DC Motor Driver ASSP MCU

HT45F4630

Revision: V1.00 Date: April 16, 2016

www.holtek.com

Features

CPU Features

- Operating voltage
  - V_{DD}=2.2V~5.5V
  - V_{CC1}=3V~12V
- 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:
  - External High Speed Crystal – HXT
  - Internal High Speed RC – HIRC
  - External 32.768kHz Crystal – LXT
  - Internal 32kHz RC – LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- Fully integrated internal 8MHz/12MHz/16MHz oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K×16
- RAM Data Memory: 128×8
- EEPROM Memory: 512×8
- Watchdog Timer function
- 18 bidirectional I/O lines
- Dual pin-shared external interrupts
- Multiple Timer Modules for time measurement, input capture, compare match output or PWM output or single pulse output function
- I^2C Interface Module
- Dual Time-Base functions for generation of fixed time interrupt signals
- 7-channel 12-bit resolution A/D converter
- Operational Amplifier
- High voltage driver control and high voltage driver combination
- Low voltage reset function
- Low voltage detect function
- Over Current Protection function
- High voltage power supply detection
- PWM output with HXT frequency
- Flash program memory can be re-programmed up to 100,000 times
- Flash program memory data retention > 10 years
- EEPROM data memory can be re-programmed up to 1,000,000 times
- EEPROM data memory data retention > 10 years
- Package type: 24-pin SSOP

General Description

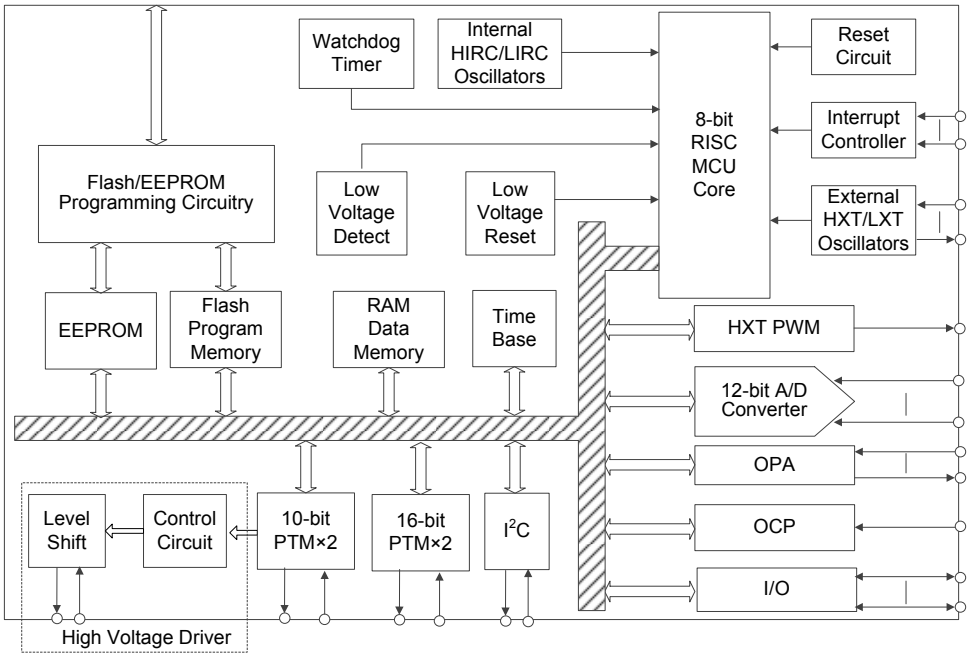
The device is a Flash Memory type 8-bit high performance RISC architecture microcontroller with integrated functions for DC motor driving applications. Offering users the convenience of Flash Memory multi-programming features, this device also includes a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 12-bit A/D converter and an operational amplifier. With regard to internal timers, the device includes multiple and extremely flexible Timer Modules providing functions for timing, pulse generation and PWM generation functions. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

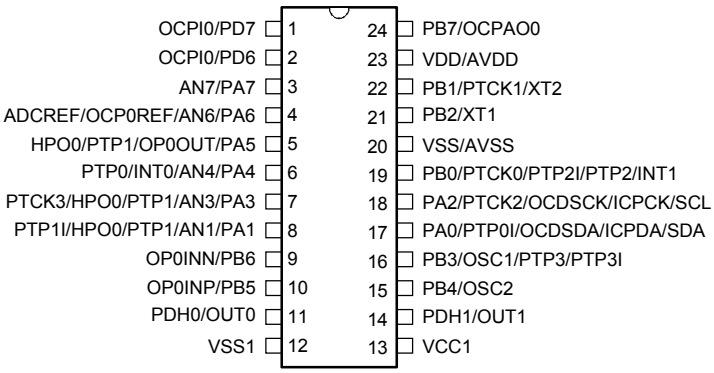
A full choice of various external and internal low and high speed oscillator functions is provided including a fully integrated system oscillator which requires no external components for its implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption. Communication with the outside world is catered for by including a fully integrated I^2C interface, this popular interface which provides designers with a means of easy communication with external peripheral hardware. Also the inclusion of flexible I/O programming features, Time-Base functions and external interrupts along with many other features further enhance device functionality and flexibility for wide range of application possibilities.

This device contains a high voltage driver control circuit and a high voltage driver combination circuit as well as high voltage power supply detection and over current protection functions which can be used in different motor driver applications.

Block Diagram



Pin Assignment



HT45F4630/HT45V4630
24SSOP-A

Note: 1. If the same pin-shared pin has multiple outputs or multiple inputs, the desired pin-shared function is determined using software control bits.

2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the HT45V4630 device which is the OCDS EV chip for the HT45F4630 device.

Pin Description

Pin Name	Function	OPT	I/T	O/T	Descriptions
PA0/PTP0I/ OCDSDA/ ICPDA/SDA	PA0	PAPU PAWU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PTP0I	PAPS0 SSCTL	ST	—	PTM0 capture input
	OCDSDA	—	ST	CMOS	OCDS address/data, for EV chip only.
	ICPDA	—	ST	CMOS	ICP address/data
	SDA	PAPS0	ST	NMOS	I²C data line
PA1/PTP1I/ HPO0/ PTP1/AN1	PA1	PAPU PAWU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PTP1I	PAPS0	ST	—	PTM1 capture input
	HPO0	PAPS0 SSCTL	—	CMOS	HXT PWM output
	PTP1	PAPS0	—	CMOS	PTM1 output
	AN1	PAPS0	AN	—	A/D Converter input channel 1
PA2/PTCK2/ OCDSCK/ ICPCK/SCL	PA2	PAPU PAWU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PTCK2	PAPS0	ST	—	PTM2 clock input
	OCDSCK	—	ST	—	OCDS clock, for EV chip only.
	ICPCK	—	ST	CMOS	ICP clock
	SCL	PAPS0	ST	NMOS	I²C clock line

Pin Name	Function	OPT	I/T	O/T	Descriptions
PA3/PTCK3/ HPO0/ PTP1/AN3	PA3	PAPU PAWU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PTCK3	PAPS0	ST	—	PTM3 clock input
	HPO0	PAPS0 SSCTL	—	CMOS	HXT PWM output
	PTP1	PAPS0	—	CMOS	PTM1 output
	AN3	PAPS0	AN	—	A/D Converter input channel 3
PA4/PTP0/INT0/ AN4	PA4	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PTP0	PAPS1	—	CMOS	PTM0 output
	INT0	PAPS1 INTEG	ST	—	External Interrupt 0 input
	AN4	PAPS1	AN	—	A/D Converter input channel 4
PA5/HPO0/ PTP1/OP0OUT	PA5	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	HPO0	PAPS1 SSCTL	—	CMOS	HXT PWM output
	PTP1	PAPS1	—	CMOS	PTM1 output
	OP0OUT	PAPS1	—	AN	Operational amplifier output
PA6/ADCREF/ OCP0REF/AN6	PA6	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	ADCREF	PAPS1	AN	—	A/D Converter reference voltage input
	OCP0REF	PAPS1	AN	—	OCP0 reference voltage input
	AN6	PAPS1	AN	—	A/D Converter input channel 6
PA7/AN7	PA7	PAPU PAWU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN7	PAPS1	AN	—	A/D Converter input channel 7
PB0/PTCK0/ PTP2I/PTP2/ INT1	PB0	PBPU PBDPS	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTCK0	PBDPS	ST	—	PTM0 clock input
	PTP2I	PBDPS	ST	—	PTM2 capture input
	PTP2	PBDPS	—	CMOS	PTM2 output
	INT1	PBDPS INTEG	ST	—	External Interrupt 1 input
PB1/PTCK1/XT2	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTCK1	PTM1C0 PTM1C1	ST	—	PTM1 clock input
	XT2	CO	—	LXT	LXT pin
PB2/XT1	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	XT1	CO	LXT	—	LXT pin
PB3/OSC1/ PTP3/PTP3I	PB3	PBPU PBDPS	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OSC1	CO	HXT	—	HXT pin
	PTP3	PBDPS	—	CMOS	PTM3 output
	PTP3I	PBDPS	ST	—	PTM3 capture input
PB4/OSC2	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OSC2	CO	—	HXT	HXT pin

Pin Name	Function	OPT	I/T	O/T	Descriptions
PB5/OP0INP	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OP0INP	PBDPS	AN	—	Operational amplifier positive input
PB6/OP0INN	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OP0INN	PBDPS	AN	—	Operational amplifier negative input
PB7/OCPAO0	PB7	PBPU PBDPS	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OCPAO0	PBDPS	—	AN	OCP operational amplifier output
PD6/OCPI0	PD6	PDPU PBDPS	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OCPI0	PBDPS SSCTL	AN	—	OCP external input
PD7/OCPI0	PD7	PDPU PBDPS	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OCPI0	PBDPS SSCTL	AN	—	OCP external input
PDH0/OUT0	PDH0	—	ST	CMOS	High Voltage Driver I/O
	OUT0	—	—	AN	High Voltage Driver Combination 0 output
PDH1/OUT1	PDH1	—	ST	CMOS	High Voltage Driver I/O
	OUT1	—	—	AN	High Voltage Driver Combination 1 output
VCC1	VCC1	—	PWR	—	High Voltage power supply
VDD/AVDD	VDD	—	PWR	—	MCU positive power supply
	AVDD	—	PWR	—	Analog positive power supply
VSS1	VSS1	—	PWR	—	High Voltage ground
VSS/AVSS	VSS	—	PWR	—	Ground
	AVSS	—	PWR	—	Analog ground

Legend: I/T: Input type; O/T: Output type;
OPT: Optional by configuration option (CO) or register option; PWR: Power;
ST: Schmitt Trigger input; CMOS: CMOS output; NMOS: NMOS output;
AN: Analog signal; CO: Configuration option;
HXT: High frequency crystal oscillator;
LXT: Low frequency crystal oscillator.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}−0.3V to V_{SS}+6.0V
Input Voltage ..... V_{SS}−0.3V to V_{DD}+0.3V
Storage Temperature ..... -50˚C to 125˚C
Operating Temperature ..... -40˚C to 85˚C
I_{OL} Total 80mA
I_{OH} Total ..... -80mA
Total Power Dissipation 500mW

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

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (HXT, HIRC)	—	f_{SYS}=8MHz	2.2	—	5.5	V
			f_{SYS}=12MHz	2.7	—	5.5	
			f_{SYS}=16MHz	4.5	—	5.5	
	Operating Voltage (LXT)		f_{SYS}=f_{LXT}=32.768kHz	2.2	—	5.5	
I_{DD}	Operating Current	3V	No load, ADC off, WDT enable f_{SYS}=f_H=8MHz, f_S=f_{SUB}=f_{LIRC}	—	1.2	2.0	mA
		5V		—	2.8	4.5	
		3V	No load, ADC off, WDT enable f_{SYS}=f_H=12MHz, f_S=f_{SUB}=f_{LIRC}	—	1.8	3.0	
		5V		—	4.0	6.0	
		5V	No load, ADC off, WDT enable f_{SYS}=f_H=16MHz, f_S=f_{SUB}=f_{LIRC}	—	4.5	7.0	
I_{STB}	Standby Current (IDLE0 Mode)	3V	No load, MCU Power Down, ADC off, WDT enable, f_{SYS}=12MHz off, f_S=f_{SUB}=f_{LIRC}	—	1.3	3.0	μA
		5V		—	2.2	5.0	
	Standby Current (IDLE1 Mode)	3V	No load, MCU Power Down, ADC off, WDT enable, f_{SYS}=f_H=12MHz on, f_S=f_{SUB}=f_{LIRC}	—	0.6	1.0	mA
		5V		—	1.2	2.0	
V_{IL}	Input Low Voltage for I/O Ports	—	—	0	—	0.3V_{DD}	V
V_{IH}	Input High Voltage for I/O Ports	—	—	0.7V_{DD}	—	V_{DD}	V
I_{OL}	Sink Current for I/O Ports	3V	V_{OL}=0.1V_{DD}	21	45	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Ports	3V	V_{OH}=0.9V_{DD}	-5	-10	—	mA
		5V		-7	-15	—	
R_{PH}	Pull-high Resistance for I/O Ports	5V	—	10	30	50	kΩ
T_{PRT}	Thermal Protection temperature	3V	PDHn (n=0~1) no load	135	155	175	°C
		5V		50	70	90	

A.C. Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock	4.5V~5.5V	—	32	—	16000	kHz
f_{HIRC}	High Speed Internal RC Oscillator (HIRC)	—	Ta= -40°C~85°C	-12%	8	+4%	MHz
			Ta= -20°C~85°C	-9%	8	+4%	
			Ta=25°C	-2%	8	+2%	
			Ta= -40°C~85°C	-12%	12	+4%	
			Ta= -20°C~85°C	-9%	12	+4%	
			Ta=25°C	-2%	12	+2%	
			Ta= -40°C~85°C	-12%	16	+4%	
			Ta= -20°C~85°C	-9%	16	+4%	
			Ta=25°C	-2%	16	+2%	
f_{HXT}	High Speed External Crystal Oscillator (HXT)	3V~5.5V	—	0.4	—	4	MHz
		4.5V~5.5V	—	0.4	—	8	
		4.5V~5.5V	—	0.4	—	12	
		4.5V~5.5V	—	0.4	—	16	
t_{RSTD}	System Reset Delay Time (Power On Reset)	—	—	25	50	100	ms
	System Reset Delay Time (Any Reset except Power On Reset)	2.2V~5.5V	—	8.3	16.7	33.3	ms
t_{SST}	System Start-up Timer Period (Wake-up from Power Down Mode and f_{SYS} Off)	—	$f_{SYS}=f_{LXT}$	1024	—	—	t_{LXT}
		—	$f_{SYS}=f_{HXT}$ ~ f_{HXT} / 64	128	—	—	t_{HXT}
		—	$f_{SYS}=f_{HIRC}$ ~ f_{HIRC} / 64	16	—	—	t_{HIRC}
		—	$f_{SYS}=f_{LIRC}$	2	—	—	t_{LIRC}
	System Start-up Timer Period (Slow Mode ↔ Normal Mode, or $f_H=f_{HIRC}$ ↔ f_{HXT}, or $f_{SUB}=f_{LIRC}$ ↔ f_{LXT})	—	f_{HXT} off → on (HTO=1)	1024	—	—	t_{HXT}
		—	f_{HIRC} off → on (HTO=1)	16	—	—	t_{HIRC}
		—	f_{LXT} off → on (LTO=1)	1024	—	—	t_{LXT}
	System Start-up Timer Period (Wake-up from Power Down Mode and f_{SYS} On)	—	$f_{SYS}=f_H$~f_H/64, $f_H=f_{HXT}$ or f_{HIRC}	2	—	—	t_H
		—	$f_{SYS}=f_{LXT}$ or f_{LIRC}	2	—	—	t_{SUB}
	System Start-up Timer Period (WDT Time-out Hardware Cold Reset)	—	—	0	—	—	t_H
t_{EERD}	EEPROM Read Time	—	—	—	45	90	μs
t_{EEWR}	EEPROM Write Time	—	—	—	2	4	ms

LXT Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	LXT Operating Voltage	—	$f_{SYS}=f_{LXT}=32.768kHz$	2.2	—	5.5	V
f_{LXT}	Low Speed External Crystal Oscillator (LXT)	2.2V~5.5V	$f_{SYS}=f_{LXT}=32.768kHz$	—	32768	—	Hz
I_{LXT}	Additional Current for LXT Enable	3V	LXTSP=0	—	—	2	μA
			LXTSP=1	—	—	3	
		5V	LXTSP=0	—	—	2	
			LXTSP=1	—	—	3	
t_{START}	LXT Start Up Timer	3V	—	—	—	1000	ms
		5V	—	—	—	1000	
Duty Cycle	Duty Cycle	—	—	40	—	60	%
R_{NEG}	Negative Resistance (Note)	2.2V	—	3×ESR	—	—	Ω

Note: C1, C2 and R_P are external components. C1=C2=10pF. R_P=10MΩ. C_L=7pF, ESR=30kΩ.

LVD & LVR Electrical Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	-5%	2.1	+5%	V
			LVR enable, voltage select 2.55V		2.55		
			LVR enable, voltage select 3.15V		3.15		
			LVR enable, voltage select 3.8V		3.8		
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 2.0V	-5%	2.0	+5%	V
			LVD enable, voltage select 2.2V		2.2		
			LVD enable, voltage select 2.4V		2.4		
			LVD enable, voltage select 2.7V		2.7		
			LVD enable, voltage select 3.0V		3.0		
			LVD enable, voltage select 3.3V		3.3		
			LVD enable, voltage select 3.6V		3.6		
			LVD enable, voltage select 4.0V		4.0		
I_{LVR}	Low Voltage Reset Current	—	LVR enable, ENLVD=0	—	60	90	μA
I_{LVD}	Low Voltage Detection Current	—	LVR disable, ENLVD=1	—	75	120	μA
			LVR enable, ENLVD=1	—	90	150	
t_{LVDS}	LVDO Stable Time	—	LVR enable, VBGEN=0, LVD off → on	—	—	15	μs
		—	LVR disable, VBGEN=0, LVD off → on	—	—	150	

Bandgap Reference Voltage Characteristics – V_{BG}

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{BG}	Bandgap Reference Voltage	—	$t_{ADCK}=f_{SYS}/8$	-5%	1.04	+5%	V
I_{BG}	Additional Current for Bandgap Reference Enable	—	LVR disable, LVD disable	—	200	300	μA
t_{BGS}	V_{BG} Turn On Stable Time	—	No load	—	—	150	μs

Note: The Bandgap reference voltage is used as the A/D converter internal signal input.

A/D Converter Electrical Characteristics

Ta= -40°C~85°C, typical is 25°C unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
AV_{DD}	A/D Converter Operating Voltage	—	$V_{REF}=AV_{DD}$	2.7	—	5.5	V
V_{AD}	A/D Converter Input Voltage	—	—	0	—	AV_{DD}/ V_{REF}	V
V_{REF}	A/D Converter Reference Voltage	3V	—	2.0	—	AV_{DD}	V
		5V	—	2.0	—	AV_{DD}	
DNL	Differential Non-linearity	3V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=0.5μs, 10-bit	-3	—	+2	LSB
		5V					
		3V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=10μs, 10-bit				
		5V					
		3V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=0.5μs, 12-bit	-4	—	+3	
		5V					
		3V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=10μs, 12-bit				
		5V					
INL	Integral Non-linearity	3V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=0.5μs, 10-bit	-4	—	+4	LSB
		5V					
		3V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=0.5μs, 12-bit	-4	—	+7	
		5V					
		3V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=10μs	-4	—	+4	
		5V					
I_{ADC}	Additional Current for A/D Converter Enable	3V	No load, t_{ADCK} =0.5μs	—	1.0	2.0	mA
		5V		—	1.5	3.0	
t_{ADCK}	A/D Clock Period	2.7V~5.5V	10-bit	0.5	—	10	μs
			12-bit	0.5	—	10	
t_{ON2ST}	A/D Converter On-to-Start Time	2.7V~5.5V	—	4	—	—	μs
t_{ADS}	A/D Sampling Time	2.7V~5.5V	12-bit A/D Converter	4			t_{ADCK}
t_{ADC}	A/D Conversion Time (Include A/D Sample and Hold Time)	2.7V~5.5V	12-bit A/D Converter	16	—	20	t_{ADCK}

Operational Amplifier Electrical Characteristics

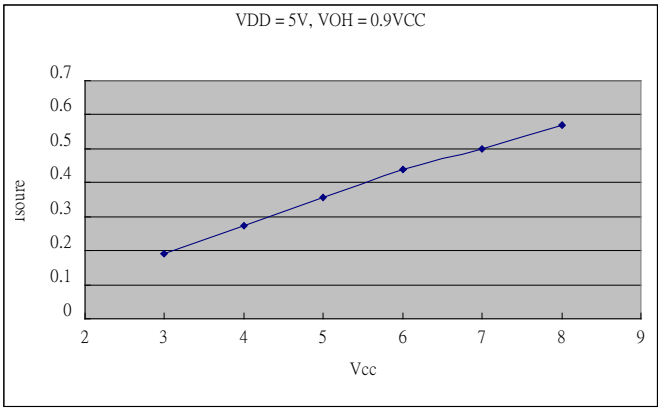
Ta= -40°C~85°C, unless otherwise specified

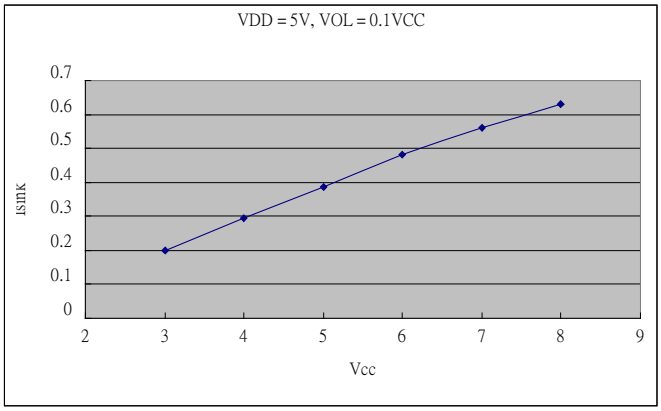
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DDO}	Operational Amplifier Operating Voltage	—	—	2.2	—	5.5	V
I_{OPA}	Operational Amplifier Operating Current	—	OP0BW=00B, no load	—	1.2	3.5	μA
			OP0BW=01B, no load	—	10	16	
			OP0BW=10B, no load	—	80	128	
			OP0BW=11B, no load	—	200	320	
V_{OS}	Operational Amplifier Input Offset Voltage	5V	Without calibration (O0OF[5:0] =100000B)	-15	—	+15	mV
		5V	With calibration	-1	—	+1	
V_{CM}	Operational Amplifier Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}-1.4V	V
I_{OS}	Operational Amplifier Input Offset Current	5V	V_{CM}=1/2V_{DD}	—	1	—	nA
PSRR	Power Supply Rejection Ratio	5V	—	58	70	—	dB
CMRR	Common Mode Rejection Ratio	5V	—	58	80	—	dB
A_{OL}	Open Loop Gain	—	—	60	80	—	dB
SR	Slew Rate +, Slew Rate -	5V	R_L=1MΩ, C_L=60pF OP0BW[1:0]=00B	1.0	1.5	—	V/ms
			R_L=1MΩ, C_L=60pF OP0BW[1:0]=01B	10	15	—	
			R_L=1MΩ, C_L=60pF OP0BW[1:0]=10B	300	500	—	
			R_L=1MΩ, C_L=60pF OP0BW[1:0] =11B	1200	1800	—	
GBW	Operational Amplifier Gain Band Bandwidth	5V	R_L=1MΩ, C_L=100pF OP0BW[1:0]=00B	2.5	5	—	kHz
			R_L=1MΩ, C_L=100pF OP0BW[1:0]=01B	20	40	—	
			R_L=1MΩ, C_L=100pF OP0BW[1:0]=10B	400	600	—	
			R_L=1MΩ, C_L=100pF OP0BW[1:0] =11B	1300	2000	—	

