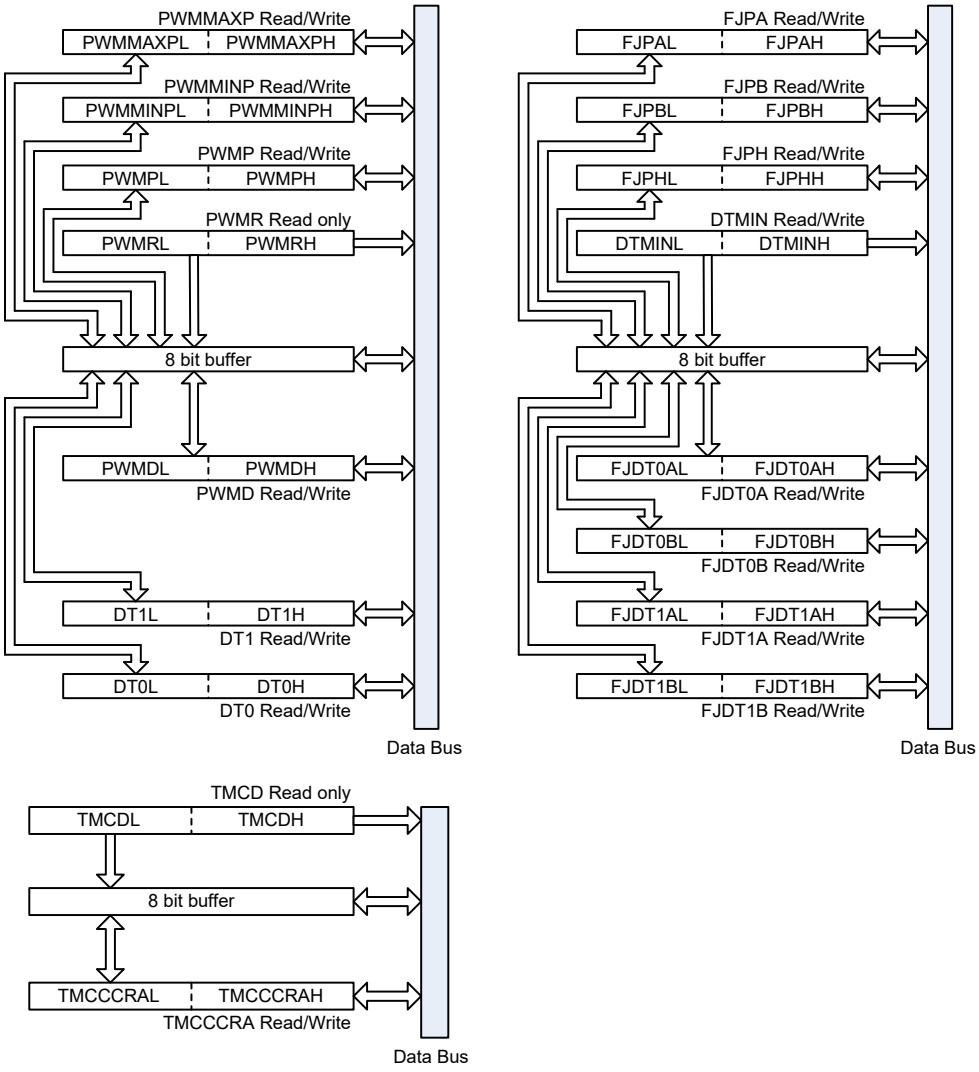
PWMON	FJWMD	FJTSS	FJOTSS	Description
0	X	X	X	General Timer Mode, load data from FJTMR1, FJTON is set by the software
↑	0	X	X	General Timer Mode, load data from FJTMR1, FJTON is set by the software
	1	X	X	If FJTON=1 and setting PWMON from 0 to 1, FJTON will be cleared by the hardware
1	0	X	X	General Timer Mode, load data from FJTMR1, FJTON is set by the software
	1	0	0	H/W frequency jitter mode, load data from FJTMR1~4, FJTON is set by OVP7OS signal triggering
			1	H/W frequency jitter mode, load data from FJTMR1~4, FJTON is set by FJTR1 signal triggering
		1	X	H/W frequency jitter mode, load data from FJTMR1~4, FJTON is set by the software
	↑	X	X	If FJTON=1 and setting FJWMD to 1, FJTON will be cleared by the hardware
	↓	X	X	Exit the frequency jitter mode, FJTON is not cleared. Operated in general timer mode, load data from FJTMR1

Programming Considerations

The following shows some points requiring special attentions during programming.

Data Register Access

The 12-bit PWMR, PWMP, PWMD, DT0, DT1, PWMMAXP, PWMMINP, FJPA, FJPB, FJPH, FJDT0A, FJDT0B, FJDT1A, FJDT1B and DTMIN registers, and the 10-bit TMCD, TMCCCRA 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.



The following steps show the read and write procedures:

- Writing Data to PWMP, PWMD, DT0, DT1, PWMMAXP, PWMMINP, FJPA, FJPB, FJPH, FJDT0A, FJDT0B, FJDT1A, FJDT1B, DTMIN and TMCCCRA
  - Step 1. Write data to Low Byte PWMPL/PWMDL/DT0L/DT1L/PWMMAXPL/PWMMINPL/FJPAL/FJPBL/FJPHL/FJDT0AL/FJDT0BL/FJDT1AL/FJDT1BL/DTMINL/TMCCCRAL
    - Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte PWMPH/PWMDH/DT0H/DT1H/PWMMAXPH/PWMMINPH/FJPAH/FJPBH/FJPHH/FJDT0AH/FJDT0BH/FJDT1AH/FJDT1BH/DTMINH/TMCCCRAH
    - 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, PWMP, PWMD, DT0, DT1, PWMMAXP, PWMMINP, FJPA, FJPB, FJPH, FJDT0A, FJDT0B, FJDT1A, FJDT1B, DTMIN and TMCCCRA
  - Step 1. Read data from the High Byte PWMRH/PWMPH/PWMDH/DT0H/DT1H/PWMMAXPH/PWMMINPH/FJPAH/FJPBH/FJPHH/FJDT0AH/FJDT0BH/FJDT1AH/FJDT1BH/DTMINH/TMCDH/TMCCCRAH
    - 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 PWMRL/PWMPL/PWMDL/DT0L/DT1L/PWMMAXPL/PWMMINPL/FJPAL/FJPBL/FJPHL/FJDT0AL/FJDT0BL/FJDT1AL/FJDT1BL/DTMINL/TMCDL/TMCCCRAL
    - This step reads data from the 8-bit buffer.

PWMP/FJPA/FJPB/FJPH Maximum Limit Description

- When PMAXPTE=0, any data can be written into the PWMP/FJPA/FJPB/FJPH regardless of the PWMMAXP limit and the related flags PMAXF/AMAXF/BMAXF/HMAXF are all 0.
- When PMAXPTE is changed from 0 to 1, the hardware will compare the value of the PWMP/FJPA/FJPB/FJPH with the PWMMAXP. If PWMP/FJPA/FJPB/FJPH>PWMMAXP, the corresponding flag PMAXF/AMAXF/BMAXF/HMAXF will be set to 1; if not, the flag will be cleared to 0.
- When PMAXPTE=1, a data write operation to the PWMMAXP/PWMP/FJPA/FJPB/FJPH will activate the maximum value limit circuit which will compare the value of the PWMP/FJPA/FJPB/FJPH with the PWMMAXP. If PWMP/FJPA/FJPB/FJPH>PWMMAXP, the corresponding flag PMAXF/AMAXF/BMAXF/HMAXF will be set to 1; if not, the flag will be cleared to 0.

The following table gives an example about writing data into the PWMP, which also applies to the FJPA/FJPB/PJPH.

Step	PMAXPTE	PWMMAXP	PWMP (To Write)	PWMP (Actual Result)	PMAXF
1	0	1000	500	500	0
2	0	1000	1024	1024	0
3	1	1000	—	1024	1
4	1	1000	1010	1024	1
5	1	1000	980	980	0
6	1	900	—	980	1
7	1	900	920	980	1
8	1	900	890	890	0
9	1	850	—	890	1
10	0	850	—	890	0

Register Data Load Timing

- PWMPCn[m+1:m]: When any signal in ERSG0~ERSG7 or the ERSGS bit is high, if writing a new data into the PWMPCn[m+1:m] bits, the hardware will not update the bits until the ERSG0~ERSG7 singals and ERSGS are all 0.
- PWMP, PWMD, DT0, DT1: When writing a new data into PWMP, PWMD, DT1 or DT0 register by the software, the new data will not be transferred to its associated counter until the PWMR counter content is 0.

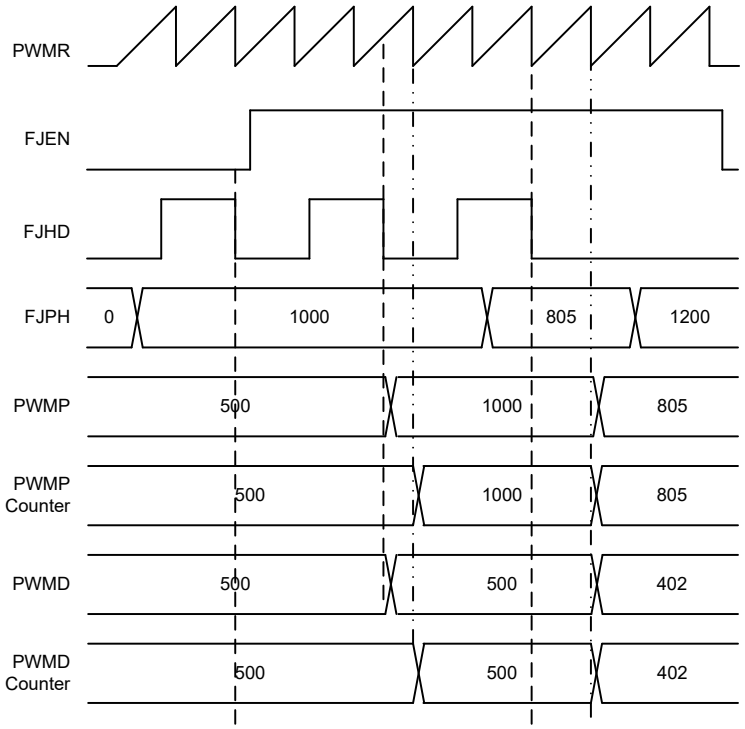
Using FJPH to load Data into PWMP/PWMD

When frequency jitter function is enabled, the contents of PWMP and PWMD will be adjusted by the hardware and cannot be changed by the software. The device provides a way for the software to change the PWMP and PWMD contents during the frequency jittering process. That is to use the FJHD bit in the FJC1 register together with the FJPH registers to change the PWMP and PWMD values. Note that the modified content will be loaded in the next PWM period, so the PWMP/PWMD will not be updated in the current period. It should be noted that if the software modification and the hardware modification occur at the same time or very close, the PWMP/PWMD will be updated with the hardware values first while the software value will be loaded in the next PWM period.

When a new PWMP data is written into the FJPH register by the software and then setting the FJHD bit from 1 to 0, the FJPH content will be transferred into the PWMP and FJPH/2 (discarding the decimal) will be transferred into the PWMD register by the hardware. The new PWM will be generated in the next period. If the time point of FJHD bit changing from 1 to 0 is very close to the PWMP reloading time when PWMR counter is equal to 0, the hardware will delay loading this software modification for one PWM cycle. The following provides an example to explain the PWMP and PWMD updates.

Mode	FJEN	Close to Hardware Reloading Time ?	Load from FJPH	Description
0	0	X	X	General PWM mode
1	1	No	Yes	FJPH and FJPH/2 values are loaded into PWMP and PWMD
2	1	Yes	No	Frquency jittering caculation values are loaded into PWMP and PWMD
3	1	Yes	Yes	Frquency jittering caculation values are loaded into PWMP and PWMD
	1	Yes	No	FJPH and FJPH/2 values are loaded into PWMP and PWMD

"x": Don't care



Frequency Jitter Start and Trigger Notes

The Fruquency jitter function can be accurately started only when there is no ERSG interrput and the ATMPF bit is 0. When the PWM circuit is turned off, the FJEN bit will also be cleared to 0 by the hardware. If the software modifies the relevant settings, the new data will not be loaded correctly until the current PWM period is fully completed.

Mode	PWMON	FJEN	PWM Protect Circuit	FJWMD	FJMDS	FJHMDS	Description
0	0	X	X	X	X	X	PWM is off
1	1	0	0	X	X	X	PWM is not updated automatically
2	1	0	1	X	X	X	ERSG Protection mode
3	1	1	0	0	0	X	S/W-Count Mode
4	1	1	0	0	1	X	S/W-Approach Mode
5	1	1	0	1	0	0	H/W-Count&Approach Mode
6	1	1	0	1	1	0	H/W-Full Approach Mode
7	1	1	0	1	X	1	H/W-Partial Approach Mode

FJTIMER Setup Notes

The FJTIMER counter is used in H/W frequency fittering mode for induction cooker applications. The FJTMR1~FJTMR4 register should be setup according to the AC half-wave frequency. If the sum of (T1+T2+T3+T4) is greater than half-wave time (10ms at 50Hz), an external trigger signal (FJWMD=1 & FJTSS=0) will occur before the FJTMR4 timing is completed. And then the hardware will load FJTMR1 data to the FJTIMER counter and restart counting. The internal counter related to frequency jittering will be cleared to 0 and restart counting and the FJTMMD[2:0] bits will change from 100 to 000 directly.

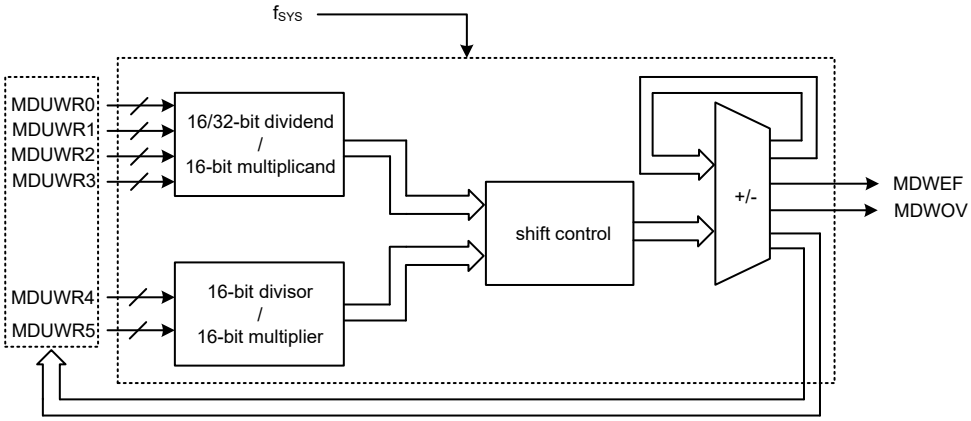
ATMPC Setting Notes

- When ATMPF=0, the ATMPC bit is set to 1 by software, when an ERSG protection signal is generated, which will set the ATMPF bit to 1, however the period auto-adjust hardware circuit will not be activated. The hardware will clear the ATMPC bit to 0 at the end of the current PWM cycle while the ATMPF bit needs to be cleared using the software.
- When ATMPF=1, the ATMPC bit is set to 1 by software, the period auto-adjust hardware circuit will be turned off at the end of the current PWM cycle, and the ATMPC bit will be cleared to 0 by the hardware. The ATMPF bit needs to be cleared using the software.
- In practical applications, the ATMPC bit is suggested to be set to 1 after the signal trigerring the ERSG protection circuit disappears, otherwise the auto-adjust hardware circuit will be turned off which will cause application problems.

