



**DC Motor Flash MCU with H-Bridge Driver**

**HT45F4833**

Revision: V1.00    Date: October 01, 2018

[www.holtek.com](http://www.holtek.com)

## Table of Contents

<b>Features .....</b>	<b>6</b>
CPU Features .....	6
Peripheral Features.....	6
<b>General Description.....</b>	<b>7</b>
<b>Block Diagram.....</b>	<b>8</b>
<b>Pin Assignment.....</b>	<b>8</b>
<b>Pin Description .....</b>	<b>9</b>
<b>Absolute Maximum Ratings.....</b>	<b>11</b>
<b>D.C. Characteristics.....</b>	<b>12</b>
<b>A.C. Characteristics.....</b>	<b>13</b>
<b>A/D Converter Electrical Characteristics .....</b>	<b>14</b>
<b>Internal Reference Voltage Electrical Characteristics.....</b>	<b>14</b>
<b>LVR/LVD Electrical Characteristics .....</b>	<b>15</b>
<b>H-Bridge Driver Electrical Characteristics .....</b>	<b>15</b>
<b>LDO Electrical Characteristics .....</b>	<b>16</b>
<b>Level Shifter Electrical Characteristics .....</b>	<b>17</b>
<b>Power-on Reset Characteristics.....</b>	<b>17</b>
<b>System Architecture .....</b>	<b>17</b>
Clocking and Pipelining.....	17
Program Counter.....	18
Stack .....	19
Arithmetic and Logic Unit – ALU .....	19
<b>Flash Program Memory.....</b>	<b>20</b>
Structure.....	20
Special Vectors .....	20
Look-up Table.....	20
Table Program Example.....	21
In Circuit Programming – ICP .....	22
On-Chip Debug Support – OCDS .....	23
<b>RAM Data Memory .....</b>	<b>23</b>
Structure.....	23
General Purpose Data Memory .....	24
Special Purpose Data Memory .....	24
<b>Special Function Register Description.....</b>	<b>26</b>
Indirect Addressing Registers – IAR0, IAR1 .....	26
Memory Pointers – MP0, MP1 .....	26
Bank Pointer – BP.....	27
Accumulator – ACC.....	27

Program Counter Low Register – PCL.....	27
Look-up Table Registers – TBLP, TBLH.....	27
Status Register – STATUS.....	28
<b>EEPROM Data Memory.....</b>	<b>30</b>
EEPROM Data Memory Structure .....	30
EEPROM Registers .....	30
Reading Data from the EEPROM .....	32
Writing Data to the EEPROM.....	32
Write Protection.....	32
EEPROM Interrupt .....	32
Programming Considerations.....	33
<b>Oscillators .....</b>	<b>34</b>
Oscillator Overview .....	34
System Clock Configurations.....	34
High Speed Internal RC Oscillator – HIRC .....	35
Internal 32kHz Oscillator – LIRC.....	35
<b>Operating Modes and System Clocks .....</b>	<b>35</b>
System Clocks .....	35
System Operation Modes.....	36
Control Register .....	37
Operating Mode Switching .....	39
Standby Current Considerations .....	43
Wake-up .....	43
<b>Watchdog Timer.....</b>	<b>44</b>
Watchdog Timer Clock Source.....	44
Watchdog Timer Control Register .....	44
Watchdog Timer Operation .....	45
<b>Reset and Initialisation.....</b>	<b>46</b>
Reset Functions .....	46
Reset Initial Conditions .....	48
<b>Input/Output Ports .....</b>	<b>50</b>
Pull-high Resistors .....	51
Port A Wake-up .....	51
I/O Port Control Registers.....	51
Pin-shared Functions .....	52
I/O Pin Structures.....	56
Programming Considerations .....	56
<b>Timer Module – TM .....</b>	<b>57</b>
Introduction .....	57
TM Operation .....	57
TM Clock Source.....	57
TM Interrupts.....	58

TM External Pins.....	58
Programming Considerations.....	59
<b>Periodic Type TM – PTM.....</b>	<b>60</b>
Periodic TM Operation .....	60
Periodic Type TM Register Description .....	60
Periodic Type TM Operating Modes .....	67
<b>Analog to Digital Converter .....</b>	<b>79</b>
A/D Converter Overview .....	79
A/D Converter Register Description .....	80
A/D Converter Operation.....	83
A/D Converter Reference Voltage.....	84
A/D Converter Input Signals.....	84
Conversion Rate and Timing Diagram .....	85
Summary of A/D Conversion Steps.....	85
Programming Considerations.....	86
A/D Conversion Function .....	87
A/D Conversion Programming Examples.....	87
<b>LDO Divider Circuit.....</b>	<b>89</b>
<b>H-Bridge Driver .....</b>	<b>89</b>
Complementary Output Control .....	90
High Voltage Output Driver .....	94
High Voltage Output Read Back Circuit .....	96
<b>Low Voltage Detector – LVD .....</b>	<b>97</b>
LVD Register .....	97
LVD Operation.....	98
<b>Interrupts .....</b>	<b>99</b>
Interrupt Registers.....	99
Interrupt Operation .....	103
Thermal Protection Interrupt .....	104
LVD Interrupt.....	104
External Interrupts.....	104
Time Base Interrupts .....	105
EEPROM Write Interrupt.....	106
Multi-function Interrupts.....	106
A/D Converter Interrupt .....	106
Timer Module Interrupts .....	107
Interrupt Wake-up Function.....	107
Programming Considerations.....	107

<b>Application Description .....</b>	<b>108</b>
Introduction .....	108
Functional Description.....	108
Hardware Block Diagram .....	109
Hardware Circuit Diagram.....	110
<b>Instruction Set.....</b>	<b>111</b>
Introduction .....	111
Instruction Timing .....	111
Moving and Transferring Data .....	111
Arithmetic Operations.....	111
Logical and Rotate Operation .....	112
Branches and Control Transfer .....	112
Bit Operations .....	112
Table Read Operations .....	112
Other Operations.....	112
<b>Instruction Set Summary .....</b>	<b>113</b>
Table Conventions.....	113
<b>Instruction Definition.....</b>	<b>115</b>
<b>Package Information .....</b>	<b>124</b>
20-pin NSOP (150mil) Outline Dimensions .....	125

## Features

### CPU Features

- Operating Voltage
  - ♦  $V_{DD}=2.2V\sim5.5V$
  - ♦  $V_{CC1}=6V\sim12V$
- Up to 0.5 $\mu$ s instruction cycle with 8MHz system clock at  $V_{DD}=5V$
- Power down and wake-up functions to reduce power consumption
- Oscillator type
  - ♦ Internal 8MHz High Speed Oscillator – HIRC
  - ♦ Internal 32kHz Low Speed Oscillator – LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- 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 $\times$ 16
- RAM Data Memory: 128 $\times$ 8
- True EEPROM Memory: 32 $\times$ 8
- Watchdog Timer function
- 13 bidirectional I/O lines
- 4 pin-shared external interrupts
- Multiple Timer Modules for time measure, compare match output, capture input, PWM output and single pulse output functions
- Dual Time-Base functions for generation of fixed time interrupt signals
- 6 external channel 12-bit resolution A/D converter
- H-Bridge Driver
  - ♦ Complementary output control
  - ♦ High voltage output driver with thermal protection
- Internal LDO output: 5V
- Low Voltage Reset function
- Low Voltage Detect function
- Flash program memory can be re-programmed up to 100,000 times
- Flash program memory data retention > 10 years
- True EEPROM data memory can be re-programmed up to 1,000,000 times
- True EEPROM data memory data retention > 10 years
- Package type: 20-pin NSOP

## **General Description**

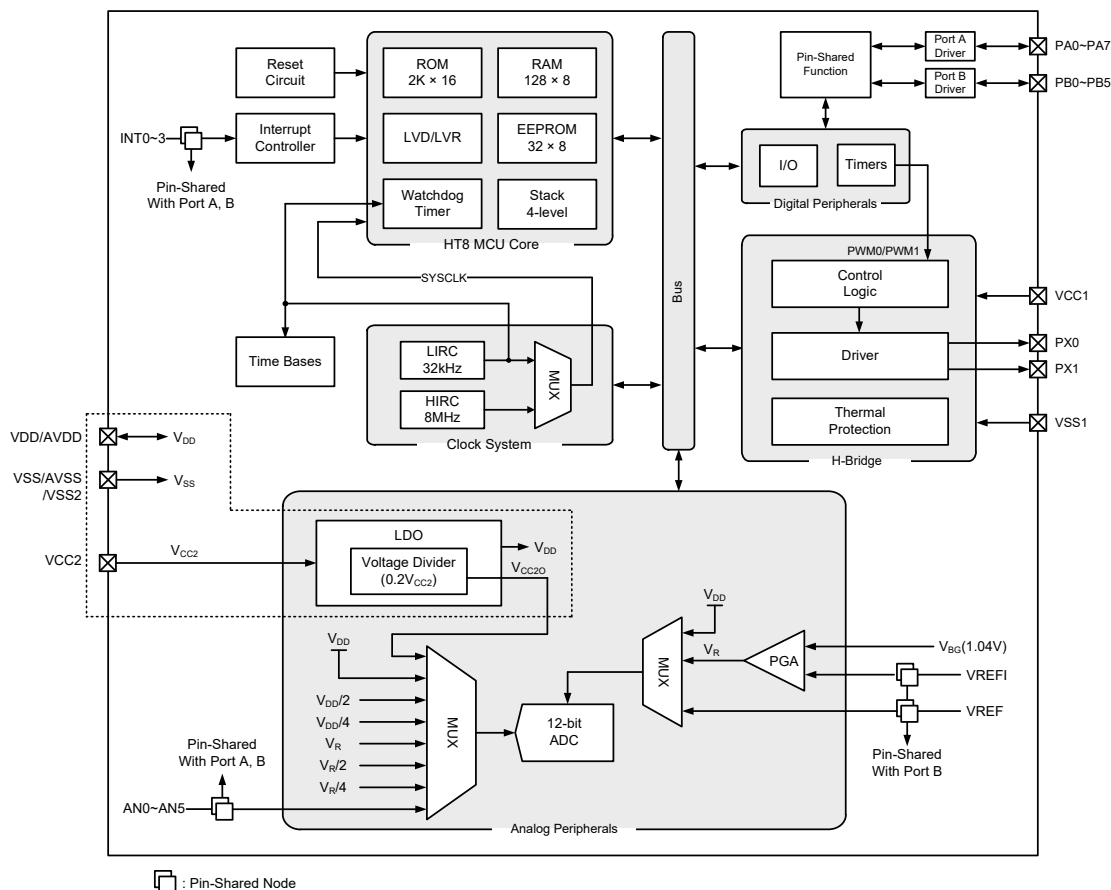
The HT45F4833 device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller, specifically designed for DC motor driver related applications. Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 12-bit A/D converter function and an internal LDO for power supply functions. Multiple extremely flexible Timer Modules provide timing, pulse generation, capture input, compare match output and PWM generation functions. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

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

The inclusion of complementary output control and high voltage output driver circuits, flexible I/O programming features, Time-Base functions along with many other features ensure that the device will find excellent use in DC motor driver related applications although the device will also lend itself for use in a range of other related applications.

## Block Diagram



## Pin Assignment

VCC2	1	20	PA5/INT1/PTP11/PTP1B/AN3
VDD/AVDD	2	19	PA4/INT2/PTCK0/PTP0/AN4
VSS/AVSS/VSS2	3	18	PA3/INT3/PTP01/PTP0B/AN5
PB0/INT0/PTCK1/PTP1/AN2	4	17	PA2/PTP0/OCDSCK/ICPCK
PB1/PTP01/AN1/VREF	5	16	PA0/PTCK1/PTP1/OCSDA/ICPDA
PB2/PTCK0/PTP1B/AN0/VREF1	6	15	PA1/PTP11/PTP0B/VCC20
PB3/INT3/PTP11/PTP1B	7	14	PA7/INT0/PTCK0/PTP0
PB4/INT2/PTCK1/PTP1	8	13	PA6/INT1/PTP01/PTP0B
VCC1	9	12	VSS1
PX1	10	11	PX0

**HT45F4833/HT45V4833**  
**20 NSOP-A**

- Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.
2. The OCSDA and OCDSCK pins are used as the OCDS dedicated pins and as such are only available for the HT45V4833 EV device.



