Emergency Light Flash MCU

HT45FH4J

Revision: V1.10 Date: December 14, 2016

www.holtek.com

Table of Contents

Features

CPU Features

- High voltage input (up to 12V) to the integrated LDO and outputs 5V to supply MCU operating voltage
- Up to 0.2μs instruction cycle with 20MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillators
  - Internal 12/16/20MHz Hight Speed RC -- HIRC
  - Internal Low Speed 32kHz RC -- LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K×16
- RAM Data Memory: 128×8
- True EEPROM Memory: 64×8
- Watchdog Timer function
- 12 bidirectional I/O lines
- Two pin-shared external interrupts
- Multiple Timer Module for time measure, input capture, compare match output, PWM output function or single pulse output function
- Two complementary PWM output with dead time control
- Over current protection (OCP) with interrupt
- Dual Time-Base functions for generation of fixed time interrupt signals
- 6 external channels 12-bit resolution A/D converter
- Low voltage reset function
- Low voltage detect function
- Package: 16-pin NSOP, 20-pin SSOP

Emergency Light Application Features

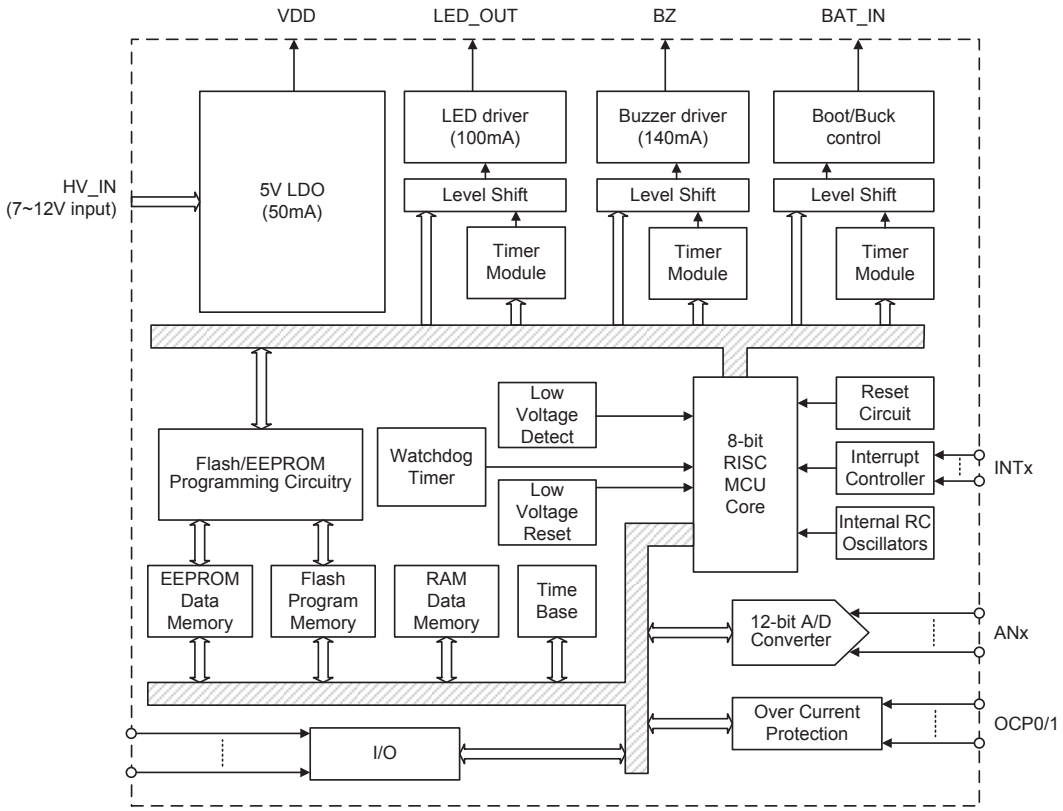
- One integrated LDO: 5V output to supply operating voltage for the MCU, LED indicator and other circuits.
- One integrated resistor divider, its DC/DC boost voltage is used for close loop control
- One internal PMOS and driving an external NMOS can implement the DC/DC boost and buck control
- Internal LED driving circuit (0.6W)
- Supports an additional external NMOS for high power LED driving (over 0.6W)
- High voltage and high current output for Buzzer driving (140mA)
- One internal switch for emergency light product battery and LED lighting self-test function

General Description

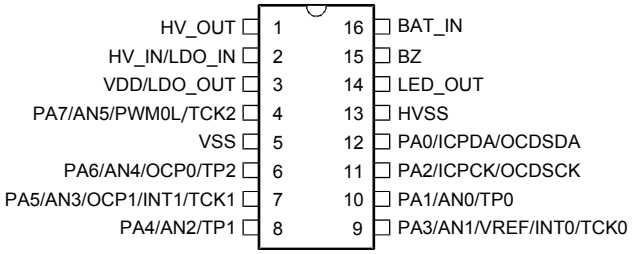
This device is an Emergency Light ASSP Flash MCU. The device includes 2K×16 of Flash Program Memory, 128 bytes of Data Memory and 64 bytes of Data EEPROM. The internal LDO provides a maximum input voltage of 12V and outputs a 5V voltage with a current of 50mA which can then be provided to the MCU and peripheral circuits. Additional features include one fully integrated high accuracy RC oscillator with three fixed frequencies of either 12MHz, 16MHz or 20MHz, an internal 6-channel 12-bit A/D converter and three 10-bit Periodic Timers. One of these Timers can be used to generate two complementary PWM outputs which are used for the required DC/DC boost and buck circuitry. A range of protection features are provided, such as over current protection, Low Voltage Detector and Low Voltage Reset, which are used for system voltage monitoring. If the system voltage falls below the LVR value, the device will be automatically reset to reduce the possibility of unstable operations.

In the traditional emergency light applications, a microcontroller usually requires additional transistors to drive LEDs, buzzers as well as the boost and buck circuits. However, this new device includes a powerful driving capability, it can directly drive LEDs with a current of 100mA and buzzers with a current of 140mA. During battery charging and discharging, the complementary PWM outputs with a dead time insertion, are used to drive an internal PMOS transistor and an external NMOS transistor to implement the synchronous rectification function. The device is able to reduce the power consumption to a minimum, improve the operating efficiency and extend the LED illumination time. As for protection features, the over current protection circuitry ensures that the LEDs and the battery remain free from damage when over current occurs.

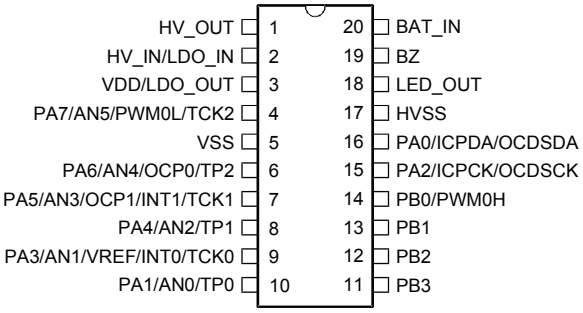
Block Diagram



Pin Assignment



HT45FH4J/HT45VH4J
16 NSOP-A



HT45FH4J/HT45VH4J
20 SSOP-A

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

2. The actual device and its equivalent OCDS EV device share the same package type, however the OCDS EV device part number is HT45VH4J. Pins OCDSCK and OCDSDA which are pin-shared with PA2 and PA0 are only used for the OCDS EV device.

Pin Descriptions

With the exception of the power pins and some relevant transformer control pins, all pins on this device can be referenced by their Port name, e.g. PA0, PA1 etc, which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Analog to Digital Converter, Timer Module pins etc. The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

As the Pin Description table shows the situation for the package with the most pins, not all pins in the table will be available on smaller package sizes.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/ICPDA/ OCDSDA	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	ICPDA	—	ST	CMOS	In-circuit programming data/address pin
	OCDSDA	—	ST	CMOS	On-chip debug support data/address pin, only for EV IC
PA1/AN0/TP0	PA1	PAPU PAWU CTRL3	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN0	CTRL3	AN	—	A/D Converter input 0
	TP0	CTRL3	ST	CMOS	TM0 I/O
PA2/ICPCK/ OCDSCK	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	ICPCK	—	ST	—	In-circuit programming clock pin
	OCDSCK	—	ST	—	On-chip debug support clock pin, only for EV IC
PA3/AN1/VREF/ INT0/TCK0	PA3	PAPU PAWU CTRL3	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN1	CTRL3	AN	—	A/D Converter input 1
	VREF	CTRL3	AN	—	A/D Converter reference voltage input
	INT0	CTRL3 INTEG INTC1	ST	—	External interrupt 0
	TCK0	CTRL3 TM0C0	ST	—	TM0 clock input
PA4/AN2/TP1	PA4	PAPU PAWU CTRL3	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN2	CTRL3	AN	—	A/D Converter input 2
	TP1	CTRL3	ST	CMOS	TM1 I/O
PA5/AN3/OCP1/ INT1/TCK1	PA5	PAPU PAWU CTRL3	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN3	CTRL3	AN	—	A/D Converter input 3
	OCP1	CTRL3	AN	—	Over current protection input
	INT1	CTRL3 INTEG INTC1	ST	—	External interrupt 1
	TCK1	CTRL3 TM1C0	ST	—	TM1 clock input

Pin Name	Function	OPT	I/T	O/T	Description
PA6/AN4/OCP0/ TP2	PA6	PAPU PAWU CTRL4	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN4	CTRL4	AN	—	A/D Converter input 4
	OCP0	CTRL4	AN	—	Over current protection input
	TP2	CTRL4	ST	CMOS	TM2 I/O
PA7/AN5/ PWM0L/ TCK2	PA7	PAPU PAWU CTRL4	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN5	CTRL4	AN	—	A/D Converter input 5
	PWM0L	CTRL4	—	CMOS	Complementary PWM0 output
	TCK2	CTRL4 TM2C0	ST	—	TM2 clock input
PB0/PWM0H	PB0	PBPU CTRL4	ST	CMOS	General purpose I/O. Register enabled pull-up
	PWM0H	CTRL4	—	CMOS	Complementary PWM0 output
PB1~PB3	PB1~PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
VDD/LDO_OUT	VDD	—	PWR	—	Positive power supply
	LDO_OUT	—	PWR	—	5V LDO output
HV_IN/LDO_IN	LDO_IN	—	PWR	—	LDO input
VSS	VSS	—	PWR	—	Negative power supply
High Voltage I/O Ports					
BZ	BZ	—	—	CMOS	Buzzer driver output
BAT_IN	BAT_IN	—	PMOS	PMOS	Battery input
LED_OUT	LED_OUT	—	—	PMOS	LED driver output
High Voltage Power					
HV_IN/LDO_IN	HV_IN	—	PWR	—	Positive power supply (for High Voltage)
HV_OUT	HV_OUT	—	PWR	—	High Voltage Output
HVSS	HVSS	—	PWR	—	Negative power supply (for High Voltage)

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

Absolute Maximum Ratings

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

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

D.C. Characteristics

Ta = 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{CC}	Analog Operation Voltage	—	HV_IN pin input voltage	5	7	12	V
V_{DD}	Operation Voltage	—	f_{SYS}=12/16/20MHz	-3%	5	+3%	V
I_{DD1}	Operation Current, Normal Mode $f_{SYS}=f_H$	5V	No load, f_H=12MHz, ADC off, WDT enable	—	2.5	3.8	mA
			No load, f_H=16MHz, ADC off, WDT enable	—	3.3	5.0	mA
			No load, f_H=20MHz, ADC off, WDT enable	—	4.2	6.3	mA
I_{DD2}	Operation Current, Slow Mode, $f_{SYS}=f_L=f_{LIRC}$, $f_{SUB}=f_{LIRC}$	5V	No load, $f_{SYS}=f_{LIRC}$, ADC off, WDT enable	—	55	95	μA
I_{DD3}	Operation Current, Normal Mode f_H=16MHz	5V	No load, $f_{SYS}=f_H/2$, ADC off, WDT enable	—	2.2	3.3	mA
			No load, $f_{SYS}=f_H/4$, ADC off, WDT enable	—	1.5	2.25	mA
			No load, $f_{SYS}=f_H/8$, ADC off, WDT enable	—	1.2	1.8	mA
			No load, $f_{SYS}=f_H/16$, ADC off, WDT enable	—	1.1	1.65	mA
			No load, $f_{SYS}=f_H/32$, ADC off, WDT enable	—	1.0	1.5	mA
			No load, $f_{SYS}=f_H/64$, ADC off, WDT enable	—	0.9	1.35	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD4}	Operation Current, Normal Mode f_H=20MHz	5V	No load, $f_{SYS}=f_H/2$, ADC off, WDT enable	—	2.7	4.1	mA
			No load, $f_{SYS}=f_H/4$, ADC off, WDT enable	—	1.6	2.4	mA
			No load, $f_{SYS}=f_H/8$, ADC off, WDT enable	—	1.5	2.3	mA
			No load, $f_{SYS}=f_H/16$, ADC off, WDT enable	—	1.3	1.95	mA
			No load, $f_{SYS}=f_H/32$, ADC off, WDT enable	—	1.2	1.8	mA
			No load, $f_{SYS}=f_H/64$, ADC off, WDT enable	—	1.15	1.75	mA
I_{IDLE0}	IDLE0 Mode Standby Current (LIRC on)	5V	No load, ADC off, WDT enable, LVR disable	—	22.2	38	μA
I_{IDLE11}	IDLE1 Mode Standby Current	5V	No load, ADC off, WDT enable, f_{SYS}=12MHz on	—	1.2	2.4	mA
I_{IDLE12}	IDLE1 Mode Standby Current	5V	No load, ADC off, WDT enable, f_{SYS}=16MHz on	—	2.0	4.0	mA
I_{IDLE13}	IDLE1 Mode Standby Current	5V	No load, ADC off, WDT enable, f_{SYS}=20MHz on	—	2.5	5.0	mA
I_{SLEEP}	SLEEP Mode Standby Current (LIRC on)	5V	No load, ADC off, WDT enable, LVR disable	—	28	40	μA
V_{IL}	Input Low Voltage for I/O Ports or Input Pins	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	V
V_{IH}	Input High Voltage for I/O Ports or Input Pins	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	V
I_{OL1}	I/O Port Sink Current (except for BZ, BAT_IN, LED_OUT, PA7)	5V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
I_{OH1}	I/O Port Source Current (except for BZ, BAT_IN, LED_OUT, PA7)	5V	$V_{OH}=0.9V_{DD}$	6	12	—	mA
I_{OL2}	I/O Port Sink Current (PA7)	5V	$V_{OL}=0.1V_{DD}$	40	80	—	mA
I_{OH2}	I/O Port Source Current (PA7)	5V	$V_{OH}=0.9V_{DD}$	40	80	—	mA
R_{PH}	Pull-high Resistance for I/O Ports	5V	—	10	30	50	kΩ
High Voltage I/O Ports							
I_{OL3}	I/O Port Sink Current (BZ)	6.5V	V_{OL}=0.1HV_OUT	115	140	—	mA
I_{OH3}	I/O Port Source Current (BZ)	6.5V	V_{OH}=0.9HV_OUT	-5%	10	—	mA
I_{OH4}	I/O Port Source Current (BAT_IN)	6.5V	V_{OH}=HV_OUT-0.5	235	—	—	mA
I_{OH5}	I/O Port Source Current (LED_OUT)	6.5V	V_{OH}=HV_OUT-0.5	100	—	—	mA

A.C. Characteristics

Ta = 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{CPU}	Operating Clock	5V±3%	—	DC	—	12	MHz
				DC	—	16	MHz
				DC	—	20	MHz
f_{SYS}	System Clock (HIRC)	5V±3%	—	—	—	12	MHz
				—	—	16	MHz
				—	—	20	MHz
f_{HIRC}	HIRC Frequency (note)	5V±3%	Ta=25°C	-2%	12/16/20	+2%	MHz
			Ta=-40°C~85°C	-5%	12/16/20	+5%	MHz
f_{LIRC}	LIRC Frequency	5V±3%	Ta=25°C	-10%	32	+10%	kHz
			Ta=-40°C ~ 85°C	-30%	32	+60%	kHz
t_{TIMER}	TCKn Input Pin Minimum Pulse Width	—	—	—	30	—	ns
t_{INT}	Interrupt Minimum Pulse Width	—	—	1	3.3	5	μs
t_{EERD}	EEPROM Read Time	—	—	—	2	4	t_{SYS}
t_{EEWR}	EEPROM Write Time	—	—	—	2	4	ms
t_{SST}	System Start-up Timer Period (Wake-up from HALT, f_{SYS} off at HALT state)	—	f_{SYS}=HIRC	16	—	—	t_{HIRC}
			f_{SYS}=LIRC	2	—	—	t_{SYS}
	System Start-up Timer Period (Wake-up from HALT, f_{SYS} on at HALT state)	—	f_{SYS}=HIRC	2	—	—	t_{HIRC}
t_{RSTD}	System Reset Delay Time (Power on Reset, LVR H/W Reset, LVR S/W Reset, WDT S/W Reset)	—	—	25	50	100	ms
	System Reset Delay Time (WDT Normal Reset)	—	—	8.3	16.7	33.3	ms

Note: 1. t_{SYS}= 1/f_{SYS}, t_{HIRC}=1/ f_{HIRC}

2. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS and located as close to the device as possible.

LVD&LVR Electrical Characteristics

Ta = 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR1}	Low Voltage Reset Voltage	—	LVR enable, 2.1V	-5%	2.1	+5%	V
V_{LVR2}			LVR enable, 2.55V		2.55		V
V_{LVR3}			LVR enable, 3.15V		3.15		V
V_{LVR4}			LVR enable, 3.8V		3.8		V
V_{LVD1}	Low Voltage Detector Voltage	—	LVDEN= 1, V_{LVD} = 2.0V	-5%	2.0	+5%	V
V_{LVD2}		—	LVDEN= 1, V_{LVD} = 2.2V		2.2		V
V_{LVD3}		—	LVDEN= 1, V_{LVD} = 2.4V		2.4		V
V_{LVD4}		—	LVDEN= 1, V_{LVD} = 2.7V		2.7		V
V_{LVD5}		—	LVDEN= 1, V_{LVD} = 3.0V		3.0		V
V_{LVD6}		—	LVDEN= 1, V_{LVD} = 3.3V		3.3		V
V_{LVD7}		—	LVDEN= 1, V_{LVD} = 3.6V		3.6		V
V_{LVD8}		—	LVDEN= 1, V_{LVD} = 4.0V		4.0		V
I_{LVR}	Additional Power Consumption if LVR is used	5V±3%	LVR disable → LVR enable	—	60	90	μA
I_{LVD}	Additional Power Consumption if LVD is used	5V±3%	LVD disable → LVD enable (LVR disable)	—	75	115	μA
			LVD disable → LVD enable (LVR enable)	—	60	90	μA
t_{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Low Voltage Width to Interrupt	—	—	60	120	240	μs
t_{LVDS}	LVDO Stable Time	—	For LVR enable, LVD off → on	—	—	5	μs
		—	For LVR disable, LVD off → on	—	—	15	μs
t_{SRESET}	Software Reset Width to Reset	—	—	45	90	120	μs

Note: V_{LVR} and V_{LVD} are the LVR and LVD voltage when the V_{DD} voltage drops.