Step	ATMPC	ATMPF	Operation Conditions
1	0	0	No PWM auto-adjust
2	1	0	No PWM auto-adjust
3	1	↑	No PWM auto-adjust, the ATMPC bit is cleared to 0 at the end of the current PWM cycle
4	↓	1	No PWM auto-adjust
5	0	↓	No PWM auto-adjust
6	0	1	PWM auto-adjust is executed according to the setting
7	1	0	The ATMPC bit is cleared to 0 at the end of the current PWM cycle. The auto-adjust hardware circuit is turned off automatically

16-bit Multiplication Division Unit – MDU

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



16-Bit MDU Block Diagram

MDU Registers

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

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

MDU Register List

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

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

"x" unknown

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

Using the register depends on the current MDU operation

The MDUWRn (n=0~5) registers cannot be read or written until the relevant MDU operation is completed. Otherwise, it will result in abnormal MDU operation results together with the MDU error flag, MDWEF, being set high.

• **MDUWCTRL Register**

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

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

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

Bit 5~0 Unimplemented, read as "0"

MDU Operation

For this MDU the multiplication or division operation is carried out in a specific way and is determined by the write access sequence of the six MDU data registers, MDUWR0~MDUWR5. The low byte data, regardless of the dividend, multiplicand, divisor or multiplier, must first be written into the corresponding MDU data register followed by the high byte data. All MDU operations will be executed after the MDUWR5 register is write-accessed together with the correct specific write access sequence of the MDUWRn. The relationship between the write access sequence and the MDU operation is shown as follows:

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

Note that it is not necessary to consecutively write data into the MDU data registers but data must be written in a correct write access sequence. Therefore, a non-write MDUWRn instruction or an interrupt, etc., can be inserted into the correct write access sequence without destroying the write operation.

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

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

The operation results will be stored in the corresponding MDU data registers and should be read out from the MDU data registers using a specific read access sequence after the operation has completed. The relationship between the operation result read access sequence and the MDU operation is shown as follows:

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

Note that it is not necessary to consecutively read data out from the MDU data registers however it must be carried out using a correct read access sequence. Therefore, a non-read MDUWRn instruction or an interrupt, etc., can be inserted into the correct read access sequence without destroying the read operation.

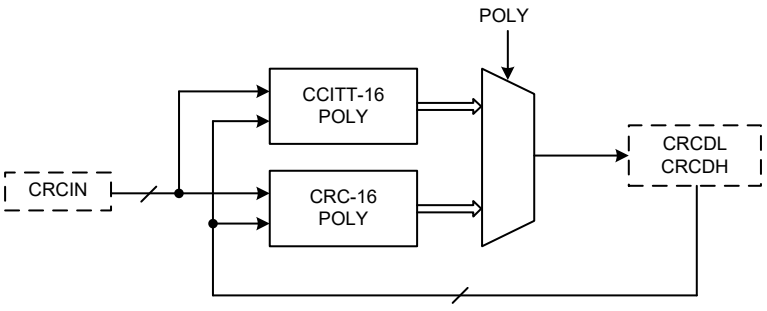
The overall important points for the MDU read/write access sequence and calculation time are summarised in the following table. Note that the device should not enter the IDLE or SLEEP mode until the MDU operation is totally completed, otherwise the MDU operation will fail.

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

MDU Operations Summary

Cyclic Redundancy Check – CRC

The Cyclic Redundancy Check, CRC, calculation unit is an error detection technique test algorithm used to verify data transmission or storage data correctness. A CRC calculation takes a data stream or a block of data as its input and generates a 16-bit output remainder. Ordinarily, a data stream is suffixed by a CRC code and used as a checksum when being sent or stored. Therefore, the received or restored data stream is calculated by the same generator polynomial as described in the following section.



CRC Block Diagram

CRC Registers

The CRC generator contains an 8-bit CRC data input register, CRCIN, and a CRC checksum register pair, CRCDH and CRCDL. The CRCIN register is used to input new data and the CRCDH and CRCDL registers are used to hold the previous CRC calculation result. A CRC control register, CRCCR, is used to select which CRC generating polynomial is used.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CRCCR	—	—	—	—	—	—	—	POLY
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D15	D14	D13	D12	D11	D10	D9	D8

CRC Register List

• CRCCR Register

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **POLY**: 16-bit CRC generating polynomial selection
0: CRC-CCITT: $X^{16}+X^{12}+X^5+1$
1: CRC-16: $X^{16}+X^{15}+X^2+1$

• CRCIN 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**: CRC input data register

• CRCDL 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**: 16-bit CRC checksum low byte data register

• CRCDH 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**: 16-bit CRC checksum high byte data register

CRC Operation

The CRC generator provides the 16-bit CRC result calculation based on the CRC16 and CCITT CRC16 polynomials. In this CRC generator, there are only these two polynomials available for the numeric values calculation. It cannot support the 16-bit CRC calculations based on any other polynomials.

The following two expressions can be used for the CRC generating polynomial which is determined using the POLY bit in the CRC control register, CRCCR. The CRC calculation result is called as the CRC checksum, CRCSUM, and stored in the CRC checksum register pair, CRCDH and CRCDL.

- CRC-CCITT: $X^{16}+X^{12}+X^5+1$
- CRC-16: $X^{16}+X^{15}+X^2+1$

CRC Computation

Each write operation to the CRCIN register creates a combination of the previous CRC value stored in the CRCDH and CRCDL registers and the new data. The CRC unit calculates the CRC data register value is based on byte by byte. It will take one MCU instruction cycle to calculate the CRC checksum.

CRC Calculation Procedures

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Execute an "Exclusive OR" operation with the 8-bit input data byte and the 16-bit CRCSUM high byte. The result is called the temporary CRCSUM.
3. Shift the temporary CRCSUM value left by one bit and move a "0" into the LSB.
4. Check the shifted temporary CRCSUM value after procedure 3.

   If the MSB is 0, then this shifted temporary CRCSUM will be considered as a new temporary CRCSUM. Otherwise, execute an "Exclusive OR" operation with the shifted temporary CRCSUM in procedure 3 and a data "8005H". Then the operation result will be regarded as the new temporary CRCSUM.

   Note that the data to be perform an "Exclusive OR" operation is "8005H" for the CRC-16 polynomial while for the CRC-CCITT polynomial the data is "1021H".
5. Repeat the procedure 3 ~ procedure 4 until all bits of the input data byte are completely calculated.
6. Repeat the procedure 2 ~ procedure 5 until all of the input data bytes are completely calculated. Then, the latest calculated result is the final CRC checksum, CRCSUM.

CRC Calculation Examples

- Write a 1-byte input data into the CRCIN register and the corresponding CRC checksum are individually calculated as the following table shown.

CRC Data Input / CRC Polynomial	00H	01H	02H	03H	04H	05H	06H	07H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	0000H	1021H	2042H	3063H	4084H	50A5H	60C6H	70E7H
CRC-16 ($X^{16}+X^{15}+X^2+1$)	0000H	8005H	800FH	000AH	801BH	001EH	0014H	8011H

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before each CRC input data is written into the CRCIN register.

- Write a 4-byte input data into the CRCIN register sequentially and the CRC checksum are sequentially listed in the following table.

CRC Data Input / CRC Polynomial	CRCIN = 78H→56H→34H→12H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	(CRCDH, CRCDL)=FF9FH→BBC3H→A367H→D0FAH
CRC-16 ($X^{16}+X^{15}+X^2+1$)	(CRCDH, CRCDL)=0110H→91F1H→F2DEH→5C43H

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before the sequential CRC data input operation.

Program Memory CRC Checksum Calculation Example

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Select the CRC-CCITT or CRC-16 polynomial as the generating polynomial using the POLY bit in the CRCCR register.
3. Execute the table read instruction to read the program memory data value.
4. Write the table data low byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.
5. Write the table data high byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.
6. Repeat the procedure 3 ~ procedure 5 to read the next program memory data value and execute the CRC calculation until all program memory data are read followed by the sequential CRC calculation. Then the value in the CRC checksum register pair is the final CRC calculation result.

Universal Serial Interface Module – USIM

The device contains a Universal Serial Interface Module, which includes the four-line SPI interface, the two-line I^2C interface and the two-line/single-wire UART interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI, I^2C or UART based hardware such as sensors, Flash or EEPROM memory, etc. The USIM interface pins are pin-shared with other I/O pins therefore the USIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As all the interface types share the same pins and registers, the choice of whether the UART, SPI or I^2C type is used is made using the UART mode selection bit, named UMD, and the SPI/I^2C operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the USIM pin-shared I/O are selected using pull-high control registers when the USIM function is enabled and the corresponding pins are used as USIM input pins.

SPI Interface

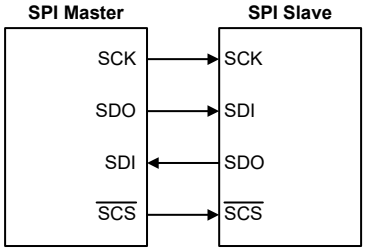
The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four-line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the 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, but the device provides only one $\overline{SCS}$ pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pin 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 SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I^2C/UART function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. 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 $\overline{SCS}$ pin only one slave device can be utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to 1 to enable $\overline{SCS}$ pin function, set CSEN bit to 0 the $\overline{SCS}$ pin will be floating state.

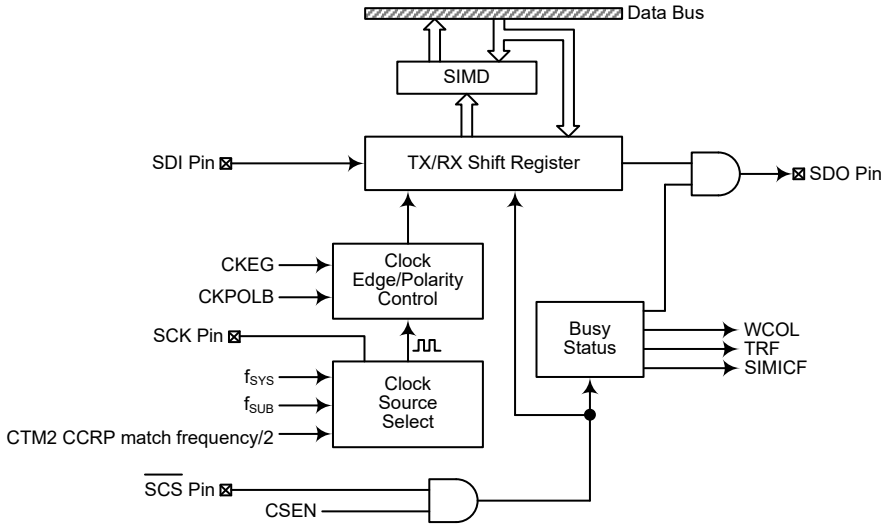


SPI Master/Slave Connection

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 CSEN and SIMEN.



SPI Block Diagram

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two control registers, SIMC0 and SIMC2. Note that the SIMC2 and SIMD registers and their POR values are only available when the SPI mode is selected by properly configuring the UMD and SIM2~SIM0 bits in the SIMC0 register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
SIMD	D7	D6	D5	D4	D3	D2	D1	D0

SPI Register List

SPI Data Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I^2C functions. Before the device writes data to the SPI bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the SPI bus, the device can read it from the SIMD register. Any transmission or reception of data from the SPI bus must be made via the SIMD register.

- **SIMD 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**: USIM SPI/I^2C data register bit 7 ~ bit 0

SPI Control Registers

There are also two control registers for the SPI interface, SIMC0 and SIMC2. The SIMC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC2 register is used for other control functions such as LSB/MSB selection, write collision flag etc.

- **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SIM2~SIM0**: USIM SPI/I^2C 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 CTM2 CCRP match frequency/2
101: SPI slave mode
110: I^2C slave mode
111: Unused mode

When the UMD bit is cleared to zero, these bits setup the SPI or I^2C operating mode of the USIM function. As well as selecting if the I^2C or SPI function, they 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 STM and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UMD**: UART mode selection bit
0: SPI or I²C mode
1: UART mode
This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be cleared to zero for SPI or I²C mode.

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
These bits are only available when the USIM is configured to operate in the I²C mode. Refer to the I²C register section.

Bit 1 **SIMEN**: USIM SPI/I²C Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the USIM SPI/I²C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I²C interface, the SDI, SDO, SCK and $\overline{SCS}$ or SDA and SCL lines will lose their SPI or I²C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I²C interface is enabled. If the USIM is configured to operate as an SPI interface via the UMD and SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the USIM is configured to operate as an I²C interface via the UMD and SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: USIM SPI Incomplete Flag
0: USIM SPI incomplete condition is not occurred
1: USIM SPI incomplete condition is occurred
This bit is only available when the USIM is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SIMEN and CSEN bits both being set high but the $\overline{SCS}$ line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set high together with the TRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the TRF bit will not be set high if the SIMICF bit is set high by software application program.

- **SIMC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
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 **D7~D6**: Undefined bits
These bits can be read or written by application program.

Bit 5 **CKPOLB**: SPI clock line base condition selection
0: The SCK line will be high when the clock is inactive
1: The SCK line will be low when the clock is inactive
The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4 **CKEG**: SPI SCK active clock edge type selection

CKPOLB=0
- 0: SCK is high base level and data capture at SCK rising edge
- 1: SCK is high base level and data capture at SCK falling edge

CKPOLB=1
- 0: SCK is low base level and data capture at SCK falling edge
- 1: SCK is low base level and data capture at SCK rising edge

The CKEG and CKPOLB 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 data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOLB bit.

Bit 3 **MLS**: SPI data shift order
- 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 **CSEN**: SPI $\overline{SCS}$ pin control
- 0: Disable
- 1: Enable

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

Bit 1 **WCOL**: SPI write collision flag
- 0: No collision
- 1: Collision

The WCOL 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 SIMD 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.