Level Shifter Electrical Characteristics

Ta=-40°C~85°C, V_{CC1}=6V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{SOURCE}	OUT0~OUT1 Output Source Current	5V	V_{OH}=0.9V_{CC1}	—	-450	—	mA
I_{SINK}	OUT0~OUT1 Output Sink Current	5V	V_{OL}=0.1V_{CC1}	—	450	—	mA
V_{HIH}	PDH0~PDH1 Input High Voltage	5V	—	0.7V_{CC1}	—	V_{CC1}	V
V_{HIL}	PDH0~PDH1 Input Low Voltage	5V	—	0	—	0.3V_{CC1}	V





Voltage Detector Electrical Characteristics

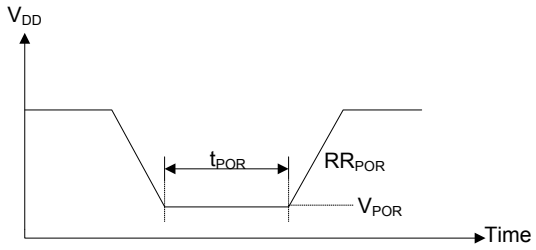
Ta= -40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DET}	V_{CC1} Detect Level	—	V_{CC1}=3V~12V $t_{ADCK}=f_{SYS}/8$	- 0.5%	$0.2V_{CC1}$	+ 0.5%	V

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

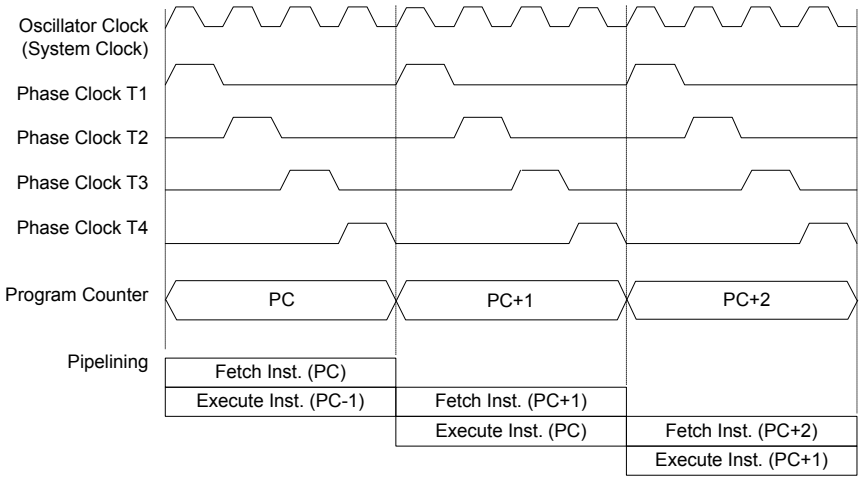


System Architecture

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

Clocking and Pipelining

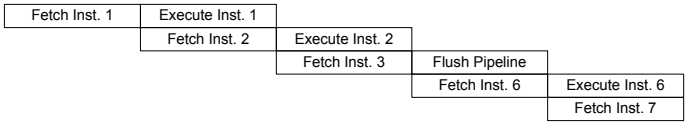
The main system clock, derived from either a HXT, LXT, HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry 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)**
PC10~PC8	PCL7~PCL0

Program Counter

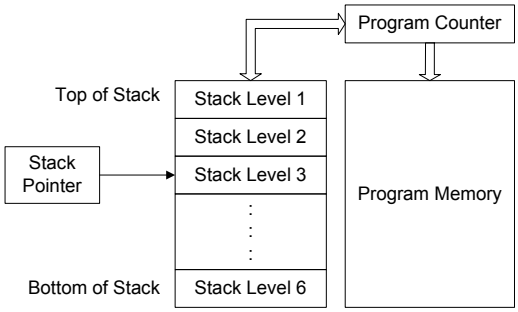
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 6 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



Arithmetic and Logic Unit – ALU

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

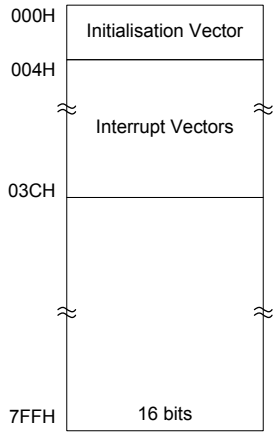
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement INCA, INC, DECA, DEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

Flash Program Memory

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

Structure

The Program Memory has a capacity of 2K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

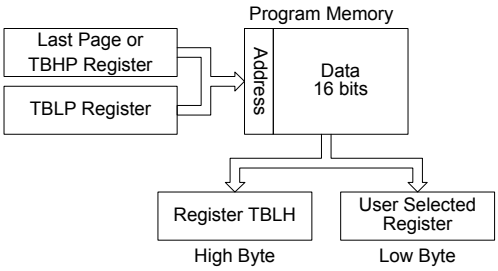


Table Program Example

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 "0700H" which refers to the start address of the last page within the 2K Program Memory of the microcontroller. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the page that TBHP pointed if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRDL [m]" instruction is executed.

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

Table Read Program Example

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

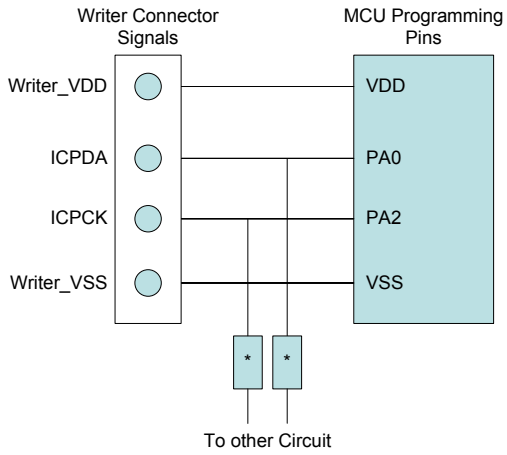
In Circuit Programming – ICP

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

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

The Program Memory and EEPROM data Memory can both 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 control 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 1k or the capacitance of * must be less than 1nF.

On Chip Debug Support – OCDS

There is an EV chip named HT45V4630 which is used to emulate the HT45F4630 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

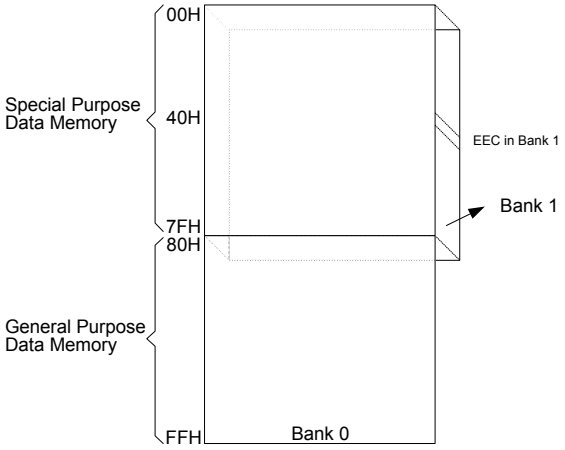
Data Memory

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

Structure

Divided into two areas, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The overall Data Memory is subdivided into two banks. The Special Purpose Data Memory registers are accessible in all banks, with the exception of the EEC register at address 40H, which is only accessible in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory for the device is the address 00H.



Data Memory Structure

General Purpose Data Memory

There are 128 bytes of general purpose data memory which are arranged in 80H~FFH of Bank 0. 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".

	Bank0	Bank1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1	
04H	BP	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH	SMOD	
0CH	TBC	
0DH	LVDC	
0EH	CTRL	
0FH	WDTC	
10H	PSS	
11H	PMSL	
12H	Unused	
13H	PDHCL	
14H	Unused	
15H	DTS0	
16H	DTS1	
17H	Unused	
18H		
19H		
1AH		
1BH	POSL	
1CH	Unused	
1DH	PRTL	
1EH	Unused	
1FH	OPCL	
20H	Unused	
21H	HVC	
22H	OCPC00	
23H	OCPC10	
24H	OCPDA0	
25H	OCPOCAL0	
26H	OCPCCAL0	
27H	Unused	
28H		
29H		
2AH		
2BH		
2CH	INTEG	
2DH	INTC0	
2EH	INTC1	
2FH	INTC2	
30H	INTC3	
31H	MFI0	
32H	MFI1	
33H	MFI2	
34H	MFI3	
35H	MFI4	
36H	LVRC	
37H	Unused	
38H	PTM2C0	
39H	PTM2C1	
3AH	PTM2DL	
3BH	PTM2DH	
3CH	PTM2AL	
3DH	PTM2AH	
3EH	PTM2RPL	
3FH	PTM2RPH	

	Bank0	Bank1
40H	Unused	EEC
41H	Unused	
42H	EED	
43H	EEA0	
44H	EEA1	
45H	Unused	
46H		
47H		
48H		
49H	PTM0C0	
4AH	PTM0C1	
4BH	PTM0DL	
4CH	PTM0DH	
4DH	PTM0AL	
4EH	PTM0AH	
4FH	PTM0RPL	
50H	PTM0RPH	
51H	PTM1C0	
52H	PTM1C1	
53H	PTM1DL	
54H	PTM1DH	
55H	PTM1AL	
56H	PTM1AH	
57H	PTM1RPL	
58H	PTM1RPH	
59H	SADC0	
5AH	SADC1	
5BH	SADOL	
5CH	SADOH	
5DH	OP0C	
5EH	OP0VOS	
5FH	IICC0	
60H	IICC1	
61H	IICD	
62H	IICA	
63H	I2CTOC	
64H	PAPS0	
65H	PAPS1	
66H	PBDPS	
67H	PAWU	
68H	PAPU	
69H	PA	
6AH	PAC	
6BH	PBPU	
6CH	PB	
6DH	PBC	
6EH	PDPU	
6FH	PD	
70H	PDC	
71H	PTM3C0	
72H	PTM3C1	
73H	PTM3DL	
74H	PTM3DH	
75H	PTM3AL	
76H	PTM3AH	
77H	PTM3RPL	
78H	PTM3RPH	
79H	Unused	
7AH	RPDH	
7BH	SSCTL	
7CH	Unused	
7DH		
7EH		
7FH		

: Unused, read as 00H

Special Purpose Data Memory

Special Function Register Description

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

Indirect Addressing Register – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data from Bank 0 while the IAR1 and MP1 register pair can access data from any bank. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of "00H" and writing to the registers indirectly will result in no operation.

Memory Pointers – MP0, MP1

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

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

Indirect Addressing Program Example

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

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

Bank Pointer – BP

For this device, the Data Memory is divided into two banks, Bank0 and Bank1. Selecting the required Data Memory area is achieved using the Bank Pointer. Bit 0 of the Bank Pointer is used to select Data Memory Banks 0~1.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the Power Down Mode, in which case, the Data Memory bank remains unaffected. It should be noted that the Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within any bank. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from Bank1 must be implemented using Indirect Addressing.

BP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	DMBP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **DMBP0**: Select Data Memory Banks
0: Bank 0
1: Bank 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 SZ, CZ, PDF and TO 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 SZ, CZ, PDF or TO 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.

- SC is the result of the "XOR" operation which is performed by the OV flag and the MSB of the current instruction operation result.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.
- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- PDF is cleared by a system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction.
- TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.

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

STATUS Register

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

"x": unknown

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

Bit 6 **CZ**: The operational result of different flags for different instructions.

For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.

For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation zero flag.

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

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

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

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

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

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

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

C is also affected by a rotate through carry instruction.

EEPROM Data memory

This device contains an area of internal EEPROM Data Memory. EEPROM, which stands for Electrically Erasable Programmable Read Only Memory, 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 are carried out in single byte operations using two address registers and a data register in Bank 0 and a single control register in Bank 1.

EEPROM Registers

Four registers control the overall operation of the internal EEPROM Data Memory. These are the address registers, EEA0 and EEA1, the data register, EED and a single control register, EEC. As both the EEA0, EEA1 and EED registers are located in Bank 0, they can be directly accessed in the same was as any other Special Function Register. The EEC register however, being located in Bank1, cannot be directly addressed directly and can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, 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
EEA0	D7	D6	D5	D4	D3	D2	D1	D0
EEA1	—	—	—	—	—	—	—	D8
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM Registers List

EEA0 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 address
Data EEPROM address bit 7 ~ bit 0

EEA1 Register

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **D8**: Data EEPROM address
Data EEPROM address bit 8

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
Data EEPROM data bit 7 ~ bit 0

EEC Register

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

Bit 7~4 Unimplemented, read as "0"

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.

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

Bit 1 **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**: EEPROM Read Control
0: Read cycle has finished
1: Activate a read cycle
This is the Data EEPROM Read Control Bit and when set high by the application program will activate a read cycle. This bit will be automatically reset to zero by the 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: The 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.

Reading Data from the EEPROM

To read data from the EEPROM, the read enable bit, RDEN, in the EEC register must first be set high to enable the read function. The EEPROM address of the data to be read must then be placed in the EEA1/EEA0 register. If the RD bit in the EEC register is now set high, a read cycle will be initiated. Setting the RD bit high will not initiate a read operation if the RDEN bit has not been set. When the read cycle terminates, the RD bit will be automatically cleared to zero, after which the 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.

Writing Data to the EEPROM

To write data to the EEPROM, the EEPROM address of the data to be written must first be placed in the EEA1/EEA0 register and the data placed in the EED register. Then 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 consecutively. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set again after the write cycle has started. Note that setting the WR bit high will not initiate a write cycle if the WREN bit has not been 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 will 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.

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 Bank Pointer, BP, will be reset to zero, which means that Data Memory Bank 0 will be selected. As the EEPROM control register is located in Bank 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 write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the EPWE bit in the relevant interrupt register. When an EEPROM write cycle ends, the EPWF request flag will be set. If the EEPROM interrupt is 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 request flag, EPWF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. 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 Bank Pointer could be normally cleared to zero as this would inhibit access to Bank 1 where the EEPROM control register exist. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process.

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

Programming Examples

- **Reading data from the EEPROM – polling method**

```
MOV  A, EEPROM_ADRES_L    ; user defined address low byte
MOV  EEA0, A
MOV  A, EEPROM_ADRES_H    ; user defined address high byte
MOV  EEA1, A
MOV  A, 040H              ; setup memory pointer MP1
MOV  MP1, A               ; MP1 points to EEC register
MOV  A, 01H               ; setup Bank Pointer
MOV  BP, 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/write
CLR  BP
MOV  A, EED               ; move read data to register
MOV  READ_DATA, A
```

- **Writing Data to the EEPROM – polling method**

```
MOV  A, EEPROM_ADRES_L    ; user defined address low byte
MOV  EEA0, A
MOV  A, EEPROM_ADRES_H    ; user defined address high byte
MOV  EEA1, A
MOV  A, EEPROM_DATA       ; user defined data
MOV  EED, A
MOV  A, 040H              ; setup memory pointer MP1
MOV  MP1, A               ; MP1 points to EEC register
MOV  A, 01H               ; setup Bank Pointer
MOV  BP, A
CLR  EMI
SET  IAR1.3               ; set WREN bit, enable write operations
SET  IAR1.2               ; start Write Cycle - set WR bit
SET  EMI
BACK:
SZ   IAR1.2               ; check for write cycle end
JMP  BACK
CLR  IAR1                 ; disable EEPROM read/write
CLR  BP
```

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 a combination of configuration options and 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. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

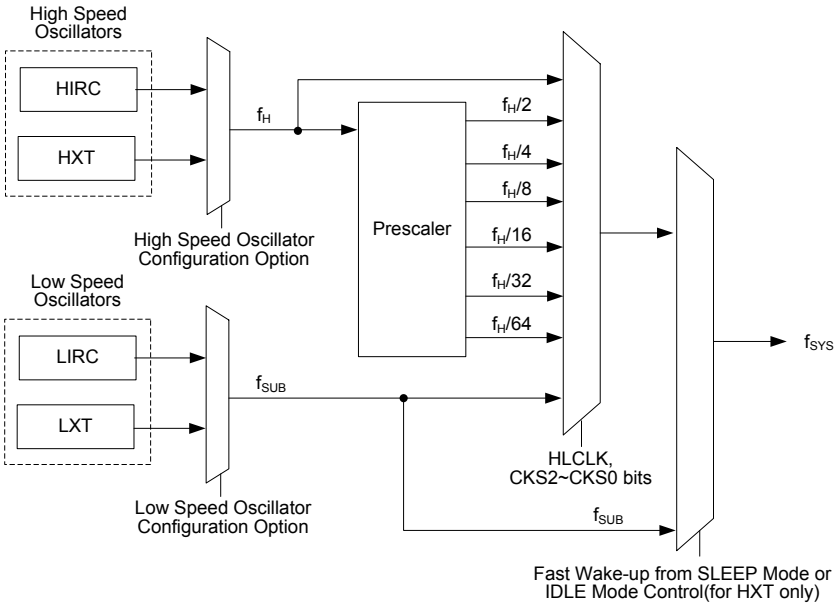
Type	Name	Frequency	Pins
External High Speed Crystal	HXT	400kHz~16MHz	OSC1/OSC2
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
Internal High Speed RC	HIRC	8/12/16MHz	—
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

System Clock Configurations

There are four methods of generating the system clock, two high speed oscillators and two low speed oscillators. The high speed oscillators are the external crystal/ceramic oscillator - HXT and the internal 8MHz, 12MHz and 16MHz RC oscillator - HIRC. The two low speed oscillators are the internal 32kHz RC oscillator - LIRC and the external 32.768kHz crystal oscillator - LXT. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and CKS2~CKS0 bits in the SMOD register and as the system clock can be dynamically selected.

The actual source clock used for each of the high speed and low speed oscillators is chosen via configuration options. The frequency of the slow speed or high speed system clock is also determined using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator. The OSC1 and OSC2 pins are used to connect the external components for the external crystal.

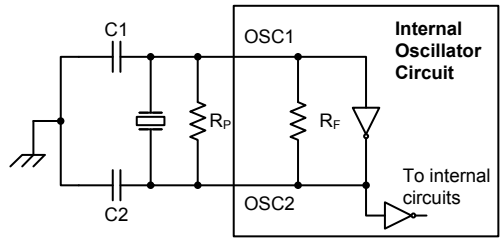


System Clock Configurations

External Crystal/Ceramic Oscillator – HXT

The External Crystal/Ceramic System Oscillator is one of the high frequency oscillator choices, which is selected via configuration option. For most crystal oscillator configurations, the simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, without requiring external capacitors. However, for some crystal types and frequencies, to ensure oscillation, it may be necessary to add two small value capacitors, C1 and C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.

For oscillator stability and to minimise the effects of noise and crosstalk, it is important to ensure that the crystal and any associated resistors and capacitors along with interconnecting lines are all located as close to the MCU as possible.



Note: 1. R_P is normally not required. C1 and C2 are required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

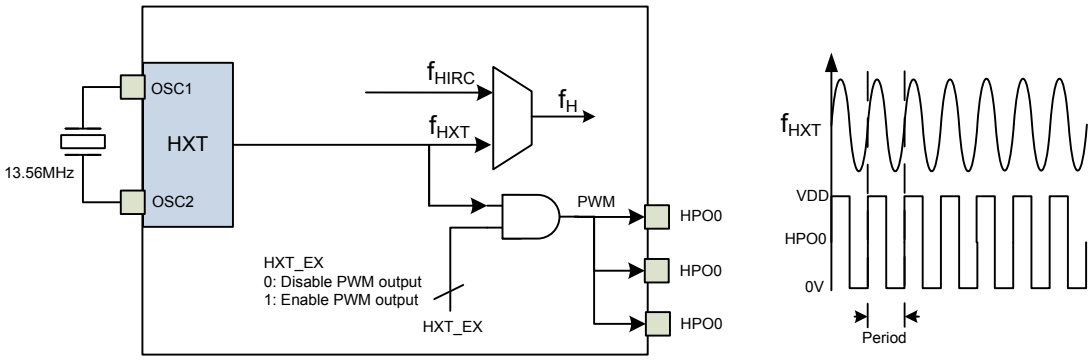
Crystal/Resonator Oscillator – HXT

HXT Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
16MHz	0pF	0pF
12MHz	0pF	0pF
8MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

HXT PWM Output

In order to enhance the current output capability, the device can generate a PWM signal with the same frequency of the external HXT oscillator, and this signal is output through multiple HPO0 pins. This function is controlled by the HXT_EX bit in the SSCTL register to switch on or off.



HXT_EX Bit	HXT PWM Output
0	Disable
1	Enable

As the HXT PWM signal can be output via multiple HPO0 pins and the HPO0 pins are pin-shared with I/O pins or other functions, the HPO0 output function must first be properly setup using the corresponding bits in the PAPS0 and PAPS1 pin-shared function selection registers before the HXT PWM function is enabled, to ensure the signal can be normally output.

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 8MHz, 12MHz and 16MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that if this internal system clock option is selected, as it requires no external pins for its operation.

External 32.768kHz Crystal Oscillator – LXT

The External 32.768kHz Crystal System Oscillator is one of the low frequency oscillator choices, which is selected via configuration option. This clock source has a fixed frequency of 32.768kHz and requires a 32.768kHz crystal to be connected between pins XT1 and XT2. The external resistor and capacitor components connected to the 32.768kHz crystal are necessary to provide oscillation. For applications where precise frequencies are essential, these components may be required to provide frequency compensation due to different crystal manufacturing tolerances. During power-up there is a time delay associated with the LXT oscillator waiting for it to start-up.

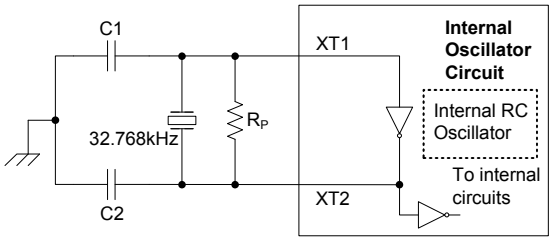
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE Mode. To do this, another clock, independent of the system clock, must be provided.

However, for some crystals, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer specification. The external parallel feedback resistor, R_P, is required.

Some configuration options determine if the XT1/XT2 pins are used for the LXT oscillator or as normal I/O or other pin-shared functional pins.

- If the LXT oscillator is not used for any clock source, the XT1/XT2 pins can be used as normal I/O or other pin-shared functional pins.
- If the LXT oscillator is used for any clock source, the 32.768kHz crystal should be connected to the XT1/XT2 pins.

For oscillator stability and to minimise the effects of noise and crosstalk, it is important to ensure that the crystal and any associated resistors and capacitors along with interconnecting lines are all located as close to the MCU as possible.



Note: 1. R_P, C1 and C2 are required.
2. Although not shown pins have a parasitic capacitance of around 7pF.

External LXT Oscillator

LXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
32.768kHz	10pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. R_P=5M~10MΩ is recommended.		

32.768kHz Crystal Recommended Capacitor Values

LXT Oscillator Low Power Function

The LXT oscillator can function in one of two modes, the Quick Start Mode and the Low Power Mode. The mode selection is executed using the LXTSP bit in the TBC register.

LXTSP Bit	LXT Operating Mode
0	Low-power
1	Quick Start

When the LXTSP bit is set to high, the LXT Quick Start Mode will be enabled. In the Quick Start Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up it can be placed into the Low-power mode by clearing the LXTSP bit to zero. The oscillator will continue to run but with reduced current consumption, as the higher current consumption is only required during the LXT oscillator start-up.