ADC Electrical Characteristics

Ta = 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
AV_{DD}	A/D Converter Operation Voltage	—	—	2.7	5	5.5	V
V_{REF}	A/D Converter Reference Voltage	—	—	2.0	—	AV_{DD}	V
V_{AD}	A/D Converter Input Voltage	—	—	0	—	AV_{DD} /V_{REF}	V
I_{ADC}	Additional Power Consumption if A/D Converter is used	5V	No Load (t_{ADCK} = 0.5μs)	—	1.5	3.0	mA
DNL	Differential Non-linearity	5V	V_{REF}=AV_{DD}=V_{DD}, t_{ADCK} = 0.5μs	-3	—	+5	LSB
			V_{REF}=AV_{DD}=V_{DD}, t_{ADCK} = 10μs	-3	—	+5	LSB
INL	Integral Non-linearity	5V	V_{REF}=AV_{DD}=V_{DD}, t_{ADCK} = 0.5μs	-5	—	+6	LSB
			V_{REF}=AV_{DD}=V_{DD}, t_{ADCK} = 10μs	-5	—	+6	LSB
t_{ADCK}	A/D Converter Clock Period	—	—	0.5	—	10	μs
t_{ADC}	A/D Conversion Time (Include Sample and Hold Time)	—	12-bit ADC	16	—	20	t_{ADCK}
t_{ADS}	A/D Converter Sampling Time	—	12-bit ADC	—	4	—	t_{ADCK}
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs

LDO Regulator Electrical Characteristics

V_{IN}=V_{OUT}+2.0V, I_O=1mA, Ta=25°C , unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IN}	Input Voltage	—	—	7	—	12	V
V_{OUT}	Output Voltage	—	—	-3%	5	3%	V
ΔV_{OUT}	Output Voltage Tolerance	7V~12V	I_O=50mA, Ta=25°C	-3.0	—	3.0	%
			I_O=50mA, Ta=-40°C~85°C (except 25°C)	-5.0	—	5.0	%
ΔV_{LOAD}	Load Regulation	7V~12V	1mA≤I_O≤50mA	—	0.18	0.36	%mA
V_{DROP}	Drop Out Voltage	7V~12V	I_O=1mA, ΔV_O=2%	—	—	100	mV
I_{SS}	Quienscent Current	7V~12V	I_O=0mA	—	25	45	μA
ΔV_{LINE}	Line Regulation	7V~12V	1.0V+V_{OUT}≤V_{IN}≤12V, I_O=1mA	—	0.2	—	%/V
$\Delta V_{OUT}/\Delta Ta$	Temperature Coefficient	7V~12V	I_O=50mA	—	0.54	—	mV/°C

Note: 1. The LDO can provide a 50mA load and a current limit function.
2. The LDO is always enabled without a control signal, it requires an external 0.1μF and more than 10μF capacitors for VIN and Vout to VSS pin.

Over Current Pretection Electrical Characteristics

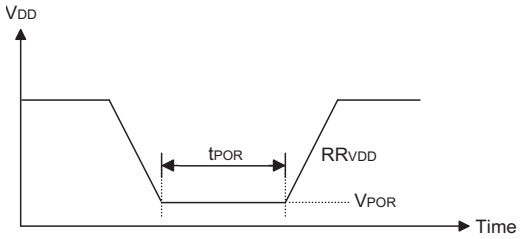
Ta = 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OP}	Operation Current	5V	ENOCP[1:0]=01, DAC V_{REF}=2.5V	—	730	1250	μA
OCP Comparator							
I_{COMP}	Comparator Operating Current	5V	No load	—	30	60	μA
V_{CMPOS}	Comparator Input Offset Voltage	5V	—	-15	—	15	mV
		5V	By calibration	-4	—	4	mV
V_{HYS}	Comparator Hysteresis Width	5V	—	20	40	60	mV
V_{CM}	Comparator Common Mode Voltage Range	5V	—	V_{SS}	—	V_{DD}-1.4	V
OCP OPA							
I_{OPA}	OPA Operating Current	5V	No load	—	200	350	μA
V_{OPAOS}	OPA Input Offset Voltage	5V	—	-15	—	15	mV
			With calibration	-4	—	4	mV
V_{CM}	OPA Common Mode Voltage Range	5V	—	V_{SS}	—	V_{DD}-1.4	V
GAIN	OPA Gain Error	5V	All conditions	-5	G	5	%
DAC for OCP							
I_{DAC}	DAC Operating Current	5V	V_{REF}=2.5V	—	250	300	μA
		5V	V_{REF}=5V	—	500	600	μA
R_O	R2R Output Resistor	5V	—	—	10	—	kΩ
DNL	DAC Differential NonLinearity	—	—	-0.5	—	0.5	LSB
INL	DAC Integral NonLinearity	—	—	-1	—	1	LSB

Power-on Reset Electrical 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_{VDD}	V_{DD} Rising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for V_{DD} Stays at V_{POR} to Ensure Power-on Reset	—	—	1	—	—	ms

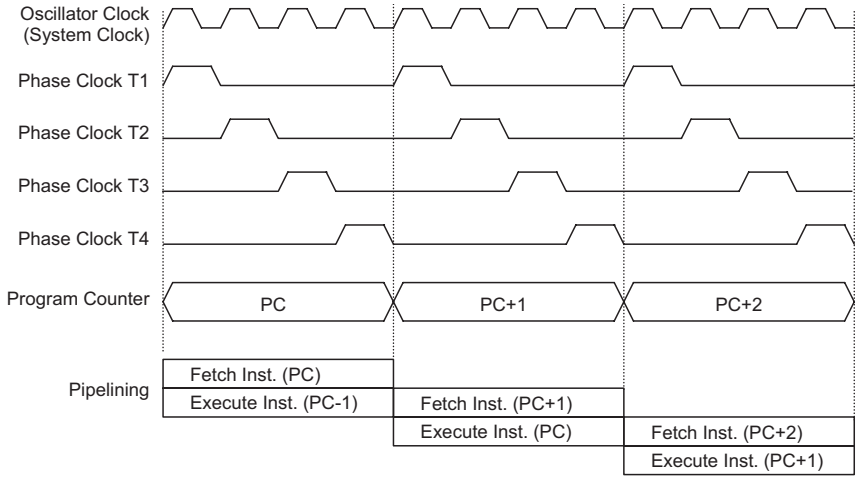


System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and Periodic 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 an HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the decoding and execution functions. In this way, one T1~T4 clock cycle forms one instruction cycle. Although the fetching and execution of instructions takes place in consecutive instruction cycles, the pipelining structure of the microcontroller ensures that instructions are effectively executed in one instruction cycle. The exception to this are instructions where the contents of the Program Counter are changed, such as subroutine calls or jumps, in which case the instruction will take one more instruction cycle to execute.

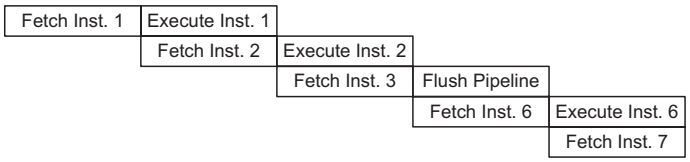


System Clock and Pipelining

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          NOP

DELAY:
```



Instruction Fetching

Program Counter

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

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

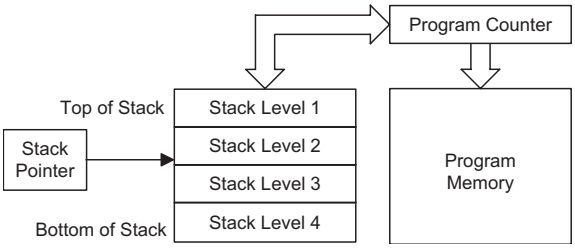
Program Counter	
Program Counter High byte	**PCL Register**
PC10~PC8	PCL7~PCL0

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 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

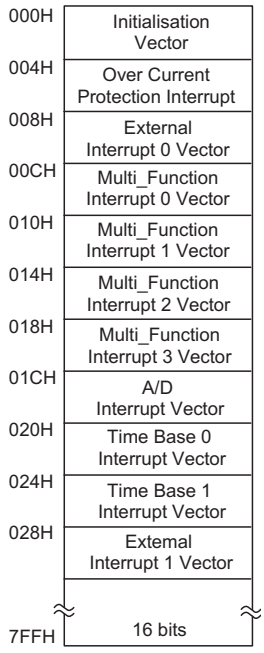
The Program Memory is the location where the user code or program is stored. For this device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, this 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.

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.



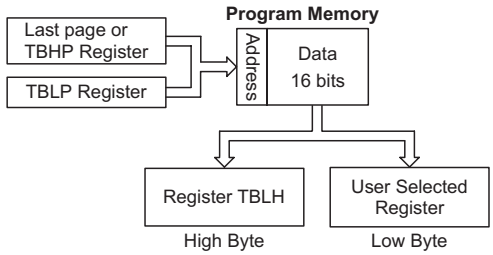
Program Memory Structure

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.



Instruction	Table Location Bits										
	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRD [m]	@10	@9	@8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: b10~b0: Table location bits

@7~@0: Table pointer (TBLP) bits

@10~@8: Table pointer (TBHP) bits

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 "700H" which refers to the start address of the last page within the 2K words Program Memory of the device. The table pointer 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 "706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address specified by the TBHP and TBLP registers if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

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

Table Read Program Example

```
tempreg1 db ?    ; temporary register #1
tempreg2 db ?    ; temporary register #2
     :
     :
mov a,06h        ; initialise low table pointer - note that this address is referenced
mov tblp,a       ; to the last page or specific page
mov a,07h        ; initialise high table pointer
mov tbhp,a
     :
     :
tabrd tempreg1   ; transfers value in table referenced by table pointer
                 ; data at program memory address "706H" transferred to tempreg1 and TBLH
dec tblp         ; reduce value of table pointer by one
tabrd tempreg2   ; transfers value in table referenced by table pointer
                 ; data at program memory address "705H" transferred 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 700h         ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
     :
     :
```

In Circuit Programming

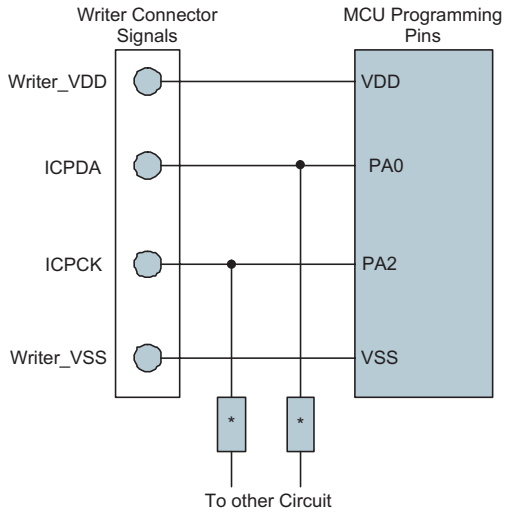
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides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

The Holtek Flash MCU to Writer Programming Pin correspondence table is as follows:

Holtek Write Pins	MCU Programming Pins	Function
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Serial Clock
VDD	VDD	Power Supply
VSS	VSS	Ground

During the programming process, the user must there take care to ensure that no other outputs are connected to these two pins.

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.



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

An EV chip exists for the purposes of device emulation. This EV chip device also provides an "On-Chip Debug" function to debug the device during the development process. The EV chip and the actual MCU devices are almost functionally compatible except for the "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 actual MCU 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 a more detailed OCDS description, 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
GND	VSS	Ground

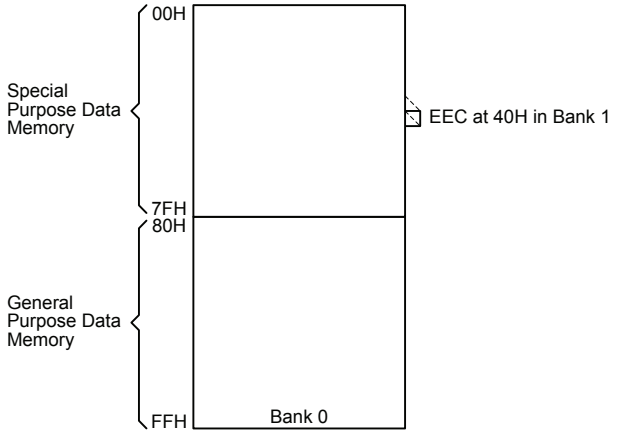
RAM Data Memory

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 sections, 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.

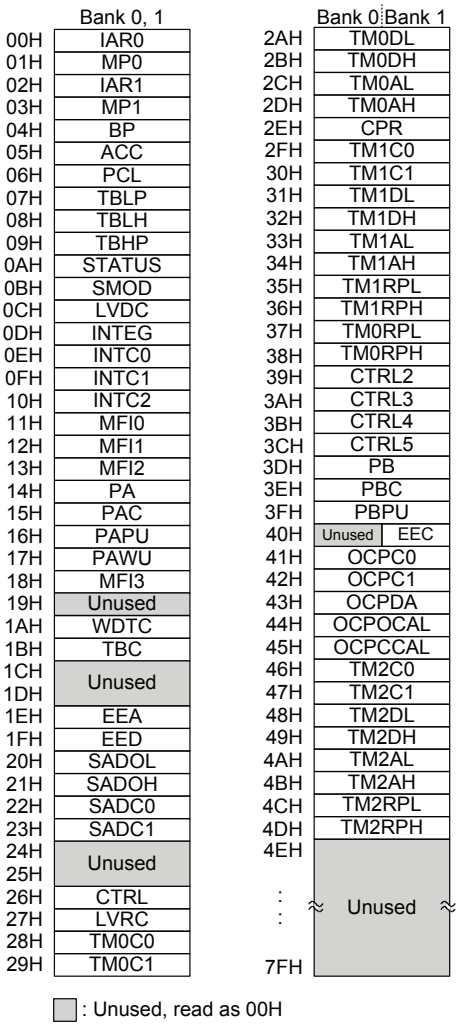
The overall Data Memory is subdivided into two banks. The Special Purpose Data Memory registers are accessible in all banks, with the exception of the EEC register at address 40H, which is only accessible in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer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



Data Memory Structure

Capacity	Banks
128×8	Bank 0: 80H~FFH

General Purpose Data Memory Structure



Special Purpose Data Memory Structure

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 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´
org00h
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 zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

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

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

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

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

STATUS Register

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

"×" unknown

Bit 7 ~ 6 Unimplemented, read as "0"

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

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

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

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

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

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

C is also affected by a rotate through carry instruction.

EEPROM Data Memory

One of the special features in the device is its internal EEPROM Data Memory. EEPROM, which stands for Electrically Erasable Programmable Read Only Memory, is by its nature a non-volatile form of 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 up to 64×8 bits. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped and is therefore not directly accessible 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 an address and data register in Bank 0 and a single control register in Bank 1.

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Bank 1, 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.

EEPROM Control Registers List

Name	Bit							
	7	6	5	4	3	2	1	0
EEA	—	—	D5	D4	D3	D2	D1	D0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEA Register

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

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 ~ 0 Data EEPROM address
Data EEPROM address bit 5 ~ bit 0

EED Register

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

Bit 7 ~ 0 EEPROM 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 can not be set to "1" at the same time in one instruction. The WR and RD can not 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 EEA 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 EEA 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. 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 DEE bit in the relevant interrupt register. However as the EEPROM is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM write cycle ends, the DEF request flag and its associated multi-function interrupt request flag will both be set. If the global, EEPROM and Multi-function interrupts are enabled and the stack is not full, a jump to the associated Multi-function Interrupt vector will take place. When the interrupt is serviced only the Multi-function interrupt flag will be automatically reset, the EEPROM interrupt flag must be manually reset by the application program. 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 Periodic 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 completed, otherwise, the EEPROM read or write operation will fail.

Programming Examples

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