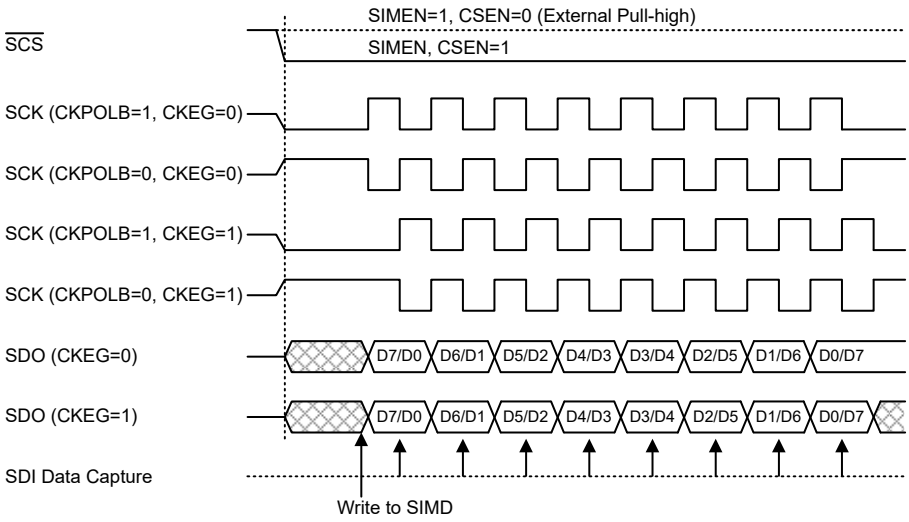
Bit 0 **TRF**: SPI Transmit/Receive complete flag
- 0: SPI data is being transferred
- 1: SPI data transmission is completed

The TRF bit is the Transmit/Receive Complete flag and is set "1" automatically when an SPI data transmission is completed, but must set to "0" by the application program. It can be used to generate an interrupt.

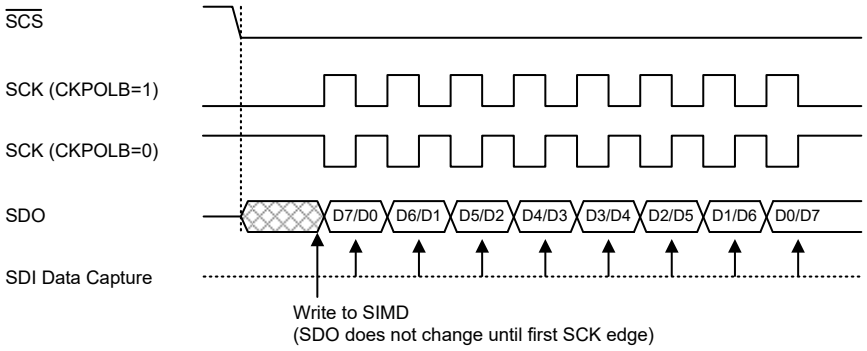
SPI Communication

After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMD register, transmission/reception will begin simultaneously. When the data transfer is completed, the TRF flag will be set high 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 SIMD register will be transmitted and any data on the SDI pin will be shifted into the SIMD register. The master should output an $\overline{SCS}$ signal to enable the slave devices before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the SCK signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagram shows the relationship between the slave data and SCK signal for various configurations of the CKPOLB and CKEG bits.

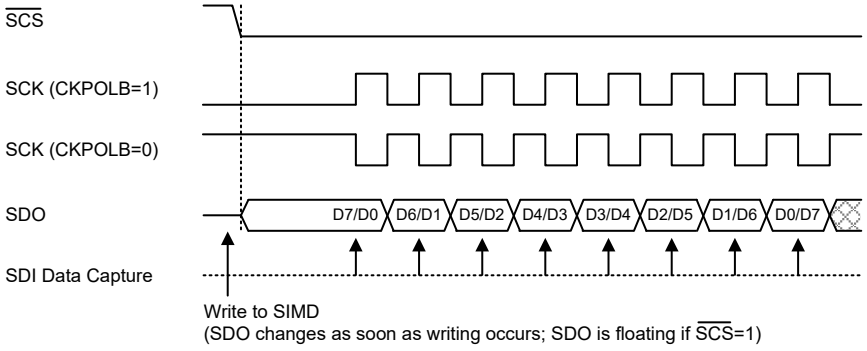
The SPI will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.



SPI Master Mode Timing

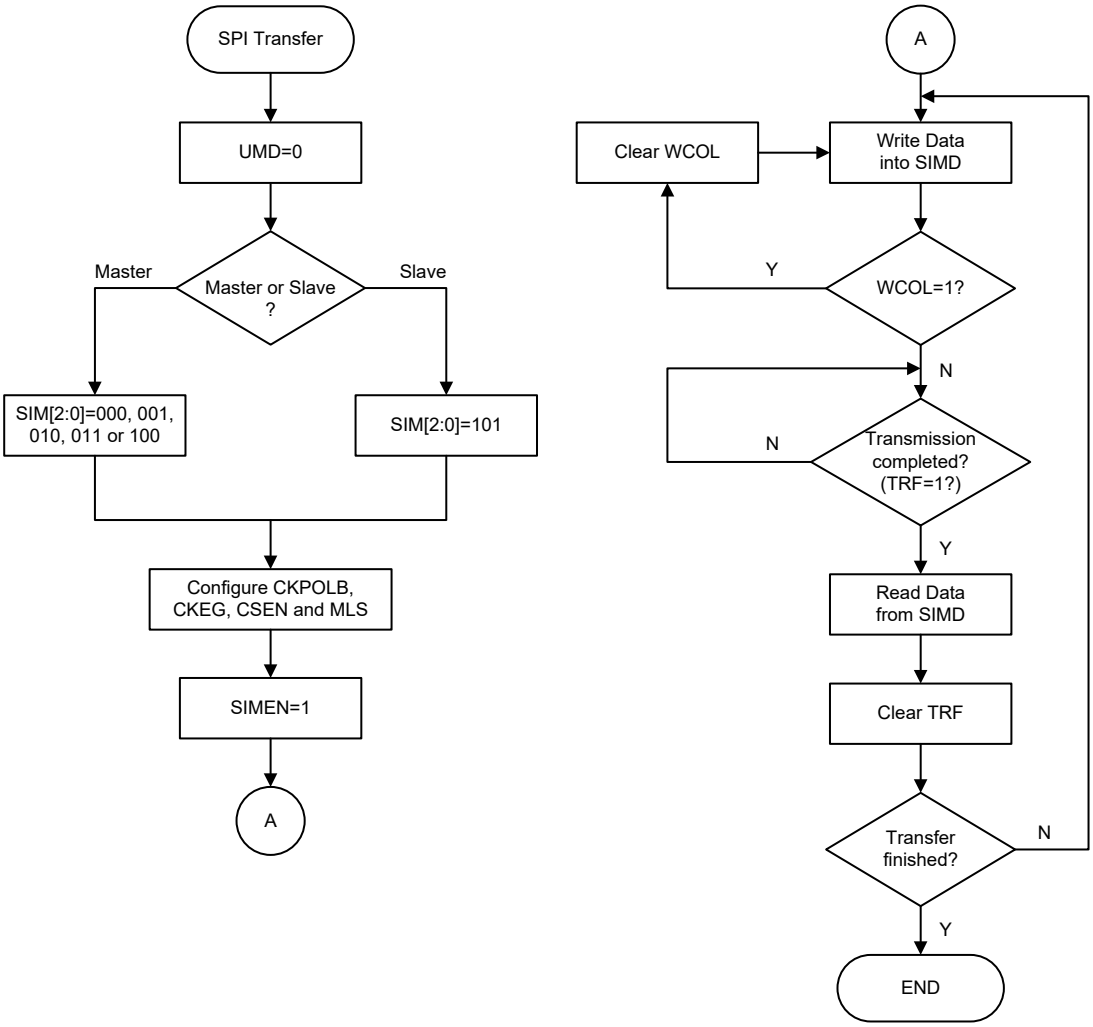


SPI Slave Mode Timing – CKEG=0



Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignores the $\overline{SCS}$ level.

SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flowchart

SPI Bus Enable/Disable

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

When the SPI bus is disabled, SCK, SDI, SDO and $\overline{SCS}$ can become I/O pins or other pin-shared functions using the corresponding pin-shared control bits.

SPI Operation Steps

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

The CSEN bit in the SIMC2 register controls the $\overline{SCS}$ pin function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{SCS}$ line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the $\overline{SCS}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the SDI line in a floating condition and the SDO line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and $\overline{SCS}$, SDI, SDO and SCK will all become I/O pins or the other functions using the corresponding pin-shared control 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 SIMD 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 UMD and SIM2~SIM0 bits in the SIMC0 control register.

- Step 2

  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave devices.

- Step 3

  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.

- Step 4

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

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

- Step 5

  Check the WCOL 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 TRF bit or wait for an USIM SPI serial bus interrupt.

- Step 7

  Read data from the SIMD register.

- Step 8

  Clear TRF.

- Step 9

  Go to step 4.

Slave Mode

- Step 1

  Select the SPI Slave mode using the UMD and SIM2~SIM0 bits in the SIMC0 control register.

- Step 2

  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.

- Step 3

  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.

- Step 4

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

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

- Step 5

  Check the WCOL 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 TRF bit or wait for an USIM SPI serial bus interrupt.

- Step 7

  Read data from the SIMD register.

- Step 8

  Clear TRF.

- Step 9

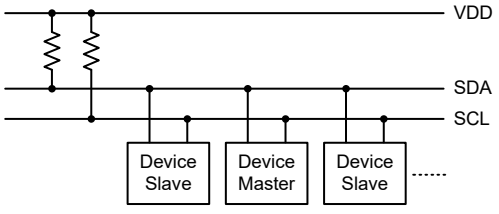
  Go to step 4.

Error Detection

The WCOL bit in the SIMC2 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 that a data collision has occurred which happens if a write to the SIMD register takes place during a data transfer operation and will prevent the write operation from continuing.

I²C Interface

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

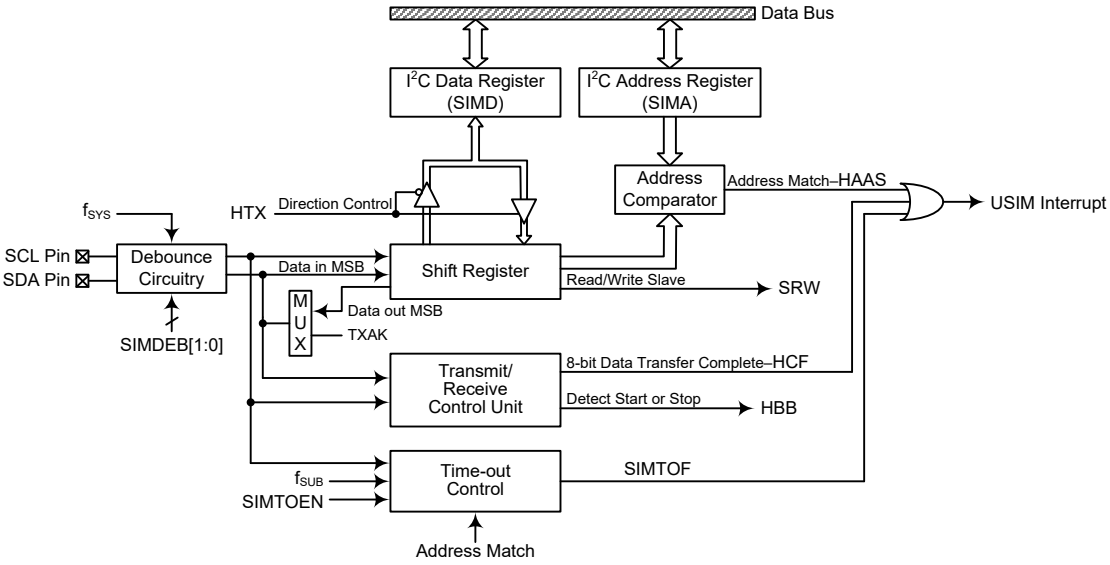


I²C Master Slave Bus Connection

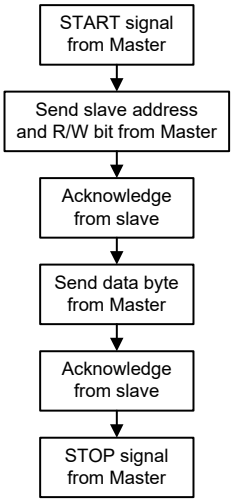
I²C Interface Operation

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

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For the device, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode. The pull-high control function pin-shared with SCL/SDA pin is still applicable even if I²C device is activated and the related internal pull-high register could be controlled by its corresponding pull-high control register.



I²C Block Diagram



I²C Interface Operation

The SIMDEB1 and SIMDEB0 bits determine the debounce time of the I²C interface. This uses the internal clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required I²C data transfer speed, there exists a relationship between the system clock, f_{SYS}, and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

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

I²C Minimum f_{SYS} Frequency Requirements

I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMTOC, one address register SIMA and one data register, SIMD. Note that the SIMC1, SIMD, SIMA and SIMTOC registers and their POR values are only available when the I²C mode is selected by properly configuring the UMD and SIM2~SIM0 bits in the SIMC0 register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C Register List

I²C Data Register

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

- **SIMD 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**: USIM SPI/I²C data register bit 7 ~ bit 0

I²C Address Register

The SIMA register is also used by the SPI interface but has the name, SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bit 7~1 of the SIMA register define the device slave address. Bit 0 is not defined. When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected.

• **SIMA Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	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 7~1 **SIMA6~SIMA0**: I²C slave address
SIMA6~SIMA0 is the 7-bit I²C slave address.

Bit 0 **D0**: Reserved bit, can be read or written by application program

I²C Control Registers

There are three control registers for the I²C interface, SIMC0, SIMC1 and SIMTOC. The SIMC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, SIMTOC, is used to control the I²C time-out function and is described in the corresponding section.

• **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C 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 CTM2 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode
When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM function. As well as selecting if the I²C or SPI function, they 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 STM and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UMD**: UART mode selection bit
0: SPI or I²C mode
1: UART mode
This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I²C mode.

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
00: Undefined
01: 2 system clock debounce
10: 4 system clock debounce
11: 4 system clock debounce
These bits are used to select the I²C debounce time when the USIM is configured as the I²C interface function by setting the UMD bit to "0" and SIM2~SIM0 bits to "110".

Bit 1 **SIMEN**: USIM SPI/I²C Enable Control
0: Disable
1: Enable

The bit is the overall on/off control for the USIM SPI/I²C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I²C interface, the SDI, SDO, SCK and SCS, or SDA and SCL lines will lose their SPI or I²C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I²C interface is enabled. If the USIM is configured to operate as an SPI interface via the UMD and SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the USIM is configured to operate as an I²C interface via the UMD and SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: USIM SPI Incomplete Flag

This bit is only available when the USIM is configured to operate in an SPI slave mode. Refer to the SPI register section.