It should be noted that, no matter what condition the LXTSP bit is set to, the LXT oscillator will always function normally. The only difference is that it will take more time to start up if in the Low-power mode.

Internal 32kHz Oscillator – LIRC

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

Operating Modes and System Clocks

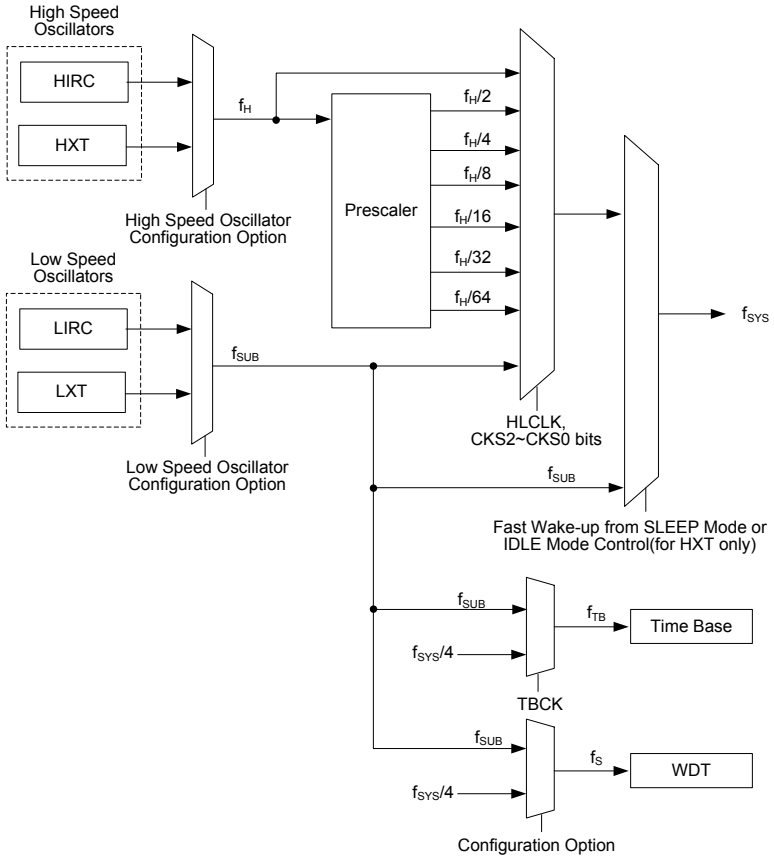
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their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

System Clocks

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

The main system clock, can come from either a high frequency f_H or low frequency f_{SUB} source, and is selected using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. The high speed system clock can be sourced from either an HXT or HIRC oscillator, selected via a configuration option. The low speed system clock source can be sourced from internal clock f_{SUB}. If f_{SUB} is selected then it can be sourced by either the LXT or LIRC oscillator, selected via a configuration option. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.

The f_{SUB} clock is used to provide a substitute clock for the microcontroller just after a wake-up has occurred to enable faster wake-up times. The f_{SUB} clock is used as a source for the Watchdog timer, the Time Base interrupt functions and for the TMs.



Device Clock Configuration

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillator will stop to conserve the power. Thus there is no f_H~f_H/64 for peripheral circuit 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 NORMAL Mode and SLOW Mode. The remaining four modes, the SLEEP0, SLEEP1, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.

Operating Mode	Description			
	CPU	f_{SYS}	f_{SUB}	f_S
NORMAL Mode	On	f_H~f_H/64	On	On
SLOW Mode	On	f_{SUB}	On	On
IDLE0 Mode	Off	Off	On	On/Off (Note)
IDLE1 Mode	Off	On	On	On
SLEEP0 Mode	Off	Off	Off	Off
SLEEP1 Mode	Off	Off	On	On

Note: The WDT clock, f_S, will be off and the WDT will stop running if the WDT clock source is selected from the system clock by configuration option regardless of the WDT is enabled or disabled. The WDT clock, f_S, will be on if the WDT clock source is selected from the f_{SUB} clock and the WDT is enabled.

NORMAL Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from one of the high speed oscillators, either the HXT or HIRC oscillators. 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 and HLCLK bits in the SMOD register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from one of the low speed oscillators, either the LXT or the LIRC. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW Mode, the f_H is off.

SLEEP0 Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP0 mode the CPU will be stopped, and the f_{SUB} and f_S clocks will be stopped too, and the Watchdog Timer function is disabled. When the device is in the SLEEP mode, the LVD function will be disabled automatically even if the ENLVD bit is high.

SLEEP1 Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP1 mode the CPU will be stopped. However the f_{SUB} and f_S clocks will continue to operate if the Watchdog Timer function is enabled with its clock source coming from the f_{SUB}. When the device is in the SLEEP mode, the LVD function will be disabled automatically even if the ENLVD bit is high.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU but some peripheral functions will remain operational such as the Watchdog Timer and TMs. In the IDLE0 Mode, the system oscillator will be stopped. In the IDLE0 Mode the Watchdog Timer clock, f_S, will either be on or off depending upon the f_S clock source. If the source is $f_{SYS}/4$ then the f_S clock will be off, and if the source comes from f_{SUB} then f_S will be on.

IDLE1 Mode

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

Control Register

The registers, SMOD and CTRL, are used for overall control of the internal clocks within the device.

SMOD Register

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

Bit 7~5 **CKS2~CKS0**: The system clock selection when HLCLK is "0"
000: f_{SUB} (f_{LXT} or f_{LIRC})
001: f_{SUB} (f_{LXT} or f_{LIRC})
010: $f_H/64$
011: $f_H/32$
100: $f_H/16$
101: $f_H/8$
110: $f_H/4$
111: $f_H/2$
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source, which can be either the LXT or LIRC, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 **FSTEN**: Fast Wake-up Control (only for HXT)
0: Disable
1: Enable
This is the Fast Wake-up Control bit which determines if the f_{SUB} clock source is initially used after the device wakes up. When the bit is high, the f_{SUB} clock source can be used as a temporary system clock to provide a faster wake up time as the f_{SUB} clock is available.

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

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

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

Bit 0 **HLCLK**: system clock selection
0: $f_H/2 \sim f_H/64$ or f_{SUB}
1: f_H
This bit is used to select if the f_H clock or the $f_H/2 \sim f_H/64$ or f_{SUB} clock is used as the system clock. When the bit is high the f_H clock will be selected and if low the $f_H/2 \sim f_H/64$ or f_{SUB} clock will be selected. When system clock switches from the f_H clock to the f_{SUB} clock and the f_H clock will be automatically switched off to conserve power.

CTRL Register

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

"x" unknown

Bit 7 **FSYSON**: f_{SYS} Control in IDLE Mode
0: Off
1: On

Bit 6~3 Unimplemented, read as "0"

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

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

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

Fast Wake-up

To minimise power consumption the device can enter the SLEEP or IDLE0 Mode, where the system clock source to the device will be stopped. However when the device is woken up again, it can take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume. To ensure the device is up and running as fast as possible a Fast Wake-up function is provided, which allows f_{SUB}, namely either the LXT or LIRC oscillator, to act as a temporary clock to first drive the system until the original system oscillator has stabilised. As the clock source for the Fast Wake-up function is f_{SUB}, the Fast Wake-up function is only available in the SLEEP1 and IDLE0 modes. When the device is woken up from the SLEEP0 mode, the Fast Wake-up function has no effect because the f_{SUB} clock is stopped. The Fast Wake-up enable/disable function is controlled using the FSTEN bit in the SMOD register.

If the HXT oscillator is selected as the NORMAL Mode system clock, and if the Fast Wake-up function is enabled, then it will take one to two t_{SUB} clock cycles of the LIRC or LXT oscillator for the system to wake-up. The system will then initially run under the f_{SUB} clock source until 128 HXT clock cycles have elapsed, at which point the HTO flag will switch high and the system will switch over to operating from the HXT oscillator.

If the HIRC oscillator or LIRC oscillator is used as the system oscillator then it will take 15~16 clock cycles of the HIRC or 1~2 cycles of the LIRC to wake up the system from the SLEEP or IDLE0 Mode. The Fast Wake-up bit, FSTEN will have no effect in these cases.

System Oscillator	FSTEN Bit	Wake-up Time (SLEEP0 Mode)	Wake-up Time (SLEEP1 Mode)	Wake-up Time (IDLE0 Mode)	Wake-up Time (IDLE1 Mode)
HXT	0	128 HXT cycles	128 HXT cycles		1~2 HXT cycles
	1	128 HXT cycles	1~2 f_{SUB} cycles (System runs with f_{SUB} first for 128 HXT cycles and then switches over to run with the HXT clock)		1~2 HXT cycles
HIRC	x	15~16 HIRC cycles	15~16 HIRC cycles		1~2 HIRC cycles
LIRC	x	1~2 LIRC cycles	1~2 LIRC cycles		1~2 LIRC cycles
LXT	x	1024 LXT cycles	1024 LXT cycles		1~2 LXT cycles

"x": don't care

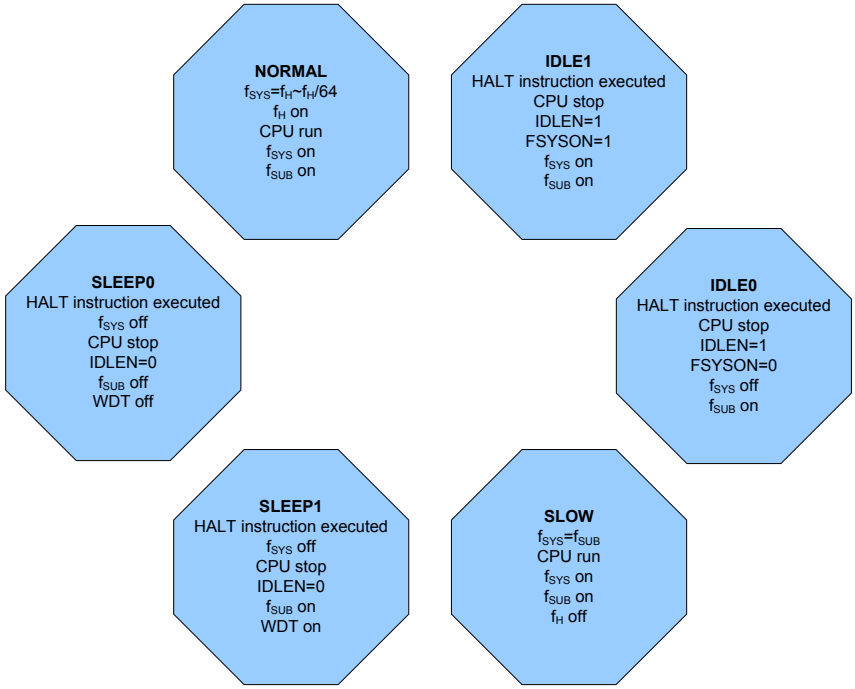
Wake-Up Times

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 NORMAL Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/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 IDLEN bit in the SMOD register and FSYSON in the CTRL register.

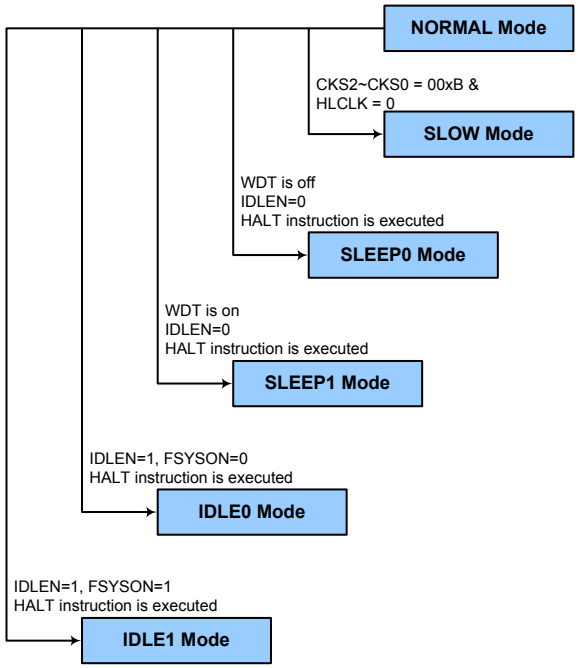
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~$f_H/64$ or f_{SUB}. If the clock is from the f_{SUB}, the high speed clock source will stop running to conserve power. When this happens it must be noted that the $f_H/16$ and $f_H/64$ internal clock sources will also stop running, which may affect the operation of other internal functions. The accompanying flowchart shows what happens when the device moves between the various operating modes.



NORMAL Mode to SLOW Mode Switching

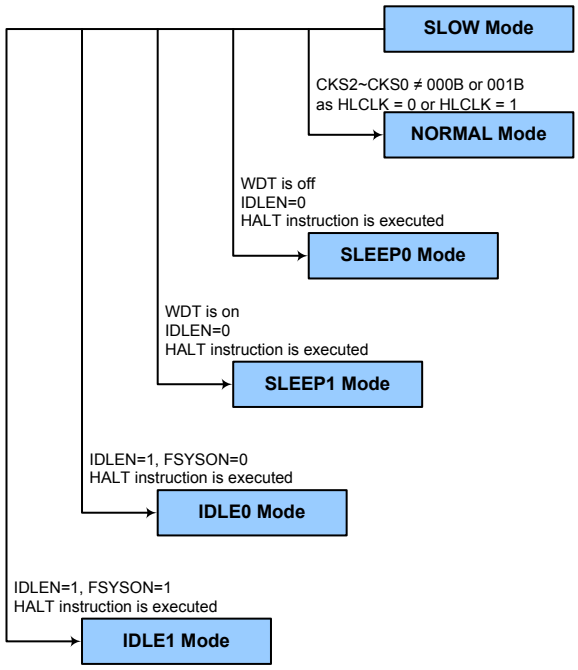
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to "0" and set the CKS2~CKS0 bits to "000" or "001" in the SMOD register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

The SLOW Mode is sourced from the LXT or the LIRC oscillators and therefore requires these oscillators to be stable before full mode switching occurs. This is monitored using the LTO bit in the SMOD register.



SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system uses either the LXT or LIRC low speed system oscillator. To switch back to the NORMAL Mode, where the high speed system oscillator is used, the HLCLK bit should be set to "1" or HLCLK bit is "0", but CKS2~CKS0 is set to "010", "011", "100", "101", "110" or "111". As a certain amount of time will be required for the high frequency clock to stabilise, the status of the HTO bit is checked. The amount of time required for high speed system oscillator stabilization depends upon which high speed system oscillator type is used.



Entering the SLEEP0 Mode

There is only one way for the device to enter the SLEEP0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0" and the WDT and LVD both off. When this instruction is executed under the conditions described above, the following will occur:

- The system clock and the f_{SUB} 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 WDT will be cleared and stopped no matter if the WDT clock source originates from the f_{SUB} clock or from the system clock.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the SLEEP1 Mode

There is only one way for the device to enter the SLEEP1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0" and the WDT or LVD on. When this instruction is executed under the conditions described above, the following will occur:

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

Entering the IDLE0 Mode

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

- The system clock will be stopped and the application program will stop at the "HALT" instruction, but the Time Base clock and f_{SUB} clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the f_{SUB} clock and the WDT is enabled. The WDT will stop if its clock source originates from the system clock.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE1 Mode

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

- The system clock and Time Base clock and 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 WDT will be cleared and resume counting if the WDT is enabled regardless of the WDT clock source which originates from the f_{SUB} clock or from the system clock.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Standby Current Considerations

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

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

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

Wake-up

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

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

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

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

Programming Considerations

The HXT and LXT oscillators both use the same SST counter. For example, if the system is woken up from the SLEEP0 Mode and both the HXT and LXT oscillators need to start-up from an off state. The LXT oscillator uses the SST counter after HXT oscillator has finished its SST period.

- If the device is woken up from the SLEEP0 Mode to the NORMAL Mode, the high speed system oscillator needs an SST period. The device will execute first instruction after HTO is "1". At this time, the LXT oscillator may not be stability if f_{SUB} is from LXT oscillator. The same situation occurs in the power-on state. The LXT oscillator is not ready yet when the first instruction is executed.
- If the device is woken up from the SLEEP1 Mode to NORMAL Mode, and the system clock source is from HXT oscillator and FSTEN is "1", the system clock can be switched to the low speed oscillator after wake up.
- There are peripheral functions, such as WDT and TMs, for which the f_{SYS} is used. If the system clock source is switched from f_H to f_{SUB}, the clock source to the peripheral functions mentioned above will change accordingly.
- The on/off condition of f_{SUB} and f_S depends upon whether the WDT is enabled or disabled as the WDT clock source is selected from f_{SUB}.

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_S, which is in turn supplied by one of two sources selected by configuration option: f_{SUB} or $f_{SYS}/4$. The f_{SUB} clock can be sourced from either the LXT or LIRC oscillators, again chosen via a configuration option. The LIRC internal oscillator has an approximate period of 32kHz at a supply voltage of 5V. However, it should be noted that this specified internal clock period can vary with V_{DD}, temperature and process variations. The LXT oscillator is supplied by an external 32.768kHz crystal. The other Watchdog Timer clock source option is the $f_{SYS}/4$ clock. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{15} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. The WDTC is set to 01010011B at device reset except WDT time-out Hardware Warm reset.

WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT function software control

If the WDT configuration option is "always enable":
10101 or 01010: Enable
Others: MCU reset

If the WDT configuration option is "controlled by the WDT control register":
10101: Disable
01010: Enable
Others: MCU reset

When these bits are changed by the environmental noise or software setting to reset the microcontroller, the reset operation will be activated after 2~3 f_{SUB} clock cycles and the WRF bit in the CTRL register will be set high.

Bit 2~0 **WS2~WS0**: WDT time-out period selection

000: $2^8/f_S$
001: $2^9/f_S$
010: $2^{10}/f_S$
011: $2^{11}/f_S$
100: $2^{12}/f_S$
101: $2^{13}/f_S$
110: $2^{14}/f_S$
111: $2^{15}/f_S$

These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

CTRL Register

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

"x" unknown

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

Bit 6~3 Unimplemented, read as "0"

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

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

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

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device.

Some of the Watchdog Timer options, such as always on select and clock source are selected using configuration options. With regard to the Watchdog Timer enable/disable function, there are also five bits, WE4~WE0, in the WDTC register to offer additional enable/disable and reset control of the Watchdog Timer. If the WDT configuration option is determined that the WDT function is always enabled, the WE4~WE0 bits still have effects on the WDT function. When the WE4~WE0 bits value is equal to 01010B or 10101B, the WDT function is enabled. However, if the WE4~WE0 bits are changed to any other values except 01010B and 10101B, which is caused by the environmental noise or software setting, it will reset the microcontroller after 2~3 LIRC clock cycles. If the WDT configuration option is determined that the WDT function is controlled by the WDT control register, the WE4~WE0 values can determine which mode the WDT operates in. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B. The WDT function will be enabled if the WE4~WE0 bits value is equal to 01010B. If the WE4~WE0 bits are set to any other values by the environmental noise or software setting, except 01010B and 10101B, it will reset the device after 2~3 LIRC clock cycles. After power on these bits will have the value of 01010B.

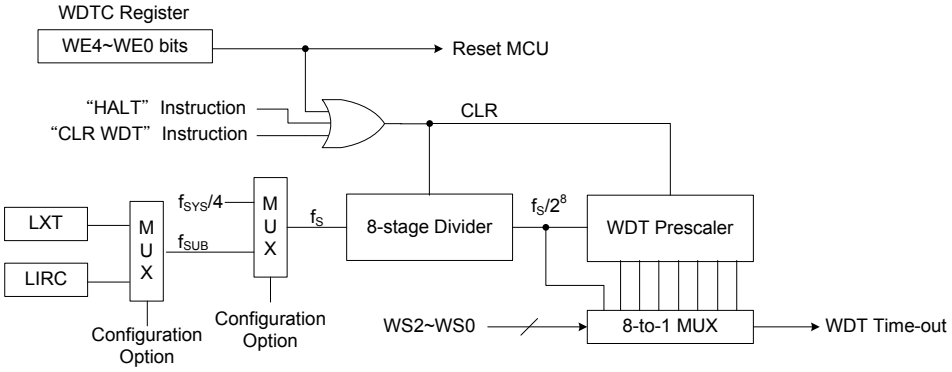
WDT Configuration Option	WE4~WE0 Bits	WDT Function
Always Enable	01010B or 10101B	Enable
	Any other values	Reset MCU
Controlled by WDT Control Register	10101B	Disable
	01010B	Enable
	Any other values	Reset MCU

Watchdog Timer Enable/Disable Control

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

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

The maximum time out period is when the 2^{15} 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 1 second for the 2^{15} division ratio, and a minimum timeout of 8ms for the 2^8 division ration.



Watchdog Timer

Reset and Initialisation

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

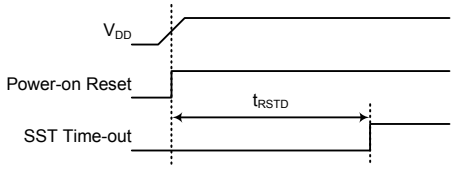
Another type of reset is when the Watchdog Timer overflows and resets. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, is implemented in situations where the power supply voltage falls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

Reset Functions

There are four ways in which a reset can occur.

Power-on Reset

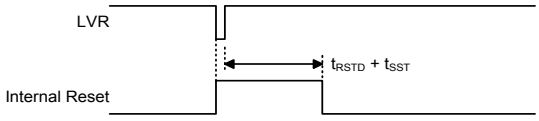
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions.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



Power-On Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the CTRL register will also be set high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVD & LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise or software setting, the LVR will reset the device after 2~3 f_{LIRC} clock cycles. When this happens, the LRF bit in the CTRL 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 power down mode.



Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Other values: MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage values above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps more than a t_{LVR} time. In this situation the register contents will remain the same after such a reset occurs.

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

CTRL Register

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

"x" unknown

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

Bit 6~3 Unimplemented, read as "0"

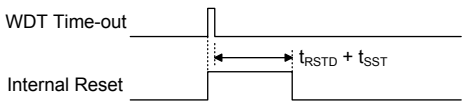
Bit 2 **LVRF**: LVR function reset flag
0: not occur
1: occurred
This bit is set high when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

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

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

Watchdog Time-out Reset during Normal Operation

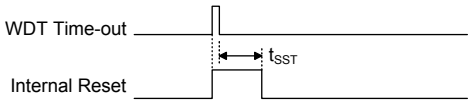
The Watchdog time-out Reset during normal operation is the same as LVR reset except that the Watchdog time-out flag TO will be set high.



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 zero and the TO flag will be set high. Refer to the A.C. Characteristics for t_{SST} details.