```
MOV   A, EEPROM_ADRES           ; user defined address
MOV   EEA, 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           ; user defined address
MOV   EEA, 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                     ; BP points to data memory bank 1
CLR   EMI
SET   IAR1.3                    ; set WREN bit, enable write operations
SET   IAR1.2                    ; start Write Cycle - set WR bit - executed immediately
                                ; after set WREN bit
SET   EMI
BACK:
SZ    IAR1.2                    ; check for write cycle end
JMP   BACK
CLR   IAR1                      ; disable EEPROM read/write
CLR   BP
```

Oscillator

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 registers.

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Fully integrated internal oscillators, requiring no external components, are provided 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.

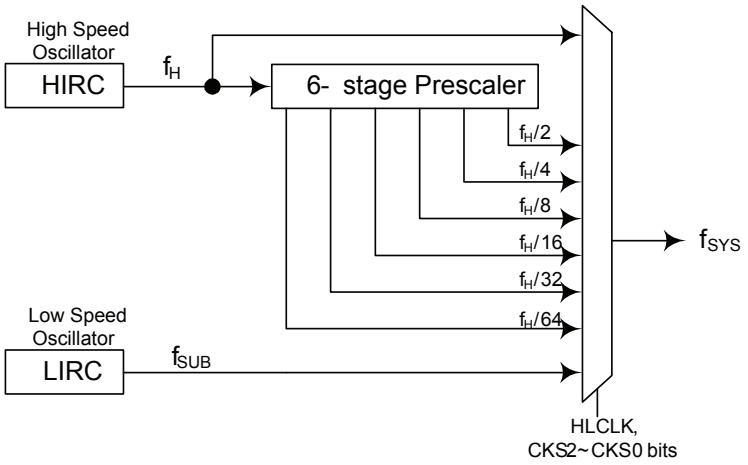
Type	Name	Freq.
Internal High Speed RC	HIRC	12/16/20MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two methods of generating the system clock, a high speed oscillator and a low speed oscillator. The high speed oscillator is the internal 12MHz, 16MHz or 20MHz RC oscillator. The low speed oscillator is the internal 32kHz RC oscillator. 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 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.



System Clock Configurations

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 either 12MHz, 16MHz or 20MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that if this internal system clock option is selected, as it requires no external pins for its operation.

Internal 32kHz Oscillator – LIRC

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

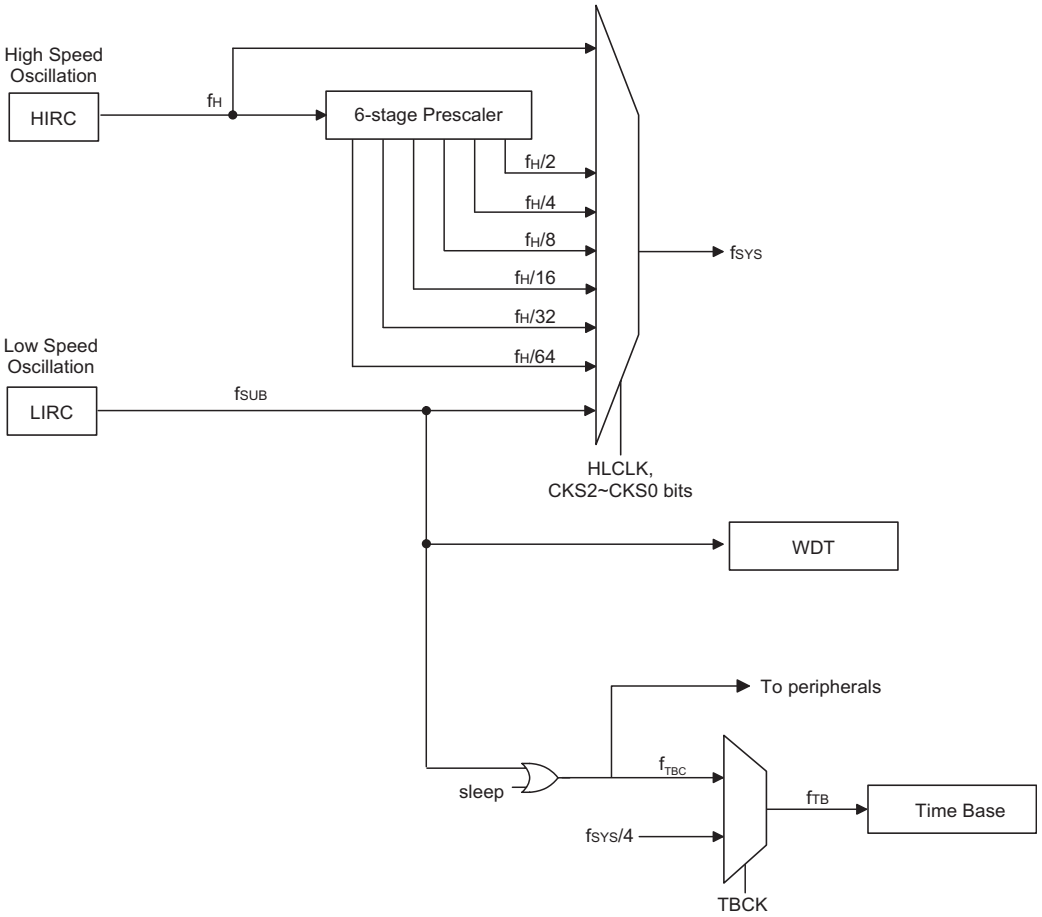
Operating Modes and System Clocks

Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice-versa, lower speed clocks reduce current consumption. As Holtek has provided this 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 the HIRC oscillator. The low speed system clock source can be sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.



System Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_L from f_H, the high speed oscillation 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 five 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 three modes, the SLEEP, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.

Operating Mode	Description			
	CPU	f_{SYS}	f_{SUB}	f_{TBC}
NORMAL mode	On	f_H~$f_H/64$	On	On
SLOW mode	On	f_{SUB}	On	On
ILDE0 mode	Off	Off	On	On
IDLE1 mode	Off	On	On	On
SLEEP mode	Off	Off	On	Off

NORMAL Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source will come from the high speed oscillator, HIRC. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 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 f_{SUB}. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW Mode, the f_H is off.

SLEEP Mode

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

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, the system oscillator will be stopped, the low frequency clock f_{SUB} will be on.

IDLE1 Mode

The IDLE1 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 high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU, 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 low frequency clock f_{SUB} will be on.

Note: If LVDEN=1 and the SLEEP or IDLE mode is entered, the LVD and bandgap functions will not be disabled, and the f_{SUB} clock will be forced to be enabled.

Control Register

The SMOD register is used to control the internal clocks within the device.

SMOD Register

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

Bit 7 ~ 5 **CKS2 ~ CKS0**: The system clock selection when HLCLK is "0"
000: f_{SUB}
001: f_{SUB}
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 LIRC, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as "0"

Bit 3 **LTO**: LIRC System OSC SST ready flag
0: Not ready
1: Ready
This is the low speed system oscillator SST ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will change to a high level after 1~2 cycles.

Bit 2 **HTO**: HIRC System OSC SST ready flag
0: Not ready
1: Ready
This is the high speed system oscillator SST ready flag which indicates when the high speed system oscillator is stable after a wake-up has occurred. 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. The flag will be low when in the SLEEP or IDLE0 Mode but after power on reset or a wake-up has occurred, the flag will change to a high level after 15~16 clock cycles if the HIRC oscillator is used.

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: $f_H/2$ ~ $f_H/64$ or f_{SUB}
1: f_H
This bit is used to select if the f_H clock or the $f_H/2$ ~ $f_H/64$ 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 $f_H/2$ ~ $f_H/64$ 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

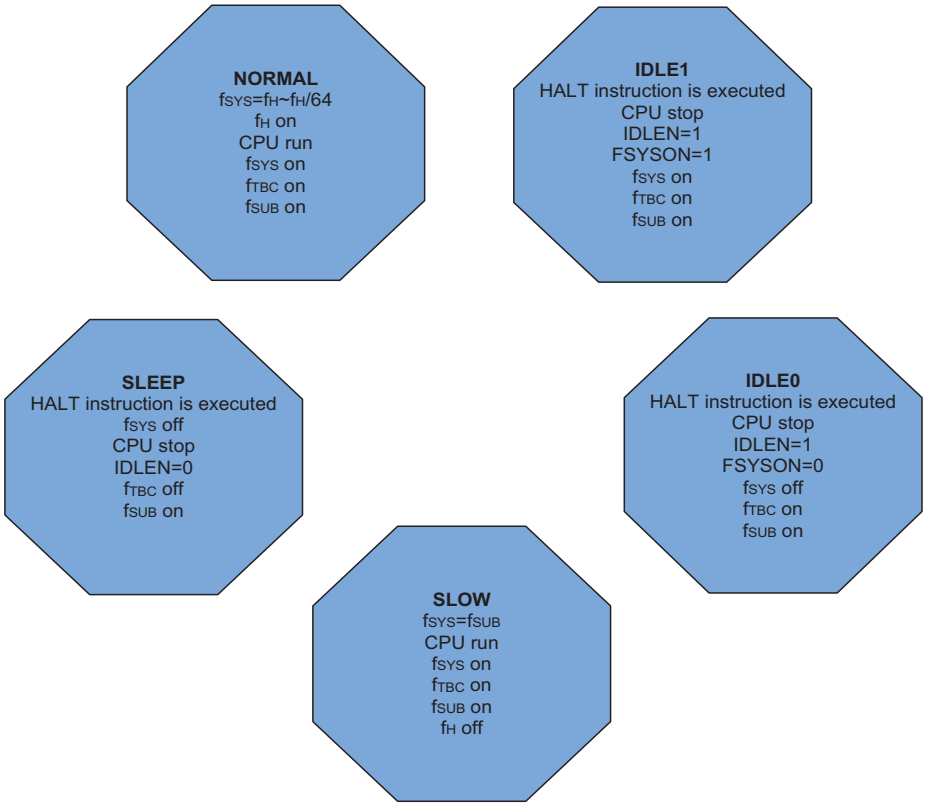
Bit 7 **FSYSON**: f_{SYS} Control in IDLE Mode
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

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

Bit 1 **LRF**: LVRC Control register software reset flag
Describe elsewhere

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



Operating Mode Switching

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

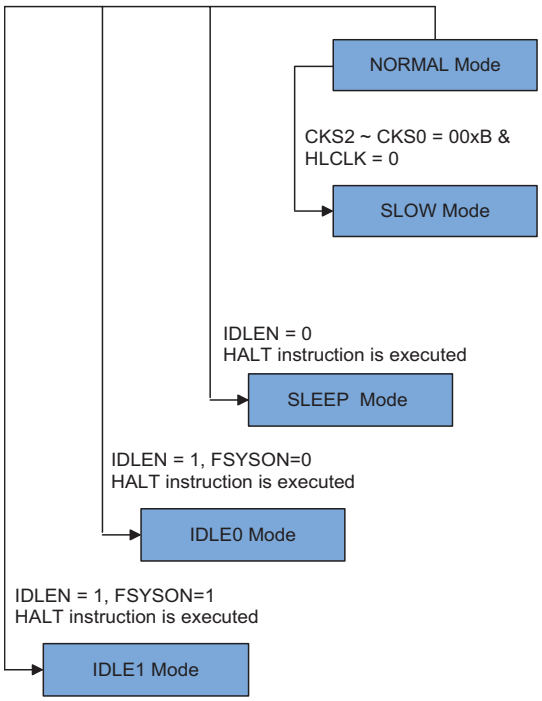
In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the 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 such as the TMs. The accompanying flowchart shows what happens when the device moves between the various operating modes.

NORMAL Mode to SLOW Mode Switching

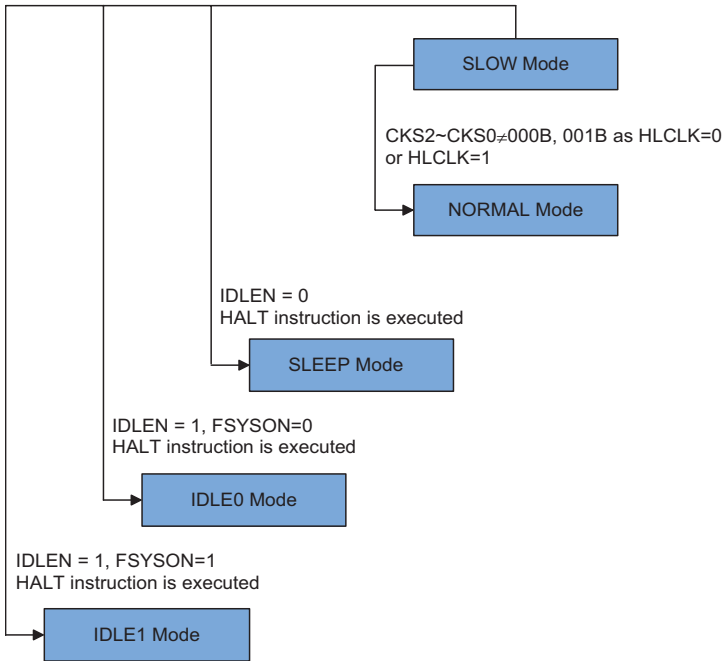
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by setting the HLCLK bit to "0" and setting the CKS2~CKS0 bits to "000" or "001" in the SMOD register. This will then use the low speed 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 LIRC oscillator and therefore requires this oscillator 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 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.



Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0". 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.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in 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 f_{TBC} and the low frequency 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.
- 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_{TBC} and the low frequency f_{SUB} 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.
- 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 devices which have different package types, as there may be unbonbed pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. In the IDLE1 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 actual source of the wake-up can be determined by examining the TO and PDF flags. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the "HALT" instruction. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal f_{SUB} clock which is in turn supplied by the LIRC oscillator. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register. 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.

Watchdog Timer Control Register

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

WDTC Register

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

Bit 7~ 3 **WE4 ~ WE0**: WDT enable bit
10101 or 01010: Enabled
Others: Reset MCU

If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the CTRL register will be set high.

Bit 2~ 0 **WS2 ~ WS0**: Select WDT Timeout Period
000: $2^8/f_{SUB}$
001: $2^{10}/f_{SUB}$
010: $2^{12}/f_{SUB}$
011: $2^{14}/f_{SUB}$
100: $2^{15}/f_{SUB}$
101: $2^{16}/f_{SUB}$
110: $2^{17}/f_{SUB}$
111: $2^{18}/f_{SUB}$

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 IDLE Mode
Describe elsewhere

Bit 6~ 3 Unimplemented, read as "0"

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

Bit 1 **LRF:** LVR Control register software reset flag
Describe 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. There are five bits, WE4~WE0, in the WDTC register to offer enable and reset control of the Watchdog Timer. When the WE4~WE0 bits value are 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 could be caused by adverse environmental conditions such as noise, it will reset the microcontroller after 2~3 LIRC clock cycles.

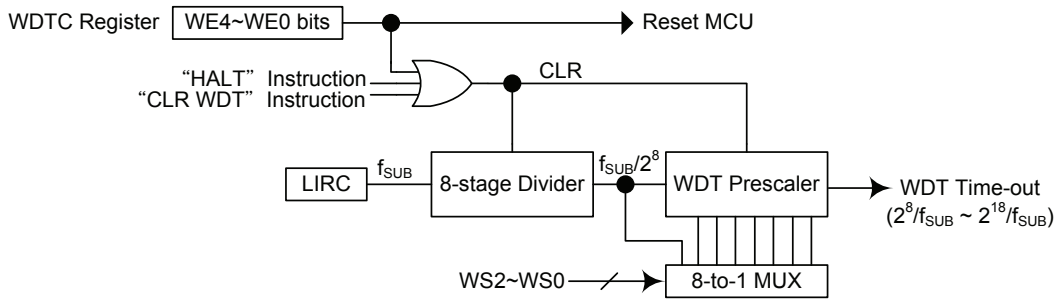
WE4 ~ WE0 Bits	WDT Function
01010B or 10101B	Enable
Any other value	Reset MCU

Watchdog Timer Enable Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT reset, which means a certain value is written into the WE4~WE0 bit filed except 01010B and 10101B, 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^{18} division ratio is selected. As an example, with a 32 kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 seconds for the 2^{18} division ratio, and a minimum timeout of 7.8ms 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.

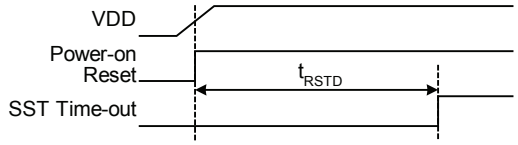
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply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

Reset Functions

There are four ways in which a microcontroller reset can occur, through events occurring internally:

Power-on Reset

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



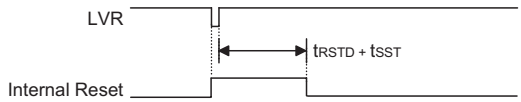
Note: t_{RSTD} is power-on delay, typical time=50ms

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 voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for greater than the value t_{LVR} specified in the LVD&LVR characteristics. If the low 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} is defined by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to any other value except some certain values defined in the LVRC register by the environmental noise, the LVR will reset the device after 2~3 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.



Note:t_{RSTD} is power-on delay, typical time=50ms

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	R	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7 ~ 0 **LVS7 ~ LVS0**: LVR Voltage Select control

01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Any other value: Generates MCU reset – register is reset to POR value

When an actual low voltage condition occurs, an MCU reset will be generated. The reset operation will be activated after 2~3 LIRC clock cycles. In this situation this register contents will remain the same after such a reset occurs.

Any register value, other than the defined values above, will also result in the generation of an MCU reset. The reset operation will be activated after 2~3 LIRC clock cycles. However in this situation this 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 IDLE Mode
Describe elsewhere

Bit 6~ 3 Unimplemented, read as "0"

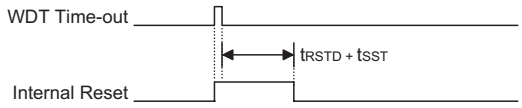
Bit 2 **LVRF:** LVR function reset flag
0: Not occur
1: Occurred
This bit is set high when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

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
Describe elsewhere

Watchdog Time-out Reset during Normal Operation

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

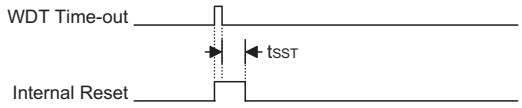


Note: t_{RSTD} is power-on delay, typical time=16.7ms

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

Note: "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	Reset (Power On)	WDT Time-out (Normal Operation)	WDT Time-out (SLEEP/IDLE)
Program Counter	0 0 0 H	0 0 0 H	0 0 0 H
MP0	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	---- ---0	---- ---0	---- ---u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	---- -xxx	---- -uuu	---- -uuu
STATUS	--00 xxxx	--1u uuuu	--11 uuuu
SMOD	110- 0010	110- 0010	uuu- uuuu
LVDC	--00 -000	--00 -000	--uu -uuu
INTEG	---- 0000	---- 0000	---- uuuu
INTC0	-00- 00-0	-00- 00-0	-uu- uu-u
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	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

Register	Reset (Power On)	WDT Time-out (Normal Operation)	WDT Time-out (SLEEP/IDLE)
MFI3	--00 --00	--00 --00	--uu --uu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PB	---- 1111	---- 1111	---- uuuu
PBC	---- 1111	---- 1111	---- uuuu
PBPU	---- 0000	---- 0000	---- uuuu
WDTC	0101 0011	0101 0011	uuuu uuuu
TBC	0011 -111	0011 -111	uuuu -uuu
EEA	--00 0000	--00 0000	--uu uuuu
EED	0000 0000	0000 0000	uuuu uuuu
EEC	---- 0000	---- 0000	---- uuuu
SADOL (ADRFS=0)	xxxx ----	xxxx ----	uuuu ----
SADOH (ADRFS=0)	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADOL (ADRFS=1)	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADOH (ADRFS=1)	---- xxxx	---- xxxx	---- uuuu
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	uuuu uuuu