- **SIMC1 Register**

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

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

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

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

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

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

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

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

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

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

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

The SRW flag is the I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C bus. When the transmitted

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

Bit 1 **IAMWU**: I²C Address Match Wake-up control
0: Disable
1: Enable

This bit should be set high to enable the I²C address match wake-up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake-up, then this bit must be cleared to zero by the application program after wake-up to ensure correction device operation.

Bit 0 **RXAK**: I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave does not receive acknowledge flag

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

I²C Bus Communication

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the SIMC1 register will be set and an USIM interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer completion or from the I²C bus time-out occurrence. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1

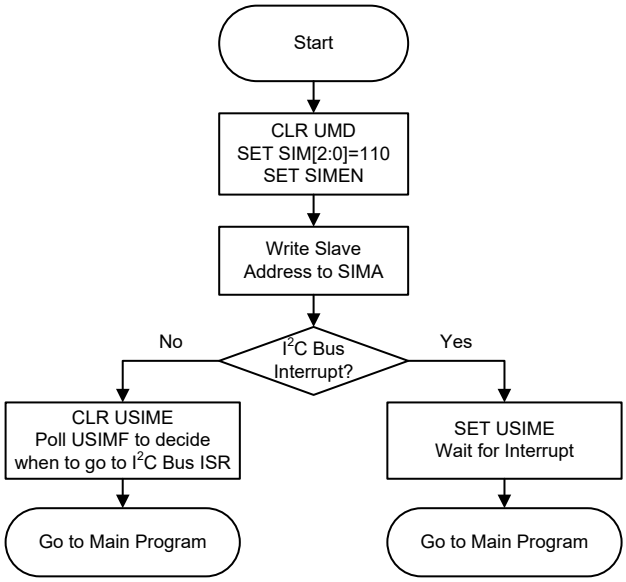
  Set the UMD, SIM2~SIM0 and SIMEN bits in the SIMC0 register to "0", "110" and "1" respectively to enable the I²C bus.

- Step 2

  Write the slave address of the device to the I²C bus address register SIMA.

- Step 3

  Set the USIME interrupt enable bit of the interrupt control register to enable the USIM interrupt.



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

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

I²C Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal USIM I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the SIMC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

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

I²C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the master device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is "1" then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the SRW flag is "0" then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

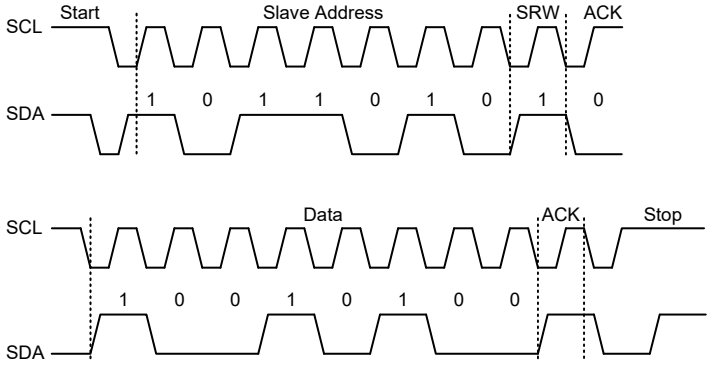
I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be set to "0".

I²C Bus Data and Acknowledge Signal

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

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the RXAK bit in the SIMC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.

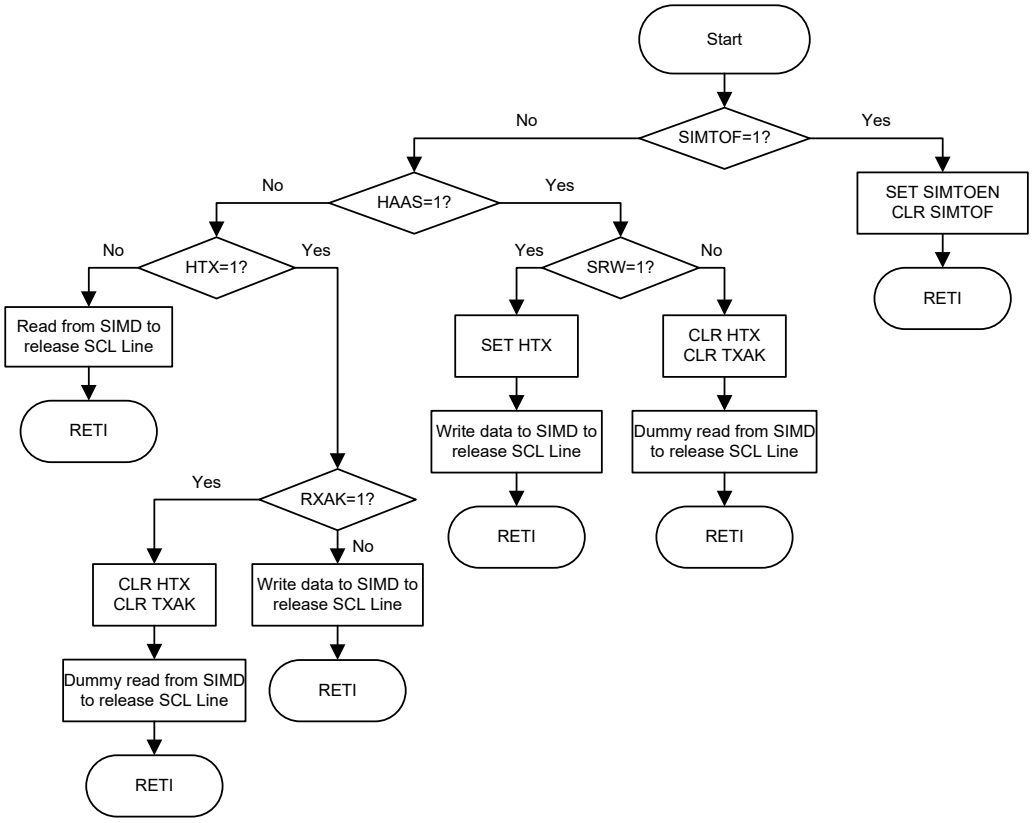


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

I²C Communication Timing Diagram

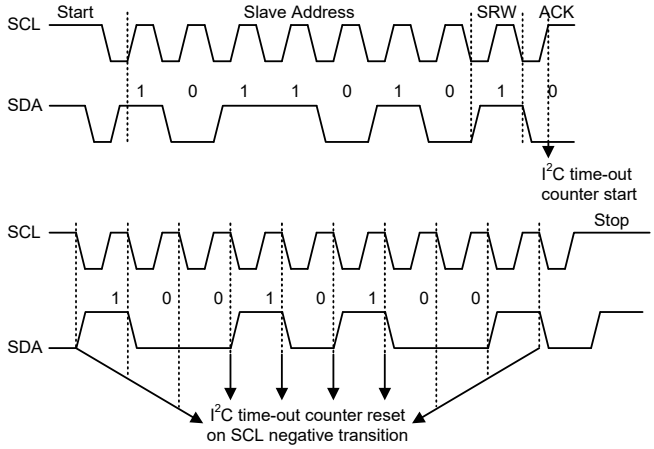
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.



I²C Bus ISR Flow Chart

I²C Time-out Control

In order to reduce the problem of I²C lockup due to reception of erroneous clock sources, a time-out function is provided. If the clock source to the I²C is not received for a while, then the I²C circuitry and registers will be reset after a certain time-out period. The time-out counter starts counting on an I²C bus "START" & "address match" condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out setup by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.



I²C Time-out

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

Registers	After I²C Time-out
SIMD, SIMA, SIMC0	No change
SIMC1	Reset to POR condition

I²C Registers after Time-out

The SIMTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using SIMTOS[5:0] bits in the SIMTOC register. The time-out time is given by the formula: ((1~64)×32)/f_{SUB}. This gives a time-out period which ranges from about 1ms to 64ms.

- **SIMTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
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 **SIMTOEN**: USIM I²C Time-out control
0: Disable
1: Enable

Bit 6 **SIMTOF**: USIM I²C Time-out flag
0: No time-out occurred
1: Time-out occurred
This bit is set high when time-out occurs and can only be cleared to zero by application program.

Bit 5~0 **SIMTOS5~SIMTOS0**: USIM I²C Time-out period selection
I²C time-out clock source is f_{SUB}/32.
I²C time-out time is equal to (SIMTOS[5:0]+1)×(32/f_{SUB}).

UART Interface

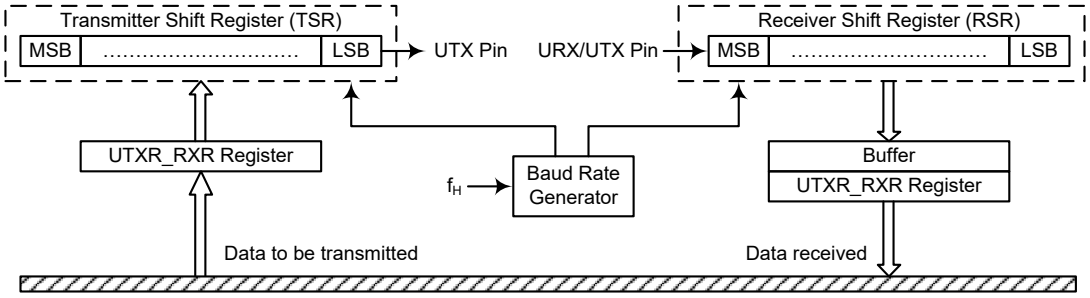
This UART interface function, which is part of the Serial Interface Module, should not be confused with the other independent UART function, which is described in another section of this datasheet.

The device contains an integrated full-duplex or half-duplex asynchronous serial communication UART interface that enables communication with external devices that contain a serial 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 shares the same internal interrupt vector with the SPI and I²C interfaces which can be used to indicate when a reception occurs or when a transmission terminates.

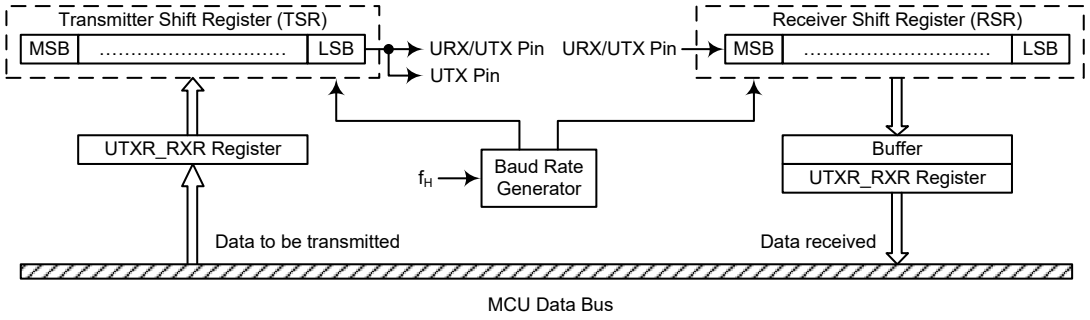
The integrated UART function contains the following features:

- Full-duplex or half-duplex (single wire mode) asynchronous communication
- 8 or 9 bits character length
- Even, odd or no parity options
- One or two stop bits
- Baud rate generator with 8-bit prescaler
- Parity, framing, noise and overrun error detection

- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 2-byte Deep FIFO Receive Data Buffer
- URX/UTX pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
  - Transmitter Empty
  - Transmitter Idle
  - Receiver Full
  - Receiver Overrun
  - Address Mode Detect



UART Data Transfer Block Diagram – USWM=0



UART Data Transfer Block Diagram – USWM=1

UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as UTX pin and URX/UTX pin, which are pin-shared with I/O or other pin functions. The UTX and URX/UTX pin function should first be selected by the corresponding pin-shared function selection register before the UART function is used. Along with the UMD bit, the UREN bit, the UTXEN or URXEN bits, if set, will setup 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 UTX or URX/UTX pin function is disabled by clearing the UMD, UREN, UTXEN or URXEN bit, the UTX or URX/UTX pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the UTX or URX/UTX 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 USWM bit in the UUCR3 register. When the USWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single URX/UTX pin can be used to transmit and receive data depending upon the corresponding control bits. When the URXEN bit is set high, the URX/UTX pin is used as a receiver pin. When the URXEN bit is cleared to zero and the UTXEN bit is set high, the URX/UTX pin will act as a transmitter pin.

It is recommended not to set both the URXEN and UTXEN bits high in the single wire mode. If both the URXEN and UTXEN bits are set high, the URXEN 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 UTX pin mentioned in this chapter should be replaced by the URX/UTX pin to understand the whole UART single wire mode function.

In the single wire mode, the data can also be transmitted on the UTX pin in a transmission operation with proper software configurations. Therefore, the data will be output on the URX/UTX and UTX 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 UTXR_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 UTX pin at a rate controlled by the Baud Rate Generator. Only the UTXR_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 URX/UTX 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 UTXR_RXR register, where it is buffered and can be manipulated by the application program. Only the UTXR_RXR register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.

It should be noted that the actual register for data transmission and reception only exists as a single shared register in the Data Memory. This shared register known as the UTXR_RXR register is used for both data transmission and data reception.

UART Status and Control Registers

There are seven control registers associated with the UART function. The UMD bit in the SIMC0 register can be used to select the UART interface. The USWM bit in the UUCR3 register is used to enable/disable the UART Single Wire Mode. The UUSR, UUCR1 and UUCR2 registers control the overall function of the UART, while the UBRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the UTXR_RXR data register. Note that UART related registers and their POR values are only available when the UART mode is selected by setting the UMD bit in the SIMC0 register to “1”.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
UUSR	UPERR	UNF	UFERR	UOERR	URIDLE	URXIF	UTIDLE	UTXIF
UUCR1	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
UUCR2	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
UUCR3	—	—	—	—	—	—	—	USWM
UTXR_RXR	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
UBRG	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0

UART Register List

- **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C Operating Mode Control
When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM function. Refer to the SPI or I²C register section for more details.

Bit 4 **UMD**: UART mode selection bit
0: SPI or I²C mode
1: UART mode
This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I²C mode.

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
Refer to the I²C register section.

Bit 1 **SIMEN**: USIM SPI/I²C Enable Control
This bit is only available when the USIM is configured to operate in an SPI or I²C mode with the UMD bit set low. Refer to the SPI or I²C register section for more details.

Bit 0 **SIMICF**: USIM SPI Incomplete Flag
Refer to the SPI register section.

- **UUSR Register**

The UUSR 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 UUSR register are read only. Further explanation on each of the flags is given below:

Bit	7	6	5	4	3	2	1	0
Name	UPERR	UNF	UFERR	UOERR	URIDLE	URXIF	UTIDLE	UTXIF
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

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

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

The UNF 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 UNF flag is set during the same cycle as the URXIF flag but will not be set in the case of as overrun. The UNF flag can be cleared to zero by a software sequence which will involve a read to the status register UUSR followed by an access to the UTXR_RXR data register.