WDT Time-out Reset during Sleep or IDLE Timing Chart

Reset Initial Conditions

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

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

"u" stands for unchanged

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

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

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

Register Name	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu
BP	---- --0	---- --0	---- ---u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	---- -xxx	---- -uuu	---- -uuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
SMOD	1110 0010	1110 0010	uuuu uuuu
TBC	0011 0111	0011 0111	uuuu uuuu
LVDC	--00 0000	--00 0000	--uu uuuu
CTRL	0--- -x00	0--- -000	u--- -uuu
LVRC	0101 0101	0101 0101	uuuu uuuu
EEA0	0000 0000	0000 0000	uuuu uuuu
EEA1	---- ---0	---- ---0	---- ---u
EEC	---- 0000	---- 0000	---- uuuu
EED	xxxx xxxx	xxxx xxxx	uuuu uuuu
WDTC	0101 0011	0101 0011	uuuu uuuu
PSS	---- --00	---- --00	---- --uu
PMSL	---- 0000	---- 0000	---- uuuu
PDHCL	---- 0000	---- 0000	---- uuuu
DTS0	0000 0000	0000 0000	uuuu uuuu
DTS1	0000 0000	0000 0000	uuuu uuuu
POSL	---- 0000	---- 0000	---- uuuu
PRTL	---- 0000	---- 0000	---- uuuu
OPCL	---- 0-0-	---- 0-0-	---- u-u-
HVC	000- ----	000- ----	uuu- ----
OCPC00	0000 ---0	0000 ---0	uuuu ---u
OCPC10	--00 0000	--00 0000	--uu uuuu
OCPDA0	0000 0000	0000 0000	uuuu uuuu
OCPOCAL0	0010 0000	0010 0000	uuuu uuuu
OCPCCAL0	0001 0000	0001 0000	uuuu uuuu
INTEG	--00 0000	--00 0000	--uu uuuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	000- 000-	000- 000-	uuu- uuu-
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	0000 0000	0000 0000	uuuu uuuu
MFI0	--00 --00	--00 --00	--uu --uu
MFI1	--00 --00	--00 --00	--uu --uu
MFI2	--00 --00	--00 --00	--uu --uu
MFI3	--00 --00	--00 --00	--uu --uu
MFI4	--00 --00	--00 --00	--uu --uu

Register Name	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PTM2C0	0000 0---	0000 0---	uuuu u---
PTM2C1	0000 0000	0000 0000	uuuu uuuu
PTM2DL	0000 0000	0000 0000	uuuu uuuu
PTM2DH	---- --00	---- --00	---- --uu
PTM2AL	0000 0000	0000 0000	uuuu uuuu
PTM2AH	---- --00	---- --00	---- --uu
PTM2RPL	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	---- --00	---- --00	---- --uu
PTM3C0	0000 0---	0000 0---	uuuu u---
PTM3C1	0000 0000	0000 0000	uuuu uuuu
PTM3DL	0000 0000	0000 0000	uuuu uuuu
PTM3DH	---- --00	---- --00	---- --uu
PTM3AL	0000 0000	0000 0000	uuuu uuuu
PTM3AH	---- --00	---- --00	---- --uu
PTM3RPL	0000 0000	0000 0000	uuuu uuuu
PTM3RPH	---- --00	---- --00	---- --uu
PTM0C0	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	uuuu uuuu
PTM0DL	0000 0000	0000 0000	uuuu uuuu
PTM0DH	0000 0000	0000 0000	uuuu uuuu
PTM0AL	0000 0000	0000 0000	uuuu uuuu
PTM0AH	0000 0000	0000 0000	uuuu uuuu
PTM0RPL	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	0000 0000	0000 0000	uuuu uuuu
PTM1C0	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	uuuu uuuu
PTM1DL	0000 0000	0000 0000	uuuu uuuu
PTM1DH	0000 0000	0000 0000	uuuu uuuu
PTM1AL	0000 0000	0000 0000	uuuu uuuu
PTM1AH	0000 0000	0000 0000	uuuu uuuu
PTM1RPL	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	0000 0000	0000 0000	uuuu uuuu
SADC0	0100 0000	0100 0000	uuuu uuuu
SADC1	---0 0000	---0 0000	---u uuuu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
OP0C	-0-- --00	-0-- --00	-u-- --uu
OP0VOS	0010 0000	0010 0000	uuuu uuuu

Register Name	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IICC0	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	0000 000-	0000 000-	uuuu uuu-
I2CTOC	0000 0000	0000 0000	uuuu uuuu
PAPS0	0000 0000	0000 0000	uuuu uuuu
PAPS1	0000 0000	0000 0000	uuuu uuuu
PBDPS	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PDPU	00-- ----	00-- ----	uu-- ----
PD	11-- ----	11-- ----	uu-- ----
PDC	11-- ----	11-- ----	uu-- ----
RPDH	---- --00	---- --uu	---- --uu
SSCTL	--00 --00	--uu --uu	--uu --uu

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

Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	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
PD	PD7	PD6	—	—	—	—	—	—
PDC	PDC7	PDC6	—	—	—	—	—	—
PDPU	PDPU7	PDPU6	—	—	—	—	—	—

"–": Unimplemented, read as "0"

I/O Logic Function Registers List

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using registers PxPU ("x" stands for A, B or D), and are implemented using weak PMOS transistors.

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

PxPU Register

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

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

0: Disable

1: Enable

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

Port A Wake-up

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

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

PAWU Register

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

Bit 7~0 **PAWU7~PAWU0**: PA7~PA0 wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

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

PxC Register

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

PxCn: I/O Port x Pin type selection

0: Output

1: Input

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

Pin-shared Functions

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

Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. 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. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAPS0	PA3S1	PA3S0	PA2S1	PA2S0	PA1S1	PA1S0	PA0S1	PA0S0
PAPS1	PA7S1	PA7S0	PA6S1	PA6S0	PA5S1	PA5S0	PA4S1	PA4S0
PBDPS	PD7S	PD6S	PB7S	PB6S	PB5S	PB3S	PB0S1	PB0S0
SSCTL	—	—	PT0IS	OCPIS	—	—	HXT_EX	EN_VDET

Pin-shared Function Selection Registers List

- **PAPS0 Register**

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

Bit 7~6 **PA3S1~PA3S0**: PA3 pin-shared function selection
00: PA3/PTCK3
01: AN3
10: PTP1
11: HPO0

Bit 5~4 **PA2S1~PA2S0**: PA2 pin-shared function selection
00: PA2/PTCK2
01: SCL
10: PA2/PTCK2
11: PA2/PTCK2

Bit 3~2 **PA1S1~PA1S0**: PA1 pin-shared function selection
00: PA1/PTPI1
01: AN1
10: PTP1
11: HPO0

Bit 1~0 **PA0S1~PA0S0**: PA0 pin-shared function selection
00: PA0/PTPI0
01: SDA
10: PA0/PTPI0
11: PA0/PTPI0

- **PAPS1 Register**

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

Bit 7~6 **PA7S1~PA7S0**: PA7 pin-shared function selection
00: PA7
01: AN7
10: PA7
11: PA7

Bit 5~4 **PA6S1~PA6S0**: PA6 pin-shared function selection
00: PA6
01: AN6
10: OCP0REF/ADCREF
11: PA6

Bit 3~2 **PA5S1~PA5S0**: PA5 pin-shared function selection
00: PA5
01: OP0OUT
10: PTP1
11: HPO0

Bit 1~0 **PA4S1~PA4S0**: PA4 pin-shared function selection
00: PA4/INT0
01: AN4
10: PTP0
11: PA4/INT0

- **PBDPS Register**

Bit	7	6	5	4	3	2	1	0
Name	PD7S	PD6S	PB7S	PB6S	PB5S	PB3S	PB0S1	PB0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PD7S**: PD7 pin-shared function selection
0: PD7
1: OCPI0

Bit 6 **PD6S**: PD6 pin-shared function selection
0: PD6
1: OCPI0

Bit 5 **PB7S**: PB7 pin-shared function selection
0: PB7
1: OCPAO0

Bit 4 **PB6S**: PB6 pin-shared function selection
0: PB6
1: OP0INN

Bit 3 **PB5S**: PB5 pin-shared function selection
0: PB5
1: OP0INP

Bit 2 **PB3S**: PB3 pin-shared function selection
0: PB3/PTP3I
1: PTP3

Bit 1~0 **PB0S1~PB0S0**: PB0 pin-shared function selection
00: PB0/PTP2I/PTCK0/INT1
01: PTP2
10: PB0/PTP2I/PTCK0/INT1
11: PB0/PTP2I/PTCK0/INT1

- **SSCTL Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PT0IS	OCPIS	—	—	HXT_EX	EN_VDET
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PT0IS**: PTP0I input source pin selection
0: From PA0
1: From internal OCP0O signal

Bit 5 **OCPIS**: OCPI0 input source pin selection
0: From PD6
1: From PD7

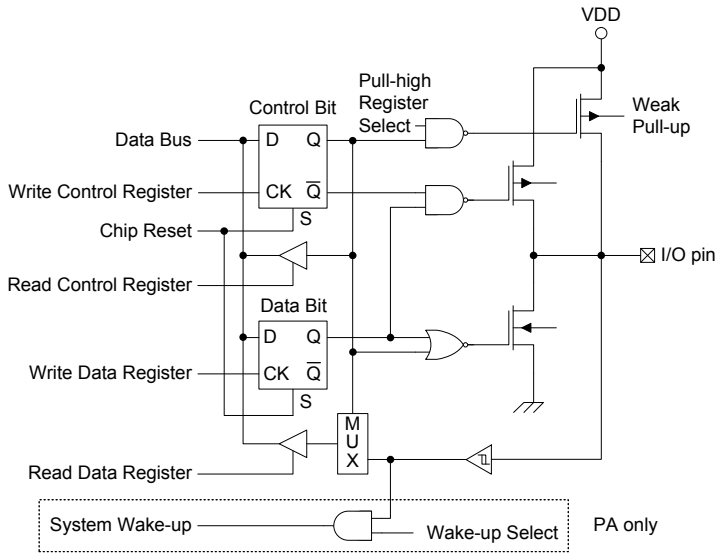
Bit 3~2 Unimplemented, read as "0"

Bit 1 **HXT_EX**: HXT PWM output control
0: Disable
1: Enable
This bit is used to control the PWM signal output with HXT frequency. Refer to the Oscillators section for more details.

Bit 0 **EN_VDET**: High voltage power supply detection function control
0: Disable
1: Enable
This bit is used to control the high voltage power supply detection function enabled or disabled, the details of which are described in the High Voltage Driver section.

I/O Pin Structures

The accompanying diagram illustrates the internal structures of the I/O logic function. As the exact logical construction of the I/O pin will differ from this drawing, it is supplied as a guide only to assist with the functional understanding of the 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, PxC, 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, Px, 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 device is the ability to control and measure time. To implement time related functions the device includes several Timer Modules, abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two individual interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

The general features of the Periodic type TM are described here with more detailed information provided in the Periodic TM section.

Introduction

The device contains four Periodic type TMs. The main features of the PTM are summarised in the accompanying table.

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

PTM Function Summary

TM Operation

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

TM Clock Source

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

TM Interrupts

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

TM External Pins

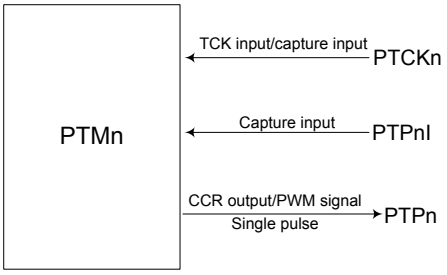
Each of the PTMs has two TM input pins, with the label PTCKn and PTPnI. The PTMn input pin, PTCKn, is essentially a clock source for the PTMn and is selected using the PTnCK2~PTnCK0 bits in the PTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The PTCKn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

The other PTMn input pin, PTPnI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTnIO1~PTnIO0 bits in the PTMnC1 register.

The PTMs each has one output pin, PTPn. 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 PTPn output pin is also the pin where the TM generates the PWM output waveform.

TM Input/Output Pin Selection

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

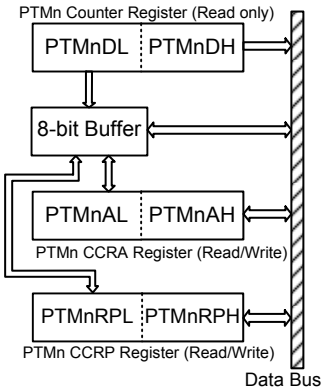


PTM Function Pin Control Block Diagram

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

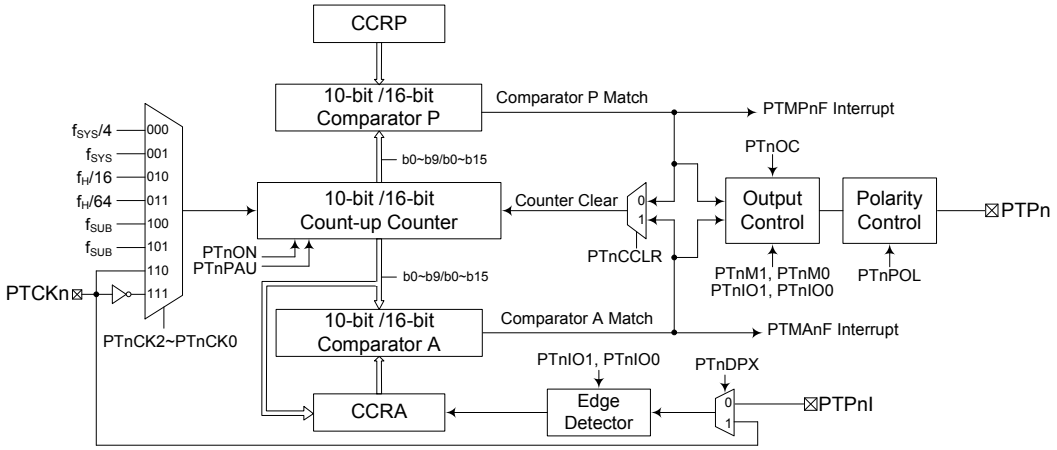


The following steps show the read and write procedures:

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

Periodic Type TM – PTM

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



Note: 1. As the PTCKn, PTPn and PTPnI pins are pin-shared with I/O pins, the corresponding pin-shared selection bits should first be properly configured.
2. For the PTM0, PTP0I input source can be selected from the external PA0 pin or the internal OCP circuit OCPO0 signal by the PTP0IS bit in the SSCTL register.
3. The size of PTM0 and PTM1 is 16-bit wide, while the size of PTM2 and PTM3 is 10-bit wide.

Periodic Type TM Block Diagram (n=0~3)

Periodic TM Operation

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

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

Periodic Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnDPX	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	D15	D14	D13	D12	D11	D10	D9	D8

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

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

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

PTMnC0 Register – n=0~3

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

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

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

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

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

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

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

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

Bit 2~0 Unimplemented, read as "0"

PTMnC1 Register – n=0~3

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

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

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

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

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

PWM 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 PTPnI or PTCKn
01: Input capture at falling edge of PTPnI or PTCKn
10: Input capture at falling/rising edge of PTPnI or PTCKn
11: Input capture disabled

Timer/Counter Mode
Unused

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

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

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

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

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

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

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

Bit 2 **PTnPOL**: PTMn PTPn Output polarity Control
0: Non-invert
1: Invert

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

Bit 1 **PTnDPX**: PTMn Capture Trigger Source Selection
0: From PTPnI pin
1: From PTCKn pin

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

PTMnDL Register – n=0~3

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

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

PTMnDH Register – n=0~1

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

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

PTMnDH Register – n=2~3

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

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

PTMnAL Register – n=0~3

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

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

PTMnAH Register – n=0~1

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

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

PTMnAH Register – n=2~3

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

Bit 7~2 Unimplemented, read as "0"

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

PTMnRPL Register – n=0~3

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

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

PTMnRPH Register – n=0~1

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

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

PTMnRPH Register – n=2~3

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

Bit 7~2 Unimplemented, read as "0"

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

Periodic Type TM Operating Modes

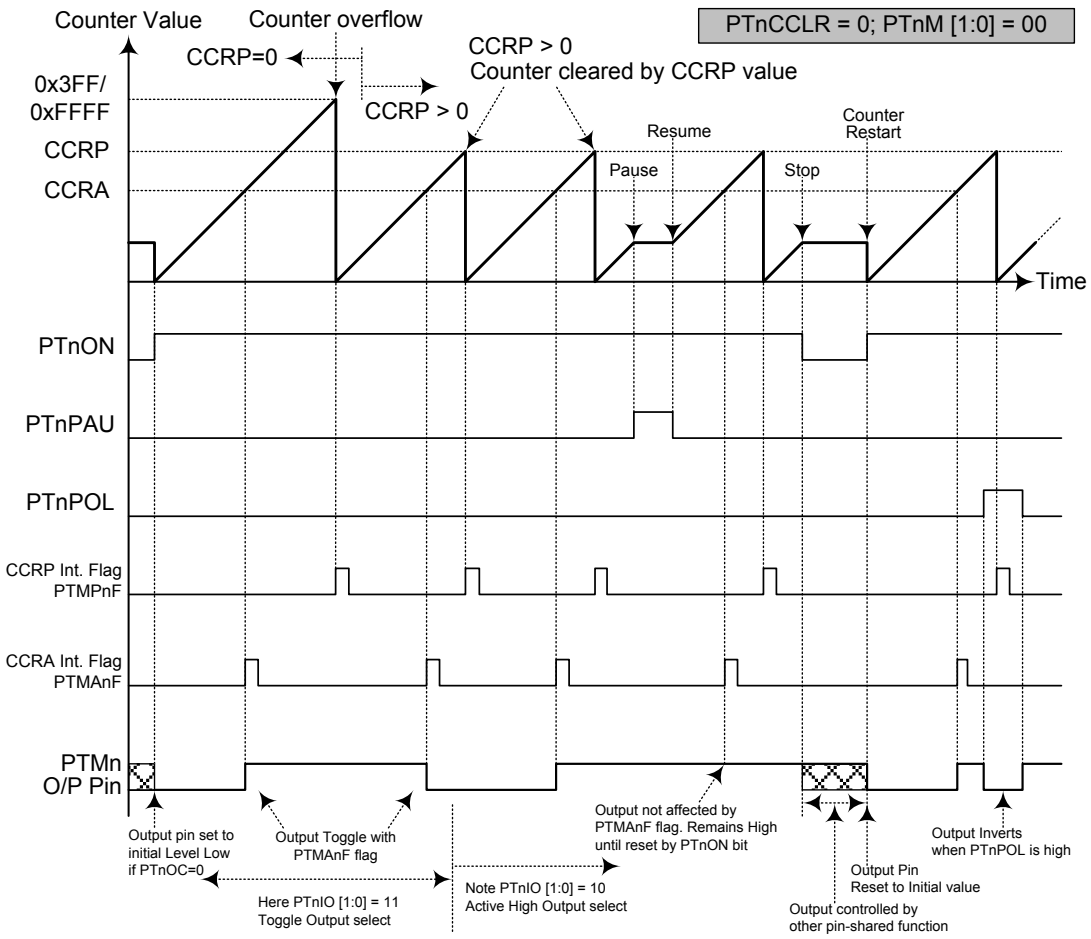
The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTnM1 and PTnM0 bits in the PTMnC1 register.

Compare Match Output Mode

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

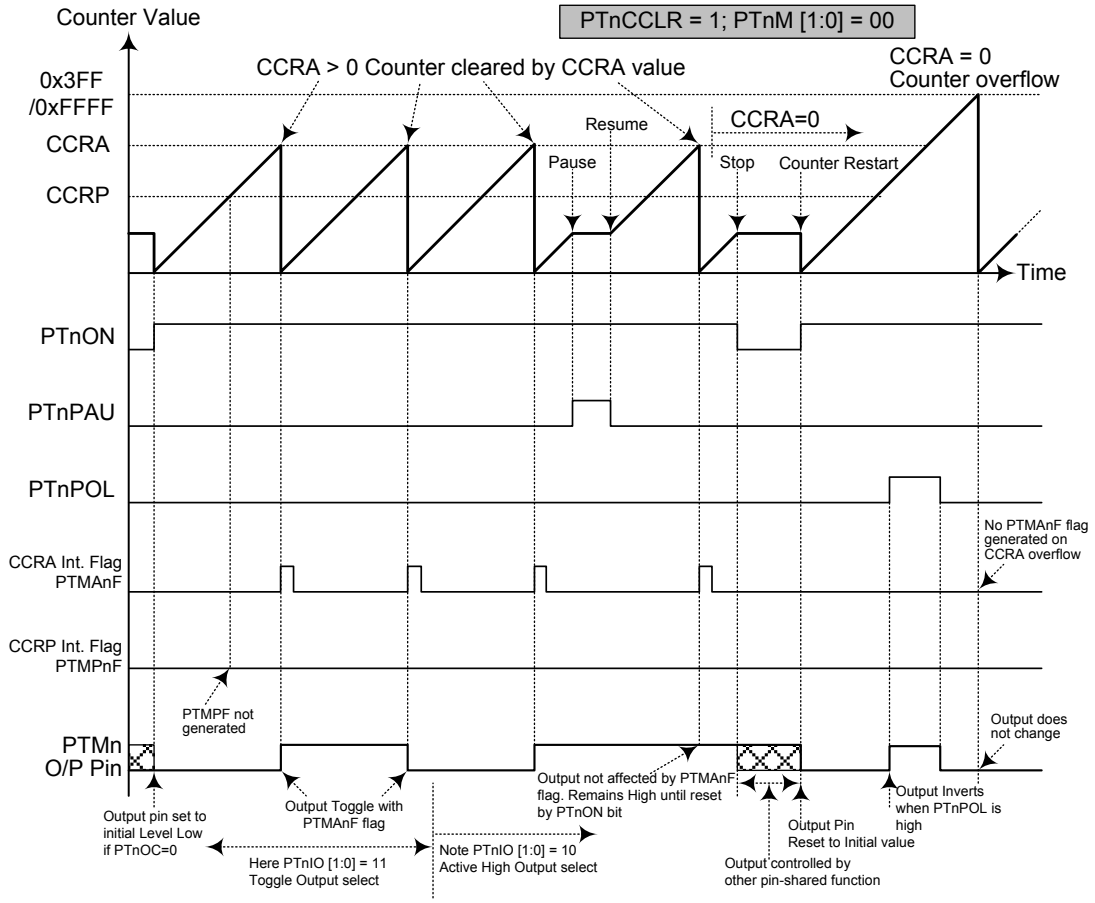
If the PTnCCLR bit in the PTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAnF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is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

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



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

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



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

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

Timer/Counter Mode

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

PWM Output Mode

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

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

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

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

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

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

The PTMn PWM output frequency=($f_{SYS}/4$)/512=$f_{SYS}/2048$=7.8125kHz, duty=128/512=25%.

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

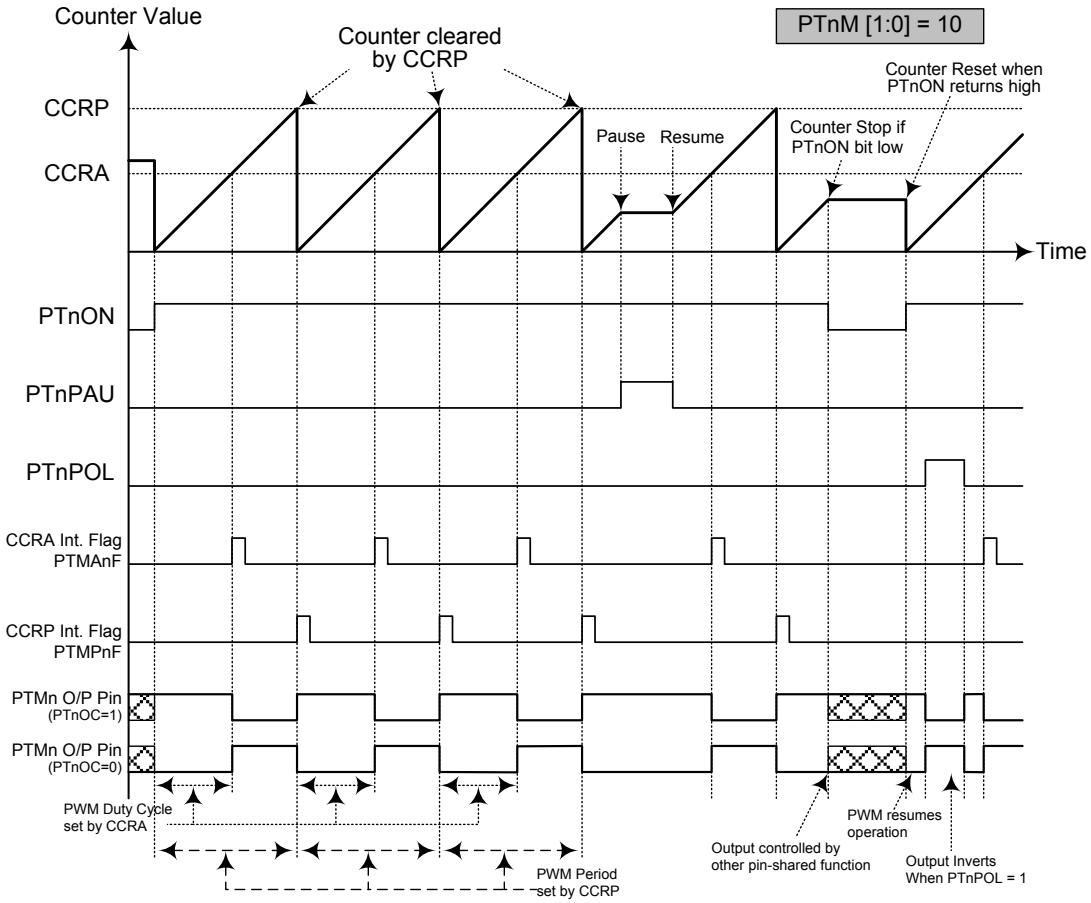
- **16-bit PTMn, PWM Output Mode**

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

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

The PTMn PWM output frequency=($f_{SYS}/4$)/512=$f_{SYS}/2048$=7.8125kHz, duty=128/512=25%.