## Pin Description

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

Pin Name	Function	OPT	I/T	O/T	Description
PA0/PTCK1/PTP1/OCDSDA/ICPDA	PA0	PAWU PAPU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTCK1	PAPS0 IFS0	ST	—	PTM1 clock input
	PTP1	PAPS0	—	CMOS	PTM1 output
	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
	ICPDA	—	ST	CMOS	ICP data/address pin
PA1/PTP11/PTP0B/VCC2O	PA1	PAWU PAPU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP11	PAPS0 IFS0	ST	—	PTM1 capture input
	PTP0B	PAPS0	—	CMOS	PTM0 inverted output
	VCC2O	PAPS0	—	AN	LDO internal resistor divider output
PA2/PTP0/OCDSCK/ICPCK	PA2	PAWU PAPU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP0	PAPS0	—	CMOS	PTM0 output
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
	ICPCK	—	ST	—	ICP clock pin
PA3/INT3/PTP0I/PTP0B/AN5	PA3	PAWU PAPU PAPS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT3	PAPS0 INTEG IFS1	ST	—	External interrupt 3 input
	PTP0I	PAPS0 IFS0	ST	—	PTM0 capture input
	PTP0B	PAPS0	—	CMOS	PTM0 inverted output
	AN5	PAPS0	AN	—	A/D Converter external input channel 5
PA4/INT2/PTCK0/PTP0/AN4	PA4	PAWU PAPU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT2	PAPS1 INTEG IFS1	ST	—	External interrupt 2 input
	PTCK0	PAPS1 IFS0	ST	—	PTM0 clock input
	PTP0	PAPS1	—	CMOS	PTM0 output
	AN4	PAPS1	AN	—	A/D Converter external input channel 4

Pin Name	Function	OPT	I/T	O/T	Description
PA5/INT1/PTP1I/PTP1B/AN3	PA5	PAWU PAPU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT1	PAPS1 INTEG IFS1	ST	—	External interrupt 1 input
	PTP1I	PAPS1 IFS0	ST	—	PTM1 capture input
	PTP1B	PAPS1	—	CMOS	PTM1 inverted output
	AN3	PAPS1	AN	—	A/D Converter external input channel 3
PA6/INT1/PTP0I/PTP0B	PA6	PAWU PAPU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT1	PAPS1 INTEG IFS1	ST	—	External interrupt 1 input
	PTP0I	PAPS1 IFS0	ST	—	PTM0 capture input
	PTP0B	PAPS1	—	CMOS	PTM0 inverted output
PA7/INT0/PTCK0/PTP0	PA7	PAWU PAPU PAPS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT0	PAPS1 INTEG IFS1	ST	—	External interrupt 0 input
	PTCK0	PAPS1 IFS0	ST	—	PTM0 clock input
	PTP0	PAPS1	—	CMOS	PTM0 output
PB0/INT0/PTCK1/PTP1/AN2	PB0	PBPU PBPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT0	PBPS0 INTEG IFS1	ST	—	External interrupt 0 input
	PTCK1	PBPS0 IFS0	ST	—	PTM1 clock input
	PTP1	PBPS0	—	CMOS	PTM1 output
	AN2	PBPS0	AN	—	A/D Converter external input channel 2
PB1/PTP0I/AN1/VREF	PB1	PBPU PBPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP0I	PBPS0 IFS0	ST	—	PTM0 capture input
	AN1	PBPS0	AN	—	A/D Converter external input channel 1
	VREF	PBPS0	AN	—	A/D Converter external reference voltage input
PB2/PTCK0/PTP1B/AN0/VREFI	PB2	PBPU PBPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK0	PBPS0 IFS0	ST	—	PTM0 clock input
	PTP1B	PBPS0	—	CMOS	PTM1 inverted output
	AN0	PBPS0	AN	—	A/D Converter external input channel 0
	VREFI	PBPS0	AN	—	A/D Converter PGA input

Pin Name	Function	OPT	I/T	O/T	Description
PB3/INT3/PTP1I/PTP1B	PB3	PBPU PBPS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT3	PBPS0 INTEG IFS1	ST	—	External interrupt 3 input
	PTP1I	PBPS0 IFS0	ST	—	PTM1 capture input
	PTP1B	PBPS0	—	CMOS	PTM1 inverted output
PB4/INT2/PTCK1/PTP1	PB4	PBPU PBPS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT2	PBPS1 INTEG IFS1	ST	—	External interrupt 2 input
	PTCK1	PBPS1 IFS0	ST	—	PTM1 clock input
	PTP1	PBPS1	—	CMOS	PTM1 output
PX0	PX0	—	—	AN	High voltage output 0
PX1	PX1	—	—	AN	High voltage output 1
VCC1	VCC1	—	PWR	—	High voltage driver power supply
VSS1	VSS1	—	PWR	—	High voltage driver negative power supply
VCC2	VCC2	—	PWR	—	High voltage positive power for LDO input voltage
VDD/AVDD	VDD	—	PWR	—	Digital positive power supply
		—	—	PWR	LDO output voltage
	AVDD	—	PWR	—	Analog positive power supply
VSS/AVSS/VSS2	VSS	—	PWR	—	Digital negative power supply
	AVSS	—	PWR	—	Analog negative power supply
	VSS2	—	PWR	—	High voltage negative power supply

Legend: I/T: Input type

OP: Optional register option

ST: Schmitt Trigger input

AN: Analog Signal

O/T: Output type

PWR: Power

CMOS: CMOS output

## Absolute Maximum Ratings

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

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

## D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage (HIRC)	—	f <sub>sys</sub> =f <sub>HIRC</sub> =8MHz	2.2	—	5.5	V
	Operating Voltage (LIRC)	—	f <sub>sys</sub> =f <sub>LIRC</sub> =32kHz	2.2	—	5.5	V
I <sub>DD</sub>	Operating Current (LIRC)	3V	No load, f <sub>sys</sub> =f <sub>LIRC</sub> =32kHz, all peripherals off, WDT enable, LVR enable	—	20	30	μA
		5V		—	30	60	
	Operating Current (HIRC)	3V	No load, f <sub>sys</sub> =f <sub>HIRC</sub> /2, all peripherals off, WDT enable, LVR enable	—	1.0	1.5	mA
		5V		—	1.5	2.0	
		3V	No load, f <sub>sys</sub> =f <sub>HIRC</sub> /4, all peripherals off, WDT enable, LVR enable	—	0.9	1.3	mA
		5V		—	1.3	1.8	
		3V	No load, f <sub>sys</sub> =f <sub>HIRC</sub> /8, all peripherals off, WDT enable, LVR enable	—	0.8	1.1	mA
		5V		—	1.1	1.6	
		3V	No load, f <sub>sys</sub> =f <sub>HIRC</sub> /16, all peripherals off, WDT enable, LVR enable	—	0.7	1.0	mA
		5V		—	1.0	1.4	
		3V	No load, f <sub>sys</sub> =f <sub>HIRC</sub> /32, all peripherals off, WDT enable, LVR enable	—	0.6	0.9	mA
		5V		—	0.9	1.2	
		3V	No load, f <sub>sys</sub> =f <sub>HIRC</sub> /64, all peripherals off, WDT enable, LVR enable	—	0.5	0.8	mA
		5V		—	0.8	1.1	
I <sub>STB</sub>	Standby Current (SLEEP Mode)	3V	No load, all peripherals off, WDT disable, LVR disable	—	0.2	0.8	μA
		5V		—	0.5	1	
		3V	No load, all peripherals off, WDT enable, LVR disable	—	1.3	5.0	μA
		5V		—	2.2	10	
	Standby Current (IDLE0 Mode)	3V	No load, all peripherals off, WDT enable, LVR disable, f <sub>SUB</sub> on	—	1.3	3.0	μA
		5V		—	5.0	10	
	Standby Current (IDLE1 Mode)	3V	No load, all peripherals off, WDT enable, LVR disable, f <sub>SUB</sub> on, f <sub>sys</sub> =f <sub>HIRC</sub> =8MHz on	—	0.8	1.6	mA
		5V		—	1.0	2.0	
V <sub>IL</sub>	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—	—	0	—	0.2V <sub>DD</sub>	
V <sub>IH</sub>	Input High Voltage for I/O Ports	5V	—	3.5	—	5	V
		—	—	0.8V <sub>DD</sub>	—	V <sub>DD</sub>	
I <sub>OL</sub>	Sink Current for I/O Pins	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	16	32	—	mA
		5V		32	65	—	
I <sub>OH</sub>	Source Current for I/O Pins	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-4	-8	—	mA
		5V		-8	-16	—	
R <sub>PH</sub>	Pull-high Resistance for I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I <sub>LEAK</sub>	Input Leakage Current	5V	V <sub>IN</sub> =V <sub>DD</sub> or V <sub>IN</sub> =V <sub>SS</sub>	—	—	±1	μA

## A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Condition				
f <sub>SYS</sub>	System Clock (HIRC)	2.2V~5.5V	f <sub>SYS</sub> =f <sub>HIRC</sub> =8MHz	—	8	—	MHz
	System Clock (LIRC)	2.2V~5.5V	f <sub>SYS</sub> =f <sub>LIRC</sub> =32kHz	—	32	—	kHz
f <sub>HIRC</sub>	High Speed Internal RC Oscillator (HIRC Trim at V <sub>DD</sub> =5V)	5V	Ta=25°C	-1%	8	+1%	MHz
			Ta=-40°C~85°C	-2%	8	+2%	
		2.2V~5.5V	Ta=25°C	-2.5%	8	+2.5%	
			Ta=-40°C~85°C	-3%	8	+3%	
f <sub>LIRC</sub>	Low Speed Internal RC Oscillator (LIRC)	5V	Ta=25°C	25.6	32	38.4	kHz
		2.2V~5.5V	Ta=25°C	12.8	32	41.6	
			Ta=-40°C~85°C	8	32	60	
t <sub>START</sub>	LIRC Start-up Time	—	—	—	—	100	μs
t <sub>SST</sub>	System Start-up Time (Wake-up from condition where f <sub>SYS</sub> is off)	—	f <sub>SYS</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub>	—	16	—	t <sub>HIRC</sub>
		—	f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>	—	2	—	t <sub>LIRC</sub>
	System Start-up Time (Wake-up from condition where f <sub>SYS</sub> is on)	—	f <sub>SYS</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub>	—	2	—	t <sub>H</sub>
		—	f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>	—	2	—	t <sub>SUB</sub>
t <sub>RSTD</sub>	System Speed Switch Time (Normal to Slow Mode or Slow to Normal Mode)	—	f <sub>HIRC</sub> switches from off → on	—	16	—	t <sub>HIRC</sub>
	System Reset Delay Time (Power-on Reset, LVR Reset, WDTC Software Reset)	—	—	25	50	150	ms
t <sub>INT</sub>	System Reset Delay Time (WDT time-out Reset)	—	—	8.3	16.7	50	ms
	External Interrupt Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>TCK</sub>	PTCKn Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>TPI</sub>	PTPnI Input Pin Minimum Pulse Width	—	—	0.1	—	—	μs
f <sub>TMCLK</sub>	PTMn Maximum Timer Clock Source Frequency	5V	—	—	—	1	f <sub>SYS</sub>
t <sub>CPW</sub>	TM Minimum Capture Pulse Width	—	—	2	—	—	t <sub>TMCLK</sub>
t <sub>SRESET</sub>	Software Reset Width to Reset	—	—	40	90	375	μs
t <sub>DEW</sub>	Data EEPROM Write Time	—	—	—	4	15	ms

- Note: 1. For the System Start-up time values, whether f<sub>SYS</sub> is on or off depends upon the mode type and the chosen f<sub>SYS</sub> system oscillator. Details are provided in the System Operating Modes section.
2. The time units, shown by the symbols t<sub>HIRC</sub>, t<sub>SYS</sub> etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example t<sub>HIRC</sub>=1/f<sub>HIRC</sub>, t<sub>SYS</sub>=1/f<sub>SYS</sub> etc.
3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t<sub>START</sub>, must be added to the t<sub>SST</sub> time in the table above.
4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.
5. 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.

## A/D Converter Electrical Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	A/D Converter Operating Voltage	—	—	2.2	—	5.5	V
V <sub>ADI</sub>	A/D Converter Input Voltage	—	—	0	—	V <sub>REF</sub>	V
V <sub>REF</sub>	A/D Converter Reference Voltage	—	—	2	—	V <sub>DD</sub>	V
DNL	Differential Non-linearity	—	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs	-3	—	3	LSB
INL	Integral Non-linearity	—	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs	-4	—	4	LSB
I <sub>ADC</sub>	Additional Current for A/D Converter Enable	2.2V	No load, t <sub>ADCK</sub> =0.5μs	—	1.0	2.0	mA
		3V		—	1.0	2.0	mA
		5V		—	1.5	3.0	mA
t <sub>ADCK</sub>	A/D Clock Period	—	—	0.5	—	10	μs
t <sub>ON2ST</sub>	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t <sub>ADS</sub>	A/D Sampling Time	—	—	—	4	—	t <sub>ADCK</sub>
t <sub>ADC</sub>	A/D Conversion Time (Include A/D Sample and Hold Time)	—	—	—	16	—	t <sub>ADCK</sub>
GERR	A/D Conversion Gain Error	—	V <sub>REF</sub> =V <sub>DD</sub>	-4	—	4	LSB
OSRR	A/D Conversion Offset Error	—	V <sub>REF</sub> =V <sub>DD</sub>	-4	—	4	LSB
I <sub>PGA</sub>	Additional Current for PGA Enable	3V	No load	—	250	500	μA
		5V		—	250	500	μA
V <sub>OR</sub>	PGA Maximum Output Voltage Range	3V	—	V <sub>SS</sub> +0.1	—	V <sub>DD</sub> -0.1	V
		5V		V <sub>SS</sub> +0.1	—	V <sub>DD</sub> -0.1	V
Ga	PGA Gain Accuracy	—	Gain=1, V <sub>RI</sub> > 0.1V	-5	—	+5	%
			Gain=2, 3, 4, V <sub>RI</sub> > 0.05V				
V <sub>RI</sub>	PGA Input Voltage Range	3V	Gain=1, Ga <±5%	V <sub>SS</sub> +0.1	—	V <sub>DD</sub> -1.4	V
		5V		V <sub>SS</sub> +0.1	—	V <sub>DD</sub> -1.4	V
		3V	Gain=2, 3, 4, Ga <±5%	V <sub>SS</sub> +0.05	—	V <sub>OR(Max)</sub> /Gain	V
		5V		V <sub>SS</sub> +0.05	—	V <sub>OR(Max)</sub> /Gain	V

## Internal Reference Voltage Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>BG</sub>	Bandgap Reference Voltage	—	—	-5%	1.04	+5%	V
I <sub>BG</sub>	Additional Current for Bandgap Reference Enable	—	LVR disable, LVD disable	—	—	25	μA

Note: 1. All the above parameters are measured under conditions of no load condition unless otherwise described.  
 2. A 0.1μF ceramic capacitor should be connected between V<sub>DD</sub> and GND.  
 3. The V<sub>BG</sub> voltage is used as the A/D converter PGA input signal.