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

The UFERR 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 to zero by a software sequence which will involve a read to the status register UUSR followed by an access to the UTXR_RXR data register.

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

The UOERR 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 UTXR_RXR receive data register. The flag is cleared to zero by a software sequence, which is a read to the status register UUSR followed by an access to the UTXR_RXR data register.

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

The URIDLE 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 URIDLE bit is "1" indicating that the UART receiver is idle and the URX/UTX pin stays in logic high condition.

Bit 2 **URXIF**: Receive UTXR_RXR data register status
0: UTXR_RXR data register is empty
1: UTXR_RXR data register has available data

The URXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the UTXR_RXR read data register is empty. When the flag is "1", it indicates that the UTXR_RXR read data register contains new data. When the contents of the shift register are transferred to the UTXR_RXR register, an interrupt is generated if URIE=1 in the UUCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags UNF, UFERR, and/or UPERR are set within the same clock cycle. The URXIF flag will eventually be cleared to zero when the UUSR register is read with URXIF set, followed by a read from the UTXR_RXR register, and if the UTXR_RXR register has no more new data available.

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

The UTIDLE 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 UTXIF flag is "1" and when there is no transmit data or break character being transmitted. When UTIDLE is equal to "1", the UTX pin becomes idle with the pin state in logic high condition. The UTIDLE flag is cleared to zero by reading the UUSR register with UTIDLE set and then writing to the UTXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

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

The UTXIF 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 UTXR_RXR data register. The UTXIF flag is cleared to zero by reading the UART status register (UUSR) with UTXIF set and then writing to the UTXR_RXR data register. Note that when the UTXEN bit is set, the UTXIF flag bit will also be set since the transmit data register is not yet full.

- **UUCR1 Register**

The UUCR1 register together with the UUCR2 and UUCR3 registers 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	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
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 **UREN**: UART function enable control
0: Disable UART. UTX and URX/UTX pins are in a floating state
1: Enable UART. UTX and URX/UTX pins function as UART pins

The UREN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the URX/UTX pin as well as the UTX pin will be set in a floating state. When the bit is equal to "1", the UART will be enabled if the UMD bit is set and the UTX and URX/UTX pins will function as defined by the USWM mode selection bit together with the UTXEN and URXEN 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 UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF bits will be cleared to zero, while the UTIDLE, UTXIF and URIDLE bits will be set high. Other control bits in UUCR1, UUCR2, UUCR3 and UBRG registers will remain unaffected. If the UART is active and the UREN bit is cleared to zero, 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 **UBNO**: 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 URX8 and UTX8 will be used to store the 9th bit of the received and transmitted data respectively.

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

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

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

Bit 3 **USTOPS**: Number of Stop bits selection for transmitter
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 transmitter. When this bit is equal to "1", two stop bits are used. If this bit is equal to "0", then only one stop bit is used.

Bit 2 **UTXBRK**: Transmit break character
0: No break character is transmitted
1: Break characters transmit
The UTXBRK bit is the Transmit Break Character bit. When this bit is "0", there are no break characters and the UTX 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 UTXBRK bit is reset.

Bit 1 **URX8**: 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 URX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **UTX8**: 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 UTX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• UUCR2 Register

The UUCR2 register is the second of the UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UART Transmitter and Receiver as well as enabling the various USIM UART mode interrupt sources. The register also serves to control the baud rate speed, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
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 **UTXEN**: UART Transmitter enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled
The bit named UTXEN 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 UTX pin will be set in a floating state.
If the UTXEN bit is equal to "1" and the UMD and UREN bit are also equal to "1", the transmitter will be enabled and the UTX pin will be controlled by the UART. Clearing the UTXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the UTX pin will be set in a floating state.

Bit 6 **URXEN**: UART Receiver enabled control
0: UART receiver is disabled
1: UART receiver is enabled

The bit named URXEN 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 URX/UTX pin will be set in a floating state. If the URXEN bit is equal to "1" and the UMD and UREN bit are also equal to "1", the receiver will be enabled and the URX/UTX pin will be controlled by the UART. Clearing the URXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the URX/UTX pin will be set in a floating state.

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

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

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

The bit named UADDEN 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 UTXRX7 if UBNO=0 or the 9th bit, which corresponds to URX8 if UBNO=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 UBNO. 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 **UWAKE**: URX/UTX pin wake-up UART function enable control
0: URX/UTX pin wake-up UART function is disabled
1: URX/UTX pin wake-up UART function is enabled

This bit is used to control the wake-up UART function when a falling edge on the URX/UTX pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no URX/UTX pin wake-up UART function if the UART clock (f_H) exists. If the UWAKE bit is set high as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the URX/UTX pin occurs. When this request happens and the corresponding interrupt is enabled, an URX/UTX 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 URX/UTX pin when the UWAKE bit is cleared to zero.

Bit 2 **URIE**: 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 UOERR or receive data available flag URXIF is set, the USIM interrupt request flag USIMF will be set. If this bit is equal to "0", the USIM interrupt request flag USIMF will not be influenced by the condition of the UOERR or URXIF flags.

Bit 1 **UTIIE**: 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 UTIDLE is set, due to a transmitter idle condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to "0", the USIM interrupt request flag USIMF will not be influenced by the condition of the UTIDLE flag.

Bit 0 **UTEIE**: 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 UTXIF is set, due to a transmitter empty condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to "0", the USIM interrupt request flag USIMF will not be influenced by the condition of the UTXIF flag.

• UUCR3 Register

The UUCR3 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, URX/UTX, together with the control of the URXEN and UTXEN bits in the UUCR2 register.

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **USWM**: Single Wire Mode enable control
0: Disable, the URX/UTX pin is used as UART receiver function only
1: Enable, the URX/UTX pin can be used as UART receiver or transmitter function controlled by the URXEN and UTXEN bits

Note that when the Single Wire Mode is enabled, if both the URXEN and UTXEN bits are high, the URX/UTX pin will just be used as UART receiver input.

• UTXR_RXR Register

The UTXR_RXR register is the data register which is used to store the data to be transmitted on the UTX pin or being received from the URX/UTX pin.

Bit	7	6	5	4	3	2	1	0
Name	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
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 **UTXRX7~UTXRX0**: UART Transmit/Receive Data bit 7 ~ bit 0

• UBRG Register

Bit	7	6	5	4	3	2	1	0
Name	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0
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 **UBRG7~UBRG0**: Baud Rate values

By programming the UBRGH bit in UUCR2 register which allows selection of the related formula described above and programming the required value in the UBRG register, the required baud rate can be setup.

Note: Baud rate=f_H/[64×(N+1)] if UBRGH=0.
Baud rate=f_H/[16×(N+1)] if UBRGH=1.

Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the baud rate register UBRG and the second is the value of the UBRGH bit in the control register UUCR2. The UBRGH bit decides if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value N in the UBRG register which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the UBRG register and has a range of between 0 and 255.

UUCR2 UBRGH Bit	0	1
Baud Rate (BR)	f_H / [64 (N+1)]	f_H / [16 (N+1)]

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

Calculating the Baud Rate and Error Values

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

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

Re-arranging this equation gives N = $[f_H/(BR\times64)]-1$

Giving a value for N = $[4000000/(4800\times64)]-1 = 12.0208$

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

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

UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding UBNO, UPRT, UPREN, and USTOPS bits in the UUCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

The basic on/off function of the internal UART function is controlled using the UREN bit in the UUCR1 register. When the UART mode is selected by setting the UMD bit in the SIMC0 register to "1", if the UREN, UTXEN and URXEN bits are set, then these two UART pins will act as normal UTX output pin and URX/UTX input pin respectively. If no data is being transmitted on the UTX pin, then it will default to a logic high value.

Clearing the UREN bit will disable the UTX and URX/UTX pin and allow these 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 UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF being cleared while bits UTIDLE, UTXIF and URIDLE will be set. The remaining control bits in the UUCR1, UUCR2, UUCR3 and UBRG registers will remain unaffected. If the UREN bit in the UUCR1 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.

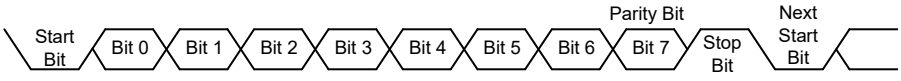
Data, Parity and Stop Bit Selection

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 UUCR1 register. The UBNO bit controls the number of data bits which can be set to either 8 or 9, the UPRT bit controls the choice of odd or even parity, the UPREN bit controls the parity on/off function and the USTOPS bit decides whether one or two stop bits are to be used. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only used for the transmitter. There is only one stop bit for the receiver.

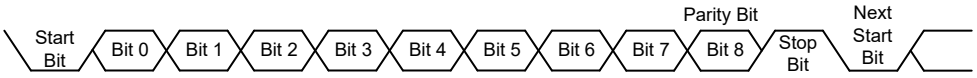
Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
Example of 9-bit Data Formats				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

Transmitter Receiver Data Format

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



8-bit data format



9-bit data format

UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the UBNO bit in the UUCR1 register. When UBNO 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 UTX8 bit in the UUCR1 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 UTXR_RXR register. The data to be transmitted is loaded into this UTXR_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 UTXR_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 UTXEN bit is set, but the data will not be transmitted until the UTXR_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 UTXR_RXR register, after which the UTXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the UTXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the UTXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The UTX output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

Transmitting Data

When the UART is transmitting data, the data is shifted on the UTX pin from the shift register, with the least significant bit first. In the transmit mode, the UTXR_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 UTX8 bit in the UUCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the UBNO, UPRT, UPREN and USTOPS bits to define the required word length, parity type and number of stop bits.
- Setup the UBRG register to select the desired baud rate.
- Set the UTXEN bit ensure that the UTX pin is used as a UART transmitter pin.
- Access the UUSR register and write the data that is to be transmitted into the UTXR_RXR register. Note that this step will clear the UTXIF bit.

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

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

1. A UUSR register access
2. A UTXR_RXR register write execution

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

During a data transmission, a write instruction to the UTXR_RXR register will place the data into the UTXR_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 UTXR_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the UTXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the UTIDLE bit will be set. To clear the UTIDLE bit the following software sequence is used:

1. A UUSR register access
2. A UTXR_RXR register write execution

Note that both the UTXIF and UTIDLE bits are cleared by the same software sequence.

Transmitting Break

If the UTXBRK bit is set high and the state keeps for a time greater than [(UBRG+1)×t_H] while UTIDLE=1, then the break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13×N '0' bits and stop bits, where N=1, 2, etc. If a break character is to be transmitted then the UTXBRK 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 UTXBRK 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 UTXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out one or two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the UBNO bit is set, the word length will be set to 9 bits with the MSB being stored in the URX8 bit of the UUCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the URX/UTX pin input 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 URX/UTX 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 URX/UTX pin input is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the URX/UTX 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 URX/UTX pin input, LSB first. In the read mode, the UTXR_RXR register forms a buffer between the internal bus and the receiver shift register. The UTXR_RXR register is a two-byte deep FIFO data buffer, where two bytes can be held in the FIFO while a third byte can continue to be received. Note that the application program must ensure that the data is read from UTXR_RXR before the third byte has been completely shifted in, otherwise this third byte will be discarded and an overrun error UOERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of UBNO, UPRT and UPREN bits to define the word length, parity type.
- Setup the UBRG register to select the desired baud rate.
- Set the URXEN bit to ensure that the URX/UTX 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 URXIF bit in the UUSR register will be set when the UTXR_RXR register has data available. There will be at most one more character available before an overrun error occurs.
- When the contents of the shift register have been transferred to the UTXR_RXR register, then if the URIE bit is set, an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

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

1. A UUSR register access
2. A UTXR_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 UBNO bit plus one stop bit. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by UBNO plus one stop bit. The URXIF bit is set, UFERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the URIDLE bit is set. A break is regarded as a character that contains only zeros with the UFERR 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 UFERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until stop bits are received. It should be noted that the URIDLE 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, UFERR, will be set.
- The receive data register, UTXR_RXR, will be cleared.
- The UOERR, UNF, UPERR, URIDLE or URXIF 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 UUSR register, otherwise known as the URIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the URIDLE flag will have a high value, which indicates the receiver is in an idle condition.

Receiver Interrupt

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

Managing Receiver Errors

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

Overrun Error – UOERR

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

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

- The UOERR flag in the UUSR register will be set.
- The UTXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the URIE bit is set.

The UOERR flag can be cleared by an access to the UUSR register followed by a read to the UTXR_RXR register.

Noise Error – UNF

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, UNF, in the UUSR register will be set on the rising edge of the URXIF bit.
- Data will be transferred from the Shift register to the UTXR_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the URXIF bit which itself generates an interrupt.

Note that the UNF flag is reset by a UUSR register read operation followed by a UTXR_RXR register read operation.

Framing Error – UFERR

The read only framing error flag, UFERR, in the UUSR 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 UFERR flag will be set. The UFERR flag and the received data will be recorded in the UUSR and UTXR_RXR registers respectively, and the flag is cleared in any reset.

Parity Error – UPERR

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

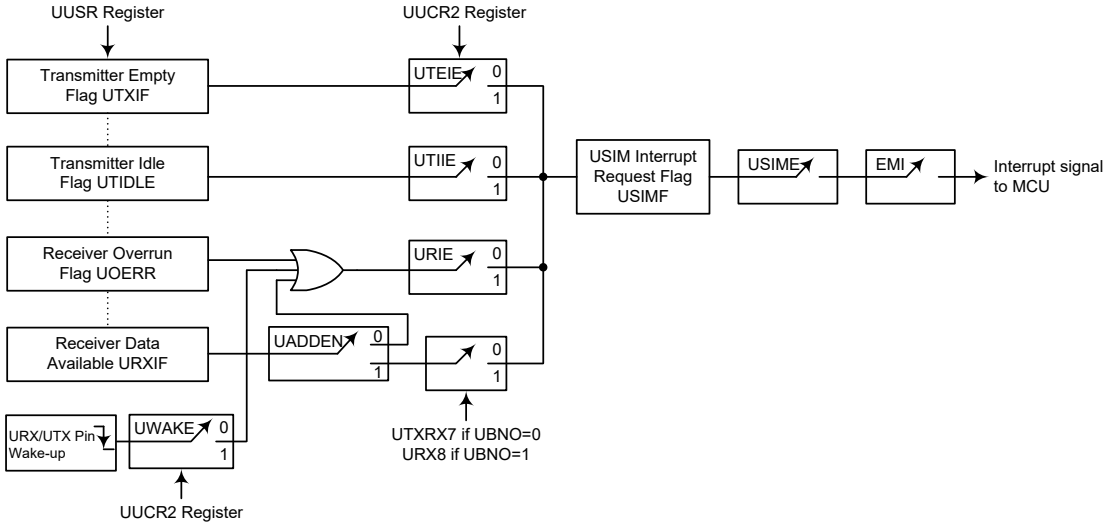
UART Interrupt Structure

Several individual UART conditions can trigger an USIM interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an URX/UTX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and the USIM 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 UUSR register flags which will generate an USIM interrupt if its associated interrupt enable control bit in the UUCR2 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 USIM UART mode interrupt sources.