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



PWM Output Mode (n=0~3)

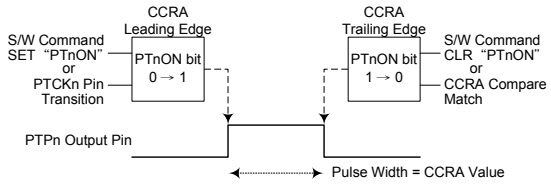
Note: 1. Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTnIO[1:0]=00 or 01
4. The PTnCCLR bit has no influence on PWM operation

Single Pulse Mode

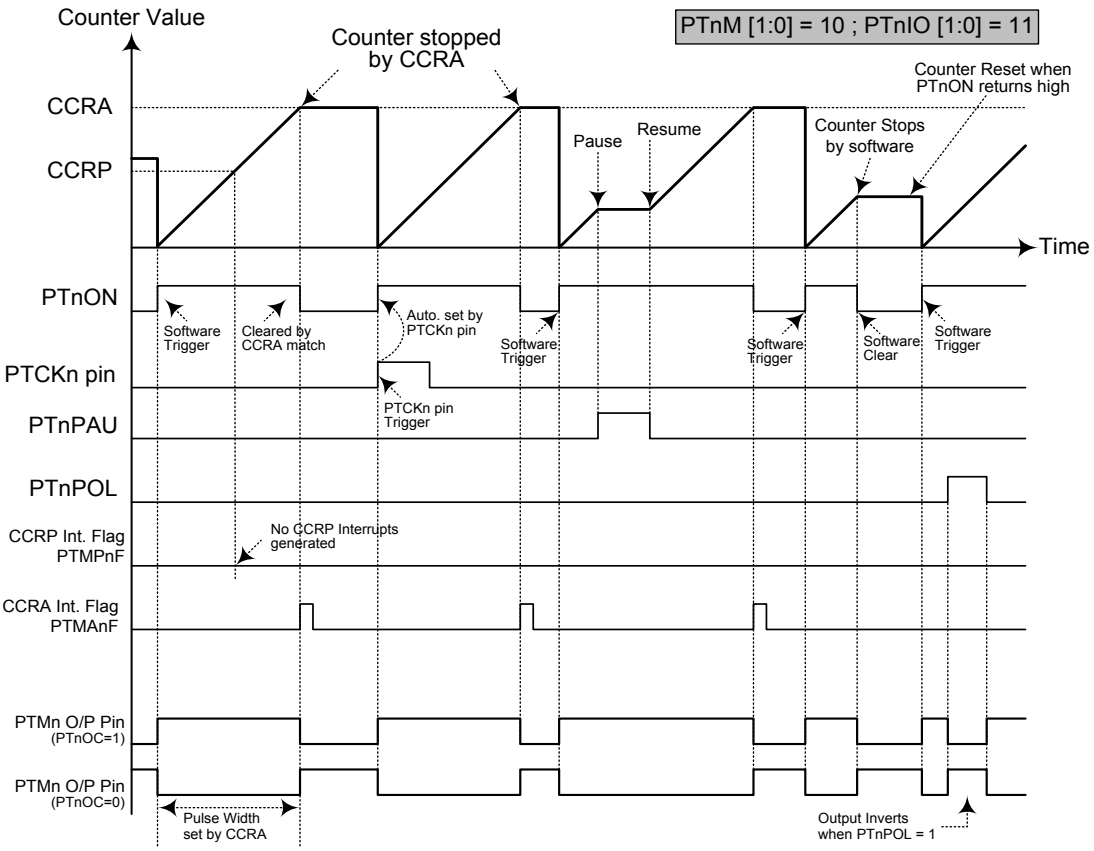
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

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

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



Single Pulse Generation (n=0~3)



Single Pulse Mode (n=0~3)

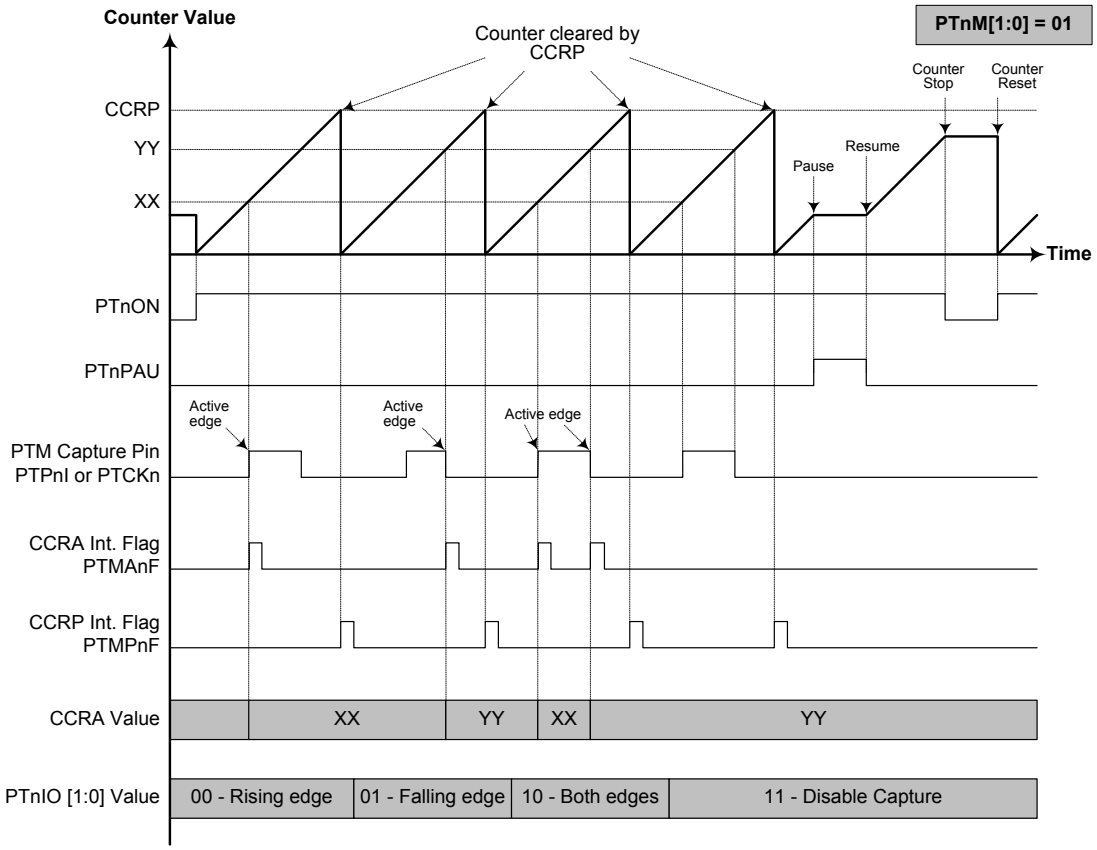
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse is triggered by the PTCKn pin or by setting the PTnON bit high
4. A PTCKn pin active edge will automatically set the PTnON bit high
5. In the Single Pulse Mode, PTnIO[1:0] must be set to "11" and cannot be changed.

Capture Input Mode

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

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

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



Capture Input Mode (n=0~3)

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

Analog to Digital Converter

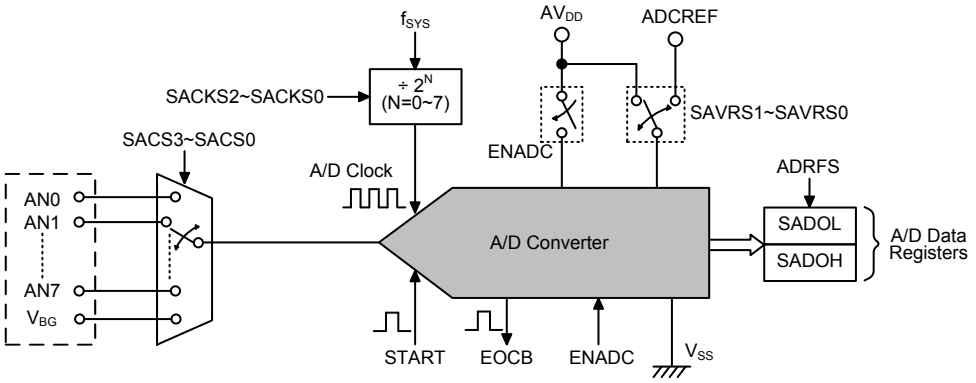
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

A/D Converter Overview

This device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, the power supply V_{CC1} divided voltage, the Over Current Protection circuit output signal or the Bandgap reference voltage, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SACS3~SACS0 bits.

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

External Input Channels	Internal Signals	Channel Select Bits
AN1, AN3, AN4, AN6, AN7	AN0: OCPAO0 signal AN5: High voltage power supply divided voltage, V_{DET} Bandgap voltage	SACS3~SACS0



Note: AN0~AN7 are internal or external signal input channels, the details can be found in the SADC0 register.

A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using four registers. A read only register pair exists to store the A/D converter data 12-bit value. The remaining two registers, SADC0 and SADC1, are control registers which setup the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	EOCB	ENADC	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	—	—	—	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

A/D Converter Registers List

A/D Converter Data Registers – SADOL, SADOH

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

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 Data Registers

A/D Converter Control Registers – SADC0, SADC1

To control the function and operation of the A/D converter, two control registers known as SADC0 and SADC1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter end of conversion status. As the device contains only one actual analog to digital converter hardware circuit, each of the external or internal analog signal inputs must be routed to the converter. The SACS3~SACS0 bits in the SADC0 register are used to determine which external analog channel input or internal analog signal is selected to be connected to the internal A/D converter.

The relevant pin-shared function selection registers, PAPS0 and PAPS1, determine which pins on I/O Ports are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

- **SADC0 Register**

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

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

Bit 6 **EOCB**: End of A/D conversion flag
0: A/D conversion ended
1: A/D conversion in progress
This read only flag is used to indicate whether the A/D conversion is completed or not. When the EOCB flag is set to 1, it indicates that the A/D conversion process is running. The bit will be cleared to 0 after the A/D conversion is complete.

Bit 5 **ENADC**: A/D converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set high to enable the A/D converter. If the bit is set low, then the A/D converter will be switched off reducing the device power consumption.

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

Bit 3~0 **SACS3~SACS0**: A/D converter analog channel input select
0000: AN0 – Internal OCP0 circuit output OCPAO0 signal
0001: AN1
0010: Reserved
0011: AN3
0100: AN4
0101: AN5 – High voltage power supply divided voltage, V_{DET}
0110: AN6
0111: AN7
1000: From Bandgap
1001~1111: Undefined, the input will be floating if selected

- **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
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 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage select
00: AV_{DD}
01: External ADCREF pin
10: AV_{DD}
11: AV_{DD}

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source select
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$
These three bits are used to select the clock source for the A/D converter.

A/D Converter Operation

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

The EOCB bit in the SADC0 register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to "0" by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the EOCB bit in the SADC0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

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

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

A/D Clock Period Examples

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

A/D Converter Reference Voltage

The reference voltage supply to the A/D converter can be supplied from the positive power supply pin, AVDD, or from an external reference source supplied on pin ADCREF. The desired selection is made using the SAVRS1 and SAVRS0 bits in the SADC1 register. As the ADCREF pin is pin-shared with other functions, when the ADCREF pin is selected as the reference voltage supply pin, the related pin-shared selection bits should first be properly configured to enable the ADCREF pin function.

A/D Converter Input Signals

All the external A/D analog channel input pins, AN1, AN3, AN4, AN6 and AN7, are pin-shared with the I/O pins as well as other functions. The corresponding control bits for each A/D external input pin in the PAPS0 and PAPS1 register determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the pin is setup to be as an A/D analog channel input, the original pin functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

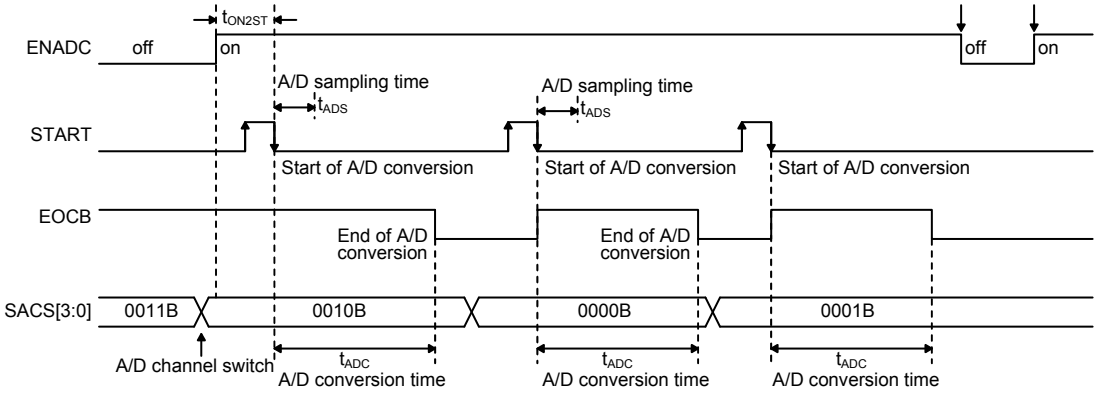
There are internal analog signals derived from AN0 – Over Current Protection analog output signal OCPAO0, AN5 – high voltage power supply detection signal V_{DET} or the Bandgap reference voltage V_{BG}, which can be connected to the A/D converter as the analog input signal by configuring the SACS3~SACS0 bits.

The A/D converter has its own reference voltage pin, ADCREF. However, the reference voltage can be supplied from the power supply pin or the external reference voltage pin, a choice which is made through the SAVRS1~SAVRS0 bits in the SADC1 register. The analog input values must not be allowed to exceed the value of the selected A/D reference voltage.

Conversion Rate and Timing Diagram

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

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 16 t_{ADCK} clock cycles where t_{ADCK} is equal to the A/D clock period.



A/D Conversion Timing

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by correctly programming bits SACKS2~SACKS0 in the SADC1 register.
- Step 2
  Enable the A/D by setting the ENADC bit in the SADC0 register to one.
- Step 3
  Select which signal is to be connected to the internal A/D converter by correctly configuring the SACS3~SACS0 bits
  Select the external channel input, AN1, AN3, AN4, AN6 or AN7, to be converted, go to Step 4.
  Select the internal analog signal, AN0, AN5 or the Bandgap reference voltage, to be converted, go to Step 5.
- Step 4
  The corresponding pins should be configured as A/D input function by configuring the relevant pin-shared function control bits. After this step, go to Step 6.
- Step 5
  Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.
- Step 6
  Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register.
- Step 7
  If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 8
  The A/D conversion procedure can now be initialized by setting the START bit in the SADC0 register from low to high and then low again.
- Step 9
  To check when the analog to digital conversion process is complete, the EOCB bit in the SADC0 register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers SADOL and SADOH can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

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

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by clearing bit ENADC to 0 in the SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/O pins, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Conversion Function

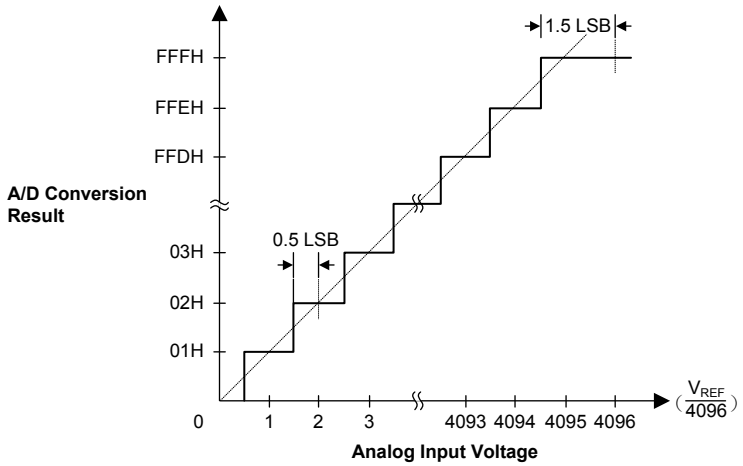
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage, V_{REF}, this gives a single bit analog input value of V_{REF} divided by 4096.

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

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

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

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



Ideal A/D Transfer Function

A/D Conversion Programming Examples

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

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

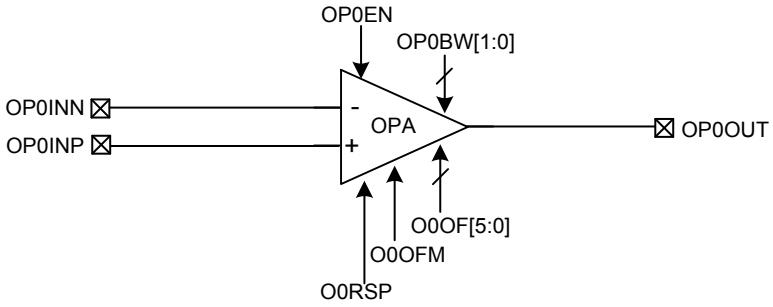
```
clr  ADE              ; disable ADC interrupt
mov  a,03H
mov  SADC1,a          ; select fSYS/8 as A/D clock
set  ENADC
mov  a,04h            ; setup PAPS0 to configure pin AN1
mov  PAPS1,a
mov  a,21h
mov  SADC0,a          ; enable and connect AN1 channel to A/D converter
:
start_conversion:
clr  START            ; high pulse on start bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
polling_EOC:
sz   EOCB             ; poll the SADC0 register EOCB bit to detect end of A/D conversion
jmp  polling_EOC      ; continue polling
mov  a,SADOL          ; read low byte conversion result value
mov  SADOL_buffer,a   ; save result to user defined register
mov  a,SADOH          ; read high byte conversion result value
mov  SADOH_buffer,a   ; save result to user defined register
:
:
jmp  start_conversion ; start next A/D conversion
```

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

```
clr  ADE              ; disable ADC interrupt
mov  a,03H
mov  SADC1,a          ; select fSYS/8 as A/D clock
set  ENADC
mov  a,04h            ; setup PAPS0 to configure pins AN1
mov  PAPS0,a
mov  a,21h
mov  SADC0,a          ; enable and connect AN1 channel to A/D converter
start_conversion:
clr  START            ; high pulse on START bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
clr  ADF              ; clear ADC interrupt request flag
set  ADE              ; enable ADC interrupt
set  EMI              ; enable global interrupt
:
:
; ADC 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
```

Operational Amplifier – OPA

The device includes an operational amplifier which has an internal input offset calibration function and adjustable bandwidth.



Operational Amplifier Block Diagram

Operational Amplifier Operation

The operational amplifier can adjust its bandwidth and provides an input offset calibration function. The calibrated data is stored in O0OF[5:0] bits. The O0OFM is calibration mode control bit and the O0RSP is used to indicate the input reference voltage comes from OP0INP or OP0INN pin in calibration mode. The OP0INP and OP0INN pins are the operational amplifier positive and negative input pair and the OP0OUT is the operational amplifier analog output voltage pin. The operational amplifier bandwidth can be set by OP0BW[1:0]. The OP0EN bit is used to enable or disable the operational amplifier.

Operational Amplifier Registers

The OP0C and OP0VOS registers control the overall operation of the operational amplifier, such as the enable or disable control, input path selection and input offset calibration.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OP0C	—	OP0EN	—	—	—	—	OP0BW1	OP0BW0
OP0VOS	O0OFM	O0RSP	O0OF5	O0OF4	O0OF3	O0OF2	O0OF1	O0OF0

Operational Amplifier Registers List

OP0C Register

Bit	7	6	5	4	3	2	1	0
Name	—	OP0EN	—	—	—	—	OP0BW1	OP0BW0
R/W	—	R/W	—	—	—	—	R/W	R/W
POR	—	0	—	—	—	—	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **OP0EN**: Operational amplifier enable or disable control
0: Disable
1: Enable

Bit 5~2 Unimplemented, read as "0"

Bit 1~0 **OP0BW1~OP0BW0**: Operational amplifier bandwidth control
Refer to the A.C. Characteristics section for the detailed information under different settings.

OP0VOS Register

Bit	7	6	5	4	3	2	1	0
Name	O0OFM	O0RSP	O0OF5	O0OF4	O0OF3	O0OF2	O0OF1	O0OF0
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 **O0OFM**: Operational Amplifier Normal Operation or Input Offset Calibration Mode selection
0: Normal Operation
1: Input Offset Calibration Mode

Bit 6 **O0RSP**: Operational Amplifier Input Offset Voltage Calibration Reference selection
0: Select negative input OP0INN as the reference input
1: Select positive input OP0INP as the reference input

Bit 5~0 **O0OF5~O0OF0**: Operational Amplifier Input Offset Voltage Calibration value

Operational Amplifier Input Offset Calibration

The Operational Amplifier provides input offset calibration function. To operate in the input offset calibration mode for the Operational Amplifier, the O0OFM bit in the OP0VOS register should first be set to "1" followed by the reference input selection by configuring the O0RSP bit. Note that because the Operational Amplifier inputs are pin-shared with I/O pins, they should be configured as Operational Amplifier inputs first.

For operational amplifier input offset calibration, the procedures are summarized in the following steps.

Step 1. Set O0OFM=1 and configure O0RSP, the Operational Amplifier will operate in the input offset Calibration mode. To make sure V_{OS} as minimise as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal operation.

Step 2. Set O0OF[5:0]=000000 and then read the OP0OUT bit.

Step 3. Increase the O0OF[5:0] value by 1 and then read the OP0OUT bit.
If the OP0OUT bit value has not changed, then repeat Step 3 until the OP0OUT bit value has changed.
If the OP0OUT bit value has changed, record the O0OF[5:0] value as V_{OS1} and then go to Step 4.