CTRL	0--- -x00	0--- -000	u--- -uuu
LVRC	0101 0101	0101 0101	uuuu uuuu
TM0C0	0000 0---	0000 0---	uuuu u---
TM0C1	0000 0000	0000 0000	uuuu uuuu
TM0DL	0000 0000	0000 0000	uuuu uuuu
TM0DH	---- --00	---- --00	---- --uu
TM0AL	0000 0000	0000 0000	uuuu uuuu
TM0AH	---- --00	---- --00	---- --uu
TM0RPL	0000 0000	0000 0000	uuuu uuuu
TM0RPH	---- --00	---- --00	---- --uu
TM1C0	0000 0---	0000 0---	uuuu u---
TM1C1	0000 0000	0000 0000	uuuu uuuu
TM1DL	0000 0000	0000 0000	uuuu uuuu
TM1DH	---- --00	---- --00	---- --uu
TM1AL	0000 0000	0000 0000	uuuu uuuu
TM1AH	---- --00	---- --00	---- --uu
TM1RPL	0000 0000	0000 0000	uuuu uuuu
TM1RPH	---- --00	---- --00	---- --uu
CPR	1000 0000	1000 0000	1uuu uuuu
OCPC0	0000 ---0	0000 ---0	uuuu ---u
OCPC1	--00 0000	--00 0000	--uu uuuu
OCPDA	0000 0000	0000 0000	uuuu uuuu
OCPOCAL	0000 0000	0000 0000	uuuu uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	WDT Time-out (SLEEP/IDLE)
OCPCCAL	0001 0000	0001 0000	uuuu uuuu
CTRL2	0-00 0001	0-00 0001	u-uu uuuu
CTRL3	0000 0000	0000 0000	uuuu uuuu
CTRL4	---0 0000	---0 0000	---u uuuu
CTRL5	-000 0000	-000 0000	-uuu uuuu
TM2C0	0000 0---	0000 0---	uuuu u---
TM2C1	0000 0000	0000 0000	uuuu uuuu
TM2DL	0000 0000	0000 0000	uuuu uuuu
TM2DH	---- --00	---- --00	---- --uu
TM2AL	0000 0000	0000 0000	uuuu uuuu
TM2AH	---- --00	---- --00	---- --uu
TM2RPL	0000 0000	0000 0000	uuuu uuuu
TM2RPH	---- --00	---- --00	---- --uu

Note: "-" not implement

"u" stands for "unchanged"

"x" stands for "unknown"

Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA~PB. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory 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	—	—	—	—	PB3	PB2	PB1	PB0
PBC	—	—	—	—	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	—	—	PBPU3	PBPU2	PBPU1	PBPU0

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 PAPU~PBPU, and are implemented using weak PMOS transistors.

Note that only when the I/O ports are configured as digital intput or NMOS output, the internal pull-high functions can be enabled using the PAPU~PBPU registers. In other conditions, internal pull-high functions are disabled.

PAPU Register

Bit	7	6	5	4	3	2	1	0
Name	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
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 I/O Port A bit7~ bit 0 Pull-High Control
0: Disable
1: Enable

PBPU Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PBPU3	PBPU2	PBPU1	PBPU0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7 ~ 4 Unimplemented, read as "0"

Bit 3 ~ 0 I/O Port B bit 3~ bit 0 Pull-High Control
0: Disable
1: Enable

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 only when the Port A pins are configured as general purpose I/Os and the device is in the HALT status, the Port A wake-up functions can be enabled using the relevant bits in the PAWU register. In other conditions, the wake-up functions are disabled.

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 I/O Port A bit 7 ~ bit 0 Wake Up Control
0: Disable
1: Enable

I/O Port Control Registers

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

PAC Register

Bit	7	6	5	4	3	2	1	0
Name	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
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

Bit 7 ~ 0 I/O Port A bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

PBC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PBC3	PBC2	PBC1	PBC0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 I/O Port B bit 3~bit 0 Input/Output Control
0: Output
1: Input

Pin-shared Functions

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

Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The device includes the CTRL3 and CTRL4 registers which can select the desired functions of the multi-function pin-shared pins.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. 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 pn-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.

Pin-shared Function Selection Registers List

Name	Bit							
	7	6	5	4	3	2	1	0
CTRL3	IOCN7	IOCN6	IOCN5	IOCN4	IOCN3	IOCN2	IOCN1	IOCN0
CTRL4	—	—	—	IOCN12	IOCN11	IOCN10	IOCN9	IOCN8

CTRL3 Register

Bit	7	6	5	4	3	2	1	0
Name	IOCN7	IOCN6	IOCN5	IOCN4	IOCN3	IOCN2	IOCN1	IOCN0
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 **IOCN7~IOCN6**: PA5 pin function selection
00: INT1/TCK1/PA5
01: AN3
10: OCP1
11: INT1/TCK1/PA5
The INT1 or TCK1 pin is furtherly selected using the corresponding function selection bits in the interrupt control register or TM control register.

Bit 5~4 **IOCN5~IOCN4**: PA4 pin function selection
00: PA4
01: AN2
10: TP1
11: PA4

Bit 3~2 **IOCN3~IOCN2**: PA3 pin function selection
00: INT0/TCK0/PA3
01: AN1
10: VREF
11: INT0/TCK0/PA3
The INT0 or TCK0 pin is furtherly selected using the corresponding function selection bits in the interrupt control register or TM control register.

Bit 1~0 **IOCN1~IOCN0**: PA1 pin function selection
00: PA1
01: AN0
10: TP0
11: PA1

CTRL4 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	IOCN12	IOCN11	IOCN10	IOCN9	IOCN8
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"

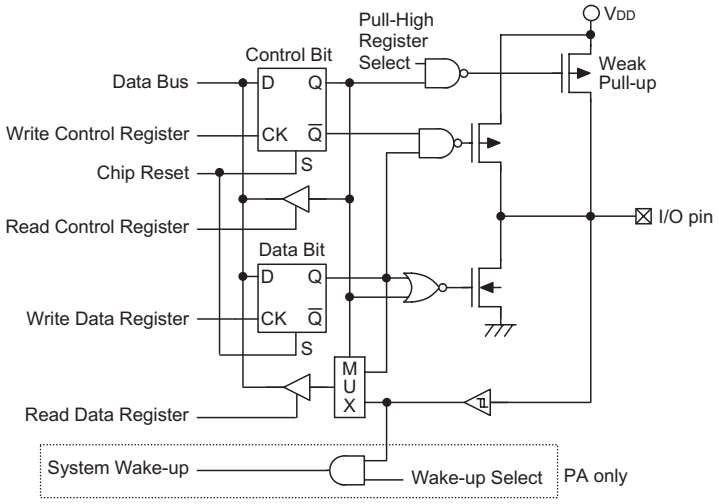
Bit 4 **IOCN12**: PB0 pin function selection
0: PB0
1: PWM0H

Bit 3~2 **IOCN11~IOCN10**: PA7 pin function selection
00: TCK2/PA7
01: AN5
10: PWM0L
11: TCK2/PA7
The TCK2 pin is furtherly selected using the corresponding function selection bits in the TM control register.

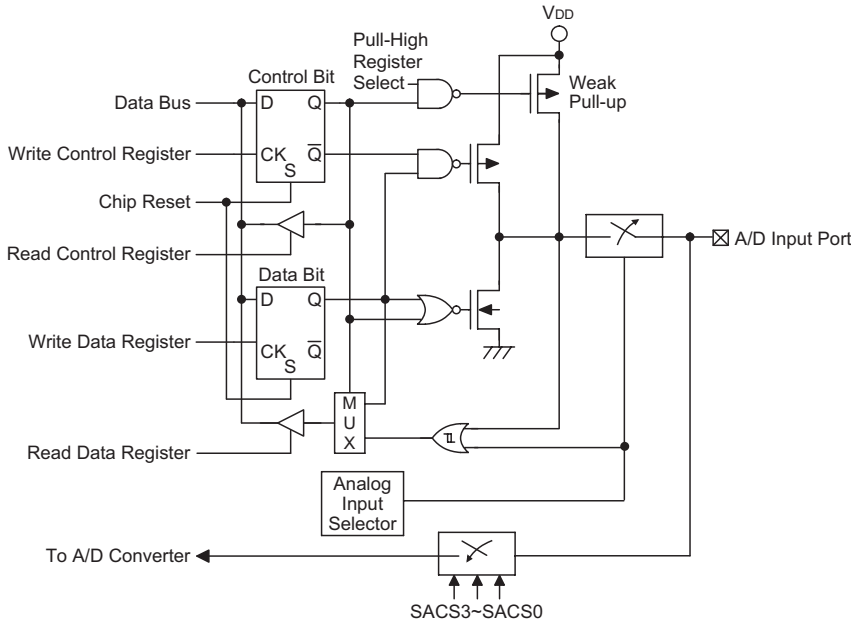
Bit 1~0 **IOCN9~IOCN8**: PA6 pin function selection
00: PA6
01: AN4
10: OCP0
11: TP2

I/O Pin Structures

The accompanying diagrams illustrate the internal structures of some generic I/O pin types. As the exact logical construction of the I/O pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins. The wide range of pin-shared structures does not permit all types to be shown.



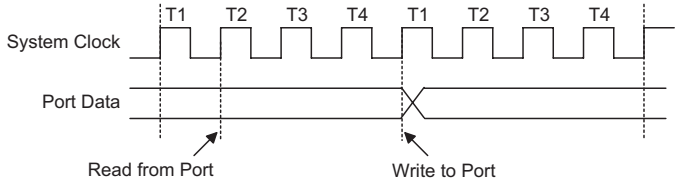
Generic Input/Output Structure



A/D Input/Output Structure

Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers, PAC~PBC, are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers, PA~PB, 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.



Read/Wite Timing

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.

Introduction

The device contains three 10-bit Periodic TMs, each TM having a reference name of TM0, TM1 and TM2. The main features of the TMs 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

TM Function Summary

TM0	TM1	TM2
10-bit PTM	10-bit PTM	10-bit PTM

TM Name/Type Reference

TM Operation

The 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 each TM can originate from various sources. The selection of the required clock source is implemented using the TnCK2~TnCK0 bits in the TM control registers. The clock source can be a ratio of either the system clock f_{SYS} or the internal high clock f_H, the f_{TBC} clock source or the external TCKn pin. The TCKn pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

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

TM External Pins

Each of the TMs, irrespective of what type, has one TM input pin, with the label TCKn. The TM input pin, is essentially a clock source for the TM and is selected using the TnCK2~TnCK0 bits in the TMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. This external TM input pin is shared with other functions but will be connected to the internal TM if selected using the TnCK2~TnCK0 bits. The TM input pin can be chosen to have either a rising or falling active edge.

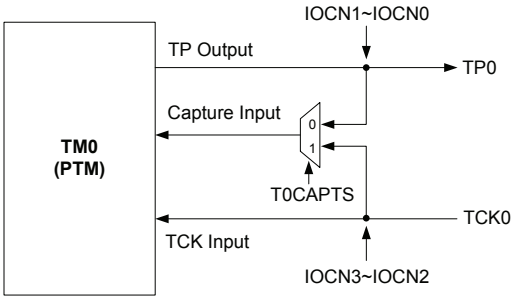
The TMs each have one output pin which is selected using the corresponding pin-shared function selection bits described in the Pin-shared Function section. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external TPn output pin is also the pin where the TM generates the PWM output waveform. As the TM output pins are pin-shared with other function, the TM output function must first be setup using relevant pin-shared function selection register.

Type	TM0	TM1	TM2	Pin Control Registers
Input	TCK0	TCK1	TCK2	CTRL3 or CTRL4
Output	TP0	TP1	TP2	CTRL3 or CTRL4

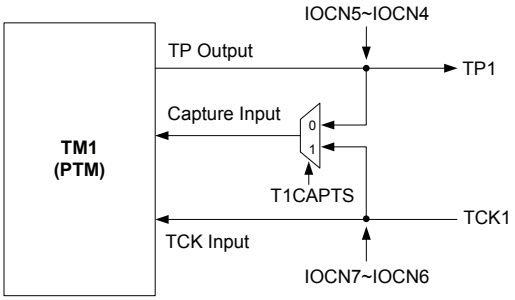
TM External Pins

TM Input/Output Pin Control

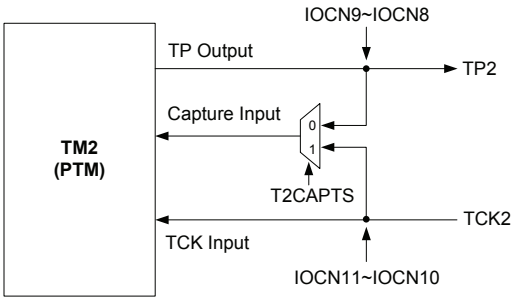
Selecting to have a TM input/output or whether to retain its other shared function is implemented using the relevant pin-shared function selection registers, with the corresponding selection bits in each pin-shared function register corresponding to a TM input/output pin. Configuring the selection bits 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.



TM0 Function Pin Control Block Diagram



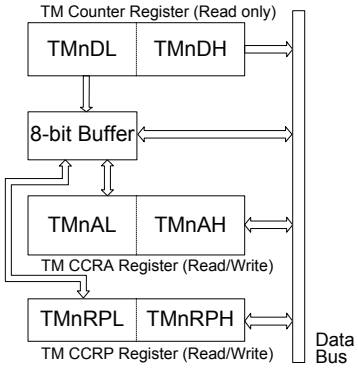
TM1 Function Pin Control Block Diagram



TM2 Function Pin Control Block Diagram

Programming Considerations

The TM Counter Registers, the Capture/Compare CCRA and the CCRP registers, being 10-bit, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed. As the CCRA and CCRP registers are implemented in the way shown in the following diagram and accessing these register pairs is carried out in a specific way described above, it is recommended to use the "MOV" instruction to access the CCRA or CCRP low byte registers, named TMnAL or TMnRPL, 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 TMnAL or TMnRPL
    – note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte TMnAH or TMnRPH
    – 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 TMnDH, TMnAH or TMnRPH
    – 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 TMnDL, TMnAL or TMnRPL
    – this step reads data from the 8-bit buffer.

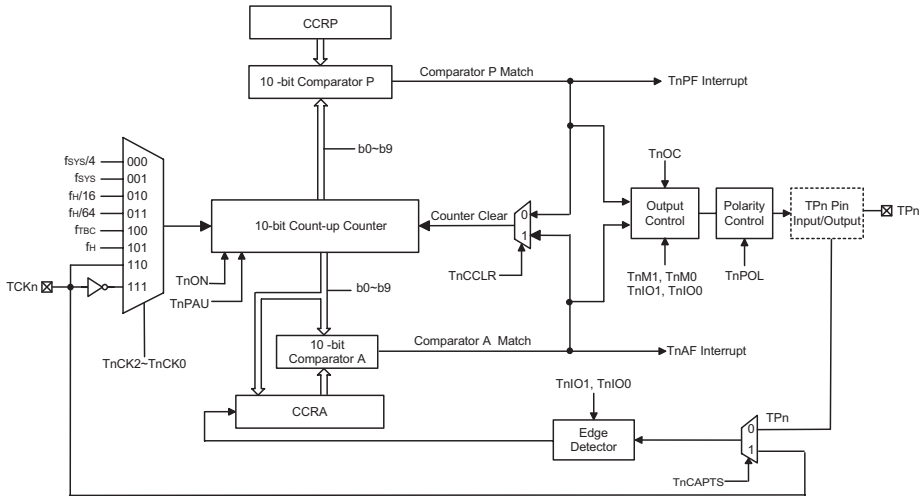
Periodic Type TM – PTM

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

Periodic TM Operation

At its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with the CCRA and CCRP registers.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the TnON 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 TM 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 the output pin. All operating setup conditions are selected using relevant internal registers.



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

Periodic Type TM Register Description

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

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMnC0	TnPAU	TnCK2	TnCK1	TnCK0	TnON	—	—	—
TMnC1	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnCAPTS	TnCCLR
TMnDL	D7	D6	D5	D4	D3	D2	D1	D0
TMnDH	—	—	—	—	—	—	D9	D8
TMnAL	D7	D6	D5	D4	D3	D2	D1	D0
TMnAH	—	—	—	—	—	—	D9	D8
TMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
TMnRPH	—	—	—	—	—	—	D9	D8

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

TMnC0 Register

Bit	7	6	5	4	3	2	1	0
Name	TnPAU	TnCK2	TnCK1	TnCK0	TnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **TnPAU**: TMn 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 TM 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 **TnCK2 ~ TnCK0**: Select TMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: f_H
110: TCKn rising edge clock
111: TCKn falling edge clock
These three bits are used to select the clock source for the TM. 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_{TBC} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **TnON**: TMn Counter On/Off Control
0: Off
1: On
This bit controls the overall on/off function of the TM. Setting the bit high enables the counter to run, clearing the bit disables the TM. Clearing this bit to zero will stop the counter from counting and turn off the TM 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 TM is in the Compare Match Output Mode then the TM output pin will be reset to its initial condition, as specified by the TM Output control bit, when the bit changes from low to high.

Bit 2 ~ 0 Unimplemented, read as "0"

TMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnCAPTS	TnCCLR
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 **TnM1~TnM0**: Select TMn Operation Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the TM. To ensure reliable operation the TM should be switched off before any changes are made to the TnM1 and TnM0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5 ~ 4 **TnIO1~TnIO0**: Select TPn output function

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

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

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

Timer/counter Mode
Unused

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

In the Compare Match Output Mode, the TnIO1 and TnIO0 bits determine how the TM output pin changes state when a compare match occurs from the Comparator A. The TM 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 these bits are both zero, then no change will take place on the output. The initial value of the TM output pin should be setup using the TnOC bit. Note that the output level requested by the TnIO1 and TnIO0 bits must be different from the initial value setup using the TnOC bit otherwise no change will occur on the TM output pin when a compare match occurs. After the TM output pin changes state it can be reset to its initial level by changing the level of the TnON bit from low to high.