## LVR/LVD Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	—	2.2	—	5.5	V
V <sub>LVR</sub>	Low Voltage Reset Voltage	—	LVR enable	-5%	3.15	+5%	V
V <sub>LVD</sub>	Low Voltage Detection Voltage	—	LVD enable, voltage select 2.0V	-5%	2.0	+5%	V
			LVD enable, voltage select 2.2V		2.2		
			LVD enable, voltage select 2.4V		2.4		
			LVD enable, voltage select 2.7V		2.7		
			LVD enable, voltage select 3.0V		3.0		
			LVD enable, voltage select 3.3V		3.3		
			LVD enable, voltage select 3.6V		3.6		
			LVD enable, voltage select 4.0V		4.0		
I <sub>LVR/LVDBG</sub>	Operating Current	3V	LVD enable, LVR enable, VBGEN=0	—	—	20	μA
		5V	VBGEN=0	—	20	25	
		3V	LVD enable, LVR enable, VBGEN=1	—	—	25	μA
		5V	VBGEN=1	—	25	30	
t <sub>LVDS</sub>	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	15	μs
			For LVR disable, VBGEN=0, LVD off → on	—	—	150	
t <sub>LVR</sub>	Minimum Low Voltage Width to Reset	—	—	140	600	1000	μs
t <sub>LVD</sub>	Minimum Low Voltage Width to Interrupt	—	—	40	150	320	μs
I <sub>LVR</sub>	Additional Current for LVR Enable	—	LVD disable, VBGEN=0	—	—	24	μA
I <sub>LVD</sub>	Additional Current for LVD Enable	—	LVR disable, VBGEN=0	—	—	24	μA

## H-Bridge Driver Electrical Characteristics

V<sub>DD</sub>=2.2V~5.5V, V<sub>IN</sub>>V<sub>DD</sub>, Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>IN</sub>	Input Voltage	—	—	6	—	12	V
I <sub>OH</sub>	Source Current for High Voltage Driving Output	—	V <sub>IN</sub> =6V, V <sub>OH</sub> =0.9×V <sub>IN</sub> , HBCC[1:0]=11B	-600	-800	—	mA
I <sub>OL</sub>	Sink Current for High Voltage Driving Output	—	V <sub>IN</sub> =6V, V <sub>OL</sub> =0.1×V <sub>IN</sub> , HBCC[1:0]=11B	600	800	—	mA
V <sub>IH</sub>	Input High Voltage for High Voltage Driving Output	—	—	0.7V <sub>IN</sub>	—	V <sub>IN</sub>	V
V <sub>IL</sub>	Input Low Voltage for High Voltage Driving Output	—	—	0	—	0.3V <sub>IN</sub>	V
T <sub>PRT</sub>	Thermal Protection Temperature	—	—	77.5	155	232.5	°C

## LDO Electrical Characteristics

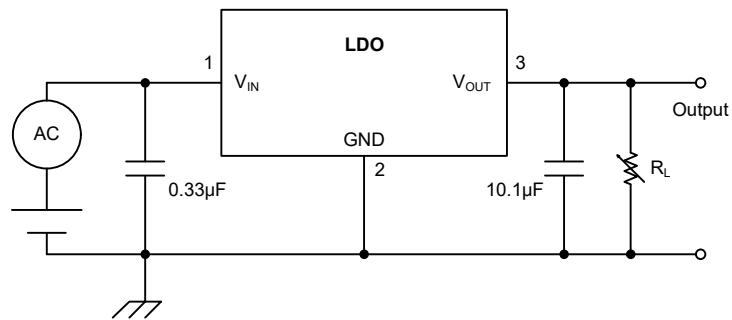
$V_{IN}=V_{OUT(Nominal)}+1V$ ,  $C_{LOAD}=10\mu F+0.1\mu F$ ,  $T_a=25^{\circ}C$

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		$V_{IN}$	Conditions				
$V_{IN}$	Input Voltage	—	—	$V_{OUT}+0.1$	6	12	V
$V_{OUT}$	Output Voltage	—	$T_a=25^{\circ}C$ , $I_{LOAD}=1mA$ , $V_{OUT}=5.0V$	-2%	5.0	+2%	V
			$T_a=-40^{\circ}C\sim 85^{\circ}C$ , $I_{LOAD}=1mA$ , $V_{OUT}=5.0V$	-5%	5.0	+5%	
$\Delta V_{LOAD}$	Load Regulation <sup>(1)</sup>	—	$1mA \leq I_{LOAD} \leq 70mA$ , $V_{IN}=V_{OUT}+1V$	—	0.015	0.033	%/mA
$V_{DROP}$	Dropout Voltage <sup>(2)</sup>	—	$\Delta V_{OUT}=2\%$ , $I_{LOAD}=1mA$	—	20	—	mV
			$\Delta V_{OUT}=2\%$ , $I_{LOAD}=10mA$	—	100	—	
			$V_{IN}=V_{OUT}+1.5V$ , $\Delta V_{OUT}=2\%$ , $I_{LOAD}=70mA$	—	600	1000	
$I_{OUT}$	Output Current	—	$V_{IN}=V_{OUT}+1V$ , $V_{OUT}=5V$ , $\Delta V_{OUT}=-3\%$	70	—	—	mA
			$V_{IN}=V_{OUT}+2V$ , $V_{OUT}=5V$ , $\Delta V_{OUT}=-3\%$	150	—	—	
$I_Q$	Quiescent Current	12V	No load	—	5	7	$\mu A$
$\Delta V_{LINE}$	Line Regulation	—	$V_{OUT}+1V \leq V_{IN} \leq 12V$ , $I_{LOAD}=1mA$	—	—	0.2	%/V
TC	Temperature Coefficient	—	$T_a=-40^{\circ}C\sim 85^{\circ}C$ , $I_{LOAD}=10mA$	—	$\pm 1.5$	$\pm 2$	mV/ $^{\circ}C$
$I_{SD}$	Shutdown Current	—	LDO disable	—	—	1	$\mu A$
$\Delta V_{OUT\_RIPPLE}$	Output Voltage Ripple	6V	$I_{LOAD}=10mA$	—	—	40	mV
RR	Ripple Rejection	—	$V_{IN}=10V_{DC}+2V_{P-P(AC)}$ , $I_{LOAD} \leq 150mA$ , $f=120Hz$	35	—	—	dB
$I_{LIMIT}$	Current Limit	6V	$\Delta V_{OUT}=-10\%$	225	380	—	mA
$t_{START}$	LDO Start-up Time	6V	$I_{LOAD}=1mA$ , $V_{OUT}$ settle to $\pm 5\%$	—	—	10	ms

Note: 1. Load regulation is measured at a constant junction temperature, using pulse testing with a low ON time and is guaranteed up to the maximum power dissipation. Power dissipation is determined by the input/output differential voltage and the output current. Guaranteed maximum power dissipation will not be available over the full input/output range. The maximum allowable power dissipation at any ambient temperature is  $P_D=(T_{J(MAX)}-T_a)/\theta_{JA}$ .

2. Dropout voltage is defined as the input voltage minus the output voltage that produces a 2% change in the output voltage from the value at appointed  $V_{IN}$ .

3. Ripple rejection ratio measurement circuit.  $RR=20 \times \log(\Delta V_{IN}/\Delta V_{OUT})$ .





## Level Shifter Electrical Characteristics

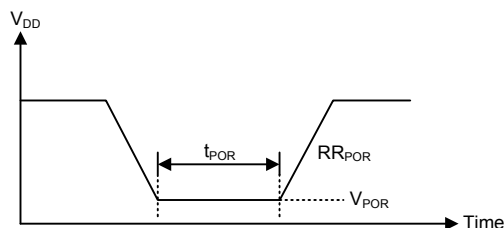
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>IN</sub>	Input Voltage	—	—	6	—	12	V
V <sub>CC2O</sub>	VCC2O Accuracy	—	V <sub>CC2</sub> =6V~12V, If A/D Converter is used, A/D clock=f <sub>SYS</sub> /8	-5%	0.2V <sub>CC2</sub>	+5%	V

## Power-on Reset Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>POR</sub>	V <sub>DD</sub> Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR <sub>POR</sub>	V <sub>DD</sub> Rising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t <sub>POR</sub>	Minimum Time for V <sub>DD</sub> Stays at V <sub>POR</sub> to Ensure Power-on Reset	—	—	1	—	—	ms



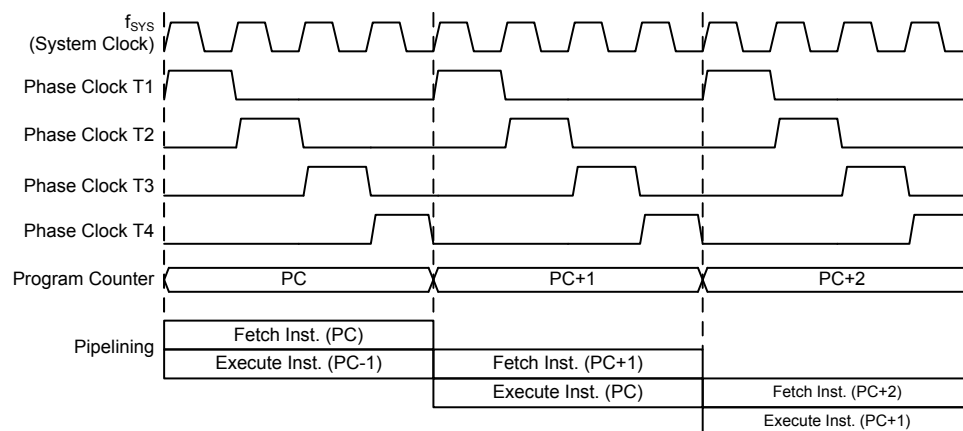
## System Architecture

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

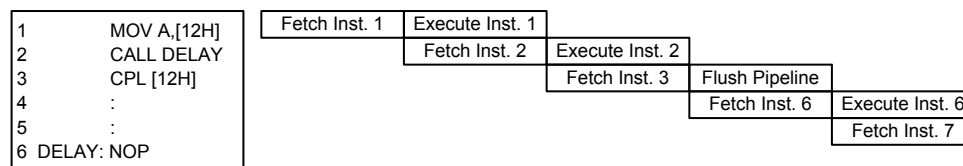
The main system clock, derived from either an HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the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



**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

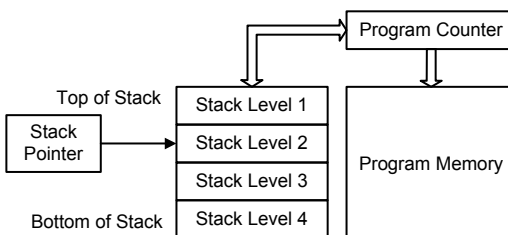
**Program Counter**

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

## Stack

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

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



## Arithmetic and Logic Unit – ALU

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

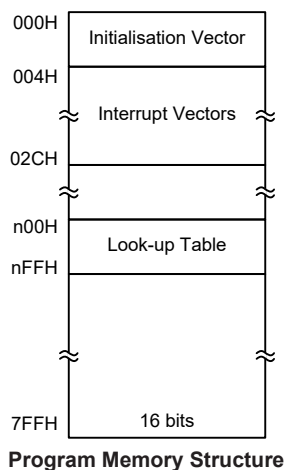
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement: INCA, INC, DECA, DEC
- Branch decision: JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET,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

## Flash Program Memory

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

### Structure

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



### Special Vectors

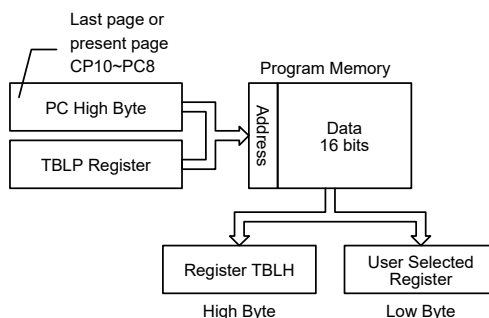
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

### Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the address of the look up data to be retrieved in the table pointer register, TBLP. This register defines 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 “TABRDC[m]” or “TABRDL[m]” instructions, respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as “0”.

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



### Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is “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 of the last page if the “TABRDL [m]” instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the “TABRDL [m]” instruction is executed.