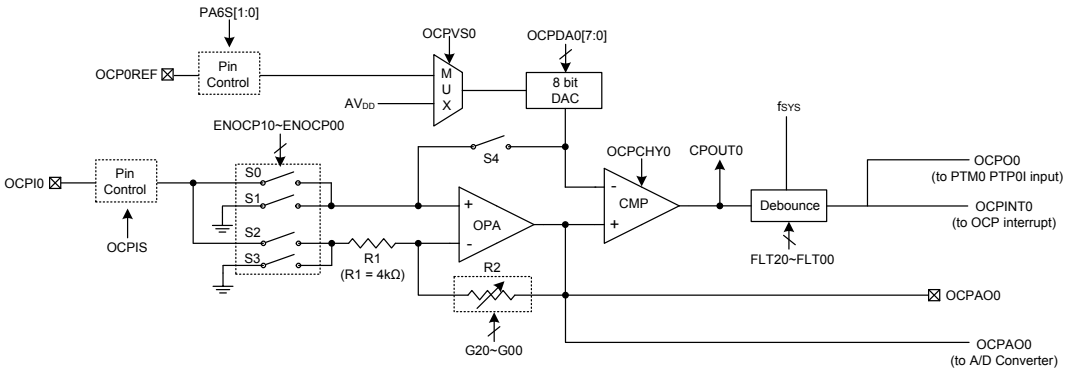
Step 4. Set O0OF[5:0]=111111 and read the OP0OUT bit.

Step 5. Decrease the O0OF[5:0] value by 1 and then read the OP0OUT bit.
If the OP0OUT bit value has not changed, then repeat Step 5 until the OP0OUT bit value has changed.
If the OP0OUT bit value has changed, record the O0OF[5:0] value as V_{OS2} and then go to Step 6.

Step 6. Restore O0OF[5:0]=V_{OS}=(V_{OS1}+V_{OS2})/2. The offset Calibration procedure is now finished.
If (V_{OS1}+V_{OS2})/2 is not an integer hen discard the decimal, where V_{OS}=V_{OUT} - V_{IN}.

Over Current Protection – OCP

The device includes an over current protection function. The OCP detects an input voltage which is proportional to the monitored source current. If the input voltage is larger than the reference voltage set by the DAC, the OCP will generate an output signal to indicate that an over current event has occurred, and triggers an interrupt to inform the MCU.



Over Current Protection Circuit

Over Current Protection Operation

The OCP circuit input signal originates from OCPI0, the source of which can be selected by the OCPIS bit in the SSCTL register. After this, four switches S0~S3 form a mode select function. An operational amplifier and two resistors form a PGA function. The PGA gain can be positive or negative determined by the input voltage connected to the positive or negative input of the PGA. The 8-bit DAC is used to generate a reference voltage. The comparator compares the reference voltage and the amplified output voltage. Finally the comparator output CPOUT0 is filtered to generate a hard decision signal OCPINT0, if an over current event occurs, the signal will trigger an interrupt to inform the MCU.

The OCPO0 signal is also the de-bounced version of CPOUT0. It can be internally connected to the PTP0I pin by the PT0IS bit and used for capture input function of the PTM0.

The OCP input signal amplified by internal operational amplifier can be directly output on the OCPAO0 pin, and also be internally connected to the A/D converter input AN0 selected by the SACS3~SACS0 bits in the SADC0 register for the amplified input voltage read.

Over Current Protection Registers

The OCPC00 and OCPC10 registers are control registers which used to control the OCP operating modes, the OPA function and de-bounce time. The OCPDA0 register is used to control D/A converter reference voltage. The OCPOCAL0 and OCPCCAL0 registers are used to cancel out the operational amplifier and comparator input offset.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SSCTL	—	—	PT0IS	OCPIS	—	—	HXT_EX	EN_VDET
OCPC00	ENOCP10	ENOCP00	OCPVS0	OCPCHY0	—	—	—	OCPO0
OCPC10	—	—	G20	G10	G00	FLT20	FLT10	FLT00
OCPDA0	D7	D6	D5	D4	D3	D2	D1	D0
OCPOCAL0	OOFM0	ORSP0	OOF50	OOF40	OOF30	OOF20	OOF10	OOF00
OCPCCAL0	CPOUT0	COFM0	CRSP0	COF40	COF30	COF20	COF10	COF00

Over Current Protection Registers List

SSCTL Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PT0IS	OCPIS	—	—	HXT_EX	EN_VDET
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PT0IS**: PTP0I input source pin selection
0: From PA0
1: From internal OCP0O signal

Bit 5 **OCPIS**: OCPI0 input source pin selection
0: From PD6
1: From PD7

Bit 3~2 Unimplemented, read as "0"

Bit 1 **HXT_EX**: HXT PWM output control
Described elsewhere.

Bit 0 **EN_VDET**: High voltage power supply detection function control
Described elsewhere.

OCPC00 Register

Bit	7	6	5	4	3	2	1	0
Name	ENOCP10	ENOCP00	OCPVS0	OCPCHY0	—	—	—	OCPO0
R/W	R/W	R/W	R/W	R/W	—	—	—	R
POR	0	0	0	0	—	—	—	0

Bit 7~6 **ENOCP10~ENOCP00**: OCP function operating mode selection
00: OCP function is disabled, S1and S3 on, S0 and S2 off
01: Non-inverting mode, S0 and S3 on, S1 and S2 off
10: Inverting mode, S1 and S2 on, S0 and S3 off
11: Calibration mode, S1 and S3 on, S0 and S2 off
Note: If the over current protection function is disabled, this results in the operational amplifier, comparator, D/A converter and debounce functions all being switched off, as well as the comparator output and OCPAO0 both being low.

Bit 5 **OCPVS0**: OCP DAC reference voltage selection
0: From AV_{DD}
1: From OCP0REF pin

Bit 4 **OCPCHY0**: OCP Comparator hysteresis function control
0: Disable
1: Enable

Bit 3~1 Unimplemented, read as "0"

Bit 0 **OCPO0**: OCP digital output bit
0: The monitored source current is not over
1: The monitored source current is over

OCPC10 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	G20	G10	G00	FLT20	FLT10	FLT00
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **G20~G00**: PGA R2/R1 ratio selection, it will define PGA gain in inverting/non-inverting mode
000: Unity gain buffer (non-inverting mode) or R2/R1=1(inverting mode)
001: R2/R1=5
010: R2/R1=10
011: R2/R1=15
100: R2/R1=20
101: R2/R1=30
110: R2/R1=40
111: R2/R1=50

Bit 2~0 **FLT20~FLT00**: OCP output filter debounce time selection
000: Bypass, without 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: $t_{DEB}=1/f_{SYS}$

OCPDA0 Register

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

Bit 7~0 **D7~D0**: OCP DAC output voltage control bits
OCP DAC Output=(DAC reference voltage/256) × N, N=OCPDA0[7:0]

OCPOCAL0 Register

Bit	7	6	5	4	3	2	1	0
Name	OOFM0	ORSP0	OOF50	OOF40	OOF30	OOF20	OOF10	OOF00
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 **OOFM0**: OCP Operational Amplifier Normal Operation or Input Offset Calibration Mode selection
0: Normal Operation
1: Input Offset Calibration Mode

Bit 6 **ORSP0**: OCP Operational Amplifier Input Offset Voltage Calibration Reference selection
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 5~0 **OOF50~OOF00**: OCP Operational Amplifier Input Offset Voltage Calibration value

OCPCCAL0 Register

Bit	7	6	5	4	3	2	1	0
Name	CPOUT0	COFM0	CRSP0	COF40	COF30	COF20	COF10	COF00
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 **CPOUT0**: OCP Comparator Output, positive logic (read only)

Bit 6 **COFM0**: OCP Comparator Normal Operation or Input Offset Calibration Mode selection
0: Normal Operation
1: Input Offset Calibration Mode

Bit 5 **CRSP0**: OCP Comparator Input Offset Calibration Reference Input select
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 4~0 **COF40~COF00**: OCP Comparator Input Offset Calibration value

Input Voltage Range

Together with different PGA operating modes, the input voltage on the OCP pin can be positive or negative for flexible operation.

- For $V_{IN} > 0$, the PGA operates in the non-inverting mode and the output voltage of the PGA is:
$$VO_{PGA}=(1+R_2/R_1)\times V_{IN} \quad (2)$$
- When the PGA operates in the non-inverting mode, it also provides a unity gain buffer function. If ENOCP10~ENOCP00=01 and G20~G00=000, the PGA gain will be 1 and the PGA is configured as a unity gain buffer. The switches S2 and S3 will be off internally and the output voltage of the PGA is:
$$VO_{PGA}=V_{IN} \quad (3)$$
- For $0 > V_{IN} > -0.4V$, the PGA operates in the inverting mode, the output voltage of the PGA is:
$$VO_{PGA}= -(R_2/R_1)\times V_{IN} \quad (4)$$
Note: If V_{IN} is negative, it cannot be lower than -0.4V which will result in current leakage.

Offset Calibration

The OCP circuit has 4 operating modes controlled by ENOCP10~ENOCP00, one of them is calibration mode. In calibration mode, the operational amplifier and comparator offset can be calibrated.

OCP Operational Amplifier Calibration

Step 1. Set ENOCP10~ENOCP00=11 and OOFM0=1, the OCP will operate in the operational amplifier input offset Calibration mode.

Step 2. Set OOF50~OOF00=000000 and then read the CPOUT0 bit.

Step 3. Let OOF50~OOF00=OOF50~OOF00+1 and then read the CPOUT0 bit.
If the CPOUT0 bit has not changed, then repeat Step 3 until the CPOUT0 bit has changed.
If the CPOUT0 bit has changed, record the OOF50~OOF00 value as V_{OS1} and then go to Step 4.

Step 4. Set OOF50~OOF00=111111 and read the CPOUT0 bit.

Step 5. Let OOF50~OOF00=OOF50~OOF00-1 and then read the CPOUT0 bit.
If the CPOUT0 bit has not changed, then repeat Step 5 until the CPOUT0 bit has changed.
If the CPOUT0 bit has changed, record the OOF50~OOF00 value as V_{OS2} and then go to Step 6.

Step 6. Restore OOF50~OOF00=$V_{OS}=(V_{OS1}+V_{OS2})/2$, the offset Calibration procedure is now finished.

OCP Comparator Calibration

Step 1. Set ENOCP10~ENOCP00=11 and COFM0=1, the OCP will now operate in the comparator input offset calibration mode.

Step 2. Set COF40~COF00=00000 and read the CPOUT0 bit.

Step 3. Let COF40~COF00=COF40~COF00+1 and then read the CPOUT0 bit.
If the CPOUT0 bit has not changed, then repeat Step 3 until the CPOUT0 bit has changed.
If the CPOUT0 bit has changed, record the COF40~COF00 value as V_{OS1} and then go to Step 4.

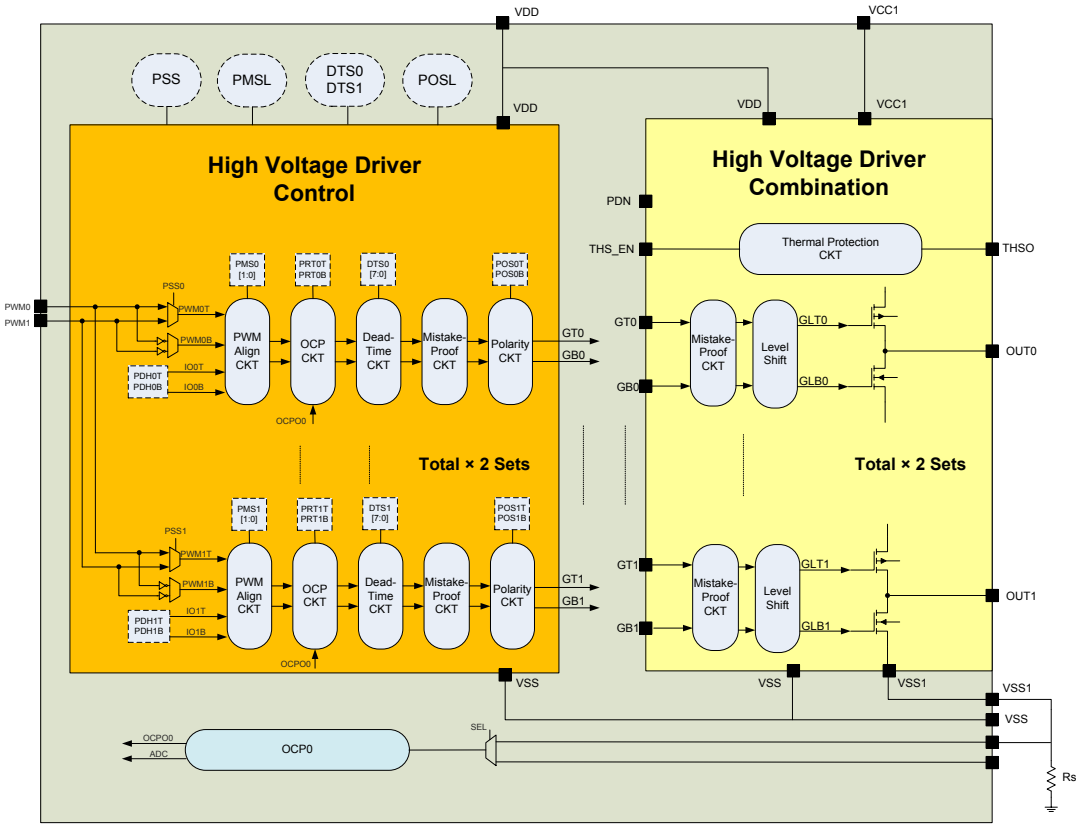
Step 4. Set COF40~COF00=11111 and then read the CPOUT0 bit.

Step 5. Let COF40~COF00=COF40~COF00-1 and then read the CPOUT0 bit.
If the CPOUT0 bit has not changed, then repeat Step 5 until the CPOUT0 bit has changed.
If the CPOUT0 bit has changed, record the COF40~COF00 value as V_{OS2} and then go to Step 6.

Step 6. Restore COF40~COF00=V_{OS}=(V_{OS1}+V_{OS2})/2, the offset Calibration procedure is now finished.

High Voltage Driver

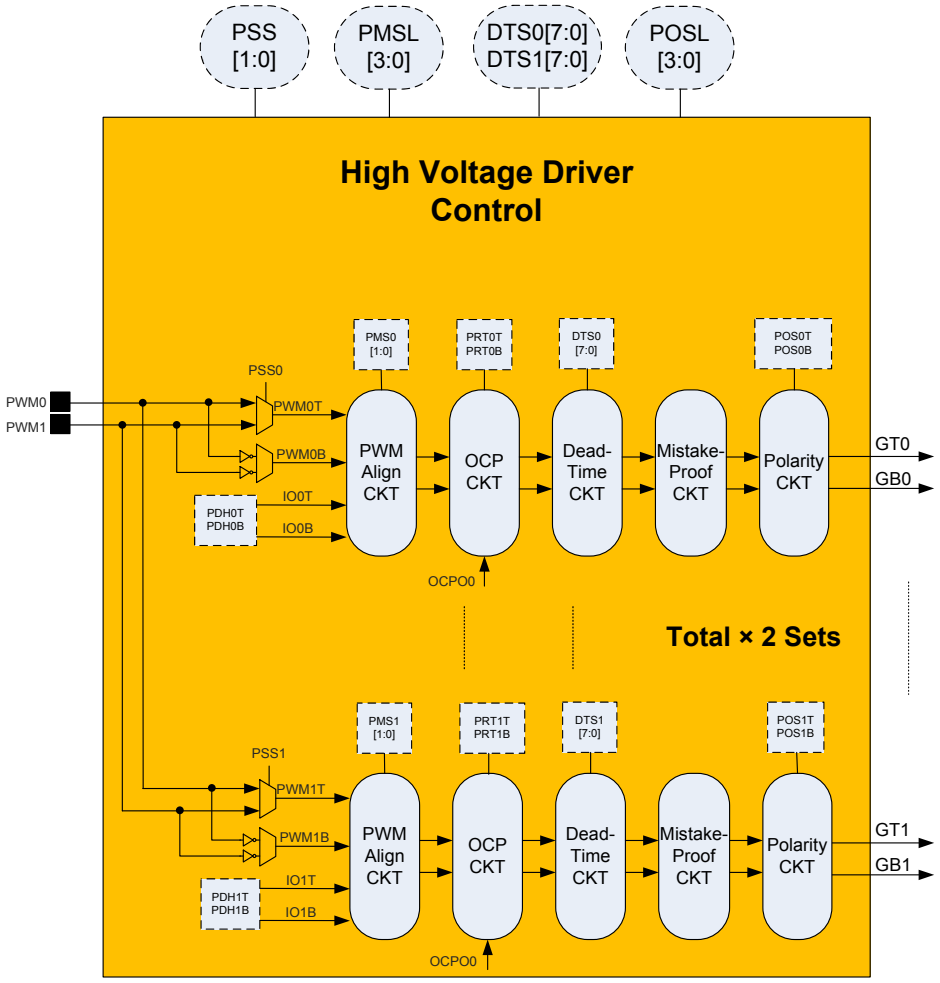
The device includes a high voltage driver function, which is composed of two sections, the high voltage driver control and the high voltage driver combination.



High Voltage Driver Structure

High Voltage Driver Control

There are two groups of high voltage driver control circuits. Each group is composed of five circuits, PWM align circuit, over current protection circuit, dead time control circuit, mistake-proof circuit and polarity control circuit.



Note: PWM0 stands for the PWM output signal from PTP2 of the PTM2. PWM1 stands for the PWM output signal from PTP3 of the PTM3.

High Voltage Driver Control Registers

The high voltage driver control is implemented using several registers. These registers are used to select the PWM input source, the PWM align mode, enable or disable the over current protection function, setup the dead time and control the output polarity, etc.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PSS	—	—	—	—	—	—	PSS1	PSS0
PMSL	—	—	—	—	PMS11	PMS10	PMS01	PMS00
PDHCL	—	—	—	—	PDH1T	PDH1B	PDH0T	PDH0B
DTS0	DT0CKS1	DT0CKS0	DT0E	DT0D4	DT0D3	DT0D2	DT0D1	DT0D0
DTS1	DT1CKS1	DT1CKS0	DT1E	DT1D4	DT1D3	DT1D2	DT1D1	DT1D0
POSL	—	—	—	—	POS1T	POS1B	POS0T	POS0B
PRTL	—	—	—	—	PRT1T	PRT1B	PRT0T	PRT0B
OPCL	—	—	—	—	OCPTE1	—	OCPTE0	—
RPDH	—	—	—	—	—	—	RPDH1	RPDH0

High Voltage Driver Control Registers List

PSS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PSS1	PSS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1 **PSS1**: Select PWM Source for OUT1 High Voltage Level Shift Driver
0: PWM source from PTM2 output
1: PWM source from PTM3 output

Bit 0 **PSS0**: Select PWM Source for OUT0 High Voltage Level Shift Driver
0: PWM source from PTM2 output
1: PWM source from PTM3 output

PMSL Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PMS11	PMS10	PMS01	PMS00
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 **PMS11~PMS10**: Select the PWM align mode of OUT1 high voltage level shift driver
00: Complementary PWM signal pair
01: Top side non-complementary PWM signal
10: Bottom side non-complementary PWM signal
11: I/O type control (PDH1T~PDH1B control top/bottom sides respectively)

Bit 1~0 **PMS01~PMS00**: Select the PWM align mode of OUT0 high voltage level shift driver
00: Complementary PWM signal pair
01: Top side non-complementary PWM signal
10: Bottom side non-complementary PWM signal
11: I/O type control (PDH0T~PDH0B control top/bottom sides respectively)

PDHCL Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PDH1T	PDH1B	PDH0T	PDH0B
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 **PDH1T~PDH1B**: Control the Top/Bottom outputs of the OUT1 high voltage level shift driver
00: Top/Bottom turn off
01: Top turn off / Bottom turn on, output low
10: Top turn on / Bottom turn off, output high
11: Top/Bottom turn off (Top/Bottom turn on is forbidden)

Bit 1~0 **PDH0T~PDH0B**: Control the Top/Bottom outputs of the OUT0 high voltage level shift driver
00: Top/Bottom turn off
01: Top turn off / Bottom turn on, output low
10: Top turn on / Bottom turn off, output high
11: Top/Bottom turn off (Top/Bottom turn on is forbidden)

OPCL Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	OCPTE1	—	OCPTE0	—
R/W	—	—	—	—	R/W	—	R/W	—
POR	—	—	—	—	0	—	0	—

Bit 7~4 Unimplemented, read as "0"

Bit 3 **OCPTE1**: Over current protection function enable control of the OUT1 high voltage level shift driver
0: Disable
1: Enable
When the bit is high, the over current protection function of the OUT1 high voltage level shift driver will be enabled, if an over current condition occurs, the Top/Bottom outputs will be controlled by the PRT1T~PRT1B bits.

Bit 2 Unimplemented, read as "0"

Bit 1 **OCPTE0**: Over current protection function enable control of the OUT0 high voltage level shift driver
0: Disable
1: Enable
When the bit is high, the over current protection function of the OUT0 high voltage level shift driver will be enabled, if an over current condition occurs, the Top/Bottom outputs will be controlled by the PRT0T~PRT0B bits.

Bit 0 Unimplemented, read as "0"

PRTL Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PRT1T	PRT1B	PRT0T	PRT0B
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 **PRT1T~PRT1B**: Control the Top/Bottom Gate outputs of the OUT1/PDH1 when an over current condition occurs
00: Top/Bottom turn off
01: Top turn off / Bottom turn on, output low
10: Top turn on / Bottom turn off, output high
11: Top/Bottom turn off
These bits are available only when the over current protection circuit of the OUT1 high voltage level shift driver is enabled by setting the OCPTE1 bit high.

Bit 1~0 **PRT0T~PRT0B**: Control the Top/Bottom Gate outputs of the OUT0/PDH0 when an over current condition occurs
00: Top/Bottom turn off
01: Top turn off / Bottom turn on, output low
10: Top turn on / Bottom turn off, output high
11: Top/Bottom turn off
These bits are available only when the over current protection circuit of the OUT0 high voltage level shift driver is enabled by setting the OCPTE0 bit high.

DTS1 Register

Bit	7	6	5	4	3	2	1	0
Name	DT1CKS1	DT1CKS0	DT1E	DT1D4	DT1D3	DT1D2	DT1D1	DT1D0
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 **DT1CKS1~DT1CKS0**: Dead time clock source (f_{DT1}) selection of the OUT1 high voltage level shift driver
00: $f_{DT1}=f_{SYS}$
01: $f_{DT1}= f_{SYS}/2$
10: $f_{DT1}=f_{SYS}/4$
11: $f_{DT1}=f_{SYS}/8$

Bit 5 **DT1E**: Dead time insertion control of the OUT1 high voltage level shift driver
0: No dead time insertion
1: Dead time insertion
If this bit is set high to enable the dead time insertion, the dead time is furtherly controlled by the DT1D4~DT1D0 bits.

Bit 4~0 **DT1D4~DT1D0**: Dead time counter for dead time unit of the OUT1 high voltage level shift driver
Dead time=(DT1D[4:0]+1) / f_{DT1}

DTS0 Register

Bit	7	6	5	4	3	2	1	0
Name	DT0CKS1	DT0CKS0	DT0E	DT0D4	DT0D3	DT0D2	DT0D1	DT0D0
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 **DT0CKS1~DT0CKS0**: Dead time clock source (f_{DT0}) selection of the OUT0 high voltage level shift driver
00: $f_{DT0}=f_{SYS}$
01: $f_{DT0}=f_{SYS}/2$
10: $f_{DT0}= f_{SYS}/4$
11: $f_{DT0}=f_{SYS}/8$

Bit 5 **DT0E**: Dead time insertion control of the OUT0 high voltage level shift driver
0: No dead time insertion
1: Dead time insertion
If this bit is set high to enable the dead time insertion, the dead time is furtherly controlled by the DT0D4~DT0D0 bits.