In the PWM Mode, the TnIO1 and TnIO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to change the values of the TnIO1 and TnIO0 bits only after the TM has been switched off. Unpredictable PWM outputs will occur if the TnIO1 and TnIO0 bits are changed when the TM is running.

Bit 3 **TnOC**: TPn Output control bit

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

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

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

Bit 2 **TnPOL**: TPn Output polarity Control
0: Non-invert
1: Invert

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

Bit 1 **TnCAPTS**: TMn capture trigger source select
0: From TPn pin
1: From TCKn pin

Bit 0 **TnCCLR**: Select TMn Counter clear condition
0: TMn Comparatror P match
1: TMn Comparatror 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 TnCCLR 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 TnCCLR bit is not used in the PWM, Single Pulse or Input Capture Mode.

TMnDL Register

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

Bit 7 ~ 0 **TMnDL**: TMn Counter Low Byte Register bit 7 ~ bit 0
TMn 10-bit Counter bit 7 ~ bit 0

TMnDH Register

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

Bit 7 ~ 2 Unimplemented, read as “0”

Bit 1 ~ 0 **TMnDH**: TMn Counter High Byte Register bit 1 ~ bit 0
TMn 10-bit Counter bit 9 ~ bit 8

TMnAL 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 **TMnAL**: TMn CCRA Low Byte Register bit 7 ~ bit 0
TMn 10-bit CCRA bit 7 ~ bit 0

TMnAH Register

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

Bit 7 ~ 2 Unimplemented, read as "0"

Bit 1 ~ 0 **TMnAH**: TMn CCRA High Byte Register bit 1 ~ bit 0
TMn 10-bit CCRA bit 9 ~ bit 8

TMnRPL 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 **TMnRPL**: TMn CCRP Low Byte Register bit 7 ~ bit 0
TMn 10-bit CCRP bit 7 ~ bit 0

TMnRPH Register

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

Bit 7 ~ 2 Unimplemented, read as "0"

Bit 1 ~ 0 **TMnRPH**: TMn CCRP High Byte Register bit 1 ~ bit 0
TMn 10-bit CCRP bit 9 ~ bit 8

Periodic Type TM Operating Modes

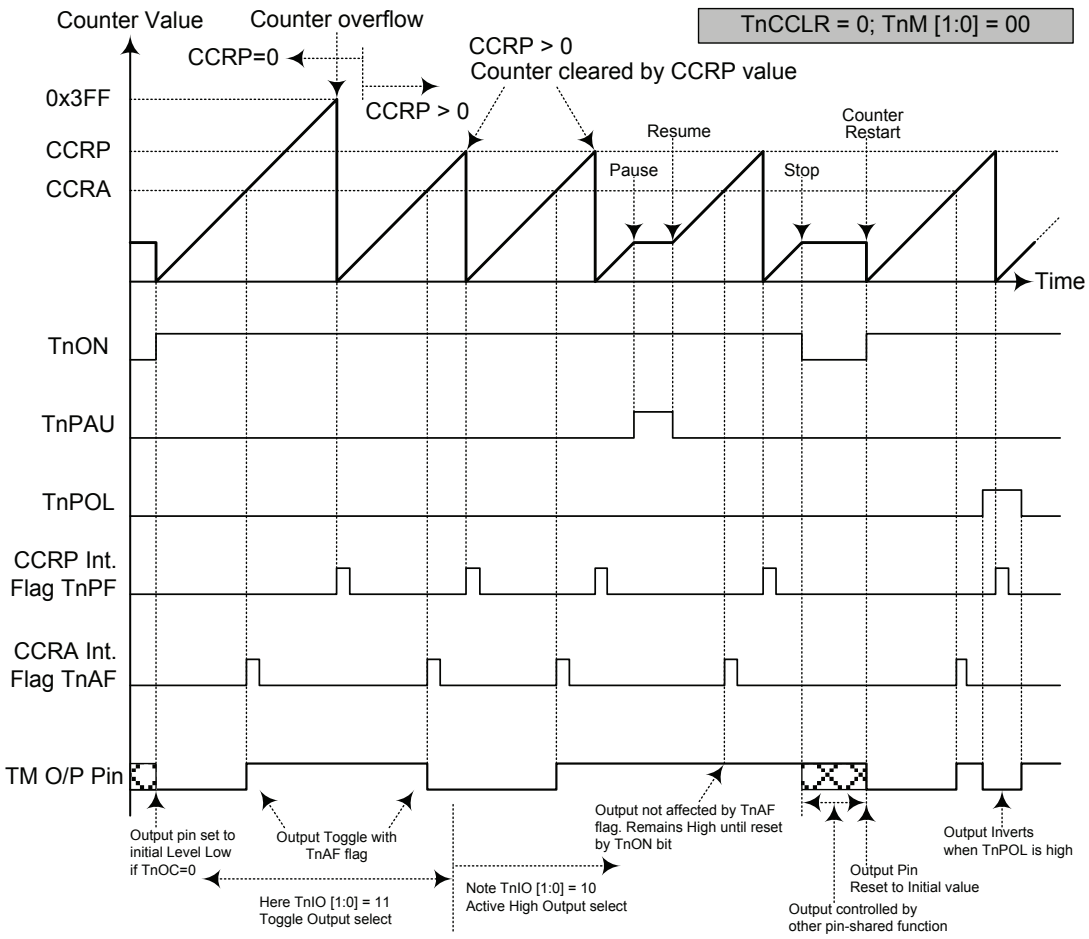
The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the TnM1 and TnM0 bits in the TMnC1 register.

Compare Match Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register, should be all cleared 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 TnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match occurs from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both the TnAF and TnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the TnCCLR bit in the TMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the TnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when TnCCLR is high no TnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”.

As the name of the mode suggests, after a comparison is made, the TM output pin, will change state. The TM output pin condition however only changes state when a TnAF interrupt request flag is generated after a compare match occurs from Comparator A. The TnPF interrupt request flag, generated from a compare match from Comparator P, will have no effect on the TM output pin. The way in which the TM output pin changes state are determined by the condition of the TnIO1 and TnIO0 bits in the TMnC1 register. The TM output pin can be selected using the TnIO1 and TnIO0 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 TM output pin, which is setup after the TnON bit changes from low to high, is setup using the TnOC bit. Note that if the TnIO1, TnIO0 bits are zero then no pin change will take place.



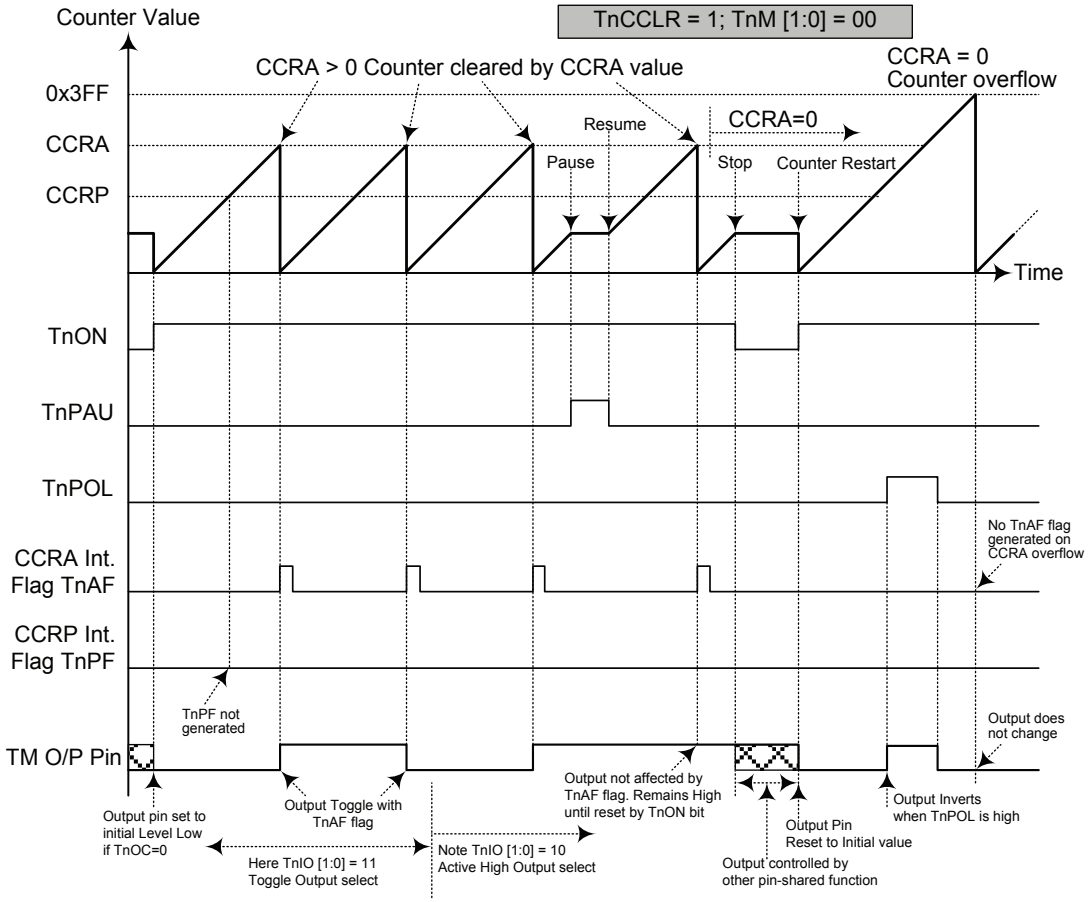
Compare Match Output Mode – TnCCLR = 0

Note: 1. With TnCCLR = 0 -- a Comparator P match will clear the counter

2. The TM output pin is controlled only by the TnAF flag

3. The output pin is reset to initial state by a TnON bit rising edge

4. n=0~2



Compare Match Output Mode – TnCCLR = 1

Note: 1. With TnCCLR = 1 -- a Comparator A match will clear the counter

2. The TM output pin is controlled only by the TnAF flag

3. The output pin is reset to initial state by a TnON rising edge

4. The TnPF flag is not generated when TnCCLR = 1

5. n=0~2

Timer/Counter Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should all 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 TnM1 and TnM0 in the TMnC1 register should be set to 10 respectively and also the TnIO1 and TnIO0 bits should be set to 10 respectively. The PWM function within the TM 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 TM 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 mode, the TnCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA 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 TnOC bit in the TMnC1 register is used to select the required polarity of the PWM waveform while the two TnIO1 and TnIO0 bits are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The TnPOL bit is used to reverse the polarity of the PWM output waveform.

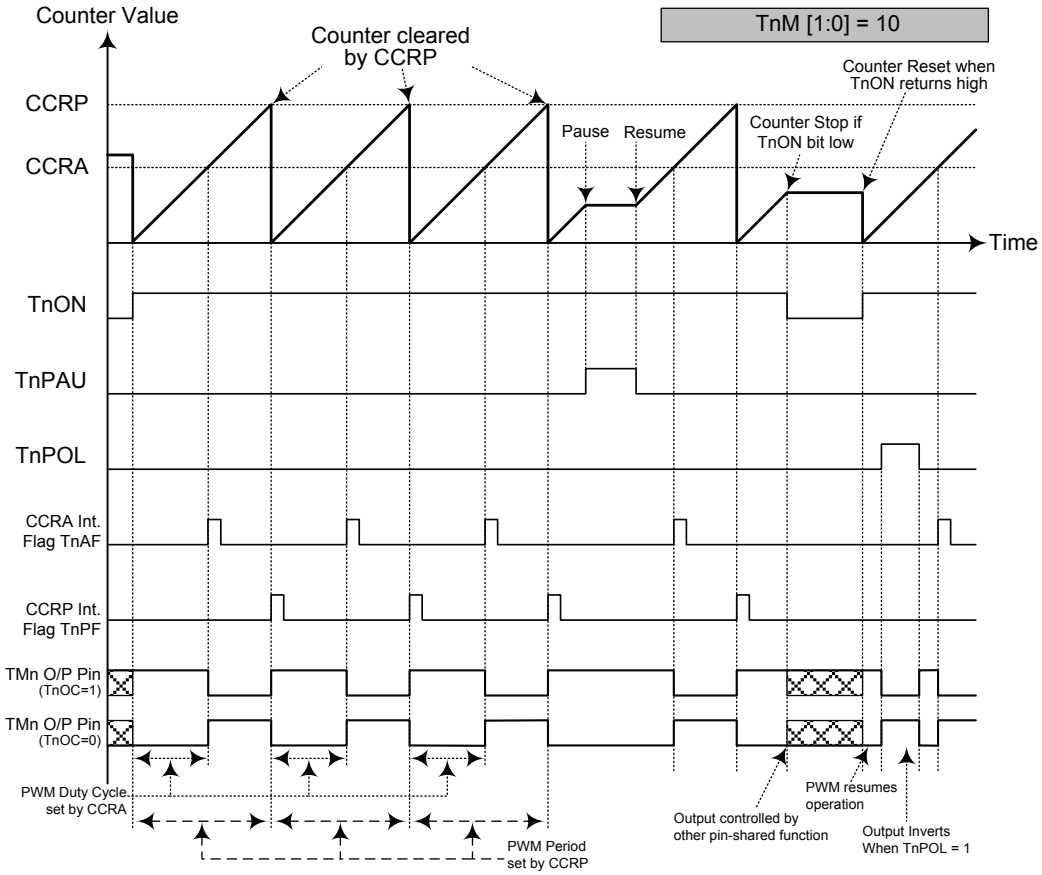
10-bit PTM, PWM Mode

Period	Duty
CCRP	CCRA

If f_H = 20MHz, TM clock source select f_H, CCRP = 200 and CCRA = 50,

The TM PWM output frequency = (f_H) / 200 = 20MHz/200 = 100 kHz, duty = 50/200 = 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 Mode

Note: 1. Here Counter cleared by CCRP

2. A counter clear sets the PWM Period

3. The internal PWM function continues running even when TnIO[1:0] = 00 or 01

4. The TnCCLR bit has no influence on PWM operation

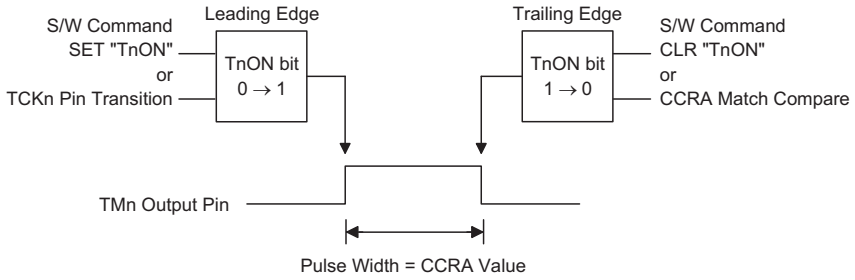
5. n=0~2

Single Pulse Output Mode

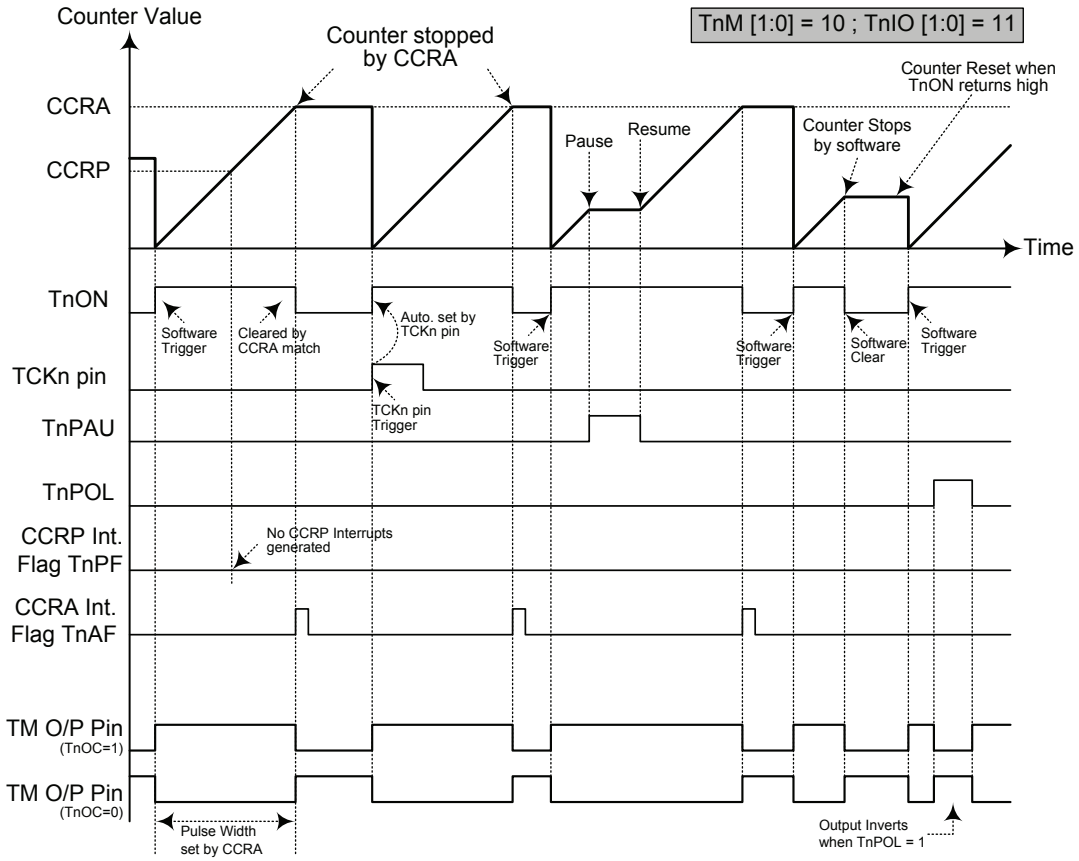
To select this mode, the required bit pairs, TnM1 and TnM0 should be set to 10 respectively and also the corresponding TnIO1 and TnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the TM output pin.

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



Single Pulse Generation (n=0~2)



Single Pulse Mode

Note: 1. Counter stopped by CCRA

2. CCRP is not used

3. The pulse is triggered by the TCKn pin or by setting the TnON bit high

4. A TCKn pin active edge will automatically set the TnON bit high

5. In the Single Pulse Mode, TnIO [1:0] must be set to "11" and can not be changed.

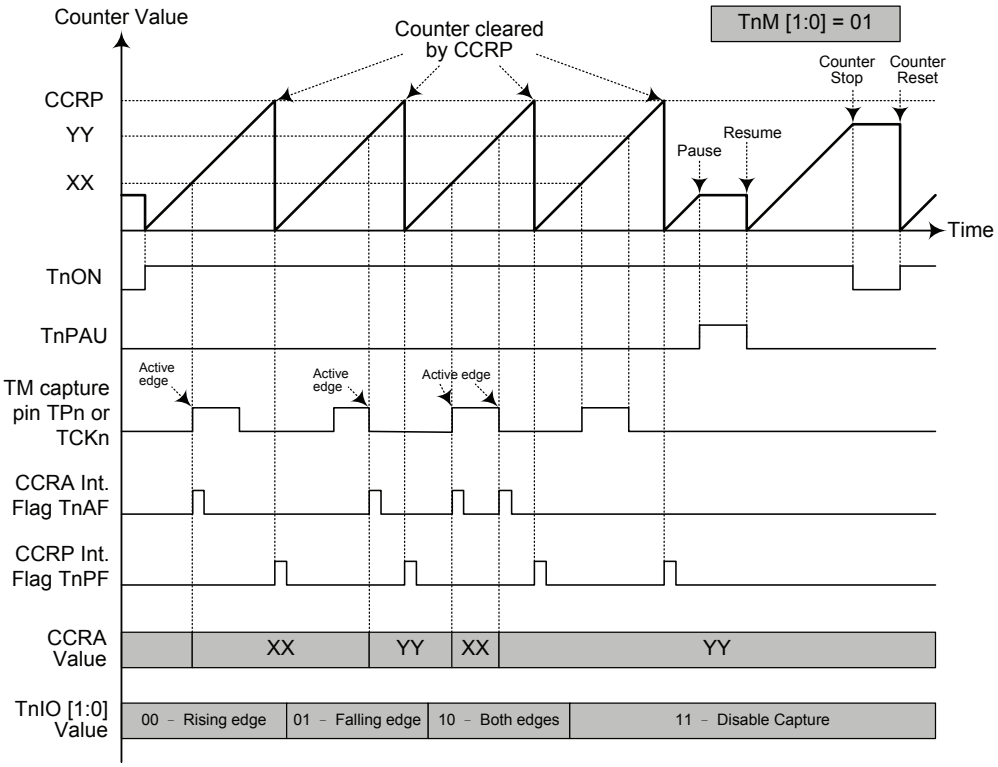
6. n=0~2

Capture Input Mode

To select this mode bits TnM1 and TnM0 in the TMnC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the TPn or TCKn pin, selected by the TnCAPTS bit in the TMnC1 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 TnIO1 and TnIO0 bits in the TMnC1 register. The counter is started when the TnON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the TPn or TCKn pin the present value in the counter will be latched into the CCRA register and a TM interrupt generated. Irrespective of what events occur on the TPn or TCKn pin the counter will continue to free run until the TnON 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 TM 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 TnIO1 and TnIO0 bits can select the active trigger edge on the TPn or TCKn pin to be a rising edge, falling edge or both edge types. If the TnIO1 and TnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the TPn or TCKn pin, however it must be noted that the counter will continue to run.

As the TPn or TCKn pin is pin shared with other functions, care must be taken if the TMn 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 TnCCLR, TnOC and TnPOL bits are not used in this Mode.



Capture Input Mode

Note: 1. TnM[1:0] = 01 and active edge set by the TnIO[1:0] bits

2. A TM Capture input pin active edge transfers counter value to CCRA

3. The TnCCLR bit is not used

4. No output function – TnOC and TnPOL bits are not used

5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero

6. n=0~2

Analog to Digital Converter

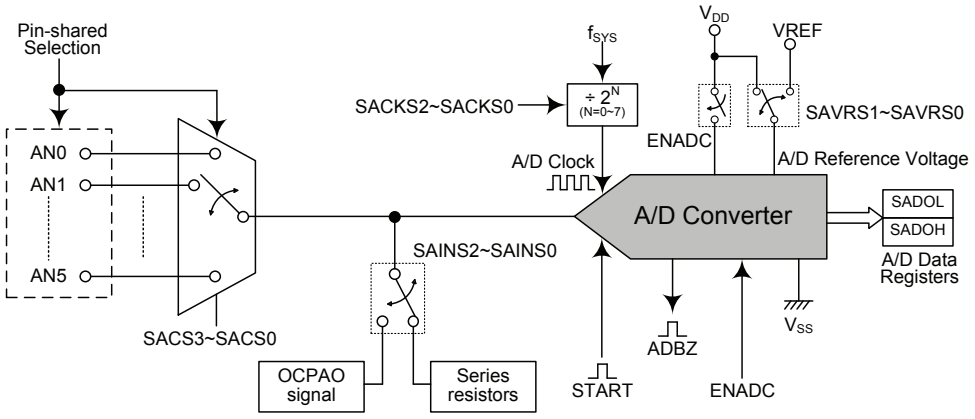
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the OCPAO signal from the OCP function or DC to DC voltage from the series resistors, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 bits together with the SACS3~SACS0 bits. Note that when the internal analog signal is to be converted, the pin-shared control bits should also be properly configured except the SAINS and SACS bit fields. More detailed information about the A/D input signal is described in the "A/D Converter Control Registers" and "A/D Converter Input Signal" sections respectively.

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

External Input Channels	A/D Channel Select Bits	Input Pins
6	SACS3~SACS0	AN0~AN5



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 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	ADBZ	ENADC	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

A/D Converter Registers List

A/D Converter Data Registers – SADOL, SADOH

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

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

A/D Converter Data Registers

A/D Converter Control Registers – SADC0, SADC1

To control the function and operation of the A/D converter, two control registers known as SADC0, 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 busy status. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input. If the SAINS2~SAINS0 bits are set to “000” or “011~111”, the external analog channel input is selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected to be converted. If the SAINS2~SAINS0 bits are set to “001”, the OCPAO signal from the OCP function is selected to be converted. If the SAINS2~SAINS0 bits are set to “010”, the DC/DC voltage of the series resistors is selected to be converted. Care must be taken when the internal analog signal is selected to be converted. If the internal analog signal is selected to be converted, the SACS3~SACS0 bits must be properly set to select a floating state. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

SAINS [2:0]	SACS [3:0]	Input Signals	Description
Except 001 and 010	0000~0101	AN0~AN5	External pin analog input
	0110~1111	—	Floating
001	0110~1111	OPA output	OCPAO signal from the OCP function
010	0110~1111	DC/DC voltage	DC/DC voltage of the series resistors

A/D Converter Input Signal Selection

SADC0 Register

Register Name	Bit							
	7	6	5	4	3	2	1	0
Name	START	ADBZ	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	0	0	0	0	0	0	0

Bit 7 **START**: Start the A/D Conversion
0→1→0: Start an A/D conversion
0→1: Reset the A/D converter and clear the ADBZ flag to "0"
1→0: Start the A/D conversion and set the ADBZ flag 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.

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

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

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

Bit 3~0 **SACS3~SACS0**: A/D converter external analog input channel select
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110~1111: Floating

SADC1 Register

Register Name	Bit							
	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~5 **SAINS2~SAINS0**: A/D converter input signal select
000: External source – External analog channel input
001: Internal source – OCPAO signal from the OCP function
010: Internal source – DC/DC voltage of the series resistors
011~111: External source – External analog channel input

Care must be taken if the SAINS2~SAINS0 bits are set to "001" or "010" to select the internal analog signal to be converted. When the internal analog signal is selected to be converted, the SACS3~SACS0 bits must be set to a value from "0110" to "1111". Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage select
00: From VREF pin
01: From VDD pin
1x: From VREF pin

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 bits are used to select the clock source for the A/D converter.

A/D Operation

The START bit 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. When the START bit is brought from low to high but not low again, the A/D conversion will not be initiated.

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

Although the A/D clock source is determined by the system clock f_{SYS}, and by bits SACKS2~SADCKS0, 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 selected system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the SACKS2~SADCKS0 bits should not be set to 000 or 11x. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values.

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)