The address detect condition, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt when an address detect condition occurs if its function is enabled by setting the UADDEN bit in the UUCR2 register. An URX/UTX pin wake-up, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt if the UART clock (f_H) source is switched off and the UWAKE and URIE bits in the UUCR2 register are set when a falling edge on the URX/UTX pin occurs. Note that in the event of an URX/UTX wake-up interrupt occurring, there will be a certain period of delay, commonly known as the System Start-up Time, for the oscillator to restart and stabilize before the system resumes normal operation.

Note that the UUSR 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 USIM interrupt enable control bit in the interrupt control register 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, UADDEN, in the UUCR2 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 URXIF flag. If the UADDEN 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 USIME and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if UBNO=1 or the 8th bit if UBNO=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 UADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the URXIF 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 UPREN to zero.

UADDEN	9th bit if UBNO=1 8th bit if UBNO=0	USIM Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

UADDEN 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 UUSR, UUCR1, UUCR2, UUCR3, transmit and receive registers, as well as the UBRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver URX/UTX pin wake-up function, which is enabled or disabled by the UWAKE bit in the UUCR2 register. If this bit, along with the UART mode selection bit, UMD, the UART enable bit, UREN, the receiver enable bit, URXEN and the receiver interrupt bit, URIE, are all set when the UART clock (f_H) is off, then a falling edge on the URX/UTX pin will trigger an URX/UTX 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 URX/UTX 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 USIM interrupt enable bit, USIME, must be set. If the EMI and USIME 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 USIM interrupt will not be generated until after this time has elapsed.

Low Voltage Detector – LVD

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

LVD Register

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

- **LVDC Register**

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

Bit 7~6 Unimplemented, read as “0”

Bit 5 **LVDO**: LVD Output Flag
0: No Low Voltage Detected
1: Low Voltage Detected

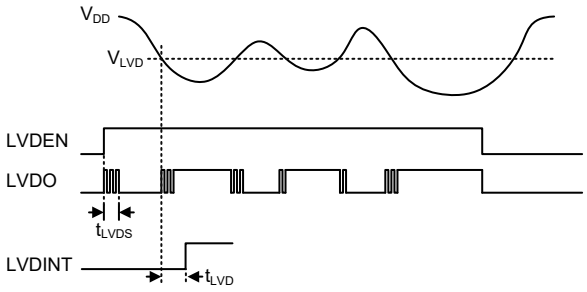
Bit 4 **LVDEN**: Low Voltage Detector Control
0: Disable
1: Enable

Bit 3 **VBGEN**: Bandgap buffer Control
0: Disable
1: Enable
Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or the VBGEN bit is set high.

Bit 2~0 **VLVD2~VLVD0**: LVD function operation condtion selection
101: 3.3V
110: 3.6V
111: 4.0V
Others: Undefined

LVD Operation

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



LVD Operation

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

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions. The external interrupt is generated by the action of the external INT0 or INT1 pin, while the internal interrupts are generated by various internal functions including TMs, OVPs, Time Bases, PWM generator, LVD, EEPROM Write operation, USIM and A/D converter, etc.

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 first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI2, MFI4~MFI8 registers which setup the Multi-function interrupts.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
External interrupt	INTnE	INTnF	n=0 or 1
Error signals (ERSG)	ERSGE	ERSGF	—
Phase protection	PHASEE	PHASEF	—
USIM	USIME	USIMF	—
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~2, 4~8
OVPn function	OVPnE	OVPnF	n=0~8
Frequency jitter function	FJTME	FJTMF	—
	FJEQUE	FJEQUF	—
Time Base	TBnE	TBnF	n=0 or 1
PWM generator	PWMPE	PWMPF	—
	PWMDE	PWMDF	—
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
10-bit TMC	TMCPE	TMCPF	—
	TMCAE	TMCAF	
Timer Module	CTMnPE	CTMnPF	n=0~2
	CTMnAE	CTMnAF	
	PTMPE	PTMPF	—
	PTMAE	PTMAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTC0	—	PHASEF	ERSGF	INT0F	PHASEE	ERSGE	INT0E	EMI
INTC1	USIMF	MF1F	MF0F	INT1F	USIME	MF1E	MF0E	INT1E
INTC2	MF4F	UNU46	ADF	MF2F	MF4E	UNU42	ADE	MF2E
INTC3	MF8F	MF7F	MF6F	MF5F	MF8E	MF7E	MF6E	MF5E
MFI0	—	OVP8F	OVP7F	OVP6F	—	OVP8E	OVP7E	OVP6E
MFI1	OVP5F	OVP4F	OVP3F	OVP2F	OVP5E	OVP4E	OVP3E	OVP2E
MFI2	—	—	FJEQUF	FJTMF	—	—	FJEQUE	FJTME
MFI4	—	TB0F	PWMDF	PWMPF	—	TB0E	PWMDE	PWMPE
MFI5	OVP1F	OVP0F	PTMAF	PTMPF	OVP1E	OVP0E	PTMAE	PTMPE
MFI6	TB1F	TMCPF	CTM0AF	CTM0PF	TB1E	TMCPE	CTM0AE	CTM0PE
MFI7	LVF	TMCAF	CTM1AF	CTM1PF	LVE	TMCAE	CTM1AE	CTM1PE
MFI8	—	DEF	CTM2AF	CTM2PF	—	DEE	CTM2AE	CTM2PE

Interrupt Register List

- **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PHASEF	ERSGF	INT0F	PHASEE	ERSGE	INT0E	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 **PHASEF**: Phase protection interrupt request flag
0: No request
1: Interrupt request

Bit 5 **ERSGF**: Error signal (ERSG) interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INT0F**: External interrupt 0 interrupt request Flag
0: No request
1: Interrupt request

Bit 3 **PHASEE**: Phase protection interrupt control
0: Disable
1: Enable

Bit 2 **ERSGE**: Error signal (ERSG) interrupt control
0: Disable
1: Enable

Bit 1 **INT0E**: External interrupt 0 interrupt control
0: Disable
1: Enable

Bit 0 **EMI**: Global interrupt control
0: Disable
1: Enable

• **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	USIMF	MF1F	MF0F	INT1F	USIME	MF1E	MF0E	INT1E
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 **USIMF**: USIM interrupt request Flag
0: No request
1: Interrupt request

Bit 6 **MF1F**: Multi-function interrupt 1 request flag
0: No request
1: Interrupt request

Bit 5 **MF0F**: Multi-function interrupt 0 request flag
0: No request
1: Interrupt request

Bit 4 **INT1F**: External interrupt 1 request flag
0: No request
1: Interrupt request

Bit 3 **USIME**: USIM interrupt control
0: Disable
1: Enable

Bit 2 **MF1E**: Multi-function interrupt 1 control
0: Disable
1: Enable

Bit 1 **MF0E**: Multi-function interrupt 0 control
0: Disable
1: Enable

Bit 0 **INT1E**: External interrupt 1 control
0: Disable
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	MF4F	UNU46	ADF	MF2F	MF4E	UNU42	ADE	MF2E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **MF4F**: Multi-function interrupt 4 request flag
0: No request
1: Interrupt request

Bit 6 **UNU46**: Reserved, this bit must be fixed at "0"

Bit 5 **ADF**: A/D converter interrupt request flag
0: No request
1: Interrupt request

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

Bit 3 **MF4E**: Multi-function interrupt 4 control
0: Disable
1: Enable

Bit 2 **UNU42**: Reserved, this bit must be fixed at "0"

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

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

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	MF8F	MF7F	MF6F	MF5F	MF8E	MF7E	MF6E	MF5E
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 **MF8F**: Multi-function interrupt 8 request flag
0: No request
1: Interrupt request

Bit 6 **MF7F**: Multi-function interrupt 7 request flag
0: No request
1: Interrupt request

Bit 5 **MF6F**: Multi-function interrupt 6 request flag
0: No request
1: Interrupt request

Bit 4 **MF5F**: Multi-function interrupt 5 request flag
0: No request
1: Interrupt request

Bit 3 **MF8E**: Multi-function interrupt 8 control
0: Disable
1: Enable

Bit 2 **MF7E**: Multi-function interrupt 7 control
0: Disable
1: Enable

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

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

• **MFI0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	OVP8F	OVP7F	OVP6F	—	OVP8E	OVP7E	OVP6E
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **OVP8F**: OVP8 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **OVP7F**: OVP7 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **OVP6F**: OVP6 interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **OVP8E**: OVP8 interrupt control
0: Disable
1: Enable

Bit 1 **OVP7E**: OVP7 interrupt control
0: Disable
1: Enable

Bit 0 **OVP6E**: OVP6 interrupt control
0: Disable
1: Enable

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	OVP5F	OVP4F	OVP3F	OVP2F	OVP5E	OVP4E	OVP3E	OVP2E
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 **OVP5F**: OVP5 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **OVP4F**: OVP4 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **OVP3F**: OVP3 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **OVP2F**: OVP2 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **OVP5E**: OVP5 interrupt control
0: Disable
1: Enable

Bit 2 **OVP4E**: OVP4 interrupt control
0: Disable
1: Enable

Bit 1 **OVP3E**: OVP3 interrupt control
0: Disable
1: Enable

Bit 0 **OVP2E**: OVP2 interrupt control
0: Disable
1: Enable

• **MFI2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	FJEQUF	FJTMF	—	—	FJEQUE	FJTME
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **FJEQUF**: FJEQU interrupt request flag
0: No request
1: Interrupt request

Bit 4 **FJTMF**: FJTMR interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **FJEQUE**: FJEQU interrupt control
0: Disable
1: Enable

Bit 0 **FJTME**: FJTMR interrupt control
0: Disable
1: Enable

• **MFI4 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TB0F	PWMDF	PWMPF	—	TB0E	PWMDE	PWMPE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"
Bit 6 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request
Bit 5 **PWMDF**: PWMD match interrupt request flag
0: No request
1: Interrupt request
Bit 4 **PWMPF**: PWMP match interrupt request flag
0: No request
1: Interrupt request
Bit 3 Unimplemented, read as "0"
Bit 2 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable
Bit 1 **PWMDE**: PWMD match interrupt control
0: Disable
1: Enable
Bit 0 **PWMPE**: PWMP match interrupt control
0: Disable
1: Enable

• **MFI5 Register**

Bit	7	6	5	4	3	2	1	0
Name	OVP1F	OVP0F	PTMAF	PTMPF	OVP1E	OVP0E	PTMAE	PTMPE
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 **OVP1F**: OVP1 interrupt request flag
0: No request
1: Interrupt request
Bit 6 **OVP0F**: OVP0 interrupt request flag
0: No request
1: Interrupt request
Bit 5 **PTMAF**: PTM Comparator A match interrupt request flag
0: No request
1: Interrupt request
Bit 4 **PTMPF**: PTM Comparator P match interrupt request flag
0: No request
1: Interrupt request
Bit 3 **OVP1E**: OVP1 interrupt control
0: Disable
1: Enable
Bit 2 **OVP0E**: OVP0 interrupt control
0: Disable
1: Enable

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

• **MFI6 Register**

Bit	7	6	5	4	3	2	1	0
Name	TB1F	TMCPF	CTM0AF	CTM0PF	TB1E	TMCPE	CTM0AE	CTM0PE
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 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TMCPF**: TMC Period interrupt request flag
0: No request
1: Interrupt request

Bit 5 **CTM0AF**: CTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTM0PF**: CTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

Bit 2 **TMCPE**: TMC Period interrupt control
0: Disable
1: Enable

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

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

• **MFI7 Register**

Bit	7	6	5	4	3	2	1	0
Name	LVF	TMCAF	CTM1AF	CTM1PF	LVE	TMCAE	CTM1AE	CTM1PE
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 **LVF**: LVD interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TMCAF**: TMCA interrupt request flag
0: No request
1: Interrupt request

Bit 5 **CTM1AF**: CTM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTM1PF**: CTM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 **LVE**: LVD interrupt control
0: Disable
1: Enable

Bit 2 **TMCAE**: TMCA interrupt control
0: Disable
1: Enable

Bit 1 **CTM1AE**: CTM1 Comparator A match interrupt control
0: Disable
1: Enable

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

• **MFI8 Register**

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

Bit 7 Unimplemented, read as "0"

Bit 6 **DEF**: EEPROM interrupt request flag
0: No request
1: Interrupt request

Bit 5 **CTM2AF**: CTM2 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTM2PF**: CTM2 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **DEE**: EEPROM interrupt control
0: Disable
1: Enable