Bit 4~0 **DT0D4~DT0D0**: Dead time counter for dead time unit of the OUT0 high voltage level shift driver
Dead time=(DT0D[4:0]+1) / f_{DT0}

POSL Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	POS1T	POS1B	POS0T	POS0B
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **POS1T**: Select the Top polarity the OUT1 high voltage level shift driver
0: Non-inverted
1: Inverted

Bit 2 **POS1B**: Select the Bottom polarity of the OUT1 high voltage level shift driver
0: Non-inverted
1: Inverted

Bit 1 **POS0T**: Select the Top polarity the OUT0 high voltage level shift driver
0: Non-inverted
1: Inverted

Bit 0 **POS0B**: Select the Bottom polarity of the OUT0 high voltage level shift driver
0: Non-inverted
1: Inverted

RPDH Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	RPDH1	RPDH0
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

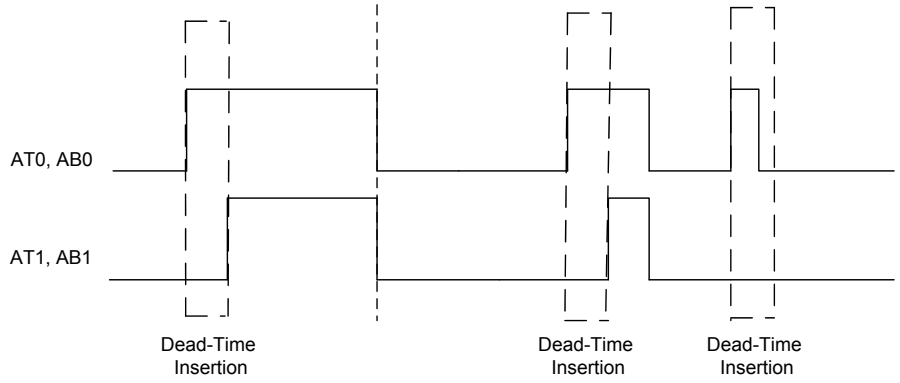
Bit 1 **RPDH1**: PDH1/OUT1 high voltage output status read back signal
0: Low
1: High

Bit 0 **RPDH0**: PDH0/OUT0 high voltage output status read back signal
0: Low
1: High

Dead Time

During the transition of the external driving transistors, there may be a moment 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. The dead time should be configured with a range of 0.3μs~5μs. Its duration is determined by the DTS0~DTS1 registers.

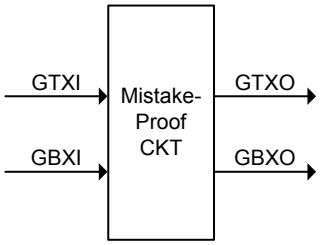
The following shows the dead time insertion timing. Note that after the dead time function is enabled, it is inserted at each rising edge only, the falling edges remain unchanged.



Dead Time Insertion Timing

Mistake-Proof for the High Voltage Driver Control

Incorrect write operations or external factors such an ESD condition, may cause incorrect on/off control resulting in the top and bottom sides of external transistors being both turned on simultaneously. A mistake-proof circuit is provided to avoid such situations by forcing both the output MOS transistors to an off state to protect the motor.

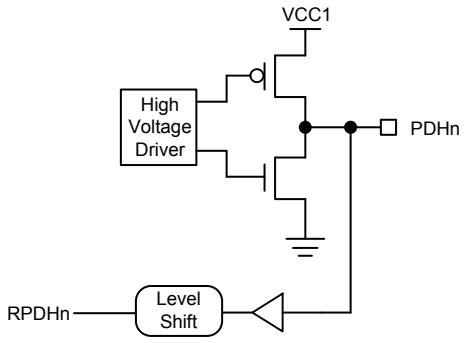


GTXI	GBXI	GTXO	GBXO
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

Note: 0 means MOS off, 1 means MOS on. The external MOS gate output status is determined by the Polarity Control circuit, and whether the output is inverted or not is controlled by the POSL register.

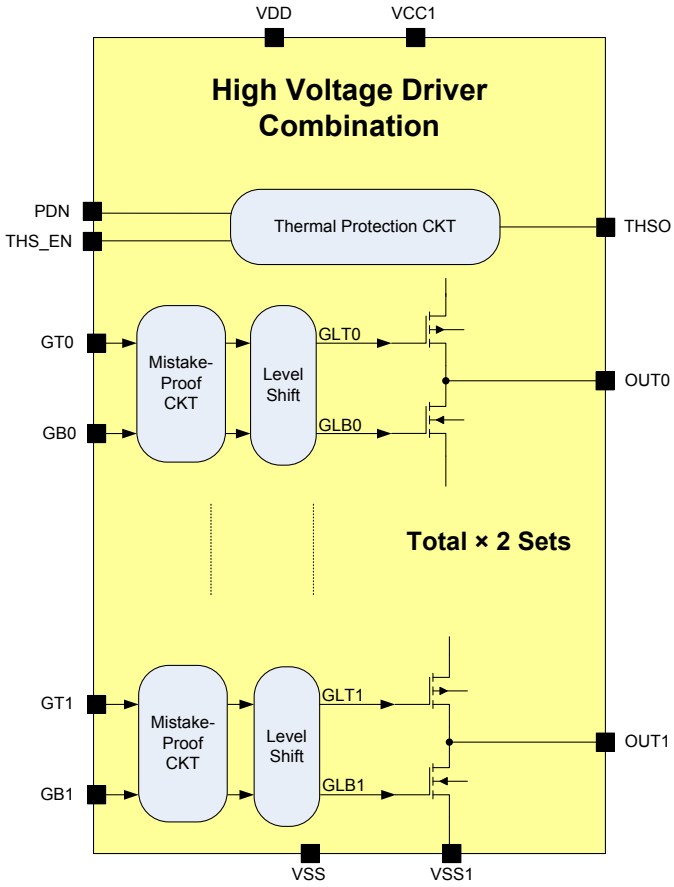
High Voltage Output Read Back

The actual output status on the PDHn/OUTn pin can be read back using the RPDH1~RPDH0 bits.



High Voltage Driver Combination

The high voltage driver combination contains two groups of high voltage driving circuits. Each group is mainly composed of a mistake-proof circuit, a level shift circuit and a thermal protection circuit. With the two integrated 12V high voltage process level shift circuits, the high voltage driver combination provides two output lines, OUT0~OUT1, which can be used for driving different polarity outputs of the DC Motor Driver Top side (PMOS) or Bottom side (NMOS) to support multiplie external driving circuits.



High Voltage Driver Combination Block Diagram

High Voltage Driver Combination Registers

The overall operation of the high voltage driver combination is controlled using the HVC register. The register controls the high voltage driver combination enable and disable operation and setups a thermal protection function.

- **HVC Register**

Bit	7	6	5	4	3	2	1	0
Name	PDN	THS_EN	THSO	—	—	—	—	—
R/W	R/W	R/W	R	—	—	—	—	—
POR	0	0	0	—	—	—	—	—

Bit 7 **PDN**: High voltage driver combination circuit on/off control
0: On
1: Off

Bit 6 **THS_EN**: Thermal protection function enable/disable control
0: Disable, no thermal protection
1: Enable, has thermal protection

Bit 5 **THSO**: Thermal protection flag
0: No over thermal codition occurs
1: Over thermal condition occurs

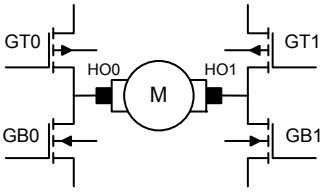
Bit 4 Unimplemented, read as "0"

Bit 3~0 Reserved bits, these bits should be kept low.

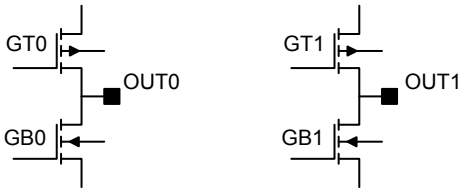
High Voltage Driver Combination Applications

The high voltage driver combination circuits are available for a variety of applications according to different product requirements. They can drive different external components using different driving currents. Each PMOS or NMOS has its individual switch, so that a variety of combinations are allowed. The following are two examples.

- H-Bridge Group: Directly drive the DC Motor

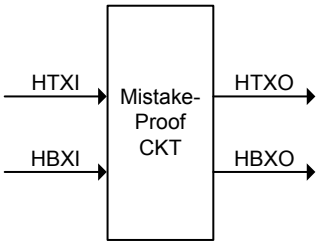


- PMOS/NMOS used independently



Mistake-Proof for the High Voltage Driver Combination

Incorrect write operations or external factors such an ESD condition, may cause incorrect on/off control resulting in the top and bottom sides of external transistor being both turned on simultaneously. A mistake-proof circuit is provided to avoid such situation.



HTXI	HBXI	HTXO	HBXO	PDHn
0	0	0	0	0
0	1	0	1	0
1	0	1	0	1
1	1	0	1	0

Note: 0 means MOS off, 1 means MOS on.

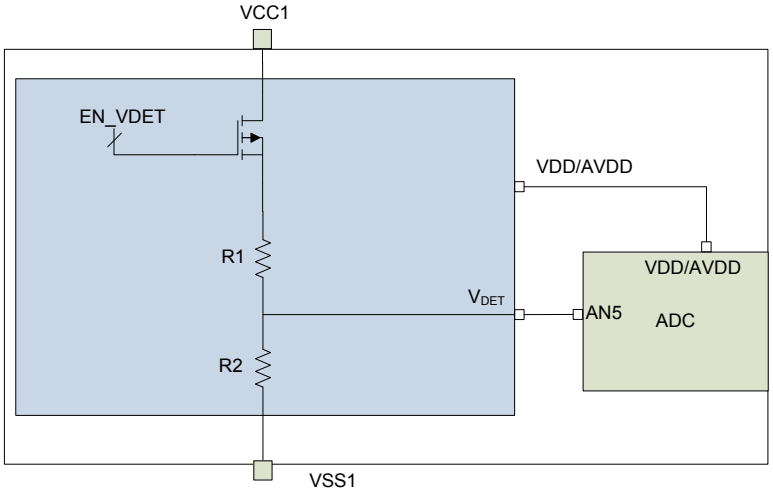
High Voltage Driver Combination Operation and Thermal Protection

The on/off function of the whole high voltage driver combination circuit is controlled using the PDN bit in the HVC register. The circuit can be powered on by clearing the PDN bit to zero, and powered off by setting the PDN bit high. The high voltage driver combination circuit contains a thermal protection function. The THS_EN bit is used to enable or disable the thermal protection function. The THSO bit is used to monitor temperature conditions, if the temperature exceeds the preset range, the bit will change from 0 to 1 to indicate an over thermal occurrence.

If the thermal protection has been enabled, the THSO bit in the HVC register can be used to check whether an over thermal condition has occurred. The THSO bit has an initial value of 0. If the detected temperature exceeds the preset value, the bit will be automatically set to 1. Additionally, the thermal protection flag in the interrupt control register will also be set high, if the corresponding interrupt has been enabled, a thermal protection interrupt will be generated to inform the MCU.

High Voltage Power Supply Detection

The device provides a high voltage power supply detection circuit for the high voltage function. The VCC1 power supply detection function is enabled or disabled by the EN_VDET bit. This detection circuit can output a power supply divided voltage using divider resistors. This divided voltage, V_{DET}, can also be read by connecting it to the A/D converter as the internal input signal AN5.

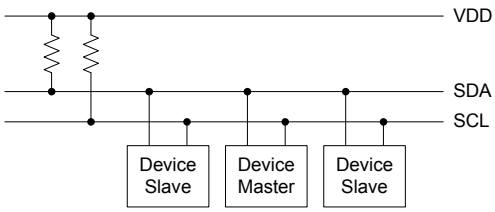


Note: R1:R2=4:1 (12K/3K), $V_{DET}=R2/(R1+R2)\times V_{CC1}=0.2V_{CC1}$.

VCC1 Power Supply Detection Circuit

I²C Interface

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



I²C Master/Slave Bus Connection

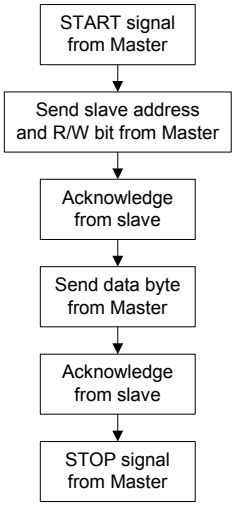
I²C Interface Operation

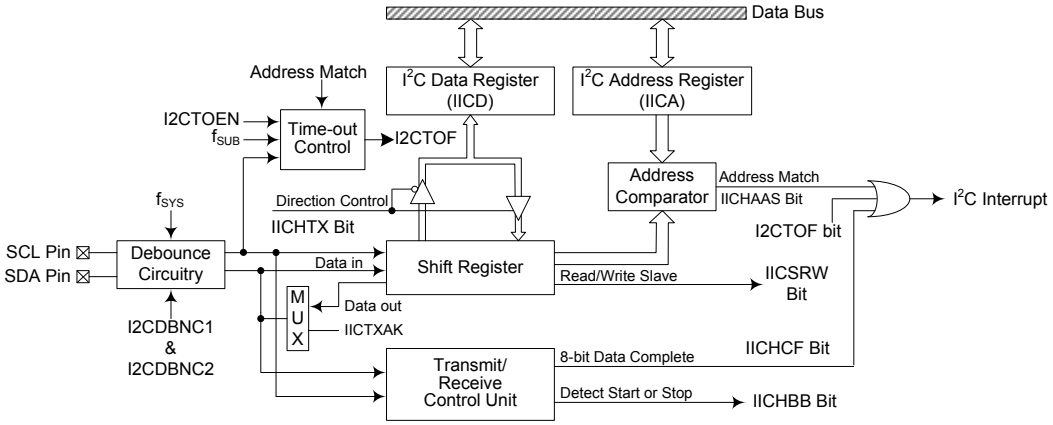
The I²C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs.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

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

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

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





I²C Block Diagram

The I2CDBNC1 and I2CDBNC0 bits determine the debounce time of the I²C interface. This uses the system clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required I²C data transfer speed, there exists a relationship between the system clock, f_{SYS}, and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

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

I²C Minimum f_{SYS} Frequency

I²C Registers

There are four control registers associated with the I²C bus, IICC0, IICC1, IICA and I2CTOC, and one data register, IICD. Further explanation on each of the bits is given below. The I2CTOC register is described in the I²C Time-out Control section.

Register Name	Bit							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	I2CDBNC1	I2CDBNC0	IICEN	—
IICC1	IICHCF	IICHAAS	IICHBB	IICHTX	IICTXAK	IICSRW	IICRNIC	IICRXAK
IICD	IICDD7	IICDD6	IICDD5	IICDD4	IICDD3	IICDD2	IICDD1	IICDD0
IICA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
I2CTOC	I2CTOEN	I2CTOF	I2CTOS5	I2CTOS4	I2CTOS3	I2CTOS2	I2CTOS1	I2CTOS0

I²C Registers List

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

IICC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	I2CDBNC1	I2CDBNC0	IICEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **I2CDBNC1~I2CDBNC0**: I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
10: 4 system clock debounce
11: 4 system clock debounce

Bit 1 **IICEN**: I²C enable
0: Disable
1: Enable

Bit 0 Unimplemented, read as "0"

IICC1 Register

Bit	7	6	5	4	3	2	1	0
Name	IICHCF	IICHAAS	IICHBB	IICHTX	IICTXAK	IICSRW	IICRNIC	IICRXAK
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

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

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

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

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

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

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

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

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

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

The IICHAAS 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 **IICHBB**: I²C Bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy

The IICHBB 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 **IICHTX**: Select I²C slave device is transmitter or receiver
0: Slave device is the receiver
1: Slave device is the transmitter

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

The IICTXAK 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 IICTXAK bit to "0" before further data is received.

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

The IICSRW 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 IICHAAS flag is set high, the slave device will check the IICSRW flag to determine whether it should be in transmit mode or receive mode. If the IICSRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the IICSRW 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 **IICRNIC**: I²C Address Match Wake Up Control
0: Disable
1: Enable

This bit should be set to 1 to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IICRNIC bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

Bit 0 **IICRXAK**: I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave do not receive acknowledge flag

The IICRXAK flag is the receiver acknowledge flag. When the IICRXAK 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 IICRXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the IICRXAK 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.

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

IID Register

Bit	7	6	5	4	3	2	1	0
Name	IICDD7	IICDD6	IICDD5	IICDD4	IICDD3	IICDD2	IICDD1	IICDD0
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 **IICDD7~IICDD0**: I²C Data Buffer bit 7~bit 0

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

IIA Register

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	0	0	0	0	0	0	0	—

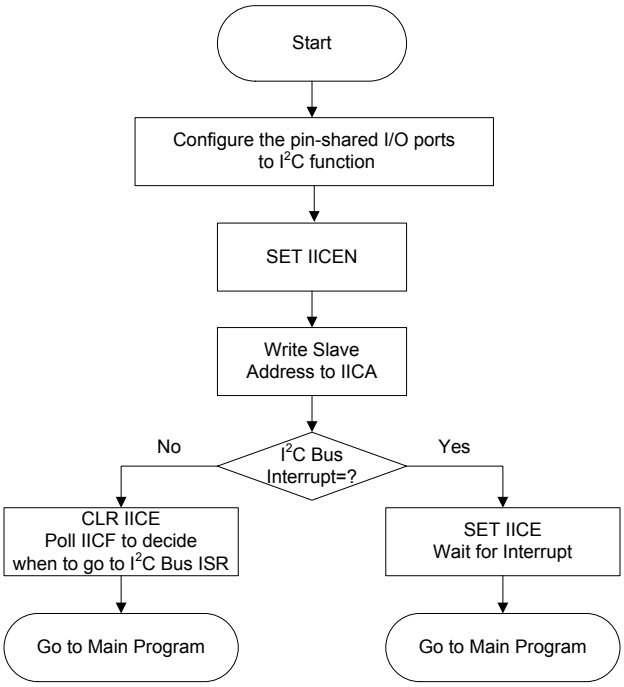
Bit 7~1 **IICA6~IICA0**: I²C slave address
IICA6~IICA0 is the I²C slave address bit 6 ~ bit 0. Bits 7~ 1 of the IICA register define the device slave address. Bit 0 is not defined. When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the IICA register, the slave device will be selected.

Bit 0 Unimplemented, read as "0"

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 IICHAAS bit in the IICC1 register will be set and an I²C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the IICHAAS and I2CTOF bits to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer completion or 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 IICSRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
  Configure the pin-shared I/O ports to I²C function.
- Step 2
  Set IICEN bit in the IICC0 register to "1" to enable the I²C bus.
- Step 3
  Write the slave address of the device to the I²C bus address register IICA.
- Step 4
  Set the IICE interrupt enable bit to enable the I²C 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 IICHBB 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.

Slave Address

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

As an I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the IICHAAS and I2CTOF 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 I²C time-out. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the SCL line.

I²C Bus Read/Write Signal

The IICSRW bit in the IICC1 register defines whether the master device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the IICSRW 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 IICSRW 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.

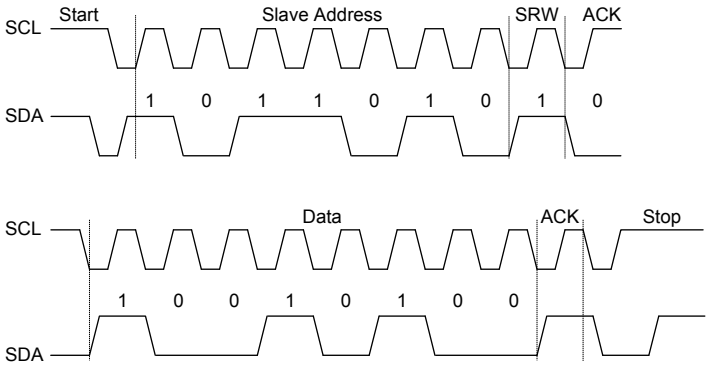
I²C Bus Slave Address Acknowledge Signal

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 IICHAAS flag is high, the addresses have matched and the slave device must check the IICSRW flag to determine if it is to be a transmitter or a receiver. If the IICSRW flag is high, the slave device should be setup to be a transmitter so the IICHTX bit in the IICC1 register should be set to "1". If the IICSRW flag is low, then the microcontroller slave device should be setup as a receiver and the IICHTX bit in the IICC1 register should be set to "0".

I²C Bus Data and Acknowledge Signal

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

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

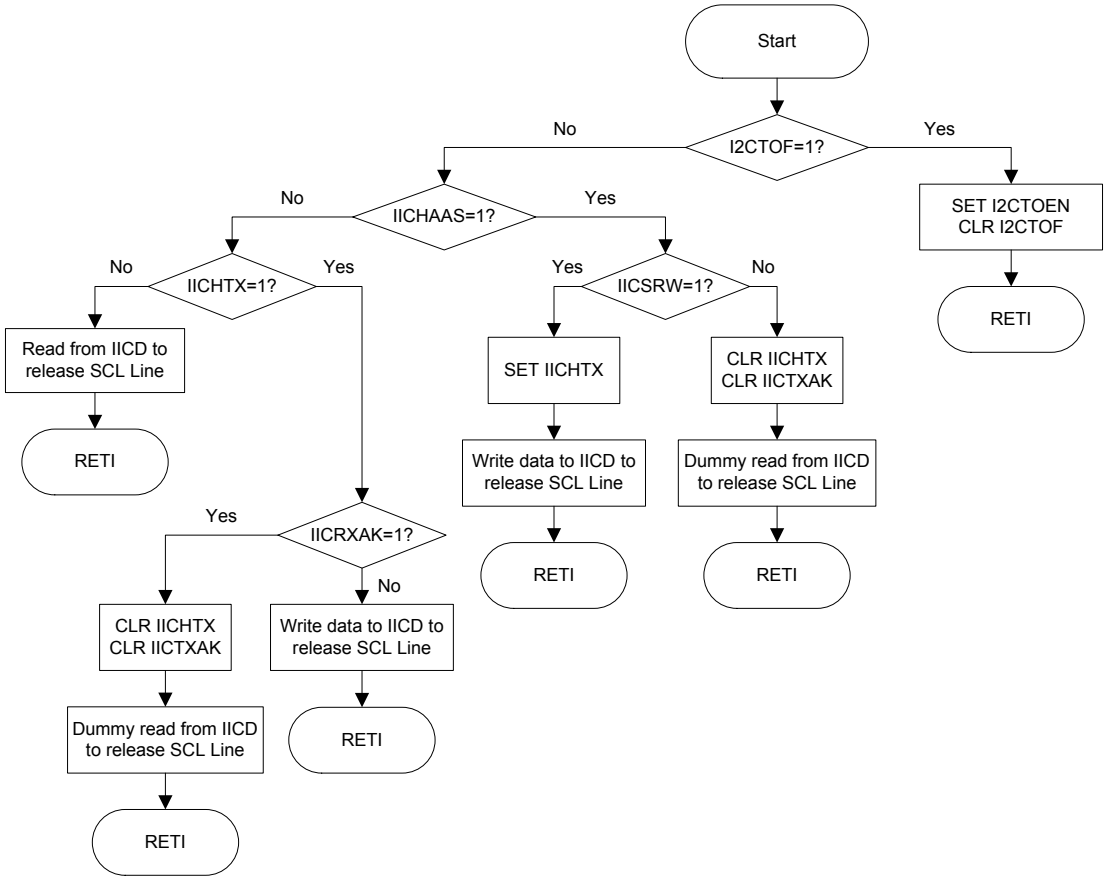


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

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

I²C Communication Timing Diagram

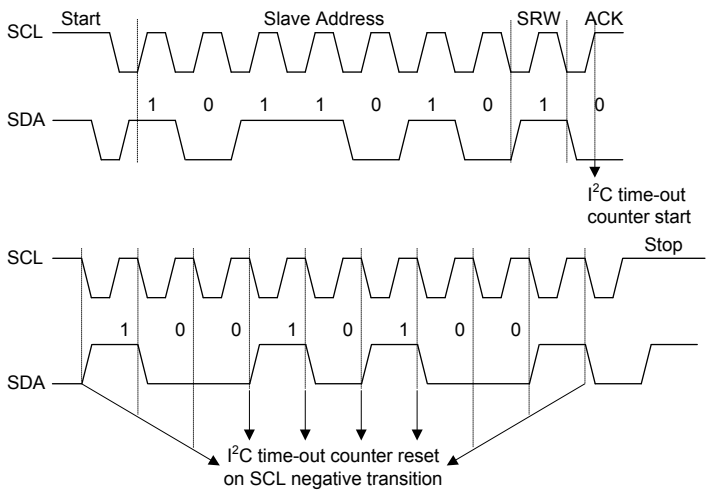


I²C Bus ISR Flow Chart

I²C Time-out Control

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

The time-out counter starts counting on an I²C bus "START" & "address match" condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out setup by the I2CTOC 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 I2CTOEN bit will be cleared to zero and the I2CTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the I²C interrupt vector. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

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

The I2CTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using bits in the I2CTOC register. The time-out time is given by the formula:

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

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

I2CTOC Register

Bit	7	6	5	4	3	2	1	0
Name	I2CTOEN	I2CTOF	I2CTOS5	I2CTOS4	I2CTOS3	I2CTOS2	I2CTOS1	I2CTOS0
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 **I2CTOEN**: I²C Time-out Control
0: Disable
1: Enable

Bit 6 **I2CTOF**: Time-out flag
0: No time-out
1: Time-out occurred

Bit 5~0 **I2CTOS5~I2CTOS0**: Time-out Definition
I²C time-out clock source is $f_{SUB}/32$.
I²C time-out time is given by: (I2CTOS[5:0]+1) × $(32/f_{SUB})$

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 two external interrupts and several internal interrupts functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Base and EEPROM.