1 MHz	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *	128μs *
2 MHz	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *
4 MHz	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *
8 MHz	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *
12 MHz	83ns *	167ns *	333ns *	667ns	1.33μs	2.67μs	5.33μs	10.67μs *
16 MHz	62.5ns *	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs
20 MHz	50ns *	100ns *	200ns *	400ns *	800ns	1.6μs	3.2μs	6.4μs

A/D Clock Period Examples

Controlling the power on/off function of the A/D converter circuitry is implemented using the 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 by configuring the corresponding pin-shared control bits, 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.

The reference voltage supply to the A/D Converter can be supplied from either the positive power supply pin, VDD, or from an external reference sources supplied on pin VREF. The desired selection is made using the SAVRS1 and SAVRS0 bits. When the SAVRS bit field is set to "01", the A/D converter reference voltage will come from the VDD pin. Otherwise, if the SAVRS bit field is set to any other value except "01", the A/D converter reference voltage will come from the VREF pin. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bits should be properly configured to disable other pin functions.

SAVRS [1:0]	Reference Voltage	Description
00	V_{REF}	A/D Converter Reference voltage comes from VREF pin
01	V_{DD}	A/D Converter Reference voltage comes from VDD pin
1x	V_{REF}	A/D Converter Reference voltage comes from VREF pin

A/D Converter Reference Voltage Selection

A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins as well as other functions. The corresponding pin-shared function selection bits for each pin in the CTRL3 and CTRL4 registers, determine whether the external pins are setup as A/D converter analog channel inputs or they have other functions. If the corresponding pin is setup to be an A/D converter analog channel input, the original pin functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the relevant pin-shared function selection bits enable an A/D analog channel input, the status of the port control register will be overridden.

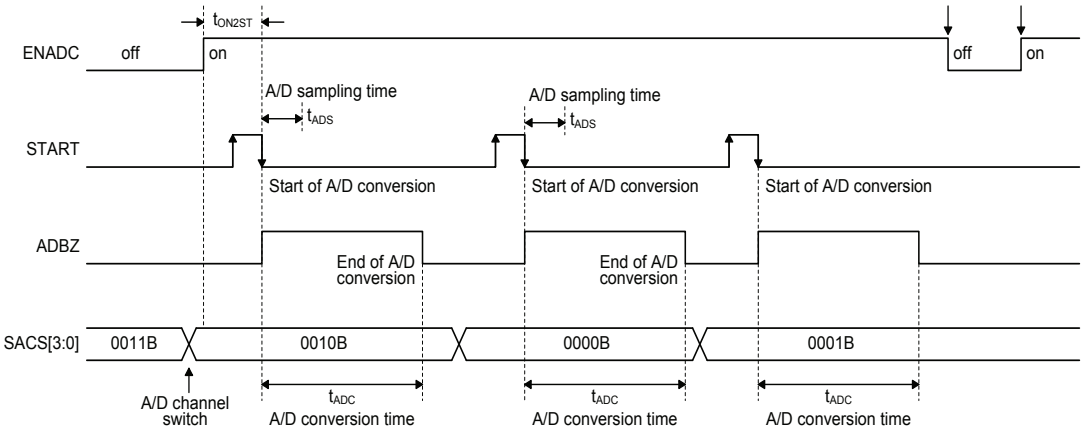
The A/D converter has its own reference voltage pin, VREF. However, the reference voltage can also be supplied from the power supply pin, a choice which is made through the SAVRS1 and SAVRS0 bits in the SADC1 register. The analog input values must not be allowed to exceed the value of V_{REF}.

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 A/D conversion which is defined as t_{ADC} are necessary.

Maximum single A/D conversion rate = A/D clock period / 16

The accompanying diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is 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 properly programming the SACKS2~SACKS0 bits in the SADC1 register.
- Step 2
  Enable the A/D converter 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 SAINS2~SAINS0 bits
  Select the external channel input to be converted, go to Step 4.
  Select the internal analog signal to be converted, go to Step 5.
- Step 4
  If the A/D input signal comes from the external channel input selecting by configuring the SAINS bit field, the corresponding pins should first be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS bit field. After this step, go to Step 6.
- Step 5
  Before the A/D input signal is selected to come from the internal analog signal by configuring the SAINS bit field, the SACS bit field must be first configured to a value from “0110” to “1111” to disconnect the external channel input. The desired internal analog signal then can be selected by configuring the SAINS bit field. After this step, go to Step 6.
- Step 6
  Select the reference voltgage source by configuring the SAVRS1~SAVRS0 bits.
- Step 7
  Select the A/D converter output data format by configuring the ADRFS bit.
- Step 8
  If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt bontrol bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 9
  The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10
  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

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

Programming Considerations

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

A/D Transfer Function

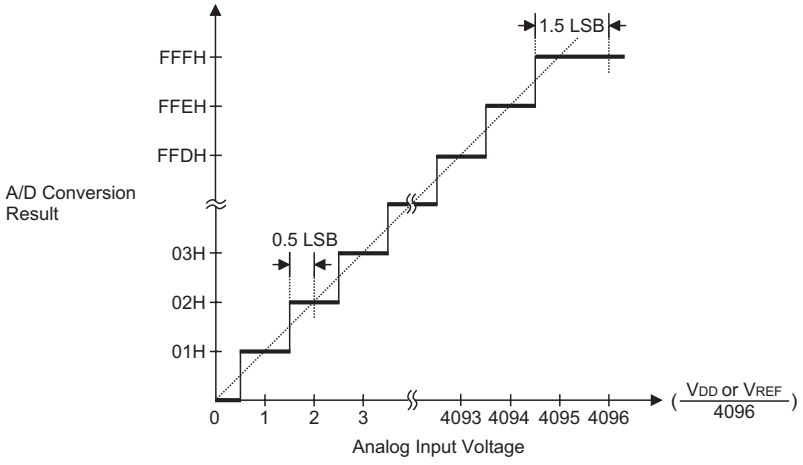
As the devices contain a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the V_{DD} or V_{REF} voltage, this gives a single bit analog input value of V_{DD} or V_{REF} divided by 4096.

$$1\ LSB = (V_{DD} \text{ or } V_{REF}) \div 4096$$

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

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

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_{DD} or V_{REF} level.



Ideal A/D Transfer Function

A/D Programming Examples

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

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

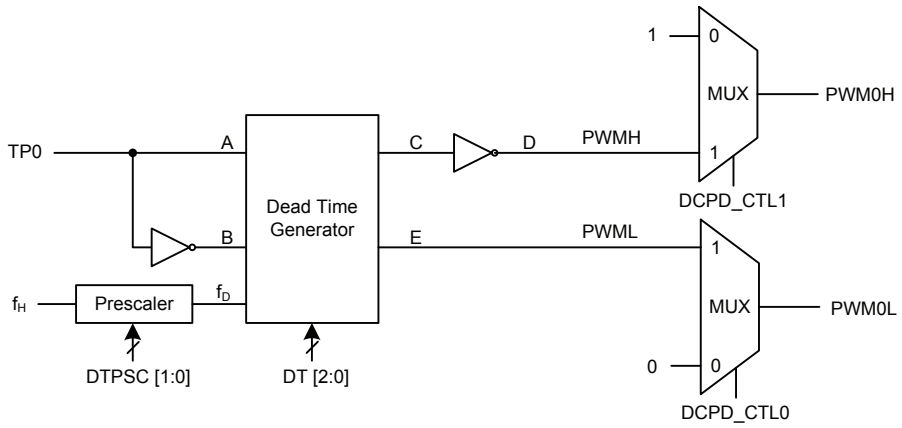
```
clr ADE                 ; disable ADC interrupt
mov a,03H
mov SADC1,a             ; select fSYS/8 as A/D clock
set ENADC
mov a,01H               ; setup CTRL3 to configure pin AN0
mov CTRL3,a
mov a,20H
mov SADC0,a             ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr START               ; high pulse on start bit to initiate conversion
set START               ; reset A/D
clr START               ; start A/D
:
polling_EOC:
sz  ADBZ                ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC         ; continue polling
:
mov a,SADOL             ; read low byte conversion result value
mov ADRL_buffer,a       ; save result to user defined register
mov a,SADOH             ; read high byte conversion result value
mov ADRH_buffer,a       ; save result to user defined register
:
jmp start_conversion    ; start next A/D conversion
```

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

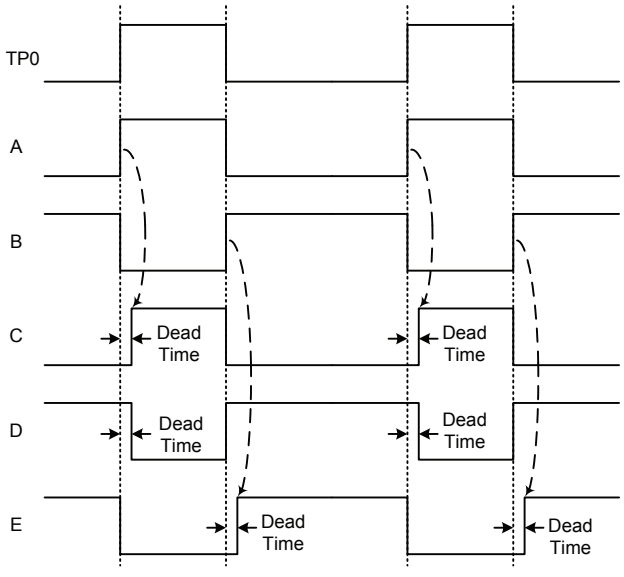
```
clr ADE                 ; disable ADC interrupt
mov a,03H
mov SADC1,a             ; select fSYS/8 as A/D clock
set ENADC
mov a,01h               ; setup CTRL3 to configure pin AN0
mov CTRL3,a
mov a,20h
mov SADC0,a             ; enable and connect AN0 channel to A/D converter
:
Start_conversion:
clr 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_ISR:                ; ADC interrupt service routine
mov acc_stack,a         ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a      ; save STATUS to user defined memory
:
mov a, SADOL            ; read low byte conversion result value
mov adrl_buffer,a       ; save result to user defined register
mov a, SADOH            ; read high byte conversion result value
mov adrh_buffer,a       ; save result to user defined register
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a            ; restore STATUS from user defined memory
mov a,acc_stack         ; restore ACC from user defined memory
reti
```

Complementary PWM output

The device provides a complementary output pair of signals which can be used as a PWM driver signal. The signal is sourced from the TM0 output signal, TP0. For PMOS type upper side driving, the PWM output is an active low signal while for NMOS type lower side driving the PWM output is an active high signal. When these complementary PWM outputs are both used to drive the upper and low sides, the dead time generator must be enabled using the DTEN bit in the CPR register, and then a dead time, which is programmable using the DTPSC and DT bit fields in the CPR register, will be inserted to prevent excessive DC currents. The dead time will be inserted whenever the rising edge of the dead time generator input signal occurs. With a dead time insertion, the output signals experience a delay before being eventually sent out to the external power transistors. The PWM0H or PWM0L signal, can be controlled by the PWMH or PWML signals repectively or remains at a certain level, these are determined by the DCPD_CTL1 and DCPD_CTL0 bits repectively. The PWM0H and PWM0L pins are pin-shared with other funcitons and can be selected as complementary PWM outputs by the revelant pin-shared control bits in the CTRL4 register.



Complementary PWM Output Block Diagram



Complementary PWM Output Waveform

CPR Register

Bit	7	6	5	4	3	2	1	0
Name	DTEN	PWMHPOL	PWMLPOL	DTPSC1	DTPSC0	DT2	DT1	DT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	0	0	0	0	0

Bit 7 **DTEN:** Dead time enable
0: Disable
1: Enable

Bit 6 **PWMHPOL:** PWMH Output polarity control
0: Non-invert
1: Invert

Bit 5 **PWMLPOL:** PWML Output polarity control
0: Non-invert
1: Invert

Bit 4 ~ 3 **DTPSC1~DTPSC0:** Dead time prescaler division ratio select
00: $f_D=f_H/1$
01: $f_D=f_H/2$
10: $f_D=f_H/4$
11: $f_D=f_H/8$

Bit 2 ~ 0 **DT2~DT0:** Dead time select
000: dead time is $[(1/f_D)-(1/f_H)] \sim (1/f_D)$
001: dead time is $[(2/f_D)-(1/f_H)] \sim (2/f_D)$
010: dead time is $[(3/f_D)-(1/f_H)] \sim (3/f_D)$
011: dead time is $[(4/f_D)-(1/f_H)] \sim (4/f_D)$
100: dead time is $[(5/f_D)-(1/f_H)] \sim (5/f_D)$
101: dead time is $[(6/f_D)-(1/f_H)] \sim (6/f_D)$
110: dead time is $[(7/f_D)-(1/f_H)] \sim (7/f_D)$
111: dead time is $[(8/f_D)-(1/f_H)] \sim (8/f_D)$

CTRL5 Register

Bit	7	6	5	4	3	2	1	0
Name	—	HV_S1	HV_S0	DCPD_CTL1	DCPD_CTL0	BZ_S1	BZ_S0	BZ_CTL
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 6~5 **HV_S1~HV_S0**: Control signal type selection for HV_EN
Described elsewhere.

Bit 4 **DCPD_CTL1**: DC to DC PWUP Control
0: PWM0H always high and build-in M4 PMOS always off
1: PWM0H signal from PWMH

Bit 3 **DCPD_CTL0**: DC to DC PWDN Control
0: PWM0L always low
1: PWM0L signal from PWML

Bit 2~1 **BZ_S1~BZ_S0**: Control signal type selection for Buzzer_EN
Described elsewhere.

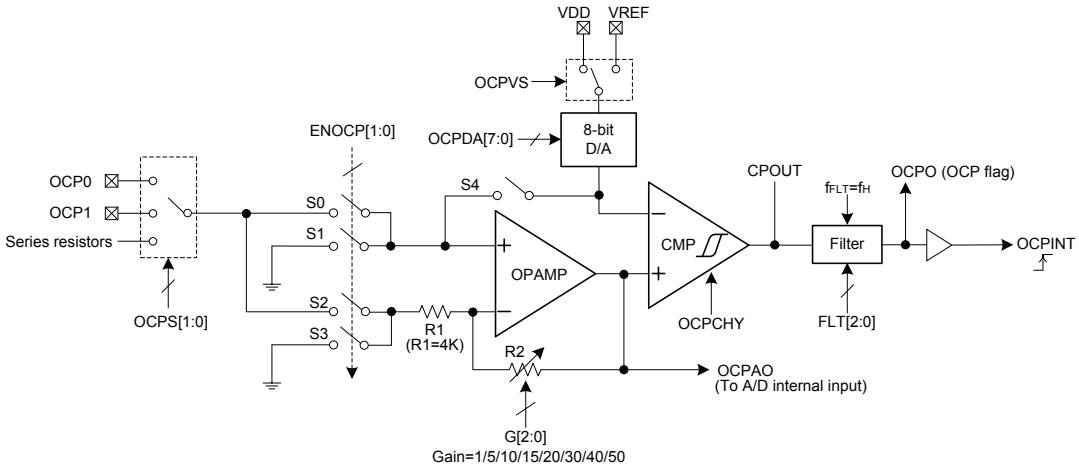
Bit 0 **BZ_CTL**: Buzzer Control
Described elsewhere.

Over Current Protection

The device includes an over current protection function which provides a protection mechanism for applications. To prevent the battery charge or load current from exceeding a specific level, the current on the OCP0 and OCP1 pins and from the series resistors are converted to a relevant voltage level according to the current value using the OCP operational amplifier. It is then compared with a reference voltage generated by an 8-bit D/A converter. When an over current event occurs, an OCP interrupt will be generated if the corresponding interrupt control is enabled.

Over Current Protection Operation

The OCP circuit is used to prevent the input current from exceeding a reference level. The current on the OCP0 or OCP1 pin or from the series resistors is converted to a voltage and then amplified by the OCP operational amplifier with a programmable gain from 1 to 50 selected by the G2~G0 bits in the OCPC1 register. This is known as the Programmable Gain Amplifier or PGA. This PGA can also be configured to operate in the non-inverting, inverting or input offset cancellation mode determined by the ENOCP1 and ENOCP0 bits in the OCPC0 register. After the current is converted and amplified to a specific voltage level, it will be compared with a reference voltage provided by an 8-bit D/A converter. The 8-bit D/A converter power can be supplied by the external power pin, VDD or VREF, selected by the OCPVS bit in the OCPC0 register. The comparator output, CPOUT, will first be filtered with a certain de-bounce time period selected by the FLT2~FLT0 bits in the OCPC1 register. Then a filtered OCP digital comparator output, OCPO, is obtained to indicate whether an over current condition occurs or not. The OCPO bit will be set to 1 if an over current condition occurs. Otherwise, the OCPO bit is zero. Once an over current event occurs, i.e., the converted voltage of the OCP input current is greater than the reference voltage, the corresponding interrupt will be generated if the relevant interrupt control bit is enabled.



Over Current Protection Block Diagram

Over Current Protection Control Registers

Overall operation of the over current protection is controlled using several registers. One register is used to provide the reference voltages for the over current protection circuit. There are two registers used to cancel out the operational amplifier and comparator input offset. The remaining two registers are control registers which control the OCP function, D/A converter reference voltage select, PGA gain select, comparator de-bounce time together with the hysteresis function. There are two control bits, OCPS1 and OCPS0, in the CTRL2 register used to configure the OCP input coming from OCP0 or OCP1 pin or from the series resistors. As for the OCP0 or OCP1 pin control, there are revelant pin-shared control bits to configure the OCP input pins. For a more detailed description regarding the input offset voltage cancellation procedures, refer to the corresponding input offset cancellation sections.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OCPC0	ENOCP1	ENOCP0	OCPVS	OCPCHY	—	—	—	OCPO