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

Bit 0 **CTM2PE**: CTM2 Comparator P match interrupt control
0: Disable
1: Enable

Interrupt Operation

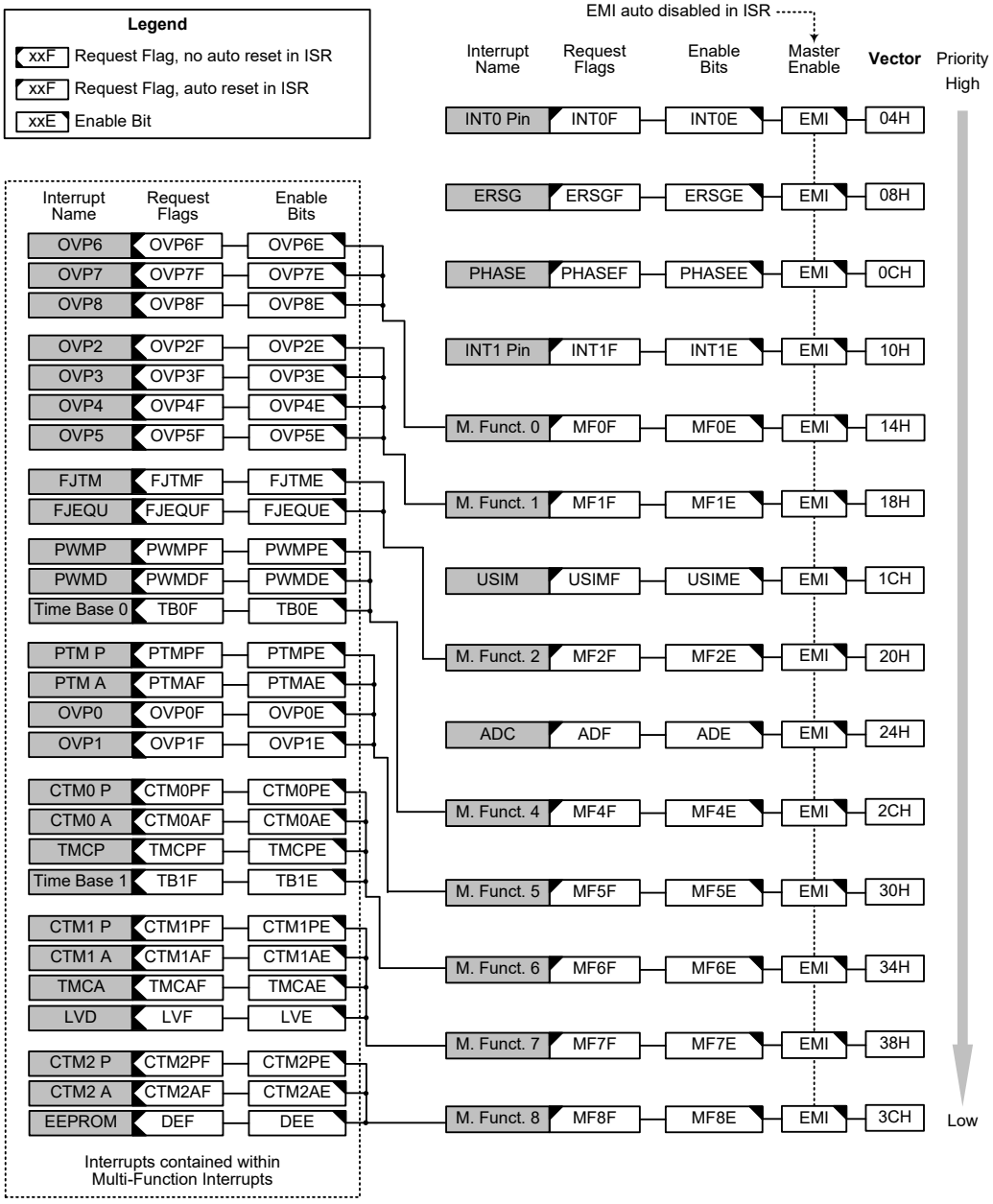
When the conditions for an interrupt event occur, such as a TM Comparator P, Comparator A match or A/D conversion completion etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector, if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a "JMP" which will jump to another section of program which is known as the interrupt service routine. Here is

located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a "RETI", which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the Accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.



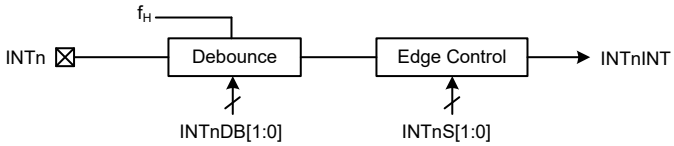
Interrupt Structure

External Interrupts

The external interrupt is controlled by signal transitions on the INTn pin. An external interrupt request will take place when the external interrupt request flag, INTnF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INTnE, 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 pin is pin-shared with the I/O pin, it can only be configured as external interrupt pin if the external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port input/output control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~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.

In order to reduce the possibility of glitches on the external interrupt line causing erroneous operations, a debounce circuit is included in each external interrupt pin. It uses the clock, f_H, to add a debounce time and the debounce time is determined by the INTnDB1 and INTnDB0 bits in the INTDB register. Note that when the INTnDB[1:0] bits are set to 01,10 or 11 to enable the corresponding pin debounce function and select a debounce time, if the f_H clock is off, then the MCU cannot be woken up through the external interrupt.



External Interrupts (n=0~1)

• INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
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: Both rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt edge control for INT0 pin
- 00: Disable
- 01: Rising edge
- 10: Falling edge
- 11: Both rising and falling edges

• **INTDB Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1DB1	INT1DB0	INT0DB1	INT0DB0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

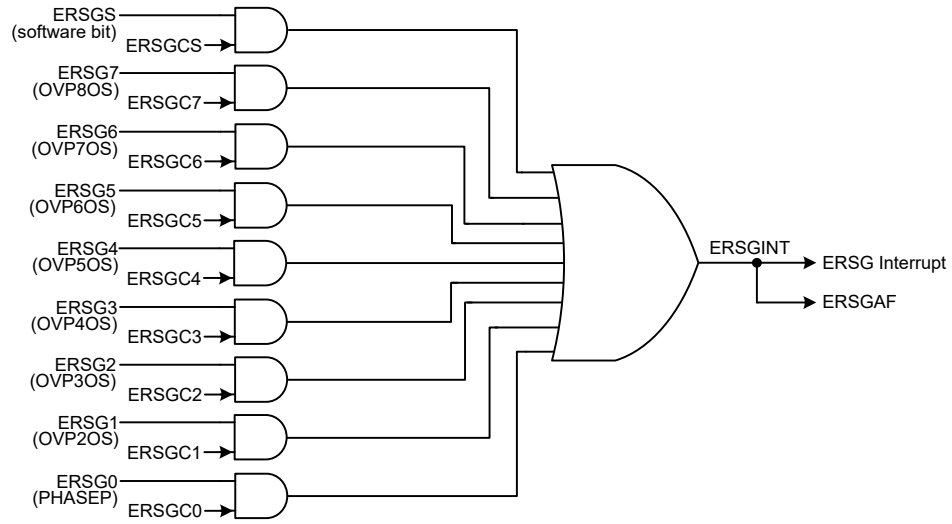
Bit 3~2 **INT1DB1~INT1DB0**: External interrupt debounce time selection for INT1 pin
00: Bypass, no debounce
01: (7~8)×t_H
10: (15~16)×t_H
11: (23~24)×t_H

Bit 1~0 **INT0DB1~INT0DB0**: External interrupt debounce time selection for INT0 pin
00: Bypass, no debounce
01: (7~8)×t_H
10: (15~16)×t_H
11: (23~24)×t_H

Note: 1. t_H=1/f_H
2. When INTnDB[1:0] bits are not equal to 00, the INTn interrupt wakeup function will be unavailable if the f_H clock is off.

Error Signal Interrupt

The Error Signal interrupt, also known as the ERSG interrupt, can be generated by totally nine trigger signals. They are the phase protection output signal (PHASEP), OVP2~OVP8 output (OVP2OS~OVP8OS) and a rising edge of the ERSCS softeware bit, as shown below.



ERSG Interrupt Structure

An ERSG interrupt will take place when the ERSG interrupt request flag, ERSGF, is set, a situation that will occur when an ERSG trigger signal occurs and the corresponding ERSGCn bit or ERSGCS is high to enable the error signal interrupt output. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and ERSG interrupt enable bit, ERSGE, must first be set. When the interrupt is enabled, the stack is not full and an error signal generates with its corresponding control bit being high, a subroutine call to the ERSG interrupt vector, will take place. When the interrupt is serviced, the ERSG interrupt request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Phase Protection Interrupt

The Phase protection interrupt is controlled by the phase protect detect 0 and phase protect detect 1 circuits. A Phase protection interrupt request will take place when the Phase protection interrupt request flag, PHASEF, is set, a situation that will occur when the detected phase 0 or phase 1 width is zero or lower than the preset value. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Phase protection interrupt enable bit, PHASEE, must first be set. When the interrupt is enabled, the stack is not full and a zero or small phase width is detected, a subroutine call to the Phase protection interrupt vector, will take place. When the interrupt is serviced, the Phase protection interrupt request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

USIM Interrupt

The Universal Serial Interface Module Interrupt, also known as the USIM interrupt, will take place when the USIM Interrupt request flag, USIMF, is set. As the USIM interface can operate in three modes which are SPI mode, I^2C mode and UART mode, the USIMF flag can be set by different conditions depending on the selected interface mode.

If the SPI or I^2C mode is selected, the USIM interrupt can be triggered when a byte of data has been received or transmitted by the USIM SPI or I^2C interface, or an I^2C slave address match occurs, or an I^2C bus time-out occurs. If the UART mode is selected, several individual UART conditions including a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up, can generate a USIM interrupt with the USIMF flag bit set high.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Universal Serial Interface Module Interrupt enable bit, USIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the respective Interrupt vector, will take place. When the interrupt is serviced, the Universal Serial Interface Module Interrupt flag, USIMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Note that if the USIM interrupt is triggered by the UART interface, after the interrupt has been serviced, the UUSR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

A/D Converter Interrupt

The A/D Converter Interrupt is controlled by the termination of a manually triggered A/D conversion or a group of automatically triggered A/D conversions. 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 finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D converter Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion 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 are eight Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the OVPn interrupts, Time base interrupts, PWM duty and period match interrupts, frequency jittering FJTIMER overflow interrupt, approach match interrupt, TM interrupts, TMC interrupts, LVD interrupt and EEPROM interrupt.

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

OVPn Interrupts

The OVPn interrupt is controlled by the Over voltage protection n function. All of the OVP interrupts are contained within the Multi-function Interrupts. An OVPn interrupt request will take place when the OVPn interrupt request flag, OVPnF, is set, a situation that will occur when an OVPn input voltage is larger than a preset voltage.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, OVPn interrupt enable bit, OVPnE, and the corresponding multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and an over voltage condition occurs, a subroutine call to the respective Multi-function Interrupt vector, will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the Multi-function interrupt request flag will be automatically cleared. As the OVPnF interrupt request flag will not be automatically cleared, it has to be cleared by the application program.

FJTMR Interrupt

The frequency jitter FJTIMER overflow Interrupt, here named as FJTMR interrupt, is contained within the Multi-function Interrupt. An FJTMR interrupt request will take place when the FJTMR interrupt request flag, FJTMF, is set, a situation that will occur when the FJTIMER counter which is used in frequency jittering function overflows.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, FJTMR interrupt enable bit, FJTME, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and the FJTIMER overflows, a subroutine call to the relevant Multi-function Interrupt vector location, will take place. When the FJTMR interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the related MFnF flag will be automatically cleared. As the FJTMF interrupt request flag will not be automatically cleared, it has to be cleared by the application program.

FJEQU Interrupt

The frequency jitter approach match interrupt, here named as FJEQU interrupt, is contained within the Multi-function Interrupt. An FJEQU interrupt will take place when the FJEQUF interrupt request flag is set, a situation that will occur when the PWMP, DT0 or DT1 approaching target A or target B in frequency jittering process has completed.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, FJEQU interrupt enable bit, FJEQUE, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and the approach match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector location, will take

place. When the FJEQU interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the related MFnF flag will be automatically cleared. As the FJEQUF interrupt request flag will not be automatically cleared, it has to be cleared by the application program.

TM Interrupts

Each Compact and Periodic TM has two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. For all of the TM types there are two interrupt request flags and two interrupt enable control bits. A TM interrupt request will take place when any of the TM request flag is set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

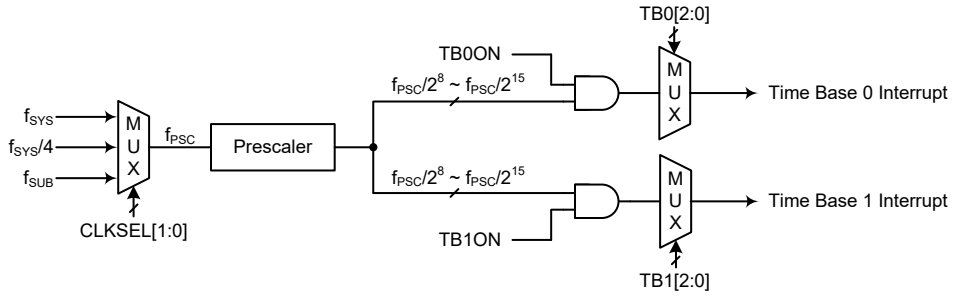
PWM Generator Interrupts

The PWM generator has two interrupts, one PWM Period match interrupt known as PWMP interrupt and one PWM Duty match interrupt known as PWMD interrupt. The PWMP and PWMD interrupts are contained in the multi-function interrupt. A PWM period or duty interrupt request will take place when the PWM period or duty interrupt request flag, PWMPF or PWMDF, is set, which occurs when the PWM Period or Duty matches. To allow the program to branch to its respectively interrupt vector address, the global interrupt enable bit, EMI, the PWM Period or Duty match interrupt enable bit, PWMPE or PWMDE, and the relevant multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and the PWM Period or Duty matches, a subroutine call to the respective Multi-function vector location will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the multi-function interrupt request flag will also be automatically reset, but the period or duty interrupt request flag, PWMDF or PWMPF, must be manually cleared by the application program.

Time Base Interrupts

The Time Base Interrupts are contained within the Multi-function Interrupt. The function of the Time Base Interrupt is to provide regular time signals in the form of an internal interrupt. It is controlled by the overflow signal from its internal timer. When this happens its interrupt request flag, TBnF, will be set. To allow the program to branch to its respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bit, TBnE, and relevant Multi-function interrupt enable bit, MFnF, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to the relevant multi-function interrupt vector location will take place. When the interrupt is serviced, the EMI bit will be cleared to disable other interrupts. However, only the related MFnF flag will be automatically cleared. As the Time Base Interrupt flag, TBnF will not be automatically cleared, it has to be cleared by the application program.

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}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL[1:0] bits in the PSCR register.



Time Base Interrupts

- **PSCR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL1	CLKSEL0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source f_{PSC} selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

- **TB0C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB0ON**: Time Base 0 Enable Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Time Base 0 time-out period selection
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

• **TB1C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	—	—	—	—	TB12	TB11	TB10
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB1ON**: Time Base 1 Enable Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB12~TB10**: Time Base 1 time-out period selection
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

TMC Interrupts

The 10-bit TMC timer/event counter has two interrupts, TMCP and TMCA interrupts. The TMCP interrupt is triggered by the 10-bit counter overflow situation and the TMCA interrupt occurs when a preset time has elapsed in Timer/Counter mode or an active edge transition is captured in Capture Input mode and then the counter value is latched into the TMCCCRA register. All of the TMC interrupts are contained within the Multi-function Interrupts. There are two interrupt request flags and two enable control bits. A TMCP or TMCA interrupt request will take place when the TMCP or TMCA request flag is set, a situation which occurs when a corresponding interrupt trigger event occurs.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TMCP or TMCA Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TMCP or TMCA interrupt trigger event occurs, a subroutine call to the relevant Multi-function Interrupt vector location, will take place. When the TMCP or TMCA interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the related MFnF flag will be automatically cleared. As the TMCPF or TMCAF interrupt request flag will not be automatically cleared, it has to be cleared by the application program.

LVD Interrupt

The Low Voltage Detector Interrupt is contained within the Multi-function Interrupt. An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the Multi-function Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the LVF flag will not be automatically cleared, it has to be cleared by the application program.