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

### Table Read Program Example

```

tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
:
mov a,06h          ; initialise low table pointer - note that this address is referenced
mov tblp,a         ; to the last page or present page
:
:
tabrdl 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
tabrdl 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 – ICP

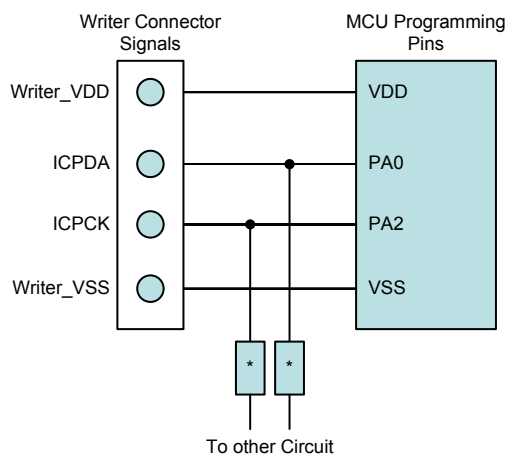
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides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

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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Serial Clock
VDD	VDD	Power Supply (5V)
VSS	VSS	Ground

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

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



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

## On-Chip Debug Support – OCDS

There is an EV chip named HT45V4833 which is used to emulate the HT45F4833. The EV chip device provides an “On-Chip Debug” function to debug the corresponding MCU device during the development process. The EV chip and the actual MCU device 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 using the EV device during debug, if there are other pin functions which are shared with the OCDSDA and OCDSCK pins, then these functions will have no effect in the EV chip. The two OCDS pins, which are pin-shared with the ICP programming pins, are still used as 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 (5V)
GND	VSS	Ground

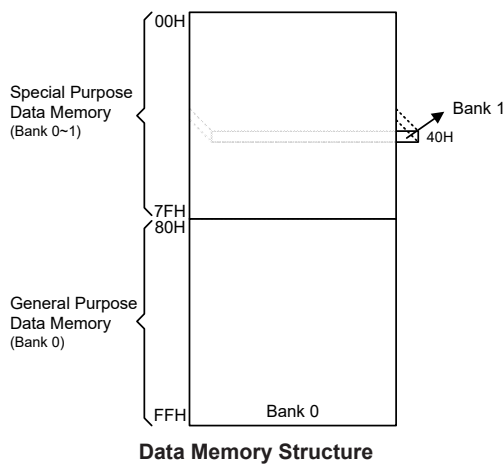
## RAM Data Memory

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

### Structure

Categorized into two types, 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 Bank 0, 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 is the address 00H.



**General Purpose Data Memory**


There are 128 bytes of General Purpose Data Memory which are located at 80H~FFH in Bank 0. All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user programing for both reading and writing operations. By using the bit operation instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

**Special Purpose Data Memory**

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



Bank 0		Bank 1	Bank 0		Bank 1
00H	IAR0		40H		EEC
01H	MP0		41H	EEA	
02H	IAR1		42H	EED	
03H	MP1		43H		
04H	BP		44H	PTM1C0	
05H	ACC		45H	PTM1C1	
06H	PCL		46H	PTM1C2	
07H	TBLP		47H	PTM1DL	
08H	TBLH		48H	PTM1DH	
09H			49H	PTM1AL	
0AH	STATUS		4AH	PTM1AH	
0BH			4BH	PTM1BL	
0CH	TBC		4CH	PTM1BH	
0DH	WDTC		4DH	PTM1RPL	
0EH			4EH	PTM1RPH	
0FH	RSTFC		4FH		
10H	SCC		50H		
11H	HIRCC		51H		
12H			52H		
13H	LVDC		53H		
14H	PA		54H		
15H	PAC		55H		
16H	PAPU		56H		
17H	PAWU		57H		
18H	PMS		58H		
19H	PXC		59H		
1AH	DTC		5AH		
1BH	POLS		5BH		
1CH	HVC		5CH		
1DH	HBC		5DH		
1EH			5EH		
1FH	INTEG		5FH		
20H	INTC0		60H		
21H	INTC1		61H		
22H	INTC2		62H		
23H	MF10		63H		
24H	MF11		64H		
25H	SADOL		65H		
26H	SADOH		66H		
27H	SADC0		67H		
28H	SADC1		68H		
29H	SACD2		69H		
2AH	PB		6AH		
2BH	PBC		6BH		
2CH	PBPU		6CH		
2DH			6DH		
2EH	PTM0C0		6EH		
2FH	PTM0C1		6FH		
30H	PTM0C2		70H		
31H	PTMODL		71H		
32H	PTMODH		72H		
33H	PTM0AL		73H		
34H	PTM0AH		74H		
35H	PTM0BL		75H		
36H	PTM0BH		76H		
37H	PTM0RPL		77H		
38H	PTM0RPH		78H		
39H			79H		
3AH	PAPS0		7AH		
3BH	PAPS1		7BH		
3CH	PBPS0		7CH		
3DH	PBPS1		7DH		
3EH	IFS0		7EH		
3FH	IFS1		7FH		

 : Unused, read as 00H

### Special Purpose Data Memory

## Special Function Register Description

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

### Indirect Addressing 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'
org 00h
start:
    mov a,04h          ; setup size of block
    mov block,a
    mov a,offset adres1 ; Accumulator loaded with first RAM address
    mov MP0,a          ; setup memory pointer with first RAM address
loop:
    clr IAR0           ; clear the data at address defined by MP0
    inc MP0            ; increment memory pointer
    sdz block          ; check if last memory location has been cleared
    jmp loop
continue:
```

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

## Bank Pointer – BP

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

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the SLEEP or IDLE mode, in which case, the Data Memory bank remains unaffected. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from Bank 1 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, TBLH

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

## Status Register – STATUS

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

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

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

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

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

• **STATUS Register**

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

"x": unknown

- Bit 7~6      Unimplemented, read as "0"
- Bit 5      **TO**: Watchdog Time-out flag  
             0: After power up or executing the "CLR WDT" or "HALT" instruction  
             1: A watchdog time-out occurred.
- Bit 4      **PDF**: Power down flag  
             0: After power up or executing the "CLR WDT" instruction  
             1: By executing the "HALT" instruction
- Bit 3      **OV**: Overflow flag  
             0: no overflow  
             1: an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa
- Bit 2      **Z**: Zero flag  
             0: The result of an arithmetic or logical operation is not zero  
             1: The result of an arithmetic or logical operation is zero
- Bit 1      **AC**: Auxiliary flag  
             0: No auxiliary carry  
             1: An operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction
- Bit 0      **C**: Carry flag  
             0: No carry-out  
             1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation  
             The C flag is also affected by a rotate through carry instruction.

## EEPROM Data Memory

The device contains an area of internal EEPROM Data Memory. EEPROM 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 32×8 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 two address registers and one 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 registers, 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 only, cannot be directly addressed and can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEA	—	—	—	EEA4	EEA3	EEA2	EEA1	EEA0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

**EEPROM Register List**

#### • EEA Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	EEA4	EEA3	EEA2	EEA1	EEA0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5      Unimplemented, read as “0”

Bit 4~0      **EEA4~EEA0**: Data EEPROM Address  
Data EEPROM address bit 4 ~ bit 0

#### • EED Register

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

Bit 7~0      **D7~D0**: Data EEPROM Data  
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: 1. The WREN, WR, RDEN and RD cannot be set to “1” at the same time in one instruction.  
 The WR and RD cannot be set to “1” at the same time.  
 2. Ensure that the  $f_{SUB}$  clock is stable before executing the write operation.  
 3. Ensure that the write operation is totally complete before changing the EEC register content.

### **Reading Data from the EEPROM**

To read data from the EEPROM, the EEPROM address of the data to be read must first be placed in the EEA register. The read enable bit, RDEN, in the EEC register must then be set high to enable the read function. 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. To initiate a write cycle 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 successfully. These two instructions must be executed in two consecutive instruction cycles. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set high again after the write cycle has started. Note that setting the WR bit high will not initiate a write cycle if the WREN bit has not been set. As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended.

### **Write Protection**

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

### **EEPROM Interrupt**

The EEPROM write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. When an EEPROM write cycle ends, the DEF request flag will be set. If the global, EEPROM Interrupt are enabled and the stack is not full, a subroutine call to the EEPROM Interrupt vector, will take place. When the EEPROM Interrupt is serviced, the EEPROM Interrupt flag DEF will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.



## 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 complete. 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 if no more read operations are required
CLR BP
MOV A, EED              ; move read data to register
MOV READ_DATA, A
```

Note: For each read operation, the address register should be re-specified followed by setting the RD bit high to activate a read cycle even if the target address is consecutive.

### 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
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 BP
```

## Oscillators

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

### Oscillator Overview

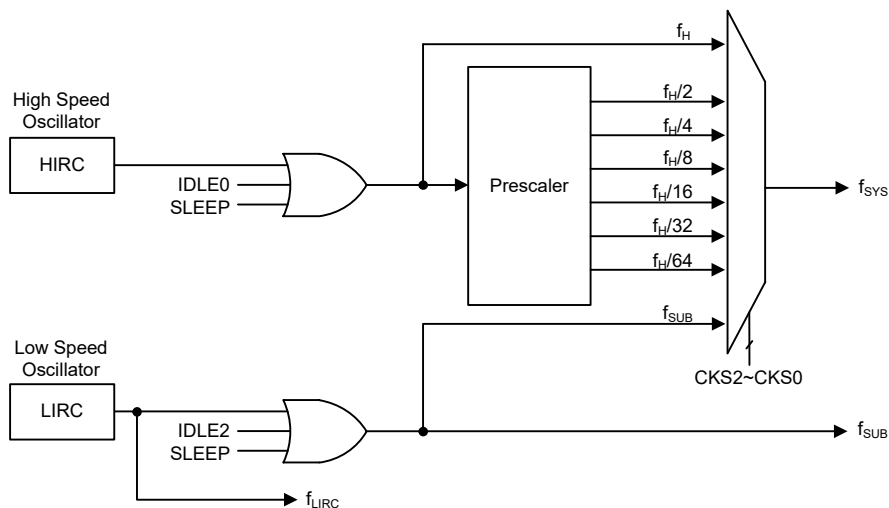
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Two fully integrated internal oscillators, requiring no external components, are provided to form a range of both fast and slow system oscillators. The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

Type	Name	Frequency
Internal High Speed RC	HIRC	8MHz
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 8MHz 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 CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.



**System Clock Configurations**

### **High Speed Internal RC Oscillator – HIRC**

The high speed internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 8MHz. 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. This internal system clock option 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, 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.

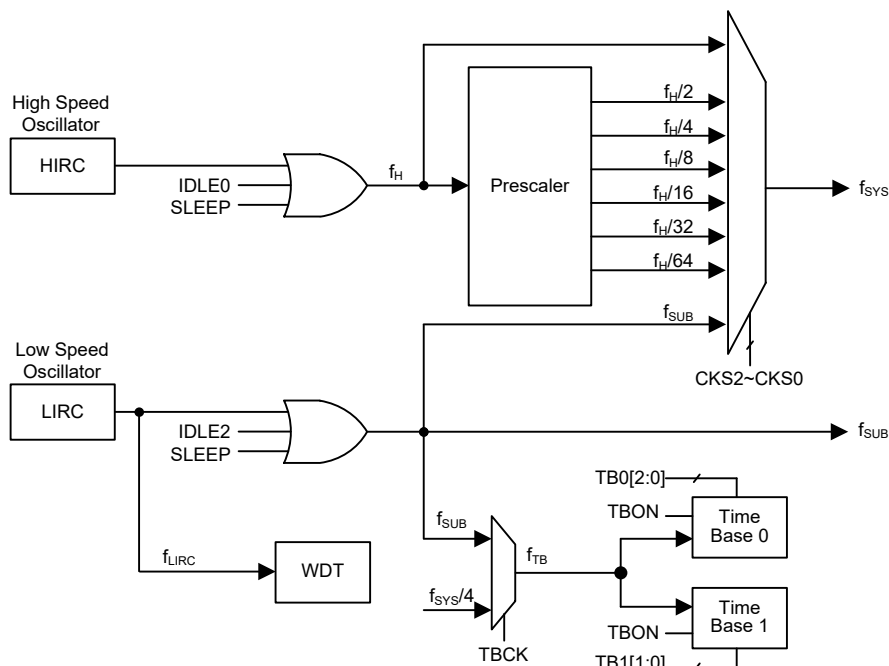
## **Operating Modes and System Clocks**

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 register programming, a clock system can be configured to obtain maximum application performance.

The main system clock, can come from either a high frequency  $f_H$  or low frequency  $f_{SUB}$  source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock 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  $f_H/2 \sim f_H/64$ .



**Device Clock Configurations**

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

## System Operation Modes

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

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

“x”: Don't care

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

### **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 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

### **SLOW Mode**

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from  $f_{SUB}$ . The  $f_{SUB}$  clock is derived from the LIRC oscillator. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW mode, the  $f_H$  clock will be switched on or off by configuring the corresponding oscillator enable bit HIRCEN.

### **SLEEP Mode**

The SLEEP Mode is entered when an HALT instruction is executed and when the FHIDEN and FSIDEN bit are low. In the SLEEP mode the CPU will be stopped, and the  $f_{SUB}$  clock provided to peripheral will be stopped too. However the  $f_{LIRC}$  clock can still continue to operate if the WDT function is enabled, the  $f_{LIRC}$  clock will be stopped too, if the Watchdog Timer function is disabled.