OCPC1	—	—	G2	G1	G0	FLT2	FLT1	FLT0
OCPDA	D7	D6	D5	D4	D3	D2	D1	D0
OCPOCAL	OOFM	ORSP	OOF5	OOF4	OOF3	OOF2	OOF1	OOF0
OCPCCAL	CPOUT	COFM	CRSP	COF4	COF3	COF2	COF1	COF0
CTRL2	SW_EN	—	LED_S1	LED_S0	LED_CTL	OCPS1	OCPS0	HV_CTL

OCP Register List

OCPC0 Register

Bit	7	6	5	4	3	2	1	0
Name	ENOCP1	ENOCP0	OCPVS	OCPCHY	—	—	—	OCPO
R/W	R/W	R/W	R/W	R/W	—	—	—	R
POR	0	0	0	0	—	—	—	0

Bit 7~6 **ENOCP1~ENOCP0**: OCP function operating mode selection
00: OCP function disabled, S1and S3 on, S0 and S2 off
01: OCP operates in non-inverter mode, S0 and S3 on, S1 and S2 off
10: OCP operates in inverter mode, S1 and S2 on, S0 and S3 off
11: OCP operates in calibration mode, S1 and S3 on, S0 and S2 off

Bit 5 **OCPVS**: OCP D/A Converter reference voltage selection
0: From VDD pin
1: From VREF pin

Bit 4 **OCPCHY**: OCP Comparator Hysteresis function control
0: Disable
1: Enable

Bit 3~1 Unimplemented, read as "0"

Bit 0 **OCPO**: OCP Comparator Filtered digital output flag
0: No Over Current condition occurs
1: Over Current condition occurs

OCPC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	G2	G1	G0	FLT2	FLT1	FLT0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **G2~G0**: PGA R2/R1 ratio selection
000: Unity gain buffer (non-inverting mode) or gain=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
These bits are used to select the R2/R1 ratio to obtain various gain values for inverting and non-inverting mode. The calculating formula of the PGA gain for the inverting and non-inverting mode is described in the "Input Voltage Range" section.

Bit 2~0 **FLT2~FLT0**: OCP output filter de-bounce time selection
000: No debounce
001: (1~2) × t_{FLT}
010: (3~4) × t_{FLT}
011: (7~8) × t_{FLT}
100: (15~16) × t_{FLT}
101: (31~32) × t_{FLT}
110: (63~64) × t_{FLT}
111: (127~128) × t_{FLT}
Note: $f_{FLT}=f_H$, $t_{FLT}=1/f_{FLT}$

OCPDA Register

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

Bit 7~0 OCP D/A Converter Data Register bit 7 ~ bit 0
OCP D/A Converter Output = (DAC reference voltage/256) × D[7:0]

OCPOCAL Register

Bit	7	6	5	4	3	2	1	0
Name	OOFM	ORSP	OOF5	OOF4	OOF3	OOF2	OOF1	OOF0
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 **OOFM**: OCP Operational Amplifier Input Offset Cancellation Mode Enable control
0: Input Offset Cancellation Mode Disabled
1: Input Offset Cancellation Mode Enabled
This bit is used to control the OCP operational amplifier input offset cancellation function. The ENOCP1 and ENOCP0 bits must first be set to "11" and then the OOFM bit must be set to 1 followed by the COFM bit being setting to 0, then the operational amplifier input offset cancellation mode will be enabled. Refer to the "Operational Amplifier Input Offset Cancellation" section for the detailed offset cancellation procedures.

Bit 6 **ORSP**: OCP Operational Amplifier Input Offset Cancellation Reference Input select
0: Operational amplifier negative input is selected
1: Operational amplifier positive input is selected

Bit 5~0 **OOF5~OOF0**: OCP Operational Amplifier Input Offset Cancellation value
This 6-bit field is used to perform the operational amplifier input offset cancellation operation and the value for the OCP operational amplifier input offset cancellation can be restored into this bit field. More detailed information is described in the "Operational Amplifier Input Offset Cancellation" section.

OCPCCAL Register

Bit	7	6	5	4	3	2	1	0
Name	CPOUT	COFM	CRSP	COF4	CF3	COF2	COF1	COF0
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 **CPOUT**: OCP Comparator or Operation Amplifier Digital Output in Input Offset Cancellation Mode
0: Positive input voltage < Negative input voltage
1: Positive input voltage > Negative input voltage
This bit is used to indicate whether the positive input voltage is greater than the negative input voltage when the OCP operates in the input offset cancellation mode. If the CPOUT is set to 1, the positive input voltage is greater than the negative input voltage. Otherwise, the positive input voltage is less than the negative input voltage.

Bit 6 **COFM**: OCP Comparator Input Offset Cancellation Mode Enable control
0: Input Offset Cancellation Mode Disabled
1: Input Offset Cancellation Mode Enabled
This bit is used to control the OCP comparator input offset cancellation function. The ENOCP1 and ENOCP0 bits must first be set to "11" and then the COFM bit must be set to 1 followed by the OOFM bit being setting to 0, then the comparator input offset cancellation mode will be enabled. Refer to the "Comparator Input Offset Cancellation" section for the detailed offset cancellation procedures.

Bit 5 **CRSP**: OCP Comparator Input Offset Cancellation Reference Input select
0: Comparator negative input is selected
1: Comparator positive input is selected

Bit 4~0 **COF4~COF0**: OCP Comparator Input Offset Cancellation value
This 5-bit field is used to perform the comparator input offset cancellation operation and the value for the OCP comparator input offset cancellation can be restored into this bit field. More detailed information is described in the "Comparator Input Offset Cancellation" section.

CTRL2 Register

Bit	7	6	5	4	3	2	1	0
Name	SW_EN	—	LED_S1	LED_S0	LED_CTL	OCPS1	OCPS0	HV_CTL
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	1

Bit 7 **SE_EN**: SW Control
Described elsewhere.

Bit 6 Unimplemented, read as "0"

Bit 5~4 **LED_S1~LED_S0**: Control signal type selection for LED_EN
Described elsewhere.

Bit 3 **LED_CTL**: LED Control
Described elsewhere.

Bit 2~1 **OCPS1~OCPS0**: OCP input selection
00: From OCP0 pin
01: From OCP1 pin
1x: From series resistors

Bit 0 **HV_CTL**: High Voltage Control
Described elsewhere.

Input Voltage Range

Together with different PGA operating modes, the input voltage on the OCP pin can be positive or negative to provide diverse applications for the device. The PGA output for the positive or negative input voltage is respectively calculated based on different formulas and described by the following.

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

$$\mathbf{V_{OUT} = (1 + \frac{R_2}{R_1}) \times V_{IN}}$$

- When the PGA operates in the non-inverting mode by setting the ENOCP1 and ENOCP0 bits to "01" with unity gain select by setting the G2~G0 to "000", the PGA will act as an unit-gain buffer whose output is equal to V_{IN} .

$$\mathbf{V_{OUT} = V_{IN}}$$

- For input voltage $0 > V_{IN} > -0.4$, the PGA operates in the inverting mode and the PGA output is obtained using the formula below. Note that if the input voltage is negative, it can not be lower than -0.4V which will result in current leakage.

$$\mathbf{V_{OUT} = -\frac{R_2}{R_1} \times V_{IN}}$$

Offset Calibration

To operate in the input offset cancellation mode for the OCP circuit, the ENOCPC1 and ENOCPC0 bits should first be set to "11". For operational amplifier and comparator input offset cancellation, the procedures are similar except for setting the respective control bits.

Operational Amplifier Input Offset Cancellation

- Step 1: Set ENOCP [1:0] = 11, OOFM=1 and COFM=0, the OCP will operate in the operational amplifier input offset cancellation mode.
- Step 2: Set OOF [5:0] = 000000 and read the CPOUT bit.
- Step 3: Increase the OOF [5:0] value by 1 and then read the CPOUT bit.
  - If the CPOUT bit state has not changed, then repeat Step 3 until the CPOUT bit state has changed.
  - If the CPOUT bit state has changed, record the OOF value as VOOS1 and then go to Step 4.
- Step 4: Set OOF [5:0] = 111111 and read the CPOUT bit.
- Step 5: Decrease the OOF [5:0] value by 1 and then read the CPOUT bit.
  - If the CPOUT bit state has not changed, then repeat Step 5 until the CPOUT bit state has changed.
  - If the CPOUT bit state has changed, record the OOF value as VOOS2 and then go to Step 6.
- Step 6: Restore the operational amplifier input offset cancellation value VOOS into the OOF [5:0] bit field. The offset cancellation procedure is now finished.

$$\text{Where } \mathbf{VOOS} = \frac{\mathbf{VOOS1} + \mathbf{VOOS2}}{\mathbf{2}}$$

Comparator input Offset Cancellation

- Step 1: Set ENOCP [1:0] = 11, COFM=1 and OOFM=0, the OCP will now operate in the comparator input offset cancellation mode. S4 is on (S4 is used for calibration mode, in normal mode operation, it is off).
- Step 2: Set COF [4:0] = 00000 and read the CPOUT bit.
- Step 3: Increase the COF [4:0] value by 1 and then read the CPOUT bit.
  - If the CPOUT bit state has not changed, then repeat Step 3 until the CPOUT bit state has changed.
  - If the CPOUT bit state has changed, record the COF value as VCOS1 and then go to Step 4.
- Step 4: Set COF [4:0] = 11111 and read the CPOUT bit.
- Step 5: Decrease the COF [4:0] value by 1 and then read the CPOUT bit.
  - If the CPOUT bit state has not changed, then repeat Step 5 until the CPOUT bit state has changed.
  - If the CPOUT bit state has changed, record the COF value as VCOS2 and then go to Step 6.
- Step 6: Restore the comparator input offset cancellation value VCOS into the COF [4:0] bit field. The offset cancellation procedure is now finished.

$$\text{Where } \mathbf{VCOS} = \frac{\mathbf{VCOS1} + \mathbf{VCOS2}}{\mathbf{2}}$$

Emergency Light Application Description

In the emergency light products, a MCU determines to buck charge or boost charge to provide the required emergency lighting power according to the conditions of the mains supply and the chargeable battery. This device has includes a range of functions related to emergency lights. Using an internal power MOS, the device can easily implement the above functions while meeting the associated Chinese national standards. The related operations are described as below.

Charge under Normal Mains Supply

An emergency light is usually powered by the mains supply with AC power being converted to DC power. When the voltage is within 12V, it can be directly connected to the HV_IN pin to provide power for the MCU and other circuits. In this case, for a 1.2V chargeable battery, buck charge can be implemented by turning on the M0 and M4 (controlled by PWM outputs) as well as the BAT_IN pin externally connected with an inductor, a schottky diode and the battery. The A/D converter can be used for charging current control. For a better buck charge result, connect an external NMOS to the PA7 pin for synchronous rectification.

Analog Battery Boost Charge under Normal Mains Supply

Turn off the M0, use the M4 and PA7 pin for complementary PWM control, and then externally connect an NMOS and an inductor to boost the battery voltage or current to a high level required by the LED lighting. After the high voltage has been generated, it can be read by the internal resistor divider which is enabled using the SW_EN bit to implement constant voltage feedback control. For a better boost charge result, connect a schottky diode in parallel between the BAT_IN pin and the HV_OUT pin. Refer to the application circuit section for more details.

Buzzer Driving

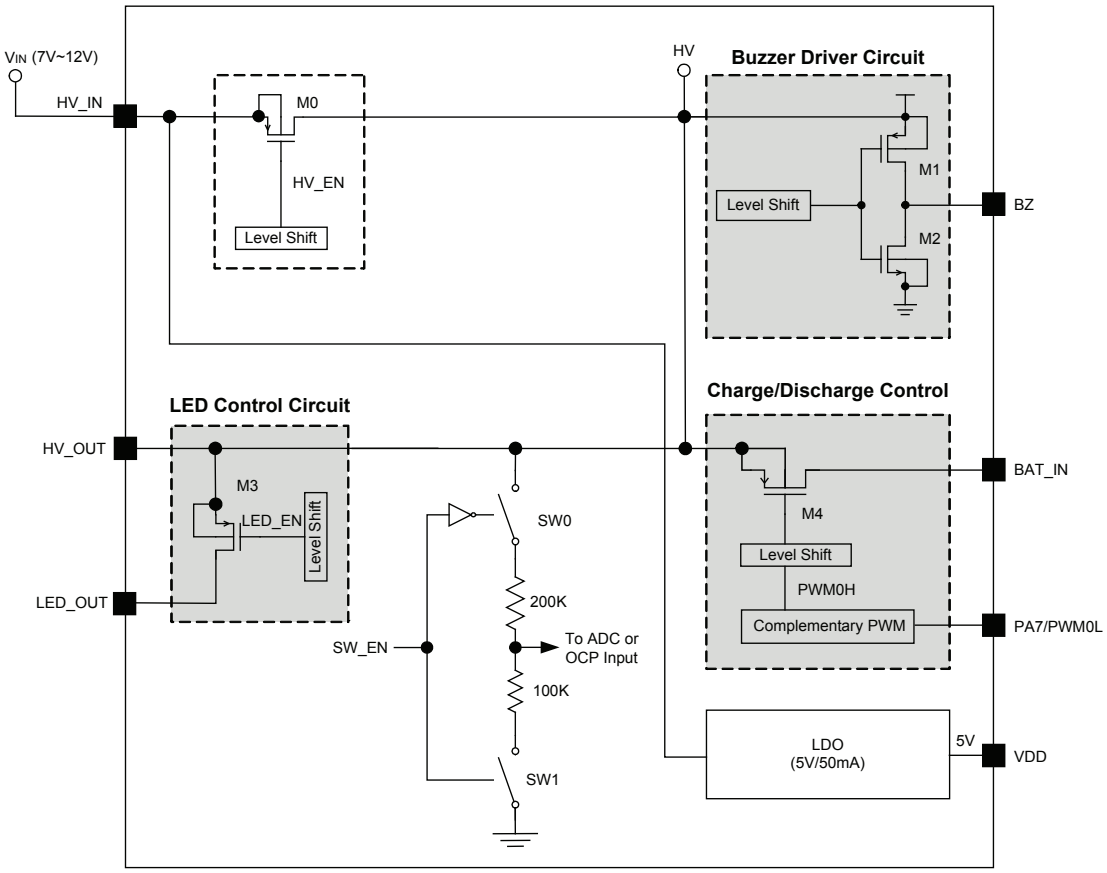
The M1 and M2 together form the buzzer dedicated output pin, it is controlled by the BZ_S0 and BZ_S1 bits to output a PWM signal or a constant high/low level.

LED Driving

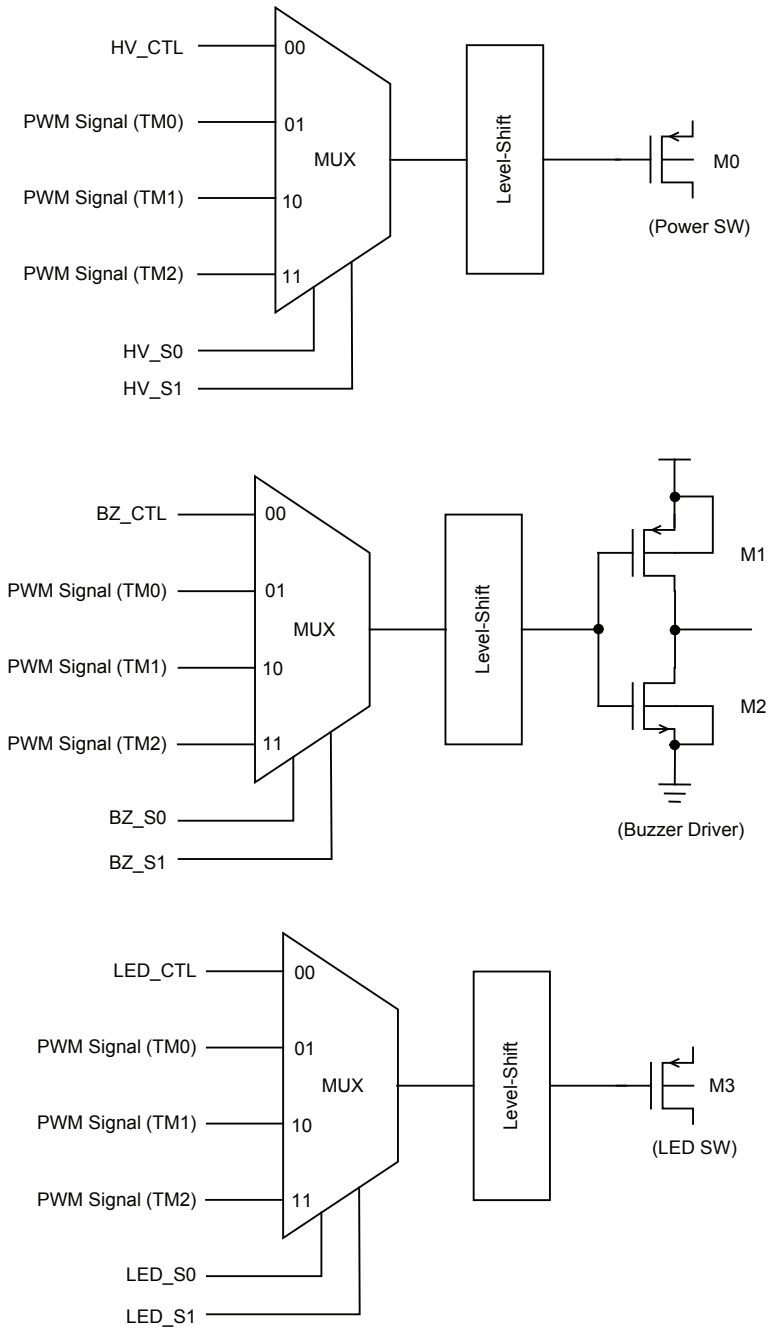
The LED_OUT pin is a LED driving output pin. The internal high voltage power is transmitted to this pin by the M3 MOS to driving the LED. Whether to use a PWM signal for LED dimming and constant current control or to enable/disable the LDO_OUT pin output is determined by the LED_S0 and LED_S1 bits.

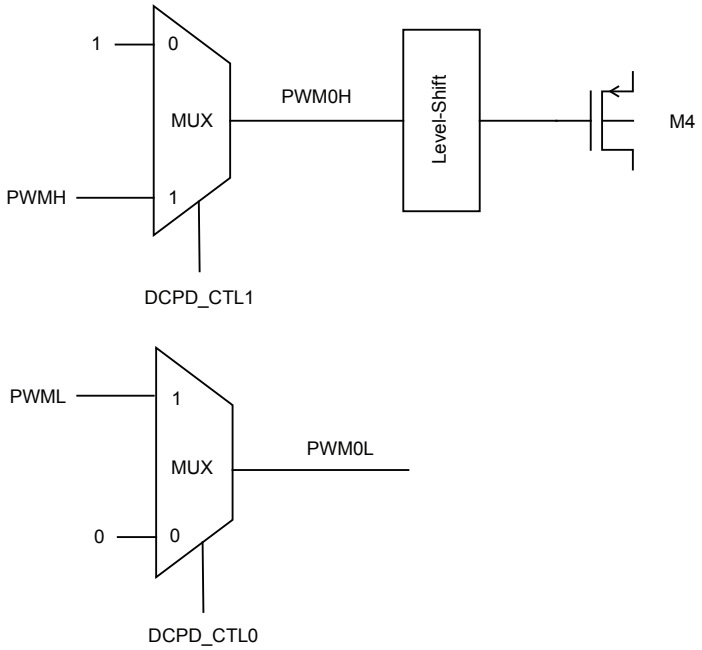
High Voltage MOS

This device integrates several high voltage MOS transistors with level shift functions, which along with the LDO regulator are used for Power control, LED control, buzzer driver control and charge/discharge control.



The driving signals for each MOS control are furtherly described in the following control circuits.





Control Registers

These two regiters are control regiters which select the driving signal for level shift function in each control circuit shown in the above block diagram.

CTRL2 Register

Bit	7	6	5	4	3	2	1	0
Name	SW_EN	—	LED_S1	LED_S0	LED_CTL	OCPS1	OCPS0	HV_CTL
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	1

Bit 7 **SE_EN**: SW Control
0: Disable
1: Enable

Bit 6 Unimplemented, read as "0"

Bit 5~4 **LED_S1~LED_S0**: Control signal type selection for LED enable
00: Normal signal (High/Low)