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. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI4 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the operational amplifier digital output interrupt and the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pin	INTnE	INTnF	n=0~1
Over Current Protection	OCP0E	OCP0F	—
Thermal Protection	THE	THF	—
Operational Amplifier	OPA0E	OPA0F	—
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~4
LVD	LVDE	LVDF	—
I^2C	IICE	IICF	—
EEPROM write operation	EPWE	EPWF	—
Time Base	TBnE	TBnF	n=0~1
PTM	PTMPnE	PTMPnF	n=0~3
	PTMAnE	PTMAnF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	OPA0S1	OPA0S0	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	OCP0F	INT1F	INT0F	OCP0E	INT1E	INT0E	EMI
INTC1	ADF	OPA0F	THF	—	ADE	OPA0E	THE	—
INTC2	MF3F	MF2F	MF1F	MF0F	MF3E	MF2E	MF1E	MF0E
INTC3	MF4F	EPWF	IICF	LVDF	MF4E	EPWE	IICE	LVDE
MFI0	—	—	PTMA2F	PTMP2F	—	—	PTMA2E	PTMP2E
MFI1	—	—	PTMA1F	PTMP1F	—	—	PTMA1E	PTMP1E
MFI2	—	—	PTMA0F	PTMP0F	—	—	PTMA0E	PTMP0E
MFI3	—	—	PTMA3F	PTMP3F	—	—	PTMA3E	PTMP3E
MFI4	—	—	TB1F	TB0F	—	—	TB1E	TB0E

Interrupt Registers List

INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	OPA0S1	OPA0S0	INT1S1	INT1S0	INT0S1	INT0S0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **OPA0S1~OPA0S0**: interrupt edge control for operational amplifier digital output
00: Disable
01: Rising edge (when positive input voltage > negative input voltage)
10: Falling edge (when negative input voltage > positive input voltage)
11: Rising and falling edges (when positive input voltage > negative input voltage or negative input voltage > positive input voltage)

Bit 3~2 **INT1S1~INT1S0**: interrupt edge control for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: interrupt edge control for INT0 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	OCP0F	INT1F	INT0F	OCP0E	INT1E	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 **OCP0F**: OCP interrupt request flag
0: No request
1: Interrupt request

Bit 5 **INT1F**: INT1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INT0F**: INT0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **OCP0E**: OCP interrupt control
0: Disable
1: Enable

Bit 2 **INT1E**: INT1 interrupt control
0: Disable
1: Enable

Bit 1 **INT0E**: INT0 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	ADF	OPA0F	THF	—	ADE	OPA0E	THE	—
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	—
POR	0	0	0	—	0	0	0	—

Bit 7 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 6 **OPA0F**: Operational Amplifier interrupt request flag
0: No request
1: Interrupt request

Bit 5 **THF**: High Voltage Driver Combination Thermal Protection interrupt request flag
0: No request
1: Interrupt request

Bit 4 Unimplemented, read as "0"

Bit 3 **ADE**: A/D Converter interrupt control
0: Disable
1: Enable

Bit 2 **OPA0E**: Operational Amplifier interrupt control
0: Disable
1: Enable

Bit 1 **THE**: High Voltage Driver Combination Thermal Protection interrupt control
0: Disable
1: Enable

Bit 0 Unimplemented, read as "0"

INTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	MF3F	MF2F	MF1F	MF0F	MF3E	MF2E	MF1E	MF0E
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 **MF3F**: Multi-function interrupt 3 request flag
0: No request
1: Interrupt request

Bit 6 **MF2F**: Multi-function interrupt 2 request flag
0: No request
1: Interrupt request

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

Bit 4 **MF0F**: Multi-function interrupt 0 request flag
0: No request
1: Interrupt request

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

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

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

Bit 0 **MF0E**: Multi-function interrupt 0 control
0: Disable
1: Enable

INTC3 Register

Bit	7	6	5	4	3	2	1	0
Name	MF4F	EPWF	IICF	LVDF	MF4E	EPWE	IICE	LVDE
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 **EPWF**: Data EEPROM interrupt request flag
0: No request
1: Interrupt request

Bit 5 **IICF**: I^2C interrupt request flag
0: No request
1: Interrupt request

Bit 4 **LVDF**: LVD interrupt request flag
0: No request
1: Interrupt request

Bit 3 **MF4E**: Multi-function interrupt 4 control
0: Disable
1: Enable

Bit 2 **EPWE**: Data EEPROM interrupt control
0: Disable
1: Enable

Bit 1 **IICE**: I^2C interrupt control
0: Disable
1: Enable

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

MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTMA2F	PTMP2F	—	—	PTMA2E	PTMP2E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PTMA2F**: PTM2 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTMP2F**: PTM2 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTMA2E**: PTM2 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTMP2E**: PTM2 Comparator P match interrupt control
0: Disable
1: Enable

MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTMA1F	PTMP1F	—	—	PTMA1E	PTMP1E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PTMA1F**: PTM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTMP1F**: PTM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTMA1E**: PTM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTMP1E**: PTM1 Comparator P match interrupt control
0: Disable
1: Enable

MFI2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTMA0F	PTMP0F	—	—	PTMA0E	PTMP0E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PTMA0F**: PTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTMP0F**: PTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTMA0E**: PTM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTMP0E**: PTM0 Comparator P match interrupt control
0: Disable
1: Enable

MFI3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTMA3F	PTMP3F	—	—	PTMA3E	PTMP3E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PTMA3F**: PTM3 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTMP3F**: PTM3 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTMA3E**: PTM3 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTMP3E**: PTM3 Comparator P match interrupt control
0: Disable
1: Enable

MFI4 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TB1F	TB0F	—	—	TB1E	TB0E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

Bit 0 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

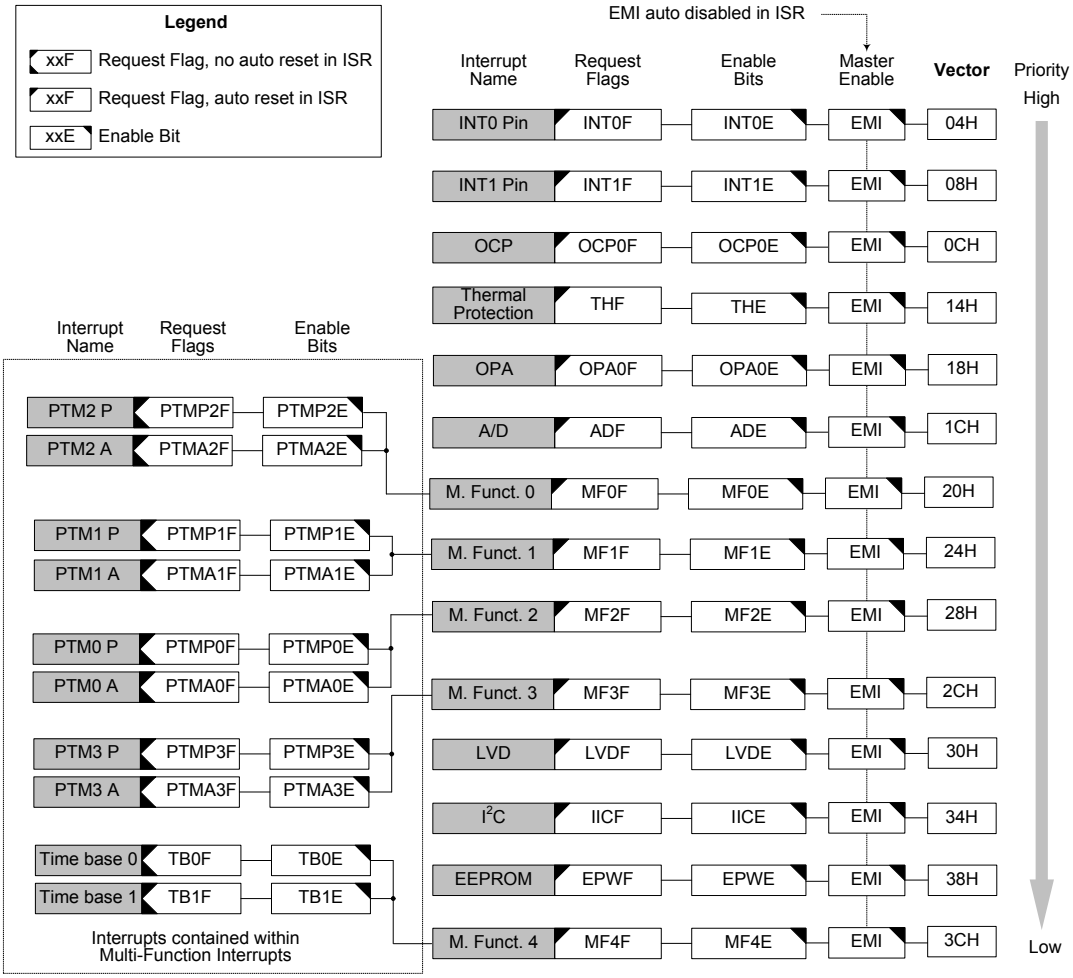
Interrupt Operation

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 Interrupt

The external interrupts are controlled by signal transitions on the pin INTn. An external interrupt request will take place when the external interrupt request flag, INTnF, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to 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 pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTnF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input. The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

Over Current Protection Interrupt

An OCP Interrupt request will take place when the OCP Interrupt request flag, OCP0F, is set, which occurs when a large current is detected. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and OCP Interrupt enable bit, OCP0E, must first be set. When the interrupt is enabled, the stack is not full and an over current is detected, a subroutine call to the OCP Interrupt vector, will take place. When the interrupt is serviced, the OCP Interrupt flag, OCP0F, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Operational Amplifier Interrupt

The Operational Amplifier interrupt is controlled by the output signal transitions. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the Operational Amplifier interrupt function and to choose the trigger edge type. An Operational Amplifier interrupt request will take place when the Operational Amplifier interrupt request flag, OPA0F, is set, which will occur when the Operational Amplifier output signal changes state, whose type is chosen by the edge select bits. When the interrupt is enabled, the stack is not full and the Operational Amplifier output generates a transition, a subroutine call to the Operational Amplifier interrupt vector, will take place. When the interrupt is serviced, the Operational Amplifier interrupt request flag, OPA0F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

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

A/D Converter Interrupt

The A/D Converter Interrupt is controlled by the termination of an A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter Interrupt vector, will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Thermal Protection Interrupt

If the High Voltage Driver Combination Thermal Protection function is enabled, a Thermal Protection Interrupt request will take place when the Thermal Protection Interrupt request flag, THF, is set, which occurs when the detected temperature exceeds the preset temperature. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Thermal Protection Interrupt enable bit, THE, must first be set. When the interrupt is enabled, the stack is not full and an over thermal condition is detected, a subroutine call to the Thermal Protection Interrupt vector, will take place. When the interrupt is serviced, the Thermal Protection Interrupt flag, THF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Multi-function Interrupt

Within this device there are multiple 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 TM and Timer Base Interrupts.

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. To allow the program to branch to its respective interrupt vector address, 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 flags will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupts, namely the TM and Timer Base Interrupts will not be automatically reset and must be manually reset by the application program.

LVD Interrupt

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

I²C Interrupt

An I²C Interrupt request will take place when the I²C Interrupt request flag, IICF, is set, which occurs when a byte of data has been received or transmitted by the I²C interface, I²C address match or I²C time-out. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, IICE, must first be set. When the interrupt is enabled, the stack is not full and any of these situations occurs, a subroutine call to the I²C Interrupt vector, will take place. When the I²C Interface Interrupt is serviced, the interrupt request flag, IICF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

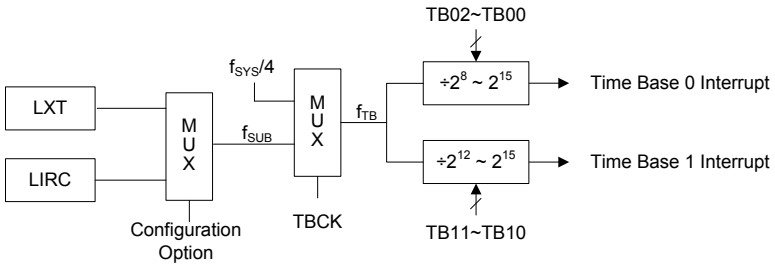
EEPROM Interrupt

An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, EPWF, 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, and EEPROM Interrupt enable bit, EPWE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the EEPROM Interrupt vector will take place. When the EEPROM Interface Interrupt is serviced, the interrupt request flag, EPWF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

Time Base Interrupts

The Time Base interrupts are contained within the Multi-function Interrupts. The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, and relevant Multi-function Interrupt enable bit, MF4E, and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the Time Base interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MF4F flag will be automatically cleared. As the Time Base interrupt request flag, TB0F or TB1F, will not be automatically cleared, they have 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_{TB}, originates from the internal clock source $f_{SYS}/4$ or f_{SUB} which can be selected by the TBCK bit in the TBC register. This f_{TB} clock passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges.



Time Base Interrupts

TBC Register

Bit	7	6	5	4	3	2	1	0
Name	TBON	TBCK	TB11	TB10	LXTSP	TB02	TB01	TB00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	1	0	1	1	1

Bit 7 **TBON**: Time Base 0 and Time Base 1 Control
0: Disable
1: Enable

Bit 6 **TBCK**: Select f_{TB} clock
0: f_{SUB}
1: $f_{SYS}/4$

Bit 5~4 **TB11~TB10**: Select Time Base 1 Time-out Period
00: $2^{12}/f_{TB}$
01: $2^{13}/f_{TB}$
10: $2^{14}/f_{TB}$
11: $2^{15}/f_{TB}$

Bit 3 **LXTSP**: LXT Quick Start Control
0: Disable
1: Enable

Bit 2~0 **TB02~TB00**: Select Time Base 0 Time-out Period
000: $2^{8}/f_{TB}$
001: $2^{9}/f_{TB}$
010: $2^{10}/f_{TB}$
011: $2^{11}/f_{TB}$
100: $2^{12}/f_{TB}$
101: $2^{13}/f_{TB}$
110: $2^{14}/f_{TB}$
111: $2^{15}/f_{TB}$

PTM Interrupts

The Periodic Type TMs have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation, all of the PTM interrupts are contained within the Multi-function Interrupts. For each of the Periodic Type TMs there are two interrupt request flags PTMPnF and PTMAnF and two enable bits PTMPnE and PTMAnE. A PTM interrupt request will take place when any of the PTM request flags are set, a situation which occurs when a PTM comparator P or A match situation happens.

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

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins, a low power supply voltage or operational amplifier input change 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.

Low Voltage Detector – LVD

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

LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of eight 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 ENLVD 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. The VBGEN bit is used to control internal Bandgap reference voltage enabled or disabled. Any of the ENLVD and VBGEN bits is set high, the Bandgap reference voltage will be enabled. 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	ENLVD	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 Detect
1: Low Voltage Detect

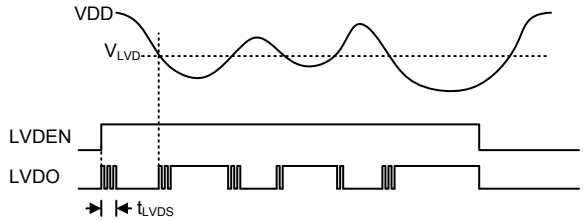
Bit 4 **ENLVD**: Low Voltage Detector Control
0: Disable
1: Enable

Bit 3 **VBGEN**: Bandgap Voltage Output Control
0: Disable
1: Enable

Bit 2~0 **VLVD2~VLVD0**: Select LVD Voltage
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.0V and 4.0V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is in the SLEEP mode, the low voltage detector will be disabled even if the ENLVD bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

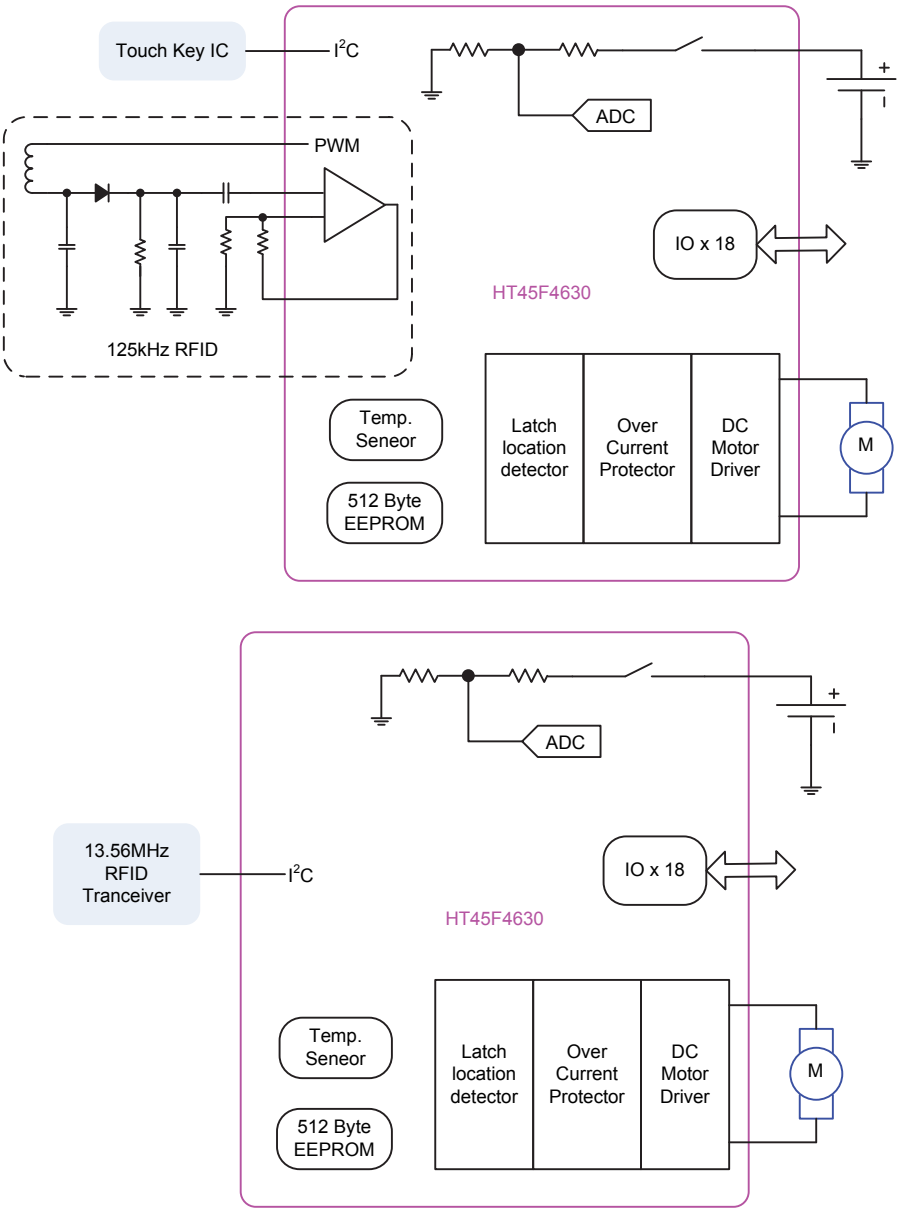
The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. In this case, the LVDF interrupt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

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later using the application program. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Options	
1	High Speed System Oscillator Selection – f_H: • HXT • HIRC
2	Low Speed System Oscillator Selection – f_{SUB}: • LXT • LIRC
3	HIRC Frequency Selection: • 16MHz • 12MHz • 8MHz
Watchdog Timer Options	
4	WDT Function: • Always enable • Controlled by WDT Control Register
5	WDT Clock Selection – f_S: • f_{SUB} • $f_{SYS}/4$

Application Circuits



Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of three kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

The following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV
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].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
TABRDC [m]	Read table (current page) to TBLH and Data Memory	2Note	None
TABRDL [m]	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
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-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.
3. For the “CLR WDT1” and “CLR WDT2” instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both “CLR WDT1” and “CLR WDT2” instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C

ADCM A,[m] Add ACC to Data Memory with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C

ADD A,[m] Add Data Memory to ACC

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m]

Affected flag(s) OV, Z, AC, C

ADD A,x Add immediate data to ACC

Description The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation ACC ← ACC + x

Affected flag(s) OV, Z, AC, C

ADDM A,[m] Add ACC to Data Memory

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m]

Affected flag(s) OV, Z, AC, C

AND A,[m] Logical AND Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC "AND" x

Affected flag(s) Z

ANDM A,[m] Logical AND ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

Description Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow \overline{[m]}$

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

DEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

Description This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

OR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

ORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

Description The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation Program Counter ← Stack

Affected flag(s) None

RET A,x Return from subroutine and load immediate data to ACC

Description The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

Description The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

RLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

RR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

Description: Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

Description: The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation:
[m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s): C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s): C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: ACC ← ACC − [m] − C

Affected flag(s): OV, Z, AC, C

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

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

SUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

Affected flag(s) OV, Z, AC, C

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − x

Affected flag(s) OV, Z, AC, C

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

Description If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

TABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer pair (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

TABRDC [m] Read table (current page) to TBLH and Data Memory

Description The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

XOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

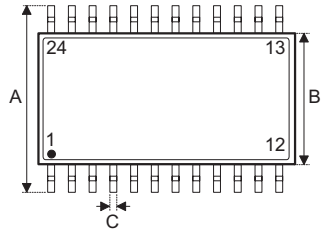
Package Information

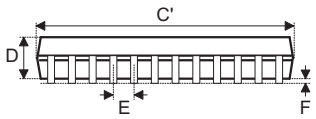
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

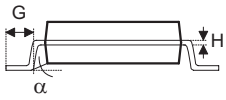
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

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

24-pin SSOP (150mil) Outline Dimensions







Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.341 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.000 BSC	—
B	—	3.900 BSC	—
C	0.20	—	0.30
C’	—	8.660 BSC	—
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°