### **IDLE0 Mode**

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

### **IDLE1 Mode**

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

### **IDLE2 Mode**

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

## **Control Register**

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

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

**System Operating Mode Control Register List**

• **SCC Register**

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

Bit 7~5      **CKS2~CKS0**: System clock selection

000:  $f_H$

001:  $f_H/2$

010:  $f_H/4$

011:  $f_H/8$

100:  $f_H/16$

101:  $f_H/32$

110:  $f_H/64$

111:  $f_{SUB}$

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

Bit 4~2      Unimplemented, read as “0”

Bit 1      **FHIDEN**: High frequency oscillator control when CPU is switched off

0: Disable

1: Enable

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

Bit 0      **FSIDEN**: Low frequency oscillator control when CPU is switched off

0: Disable

1: Enable

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

• **HIRCC Register**

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

Bit 7~2      Unimplemented, read as “0”

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

0: HIRC unstable

1: HIRC stable

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

Bit 0      **HIRCEN**: HIRC oscillator enable control

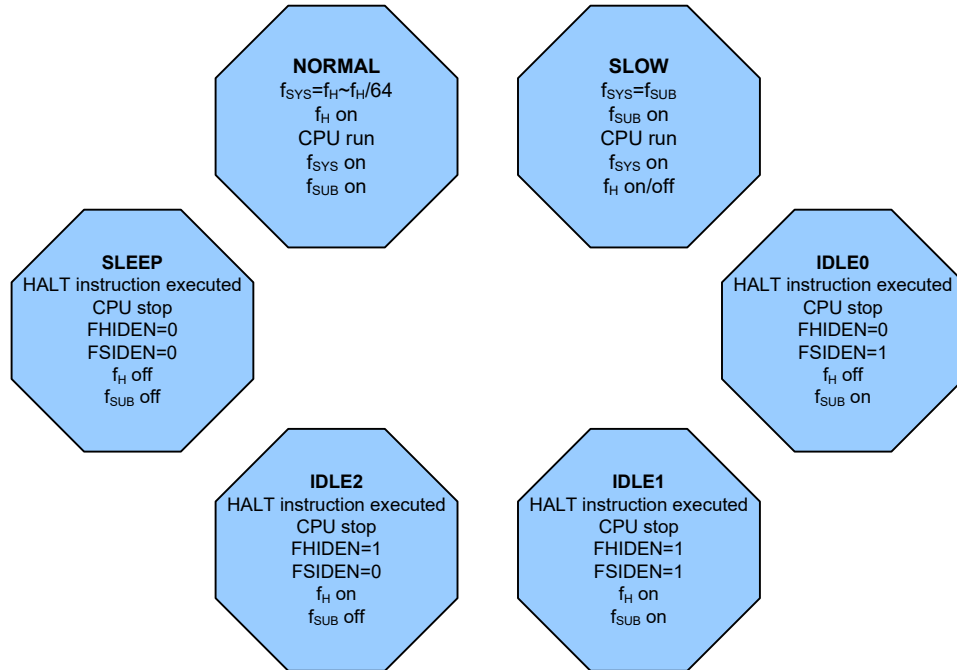
0: Disable

1: Enable

## Operating Mode Switching

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

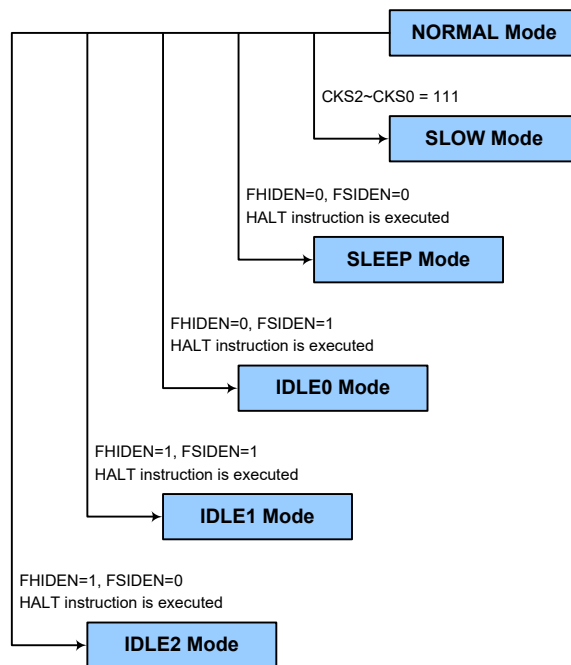
In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the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



### NORMAL Mode to SLOW Mode Switching

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

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

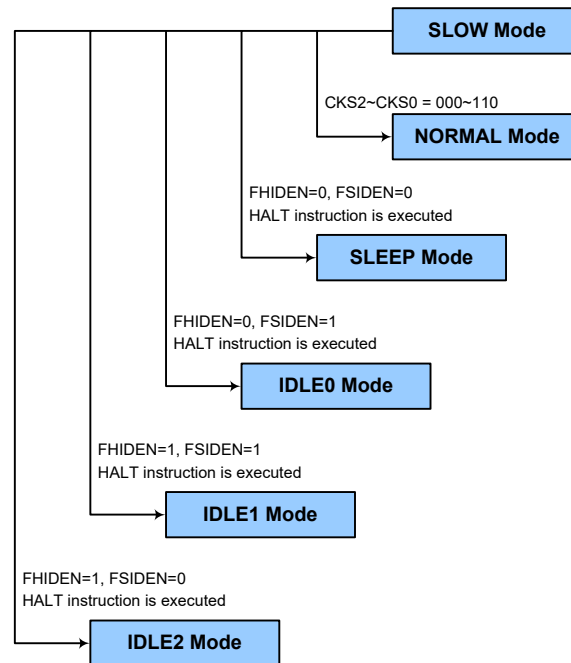




### SLOW Mode to NORMAL Mode Switching

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

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



### Entering the SLEEP Mode

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

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

**Entering the IDLE0 Mode**

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

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

**Entering the IDLE1 Mode**

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

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

**Entering the IDLE2 Mode**

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

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

## **Standby Current Considerations**

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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

In the IDLE1 and IDLE2 Mode the high speed oscillator is on, if the peripheral function clock source is derived from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

## **Wake-up**

To minimise power consumption the device can enter the SLEEP or any IDLE Mode, where the CPU will be switched off. However, when the device is woken up again, it will take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

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

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

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

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

## Watchdog Timer

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

### Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock,  $f_{LIRC}$  which is sourced from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with  $V_{DD}$ , temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of  $2^8$  to  $2^{15}$  to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

### Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period as well as the enable/disable and reset MCU operation.

#### • WDTC Register

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

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

10101: Disable

01010: Enable

Other values: Reset MCU

When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time,  $t_{SRESET}$ , and the WRF bit in the RSTFC register will be set high.

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

000:  $2^8/f_{LIRC}$

001:  $2^9/f_{LIRC}$

010:  $2^{10}/f_{LIRC}$

011:  $2^{11}/f_{LIRC}$

100:  $2^{12}/f_{LIRC}$

101:  $2^{13}/f_{LIRC}$

110:  $2^{14}/f_{LIRC}$

111:  $2^{15}/f_{LIRC}$

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

• **RSTFC Register**

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

“x”: unknown

Bit 7~3 Unimplemented, read as “0”

Bit 2 **LVRF**: LVR function reset flag  
Described in the Low Voltage Reset section

Bit 1 Unimplemented, read as “0”

Bit 0 **WRF**: WDTC register software reset flag  
0: Not occurred  
1: Occurred

This bit is set high by the WDTC register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

## Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control and reset control of the Watchdog Timer. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B while the WDT function will be enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after a delay time,  $t_{\text{SRESET}}$ . After power on these bits will have a value of 01010B.

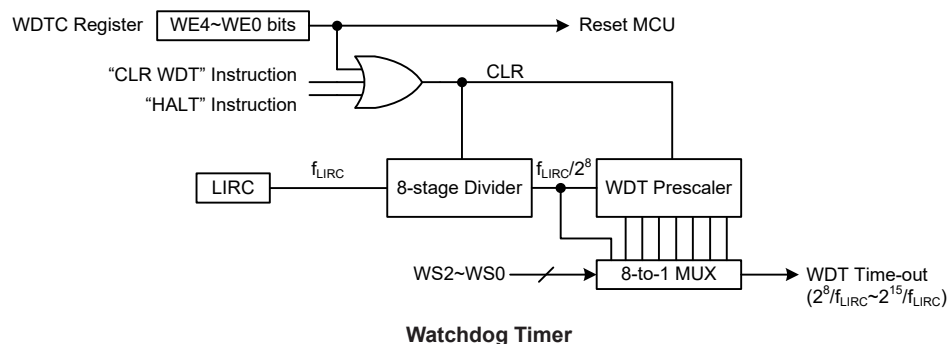
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

### Watchdog Timer Enable/Disable Control

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

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

The maximum time-out period is when the  $2^{15}$  division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 1 second for the  $2^{15}$  division ratio, and a minimum timeout of 8ms for the  $2^8$  division ration.



## Reset and Initialisation

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

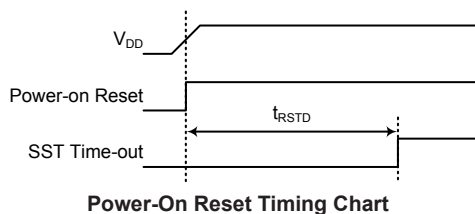
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

### Reset Functions

There are several ways in which a microcontroller reset can occur through events occurring internally.

#### Power-on Reset

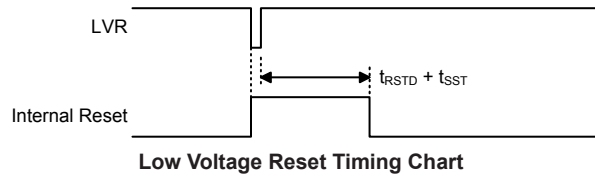
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions.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



### Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level.

The LVR function is always enabled in the Normal or Slow mode with a specific LVR voltage  $V_{LVR}$ . If the supply voltage of the device drops to within a range of  $0.9V \sim V_{LVR}$  such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between  $0.9V \sim V_{LVR}$  must exist for a time greater than that specified by  $t_{LVR}$  in the LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual  $V_{LVR}$  value is 3.15V. Note that the LVR function will be automatically disabled when the device enters the IDLE or SLEEP mode.



### • RSTFC Register

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

“x”: unknown

Bit 7~3 Unimplemented, read as “0”

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

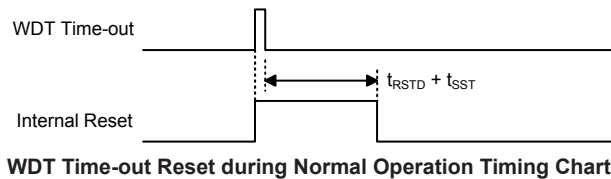
This bit is set high when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

Bit 1 Unimplemented, read as “0”

Bit 0 **WRF**: WDT Control register software reset flag  
 Described in the Watchdog Timer Control Register section

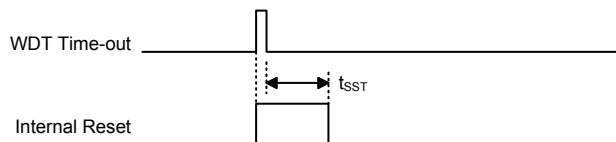
### Watchdog Time-out Reset during Normal Operation

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



### Watchdog Time-out Reset during SLEEP or IDLE Mode

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



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

### Reset Initial Conditions

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

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

“u”: unchanged

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

Item	Condition after Reset
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Bases	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. Note that where more than one package type exists the table will reflect the situation for the larger package type.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE or SLEEP)
IAR0	x x x x x x x x	x x x x x x x x	u u u u u u u u
MP0	x x x x x x x x	x x x x x x x x	u u u u u u u u
IAR1	x x x x x x x x	u u u u u u u u	u u u u u u u u
MP1	x x x x x x x x	u u u u u u u u	u u u u u u u u
BP	- - - - - 0	- - - - - 0	- - - - - u
ACC	x x x x x x x x	u u u u u u u u	u u u u u u u u
PCL	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
TBLP	x x x x x x x x	u u u u u u u u	u u 1 1 u u u u



Register	Reset (Power On)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE or SLEEP)
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	--00 xxxx	--1u xxxx	--11 uuuu
TBC	0011 -111	0011 -111	uuuu -uuu
WDTC	0101 0011	0101 0011	uuuu uuuu
RSTFC	---- -x-0	---- -u-u	---- -u-u
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- --01	---- --01	---- --uu
LVDC	--00 0000	--00 0000	--uu uuuu
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
PMS	---- 0000	---- 0000	---- uuuu
PXC	---- 0000	---- 0000	---- uuuu
DTC	0000 0000	0000 0000	uuuu uuuu
POLS	---- 0000	---- 0000	---- uuuu
HVC	--00 0000	--00 0000	--uu uuuu
HBC	---- --xx	---- --xx	---- --uu
INTEG	0000 0000	0000 0000	uuuu uuuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
MFIO	--00 --00	--00 --00	--uu --uu
MF11	-000 -000	-000 -000	-uuu -uuu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRF=0)
			uuuu uuuu (ADRF=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRF=0)
			---- uuuu (ADRF=1)
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 -000	0000 -000	uuuu -uuu
SADC2	0-00 0000	0-00 0000	u-uu uuuu
PB	---1 1111	---1 1111	---u uuuu
PBC	---1 1111	---1 1111	---u uuuu
PBPU	---0 0000	---0 0000	---u uuuu
PTM0C0	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	uuuu uuuu
PTM0C2	---- -000	---- -000	---- -uuu
PTM0DL	0000 0000	0000 0000	uuuu uuuu
PTM0DH	0000 0000	0000 0000	uuuu uuuu
PTM0AL	0000 0000	0000 0000	uuuu uuuu
PTM0AH	0000 0000	0000 0000	uuuu uuuu
PTM0BL	0000 0000	0000 0000	uuuu uuuu
PTM0BH	0000 0000	0000 0000	uuuu uuuu
PTM0RPL	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	0000 0000	0000 0000	uuuu uuuu
PAPS0	0000 0000	0000 0000	uuuu uuuu
PAPS1	0000 0000	0000 0000	uuuu uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE or SLEEP)
PBPS0	0000 0000	0000 0000	uuuu uuuu
PBPS1	---- --00	---- --00	---- --uu
IFS0	0000 0000	0000 0000	uuuu uuuu
IFS1	---- 0000	---- 0000	---- uuuu
EEC	---- 0000	---- 0000	---- uuuu
EEA	---0 0000	---0 0000	---u uuuu
EED	0000 0000	0000 0000	uuuu uuuu
PTM1C0	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	uuuu uuuu
PTM1C2	---- -000	---- -000	---- -uuu
PTM1DL	0000 0000	0000 0000	uuuu uuuu
PTM1DH	---- --00	---- --00	---- --uu
PTM1AL	0000 0000	0000 0000	uuuu uuuu
PTM1AH	---- --00	---- --00	---- --uu
PTM1BL	0000 0000	0000 0000	uuuu uuuu
PTM1BH	---- --00	---- --00	---- --uu
PTM1RPL	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	---- --00	---- --00	---- --uu

Note: “u” stands for unchanged  
“x” stands for unknown  
“-” stands for unimplemented

## Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA~PB. 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	—	—	—	PB4	PB3	PB2	PB1	PB0
PBC	—	—	—	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	—	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0

“—”: Unimplemented, read as “0”

### I/O Logic Function Register List

## Pull-high Resistors

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

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

### • PxPUn Register

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

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

0: Disable

1: Enable

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

## Port A Wake-up

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

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

### • PAWU Register

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

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

0: Disable

1: Enable

## I/O Port Control Registers

Each I/O port has its own control register known as PAC~PBC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a “1”. This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a “0”,

the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

#### • PxC Register

Bit	7	6	5	4	3	2	1	0
Name	PxC7	PxC6	PxC5	PxC4	PxC3	PxC2	PxC1	PxC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

**PxCn:** I/O Port x Pin type selection

0: Output

1: Input

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

### Pin-shared Functions

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

#### Pin-shared Function Selection Registers

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

When the pin-shared input function is selected to be used, the corresponding input and output functions selection should be properly managed. For example, if the PTCKn line is used, the corresponding output pin-shared function should be configured as the PTCKn function by configuring the PxPSn register and the PTCKn signal input should be properly selected using the IFSi register. However, if the external interrupt function is selected to be used, the relevant output pin-shared function should be selected as an I/O function and the interrupt input signal should be selected.