01: PWM signal from TM0
10: PWM signal from TM1
11: PWM signal from TM2
When these bits are set to 00, the control signal is determined by the LED_CTL bit.

Bit 3 **LED_CTL**: LED Control
0: Off
1: On

Bit 2~1 **OCPS1~OCPS0**: OCP input selection
Described elsewhere.

Bit 0 **HV_CTL**: High Voltage Control
0: Off
1: On

CTRL5 Register

Bit	7	6	5	4	3	2	1	0
Name	—	HV_S1	HV_S0	DCPD_CTL1	DCPD_CTL0	BZ_S1	BZ_S0	BZ_CTL
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 6~5 **HV_S1~HV_S0**: Control signal type selection for HV enable
00: Normal signal (High/Low)
01: PWM signal from TM0
10: PWM signal from TM1
11: PWM signal from TM2
When these bits are set to 00, the control signal is determined by the HV_CTL bit.

Bit 4 **DCPD_CTL1**: DC to DC PWUP Control
0: PWM0H always high
1: PWM0H signal from PWMH
When this bit is cleared to zero, PWM0H is always high level, M4 is turned off. When this bit is set high, M4 is controlled by PWM0H whose signal is from PWMH, it means that M4 is only controlled by TM0, it can not use TM1 and TM2. Refer to the Complementary PWM Output section for more details.

Bit 3 **DCPD_CTL0**: DC to DC PWDN Control
0: PWM0L always low
1: PWM0L signal from PWML

Bit 2~1 **BZ_S1~BZ_S0**: Control signal type selection for Buzzer enable
00: Normal signal (High/Low)
01: PWM signal from TM0
10: PWM signal from TM1
11: PWM signal from TM2
When these bits are set to 00, the control signal is determined by the BZ_CTL bit.

Bit 0 **BZ_CTL**: Buzzer Control
0: Off
1: On

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions. The external interrupt is generated by the action of the external INT0 and INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, Over Current Protection function, Time Base, LVD, EEPROM and the A/D converter.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The interrupt registers fall into three categories. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0~MFI3 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pin	INTnE	INTnF	n=0 or 1
OCP	OCPE	OCPF	—
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~3
Time Base	TBnE	TBnF	n=0 or 1
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
TM	TnPE	TnPF	n=0~2
	TnAE	TnAF	

Interrupt Register Bit Naming Conventions

Interrupt Register Contents

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	INT0F	OCPF	—	INT0E	OCPE	—	EMI
INTC1	MF3F	MF2F	MF1F	MF0F	MF3E	MF2E	MF1E	MF0E
INTC2	INT1F	TB1F	TB0F	ADF	INT1E	TB1E	TB0E	ADE
MFI0	—	—	T0AF	T0PF	—	—	T0AE	T0PE
MFI1	—	—	T1AF	T1PF	—	—	T1AE	T1PE
MFI2	—	—	T2AF	T2PF	—	—	T2AE	T2PE
MFI3	—	—	DEF	LVF	—	—	DEE	LVE

INTEG Register

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

Bit 7 ~ 4 Unimplemented, read as "0"

Bit 3 ~ 2 **INT1S1, INT1S0**: Defines INT1 interrupt active edge
00: Disabled Interrupt
01: Rising Edge Interrupt
10: Falling Edge Interrupt
11: Dual Edge Interrupt

Bit 1 ~ 0 **INT0S1, INT0S0**: Defines INT0 interrupt active edge
00: Disabled Interrupt
01: Rising Edge Interrupt
10: Falling Edge Interrupt
11: Dual Edge Interrupt

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	INT0F	OCPF	—	INT0E	OCPE	—	EMI
R/W	—	R/W	R/W	—	R/W	R/W	—	R/W
POR	—	0	0	—	0	0	—	0

Bit 7 Unimplemented, read as "0"

Bit 6 **INT0F**: External interrupt 0 request flag
0: No request
1: Interrupt request

Bit 5 **OCPF**: Over current protection interrupt request flag
0: No request
1: Interrupt request

Bit 4 Unimplemented, read as "0"

Bit 3 **INT0E**: External interrupt 0 control
0: Disable
1: Enable

Bit 2 **OCPE**: Over current protection interrupt control
0: Disable
1: Enable

Bit 1 Unimplemented, read as "0"

Bit 0 **EMI**: Global Interrupt Control
0: Disable
1: Enable

INTC1 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

INTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	INT1F	TB1F	TB0F	ADF	INT1E	TB1E	TB0E	ADE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **INT1F**: External interrupt 1 request flag
0: No request
1: Interrupt request

Bit 6 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **ADF**: A/D converter interrupt request flag
0: No request
1: Interrupt request

Bit 3 **INT1E**: External interrupt 1 control
0: Disable
1: Enable

Bit 2 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

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

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

MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	T0AF	T0PF	—	—	T0AE	T0PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **T0AF**: TM0 comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T0PF**: TM0 comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **T0AE**: TM0 comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **T0PE**: TM0 comparator P match interrupt control
0: Disable
1: Enable

MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	T1AF	T1PF	—	—	T1AE	T1PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 **T1AF**: TM1 comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T1PF**: TM1 comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 ~ 2 Unimplemented, read as "0"

Bit 1 **T1AE**: TM1 comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **T1PE**: TM1 comparator P match interrupt control
0: Disable
1: Enable

MFI2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	T2AF	T2PF	—	—	T2AE	T2PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 **T2AF**: TM2 comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T2PF**: TM2 comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 ~ 2 Unimplemented, read as "0"

Bit 1 **T2AE**: TM2 comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **T2PE**: TM2 comparator P match interrupt control
0: Disable
1: Enable

MFI3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	DEF	LVF	—	—	DEE	LVE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 **DEF**: Data EEPROM interrupt request flag
0: No request
1: Interrupt request

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

Bit 3 ~ 2 Unimplemented, read as "0"

Bit 1 **DEE**: Data EEPROM interrupt control
0: Disable
1: Enable

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

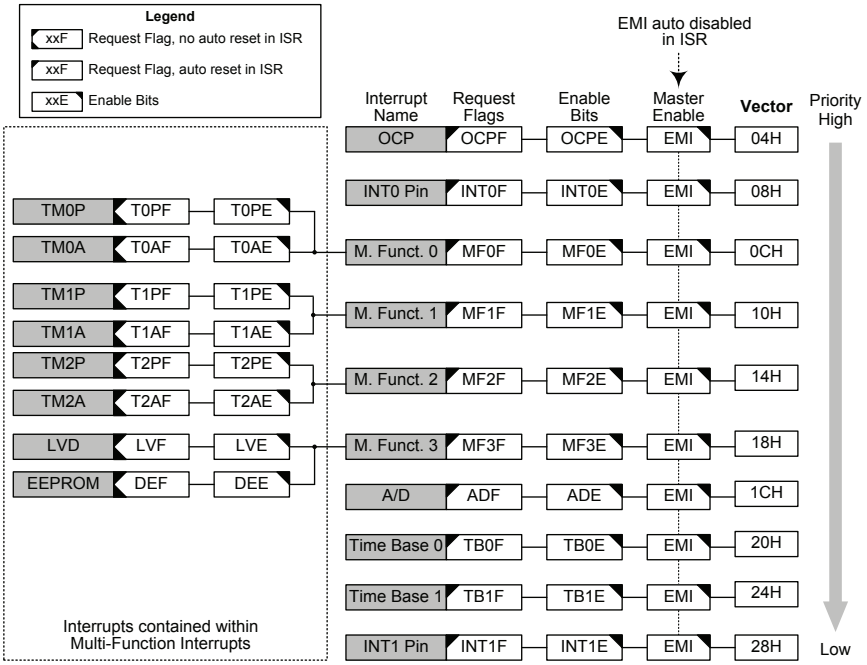
Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A match or A/D conversion completion etc, the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a "JMP" which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a "RETI", which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

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 pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, 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, INT0E~INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port control register as well as the relevant pin-shared function selection bits. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

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

OCP Interrupt

An OCP interrupt request will take place when the Over Current Protection Interrupt request flag, OCPF, is set, which occurs when the Over Current Protection function detects an over current condition. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Over Current Protection Interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the OCP Interrupt vector, will take place. When the Over Current Protection Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the interrupt request flag will be also automatically cleared.

Multi-function Interrupt

Within the device there are four 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 Interrupts, LVD Interrupt and EEPROM Interrupt. A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags, MFnF are set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. 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 Interrupts, LVD Interrupt and EEPROM Interrupt will not be automatically reset and must be manually reset by the application program.

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.

Time Base 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 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 interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Their clock sources originate from the internal clock source f_{TB}. This f_{TB} input 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. The clock source that generates f_{TB}, which in turn controls the Time Base interrupt period, can originate from several different sources, as shown in the System Operating Mode section.

TBC Register

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

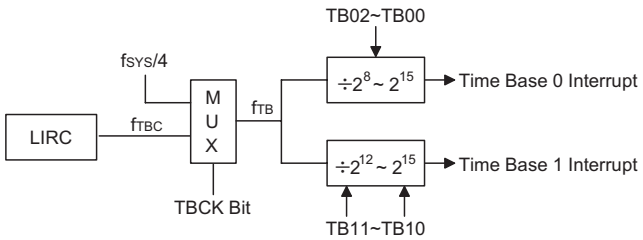
Bit 7 **TBON**: TB0 and TB1 Control bit
0: Disable
1: Enable

Bit 6 **TBCK**: Select f_{TB} Clock
0: f_{TBC}
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 Unimplemented, read as "0"

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: $2^{15}/f_{TB}$



Time Base Interrupt

EEPROM Interrupt

The EEPROM interrupt is contained within the Multi-function Interrupt. An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit, MF3E, 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 respective EEPROM Interrupt vector, will take place. When the EEPROM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

LVD Interrupt

The LVD interrupt is contained within the Multi-function Interrupt. An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Low Voltage Interrupt enable bit, LVE, and associated Multi-function interrupt enable bit, MF3E, 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, however only the Multi-function interrupt request flag will be also automatically cleared. As the LVD interrupt request flag, LVF, will not be automatically cleared, it has to be cleared by the application program.

TM Interrupts

The Periodic Type TMs each has two interrupts. All of the TM interrupts are contained within the Multi-function Interrupts. For the Periodic Type TMs there are two interrupt request flags TnPF and TnAF and two enable bits TnPE and TnAE. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or comparator A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the respective TM Interrupt enable bit, and associated Multi-function interrupt enable bit, MFnF, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant TM Interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins or a low power supply voltage may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program.

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MF0F~MF3F, 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 enables the device to monitor the power supply voltage, V_{DD}, and provides a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

LVD Register

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

LVDC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	—	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	—	R/W	R/W	R/W
POR	—	—	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

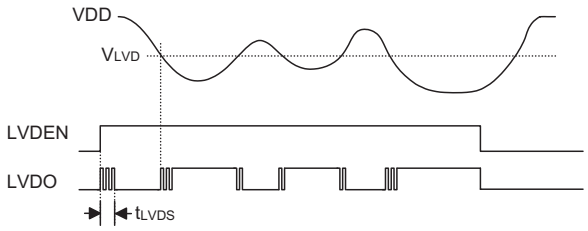
Bit 4 **LVDEN**: Low Voltage Detector Control
0: Disable
1: Enable

Bit 3 Unimplemented, read as "0"

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. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

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

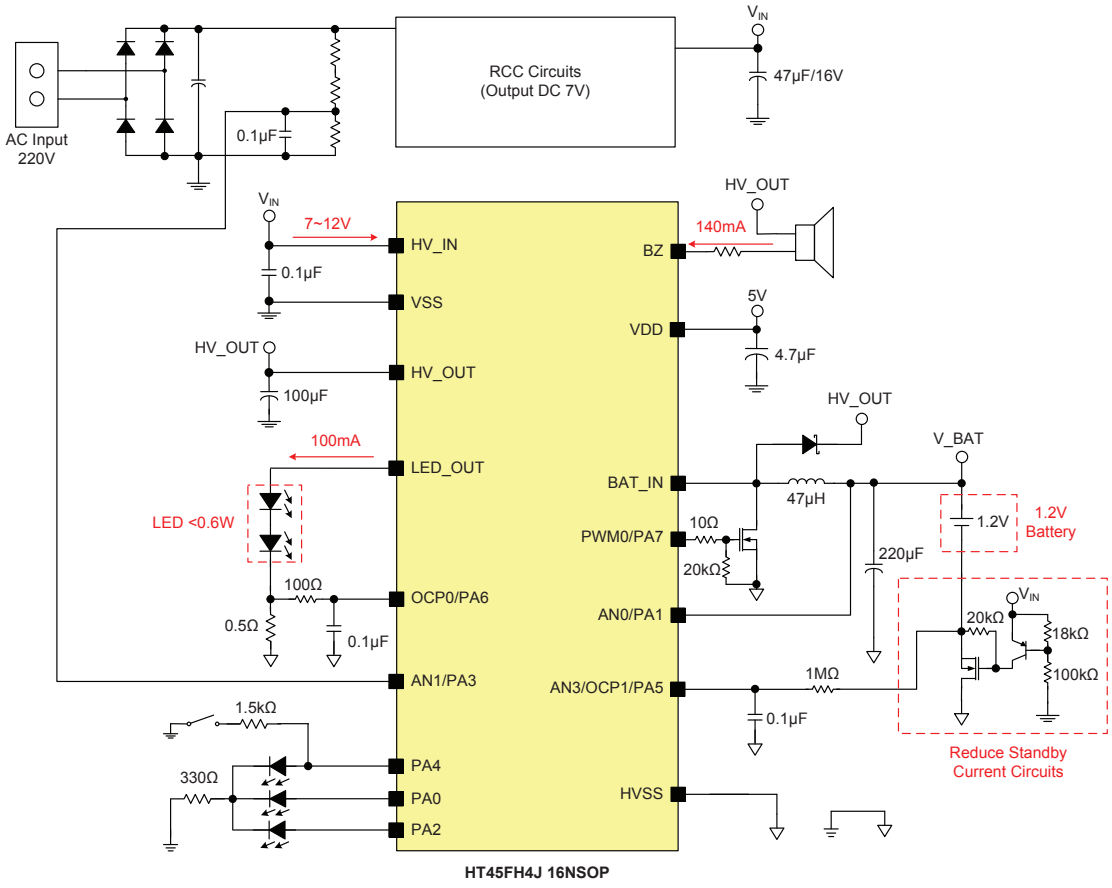
Configuration Option

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.

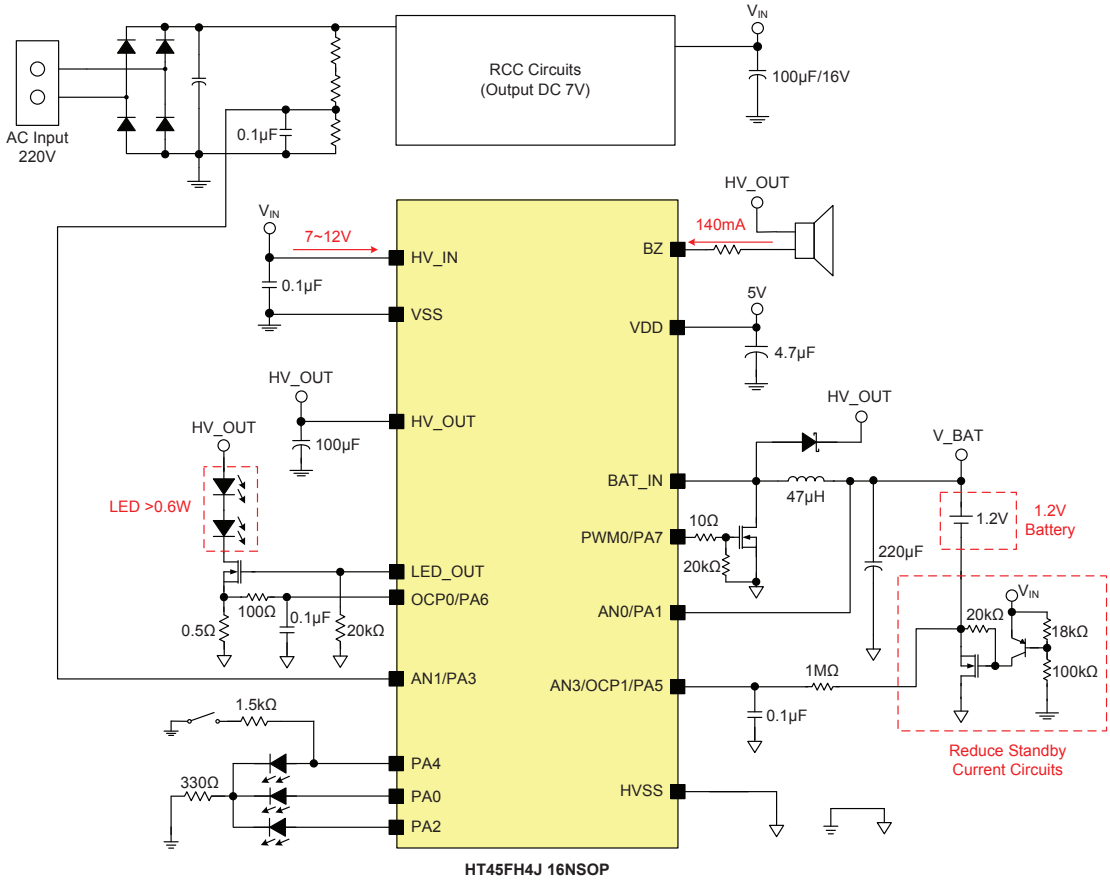
No.	Options
Oscillator Option	
1	HIRC Frequency Selection: 1. 20MHz 2. 16MHz 3. 12MHz

Application Circuit

Emergency Light Application Circuit (LED under 0.6W)



Emergency Light Application Circuit (LED over 1W)



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			
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			
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 and the carry flag are 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

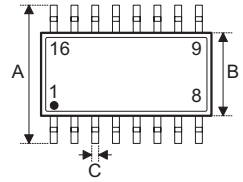
Package Information

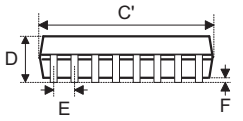
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

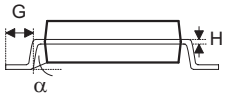
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

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

16-pin NSOP (150mil) Outline Dimensions



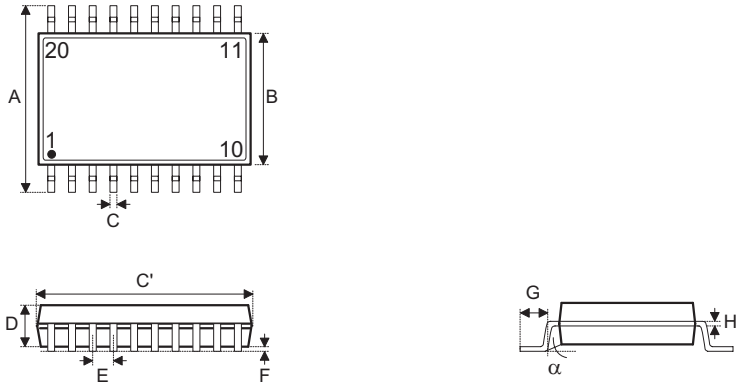




Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C’	—	0.390 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6 BSC	—
B	—	3.9 BSC	—
C	0.31	—	0.51
C’	—	9.9 BSC	—
D	—	—	1.75
E	—	1.27 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

20-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.155 BSC	—
C	0.008	—	0.012
C’	—	0.341 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.0098
G	0.016	—	0.05
H	0.004	—	0.01
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6 BSC	—
B	—	3.9 BSC	—
C	0.20	—	0.30
C‘	—	8.66 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°