EEPROM Interrupt

The EEPROM Interrupt is contained within the Multi-function Interrupt. An EEPROM erase or write Interrupt request will take place when the EEPROM Write Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, EEPROM Write Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and an EEPROM erase or write cycle ends, a subroutine call to the relevant Multi-function Interrupt vector will take place. When the EEPROM Write Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the Multi-function interrupt request flag will be automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins, a low power supply voltage 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, MFnF, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

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

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

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

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

Application Descriptions – Half-bridge Induction Cooker

Generally half-bridge induction cookers have mainly four functions, which are power control, phase detection, protection functions as well as power measurement and display. A group of complementary PWM signals, PWM0 and PWM1, which have individually programmable dead-time insertion and integrated frequency jittering functions, is used to implement power control. In the serial resonant circuit, the current sensors and PWM signals are used for phase detection. In the process of power control, it will be necessary to detect IGBT resonant over-current, IGBT resonant short-circuit current, AC over-current and AC voltage zero-crossing. When a signal abnormality is detected, the protect function will be activated and each signal has four optional protect levels. The A/D converter which provides 2-channel automatic conversion function is included in this device for implementing the power measurement and display. This Holtek special half-bridge induction cooker MCU, HT45F0074A, can provide all the control signals to implement the above-mentioned functions.

Power Control Operating Principle

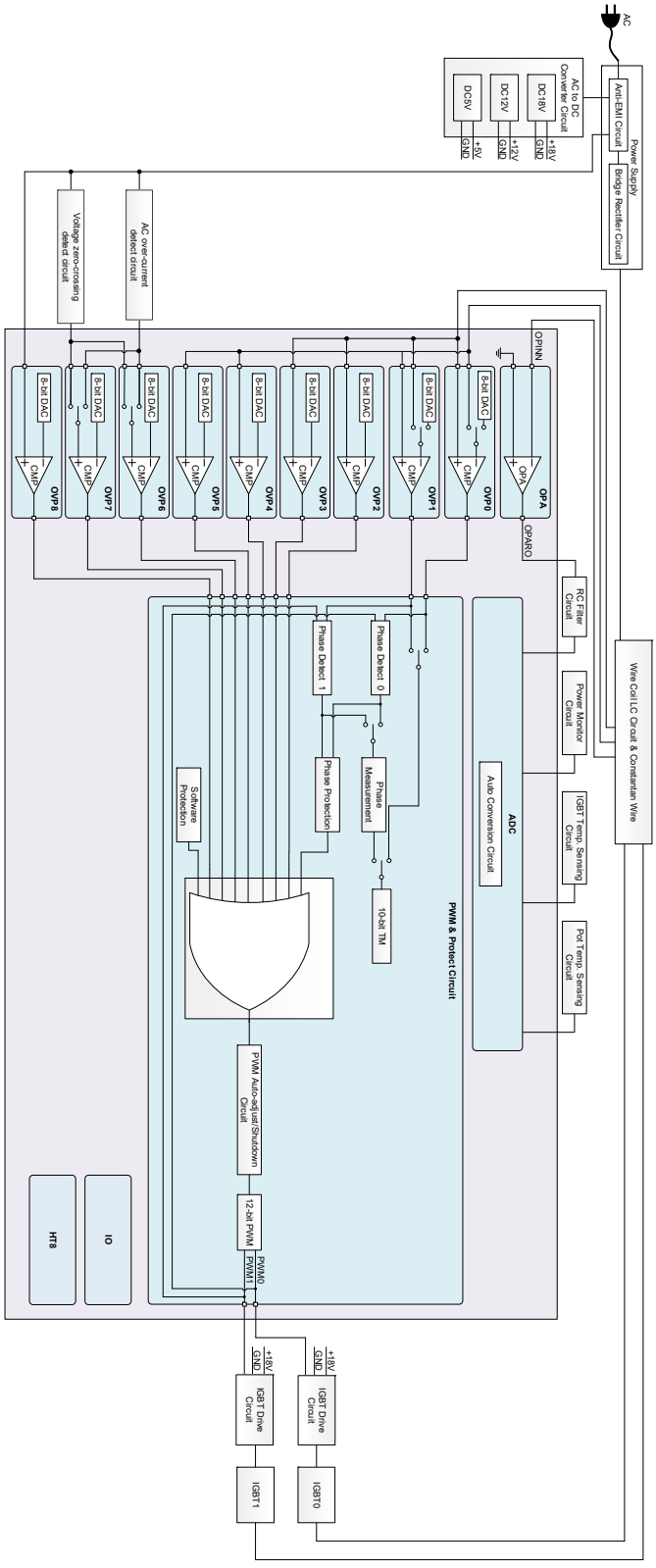
The power of half-bridge induction cookers is determined by the complementary PWM output frequency. When the PWM frequency is higher than the resonant frequency, the more the voltage phase leads the current phase, the smaller the resonant current will be and the smaller the power of the half-bridge induction cooker will be. When the PWM frequency is closer to the resonant frequency, the smaller the voltage phase leads the current phase, the larger the resonant current will be and the larger the power of the half-bridge induction cooker will be.

The deadtime control circuits can insert a deadtime into the PWM0 and PWM1 outputs. During the transition of the external driving transistors, there may be short periods of time when both the top and bottom sides are simultaneously on, which will result in a momentary short circuit. To avoid this situation, a dead time can be inserted.

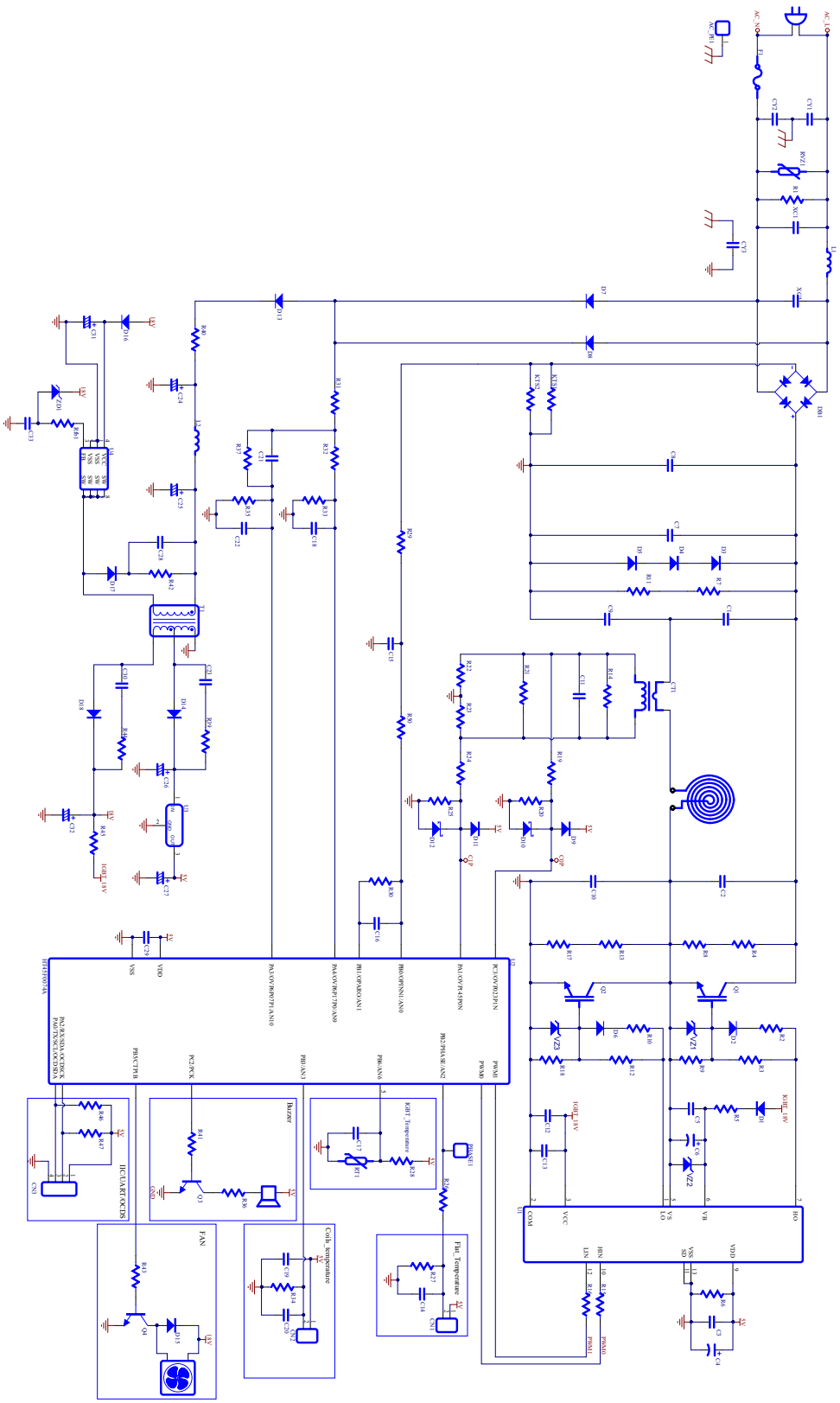
Phase Detection and Protection Operating Principle

Two groups of phase detection circuits, Phase Detect 0 and Phase Detect 1 are provided. Phase detect 0 is used to compare the PWM0 and C0X (OVP0OS) signals and output the Phase 0 signal, while Phase Detect 1 is to compare the PWM1 and C1X (OVP1OS) signals and output the Phase 1 signal. If the OVPnOS signal (current signal) phase lags behind the PWMn signal (voltage signal), the phase detect circuit will output a Phase n signal. The Phase n signal width can be used to calculate the phase difference between the current and voltage phases, so as to adjust the output power of induction cookers. When there is no phase difference between the current signal and voltage signal, the phase detect circuit output phase width will be zero, at which point the output power of half-bridge induction cookers will be the largest. If the current signal leads the voltage signal, the output phase width of the phase detection circuit cannot be detected and then the corresponding phase protect circuit will be activated. The protect circuit will execute a protection mechanism according to the settings. There are four kinds of protection mechanisms which are PWM shutdown immediately, PWM not shutdown until the current whole period is complete, PWM is adjusted in each period until it is less than the set value and will be then shutdown, PWM is adjusted in each period until it is less than the set value and will not be shutdown.

Hardware Block Diagram



Hardware Circuit



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

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

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

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

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

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

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

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

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

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

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← [m]

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← [m]

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC "OR" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − C

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

SBC A, x Subtract immediate data from ACC with Carry

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

Operation ACC ← ACC – [m] – C

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

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

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

Operation [m] ← ACC − [m] − C

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

SDZ [m] Skip if decrement Data Memory is 0

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

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

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

Operation Skip if [m]≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

ITABRD [m] Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

ITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC "XOR" [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC "XOR" [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC "XOR" x

Affected flag(s) Z

Extended Instruction Definition

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

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

LADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

LADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

LCLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

LCLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

LCPL [m] Complement Data Memory

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

Operation [m] ← [m]

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← [m]

Affected flag(s) Z

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

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

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

Affected flag(s) C

LDEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

LINC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

LMOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

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

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

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

LRL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

LRLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

LRR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

LRRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − C

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

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

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

Operation [m] ← ACC − [m] − C

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

LSDZ [m] Skip if decrement Data Memory is 0

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

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

LSET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

LSET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

LSIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

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

Operation Skip if [m] ≠ 0

Affected flag(s) None

LSUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

LSWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

LSZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

LTABRD [m] Read table (specific page) to TBLH and Data Memory

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

LITABRD [m] Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

LITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC "XOR" [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC "XOR" [m]

Affected flag(s) Z

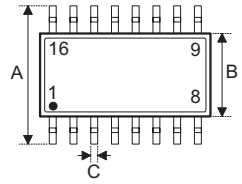
Package Information

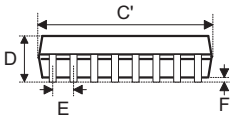
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

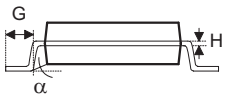
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

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

16-pin NSOP (150mil) Outline Dimensions



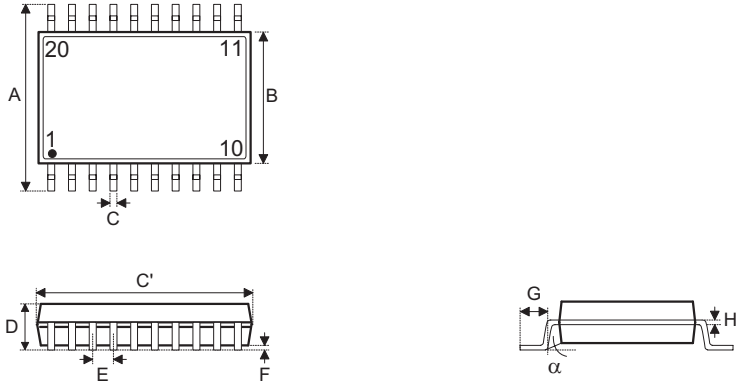




Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.012	—	0.020
C'		0.390 BSC	
D	—	—	0.069
E		0.050 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		6.000 BSC	
B		3.900 BSC	
C	0.31	—	0.51
C'		9.900 BSC	
D	—	—	1.75
E		1.270 BSC	
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

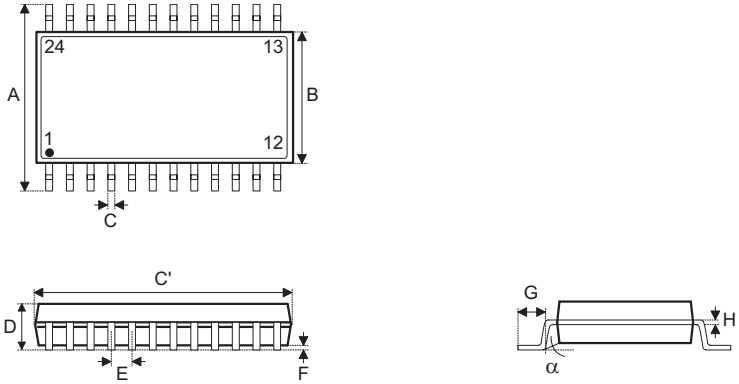
20-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.406 BSC	
B		0.295 BSC	
C	0.012	—	0.020
C’		0.504 BSC	
D	—	—	0.104
E		0.050 BSC	
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		10.30 BSC	
B		7.50 BSC	
C	0.31	—	0.51
C’		12.80 BSC	
D	—	—	2.65
E		1.27 BSC	
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

24-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.406 BSC	
B		0.295 BSC	
C	0.012	—	0.020
C'		0.606 BSC	
D	—	—	0.104
E		0.050 BSC	
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		10.30 BSC	
B		7.50 BSC	
C	0.31	—	0.51
C'		15.40 BSC	
D	—	—	2.65
E		1.27 BSC	
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°