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAPS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAPS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBPS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBPS1	—	—	—	—	—	—	PBS11	PBS10
IFS0	PTCK1S1	PTCK1S0	PTP1IS1	PTP1IS0	PTCK0S1	PTCK0S0	PTP0IS1	PTP0IS0
IFS1	—	—	—	—	INT3S	INT2S	INT1S	INT0S

Pin-shared Function Selection Register List

• PAPS0 Register

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

Bit 7~6 **PAS07~PAS06:** PA3 pin-shared function selection

00: PA3/INT3/PTP0I

01: PTP0B

10: AN5

11: PA3/INT3/PTP0I

Bit 5~4 **PAS05~PAS04:** PA2 pin-shared function selection

00: PA2

01: PTP0

10: PA2

11: PA2

Bit 3~2 **PAS03~PAS02:** PA1 pin-shared function selection

00: PA1/PTP1I

01: PTP0B

10: VCC2O

11: PA1/PTP1I

Bit 1~0 **PAS01~PAS00:** PA0 pin-shared function selection

00: PA0/PTCK1

01: PTP1

10: PA0/PTCK1

11: PA0/PTCK1

• PAPS1 Register

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

Bit 7~6 **PAS17~PAS16:** PA7 pin-shared function selection

00: PA7/INT0/PTCK0

01: PTP0

10: PA7/INT0/PTCK0

11: PA7/INT0/PTCK0

Bit 5~4 **PAS15~PAS14:** PA6 pin-shared function selection

00: PA6/INT1/PTP0I

01: PTP0B

10: PA6/INT1/PTP0I

11: PA6/INT1/PTP0I

- Bit 3~2     **PAS13~PAS12:** PA5 pin-shared function selection  
                  00: PA5/INT1/PTP1I  
                  01: PTP1B  
                  10: AN3  
                  11: PA5/INT1/PTP1I
- Bit 1~0     **PAS11~PAS10:** PA4 pin-shared function selection  
                  00: PA4/INT2/PTCK0  
                  01: PTP0  
                  10: AN4  
                  11: PA4/INT2/PTCK0

• **PBPS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7~6     **PBS07~PBS06:** PB3 pin-shared function selection  
                  00: PB3/INT3/PTP1I  
                  01: PTP1B  
                  10: PB3/INT3/PTP1I  
                  11: PB3/INT3/PTP1I
- Bit 5~4     **PBS05~PBS04:** PB2 pin-shared function selection  
                  00: PB2/PTCK0  
                  01: PTP1B  
                  10: AN0  
                  11: VREFI
- Bit 3~2     **PBS03~PBS02:** PB1 pin-shared function selection  
                  00: PB1/PTP0I  
                  01: PB1/PTP0I  
                  10: AN1  
                  11: VREF
- Bit 1~0     **PBS01~PBS00:** PB0 pin-shared function selection  
                  00: PB0/INT0/PTCK1  
                  01: PTP1  
                  10: AN2  
                  11: PB0/INT0/PTCK1

• **PBPS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PBS11	PBS10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

- Bit 7~2     Unimplemented, read as “0”
- Bit 1~0     **PBS11~PBS10:** PB4 pin-shared function selection  
                  00: PB4/INT2/PTCK1  
                  01: PTP1  
                  10: PB4/INT2/PTCK1  
                  11: PB4/INT2/PTCK1

• **IFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTCK1S1	PTCK1S0	PTP1IS1	PTP1IS0	PTCK0S1	PTCK0S0	PTP0IS1	PTP0IS0
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     **PTCK1S1~PTCK1S0**: PTCK1 input source pin selection  
                     00: PB0  
                     01: PB4  
                     10: PA0  
                     11: PB0
- Bit 5~4     **PTP1IS1~PTP1IS0**: PTP1I input source pin selection  
                     00: PA5  
                     01: PA1  
                     10: PB3  
                     11: PA5
- Bit 3~2     **PTCK0S1~PTCK0S0**: PTCK0 input source pin selection  
                     00: PB2  
                     01: PA4  
                     10: PA7  
                     11: PB2
- Bit 1~0     **PTP0IS1~PTP0IS0**: PTP0I input source pin selection  
                     00: PA3  
                     01: PA6  
                     10: PB1  
                     11: PA3

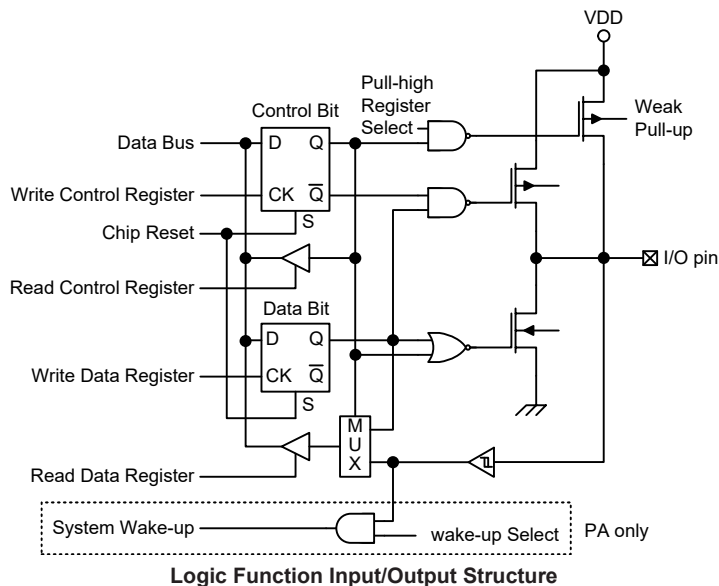
• **IFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT3S	INT2S	INT1S	INT0S
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

- Bit 7~4     Unimplemented, read as “0”
- Bit 3     **INT3S**: INT3 input source pin selection  
                     0: PA3  
                     1: PB3
- Bit 2     **INT2S**: INT2 input source pin selection  
                     0: PA4  
                     1: PB4
- Bit 1     **INT1S**: INT1 input source pin selection  
                     0: PA5  
                     1: PA6
- Bit 0     **INT0S**: INT0 input source pin selection  
                     0: PB0  
                     1: PA7

## I/O Pin Structures

The accompanying diagram illustrates the internal structure of the I/O logic function. As the exact logical construction of the I/O pin will differ from this drawing, it is supplied as a guide only to assist with the functional understanding of the I/O logic function. The wide range of pin-shared structures does not permit all types to be shown.



## Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the “SET [m].i” and “CLR [m].i” instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

Port A has the additional capability of providing wake-up 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 Module – 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 two 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 pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

The brief features of the Periodic TMs are described here with more detailed information provided in the Periodic Type TM section.

### Introduction

The device contains one 10-bit and one 16-bit Periodic Type TMs with each TM having a reference name of PTM0 or PTM1. The main features of the PTMs are summarised in the accompanying table.

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

**TM Function Summary**

PTM0	PTM1
16-bit PTM	10-bit PTM

**TM Name/Type Reference**

### TM Operation

The Periodic 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. The internal TM counter is driven by a user selectable internal clock source.

### TM Clock Source

The clock source which drives the main counter in the TM can originate from various sources. The selection of the required clock source is implemented using the PTnCK2~PTnCK0 bits in the PTMnC0 control register, where “n” stands for the specific TM serial number. The clock source can be a ratio of the system clock  $f_{SYS}$  or the internal high clock  $f_H$  or the  $f_{SUB}$  clock source.

## TM Interrupts

Each of the Periodic type 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.

## TM External Pins

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

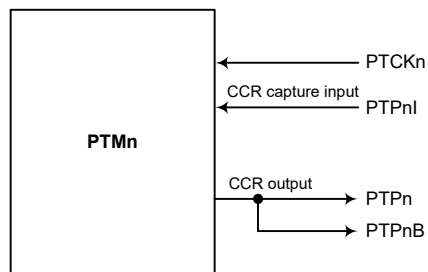
The other PTMn input pin, PTPnI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTnIO1~PTnIO0 bits in the PTMnC1 register. There is another capture input, PTCKn, for PTMn capture input mode, which can be used as the external trigger input source except the PTPnI pin.

The PTMs each has two output pins with the label PTPn and PTPnB. The PTPnB pin outputs the inverted signal of the PTPn. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external PTPn and PTPnB output pins are also the pins where the TM generates the PWM output waveform.

As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using the relevant pin-shared function selection bits described in the Pin-shared Function section. The details of the pin-shared function selection are described in the pin-shared function section.

PTM0		PTM1	
Input	Output	Input	Output
PTCK0, PTP0I	PTP0, PTP0B	PTCK1, PTP1I	PTP1, PTP1B

**TM External Pins**

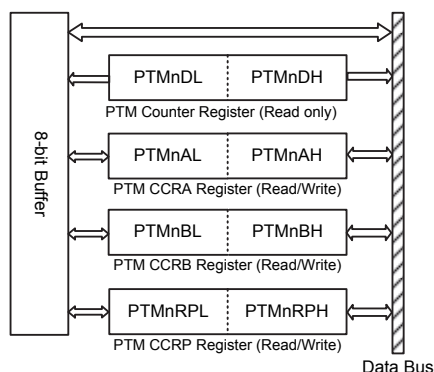


**PTM Function Pin Block Diagram (n=0~1)**

## Programming Considerations

The TM Counter Registers, the Capture/Compare CCRA, CCRB and CCRP registers, being 10-bit or 16-bit, all have a low and high byte structure. The high byte can be directly accessed, but as the low byte can only be accessed via an internal 8-bit buffer, reading or writing to this register pair must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed.

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

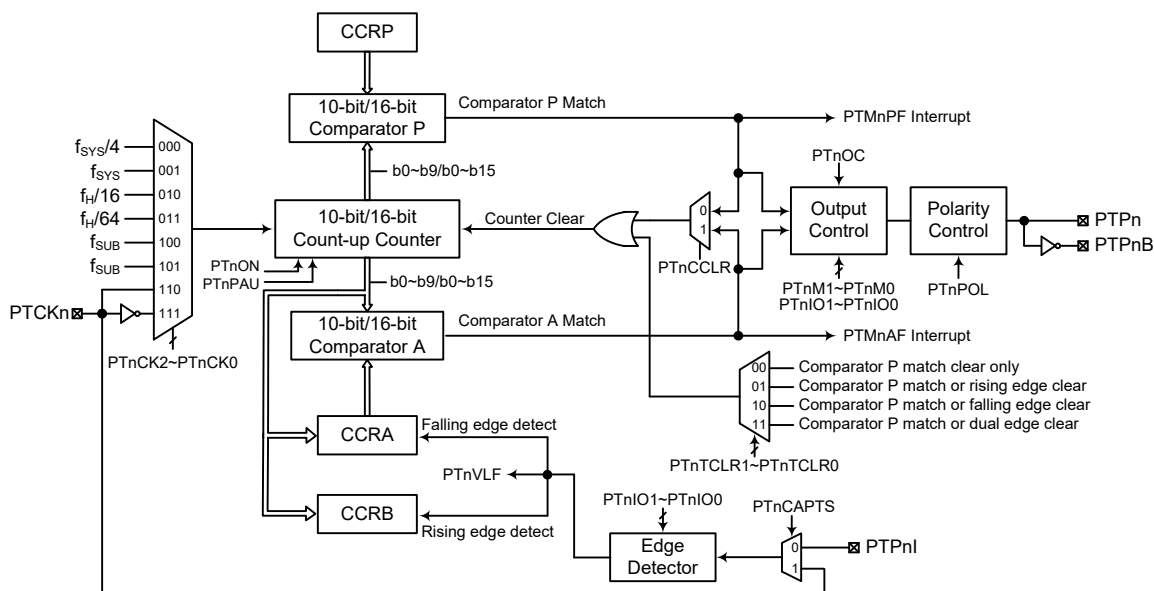


The following steps show the read and write procedures:

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

## Periodic Type TM – PTM

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



Note: 1: 16-bit for PTM0, 10-bit for PTM1.

- The PTMn external pins are pin-shared with other functions, so before using the PTMn function, the pin-shared function registers must be set properly.

**Periodic Type TM Block Diagram (n=0~1)**

### Periodic TM Operation

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

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

### Periodic Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTM0C0	PT0PAU	PT0CK2	PT0CK1	PT0CK0	PT0ON	—	—	—
PTM0C1	PT0M1	PT0M0	PT0IO1	PT0IO0	PT0OC	PT0POL	PT0CAPTS	PT0CCLR
PTM0C2	—	—	—	—	—	PT0TCLR1	PT0TCLR0	PT0VLF
PTM0DL	D7	D6	D5	D4	D3	D2	D1	D0
PTM0DH	D15	D14	D13	D12	D11	D10	D9	D8
PTM0AL	D7	D6	D5	D4	D3	D2	D1	D0
PTM0AH	D15	D14	D13	D12	D11	D10	D9	D8
PTM0BL	D7	D6	D5	D4	D3	D2	D1	D0
PTM0BH	D15	D14	D13	D12	D11	D10	D9	D8
PTM0RPL	PT0RP7	PT0RP6	PT0RP5	PT0RP4	PT0RP3	PT0RP2	PT0RP1	PT0RP0
PTM0RPH	PT0RP15	PT0RP14	PT0RP13	PT0RP12	PT0RP11	PT0RP10	PT0RP9	PT0RP8

**16-bit Periodic TM Register List**

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTM1C0	PT1PAU	PT1CK2	PT1CK1	PT1CK0	PT1ON	—	—	—
PTM1C1	PT1M1	PT1M0	PT1IO1	PT1IO0	PT1OC	PT1POL	PT1CAPTS	PT1CCLR
PTM1C2	—	—	—	—	—	PT1TCLR1	PT1TCLR0	PT1VLF
PTM1DL	D7	D6	D5	D4	D3	D2	D1	D0
PTM1DH	—	—	—	—	—	—	D9	D8
PTM1AL	D7	D6	D5	D4	D3	D2	D1	D0
PTM1AH	—	—	—	—	—	—	D9	D8
PTM1BL	D7	D6	D5	D4	D3	D2	D1	D0
PTM1BH	—	—	—	—	—	—	D9	D8
PTM1RPL	PT1RP7	PT1RP6	PT1RP5	PT1RP4	PT1RP3	PT1RP2	PT1RP1	PT1RP0
PTM1RPH	—	—	—	—	—	—	PT1RP9	PT1RP8

**10-bit Periodic TM Register List**

• **PTMnC0 Register (n=0~1)**

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

Bit 7 **PTnPAU**: PTMn Counter Pause Control

0: Run

1: Pause

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

Bit 6~4 **PTnCK2~PTnCK0**: Select PTMn Counter clock

000:  $f_{SYS}/4$

001:  $f_{SYS}$

010:  $f_H/16$

011:  $f_H/64$

100:  $f_{SUB}$

101:  $f_{SUB}$

110: PTCKn rising edge clock

111: PTCKn falling edge clock

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

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

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

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

Bit 2~0 Unimplemented, read as “0”

• **PTMnC1 Register (n=0~1)**

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

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

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

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

Compare Match Output Mode

- 00: No change
- 01: Output low
- 10: Output high
- 11: Toggle output

PWM Output Mode/Single Pulse Output Mode

- 00: PWM output inactive state
- 01: PWM output active state
- 10: PWM output
- 11: Single pulse output

Capture Input Mode

PTnTCLR[1:0]=00B:

- 00: Input capture at rising edge of PTPnI or PTCKn, and the counter value will be latched into CCRA
- 01: Input capture at falling edge of PTPnI or PTCKn, and the counter value will be latched into CCRA
- 10: Input capture at both falling and rising edge of PTPnI or PTCKn, and the counter value will be latched into CCRA
- 11: Input capture disabled

PTnTCLR[1:0]=01B, 10B or 11B:

- 00: Input capture at rising edge of PTPnI or PTCKn, and the counter value will be latched into CCRB
- 01: Input capture at falling edge of PTPnI or PTCKn, and the counter value will be latched into CCRA
- 10: Input capture at both falling and rising edge of PTPnI or PTCKn, and the counter value will be latched into CCRA at falling edge and into CCRB at rising edge
- 11: Input capture disabled

Timer/Counter Mode

Unused

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

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

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

Bit 3

**PTnOC:** PTMn PTPn Output control bit

Compare Match Output Mode

- 0: Initial low
- 1: Initial high

PWM Output Mode/Single Pulse Output Mode

- 0: Active low
- 1: Active high

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

Bit 2

**PTnPOL:** PTMn PTPn Output polarity Control

- 0: Non-invert
- 1: Invert

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

Bit 1

**PTnCAPTS:** PTMn Capture Trigger Source Selection

- 0: From PTPnI pin
- 1: From PTCKn pin

Bit 0      **PTnCCLR**: Select PTMn Counter clear condition  
             0: PTMn Comparator P match  
             1: PTMn Comparator A match

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

• **PTMnC2 Register (n=0~1)**

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

Bit 7~3      Unimplemented, read as “0”

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

Bit 0      **PTnVLF**: PTMn counter value latch edge flag  
             0: Falling edge trigger the counter value latch  
             1: Rising edge trigger the counter value latch

When the PTnTCLR1~PTnTCLR0 bits equal to 00B, ignore this flag status.

• **PTMnDL Register (n=0~1)**

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

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

• **PTM0DH Register**

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

Bit 7~0      **D15~D8**: PTM0 Counter High Byte Register bit 7 ~ bit 0  
             PTM0 16-bit Counter bit 15 ~ bit 8



• **PTM1DH Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **D9~D8**: PTM1 Counter High Byte Register bit 1 ~ bit 0  
PTM1 10-bit Counter bit 9 ~ bit 8

• **PTMnAL Register (n=0~1)**

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

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

• **PTM0AH Register**

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

Bit 7~0 **D15~D8**: PTM0 CCRA High Byte Register bit 7 ~ bit 0  
PTM0 16-bit CCRA bit 15 ~ bit 8

• **PTM1AH Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **D9~D8**: PTM1 CCRA High Byte Register bit 1 ~ bit 0  
PTM1 10-bit CCRA bit 9 ~ bit 8

• **PTMnBL Register (n=0~1)**

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

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

• **PTM0BH Register**

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

Bit 7~0      **D15~D8**: PTM0 CCRB High Byte Register bit 7 ~ bit 0  
PTM0 16-bit CCRB bit 15 ~ bit 8

• **PTM1BH Register**

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

Bit 7~2      Unimplemented, read as “0”  
Bit 1~0      **D9~D8**: PTM1 CCRB High Byte Register bit 1 ~ bit 0  
PTM1 10-bit CCRB bit 9 ~ bit 8

• **PTMnRPL Register (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	PTnRP7	PTnRP6	PTnRP5	PTnRP4	PTnRP3	PTnRP2	PTnRP1	PTnRP0
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      **PTnRP7~PTnRP0**: PTMn CCRP Low Byte Register bit 7 ~ bit 0  
PTMn 10-bit/16-bit CCRP bit 7 ~ bit 0

• **PTM0RPH Register**

Bit	7	6	5	4	3	2	1	0
Name	PT0RP15	PT0RP14	PT0RP13	PT0RP12	PT0RP11	PT0RP10	PT0RP9	PT0RP8
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      **PT0RP15~PT0RP8**: PTM0 CCRP High Byte Register bit 7 ~ bit 0  
PTM0 16-bit CCRP bit 15 ~ bit 8

• **PTM1RPH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PT1RP9	PT1RP8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2      Unimplemented, read as “0”  
Bit 1~0      **PT1RP9~PT1RP8**: PTM1 CCRP High Byte Register bit 1 ~ bit 0  
PTM1 10-bit CCRP bit 9 ~ bit 8

## **Periodic Type TM Operating Modes**

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

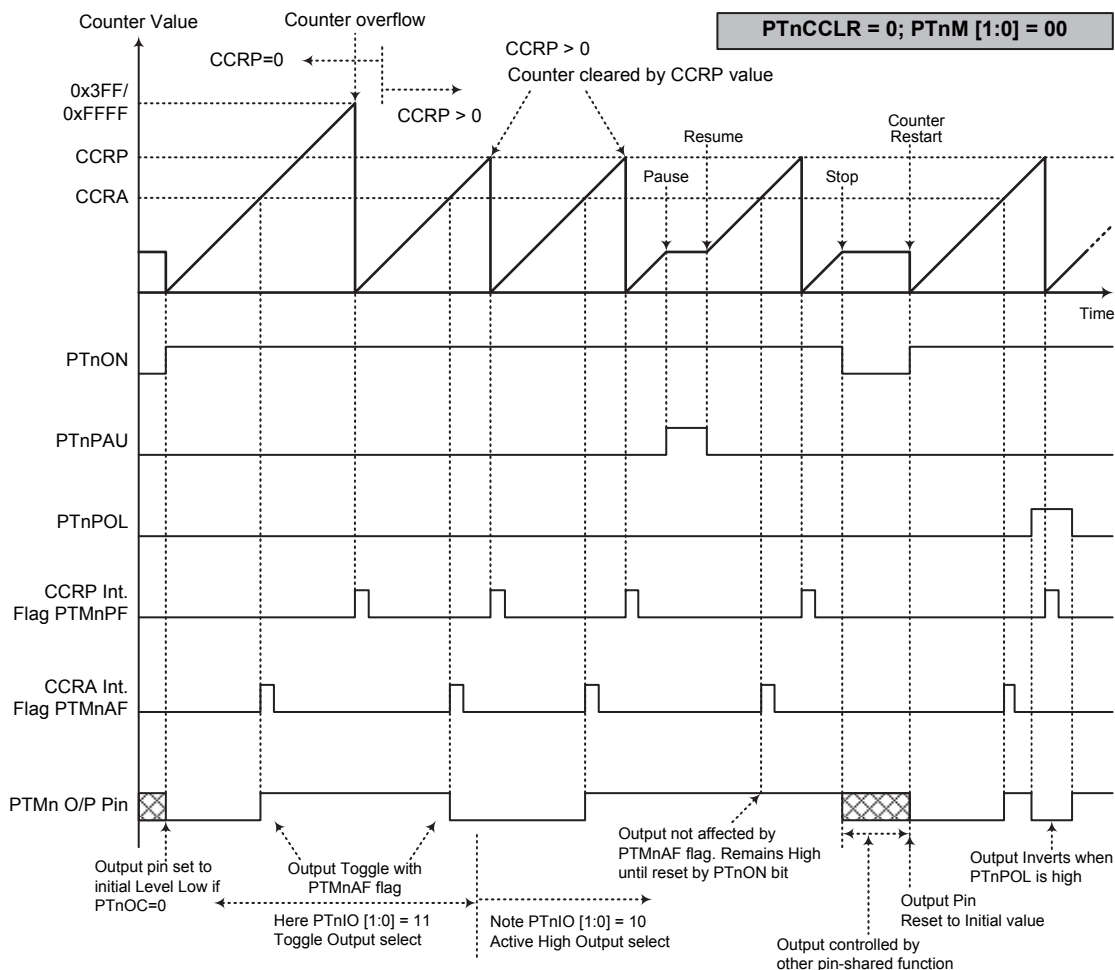
### **Compare Match Output Mode**

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

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

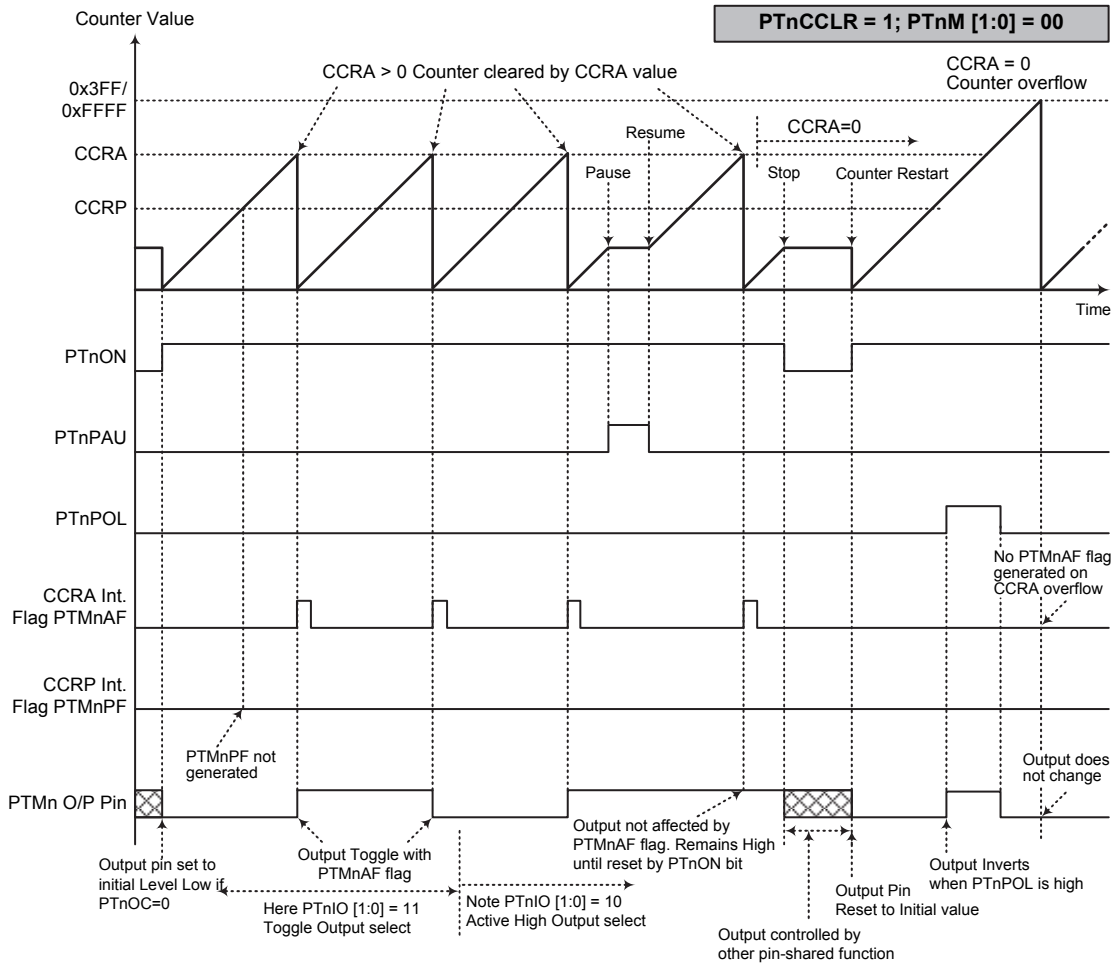
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, or 16-bit, FFFF Hex, value, however here the PTMnAF interrupt request flag will not be generated.

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



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

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



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

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

### Timer/Counter Mode

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

### PWM Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 10 respectively. The PWM function within the PTMn is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the PTMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

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

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

#### • 16-bit PTM0, PWM Output Mode, Edge-aligned Mode

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

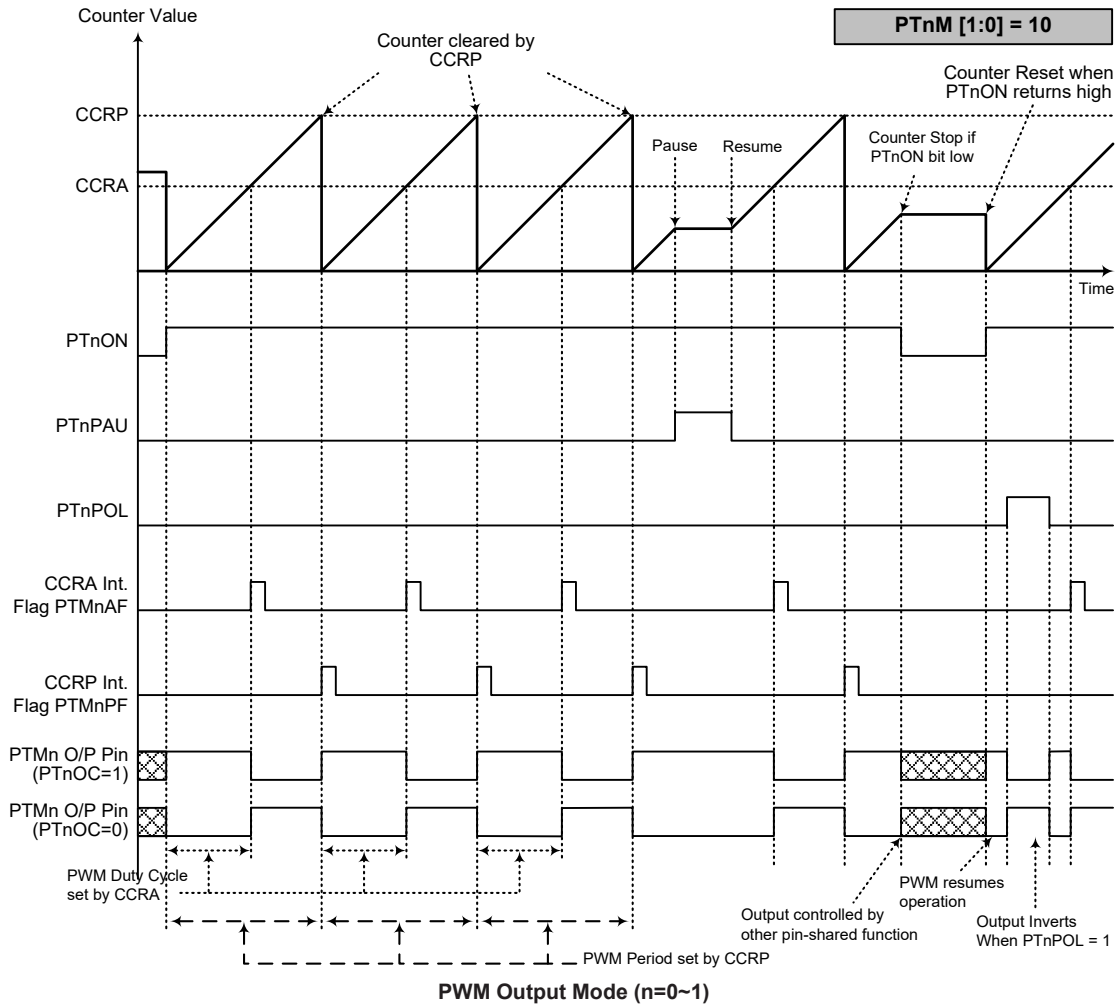
#### • 10-bit PTM1, PWM Output Mode, Edge-aligned Mode

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

If  $f_{SYS}=8\text{MHz}$ , PTMn clock source select  $f_{SYS}/4$ , CCRP=512 and CCRA=128,

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

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



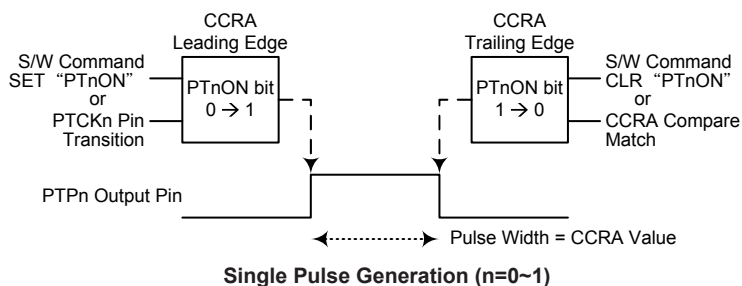
- Note: 1. Counter cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues running even when PTnIO[1:0]=00 or 01  
 4. The PTnCCLR bit has no influence on PWM operation

### Single Pulse Output Mode

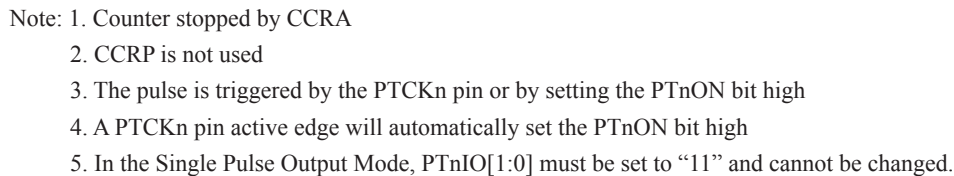
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

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

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







### Capture Input Mode

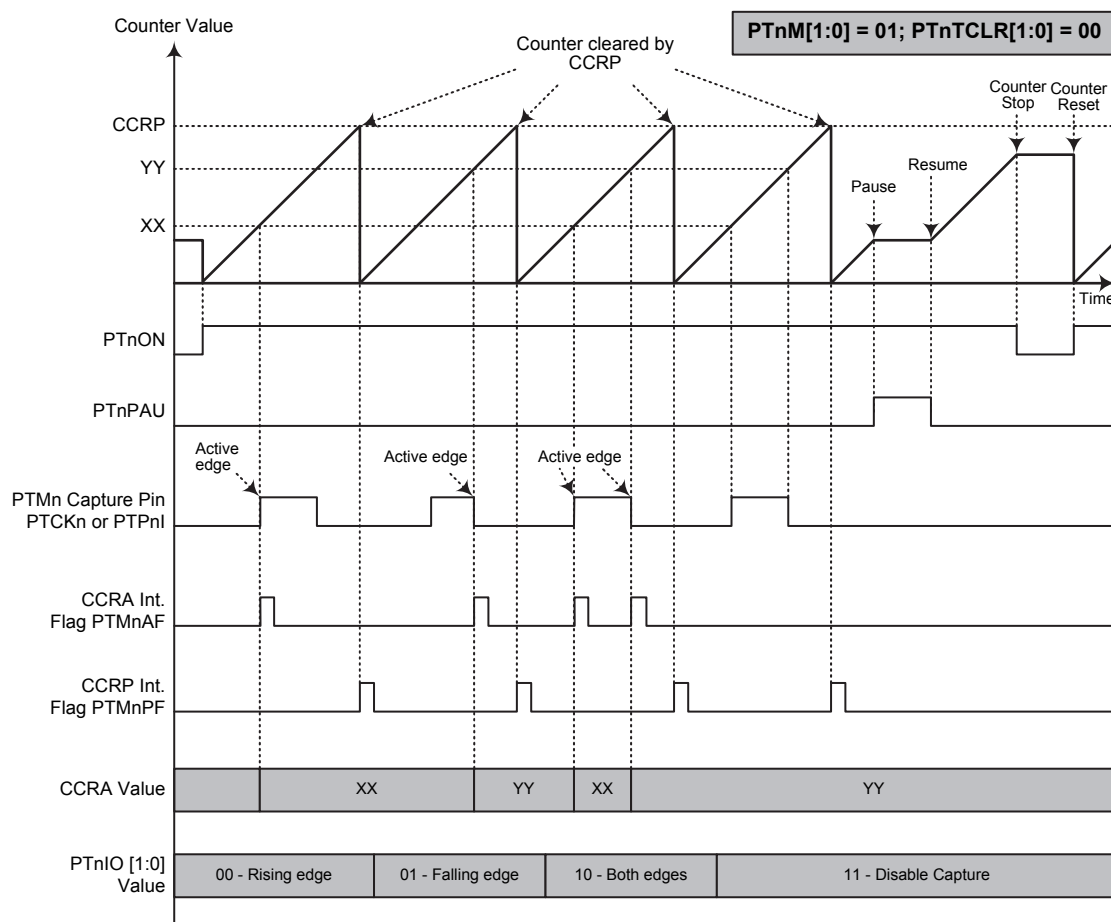
To select this mode bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied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

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

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

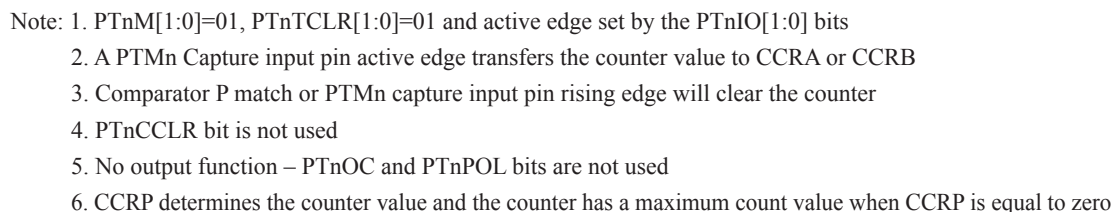
If the capture pulse width is less than two timer clock cycles, it may be ignored by hardware. The timer clock source must be equal to or less than 50MHz, otherwise the counter may fail to count.

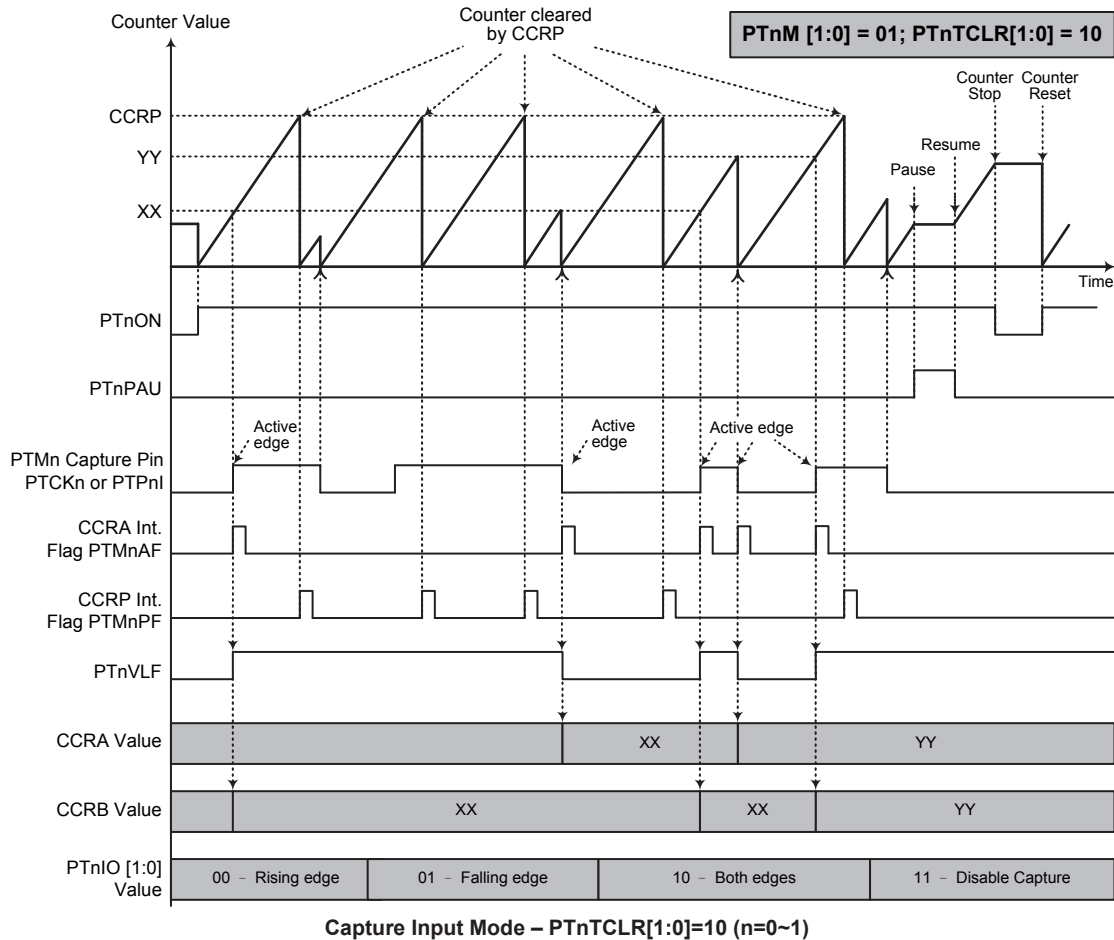
As the PTPnI or PTCKn pin is pin-shared with other functions, care must be taken if the PTMn is in the Capture Input Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The PTnCCLR, PTnOC and PTnPOL bits are not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



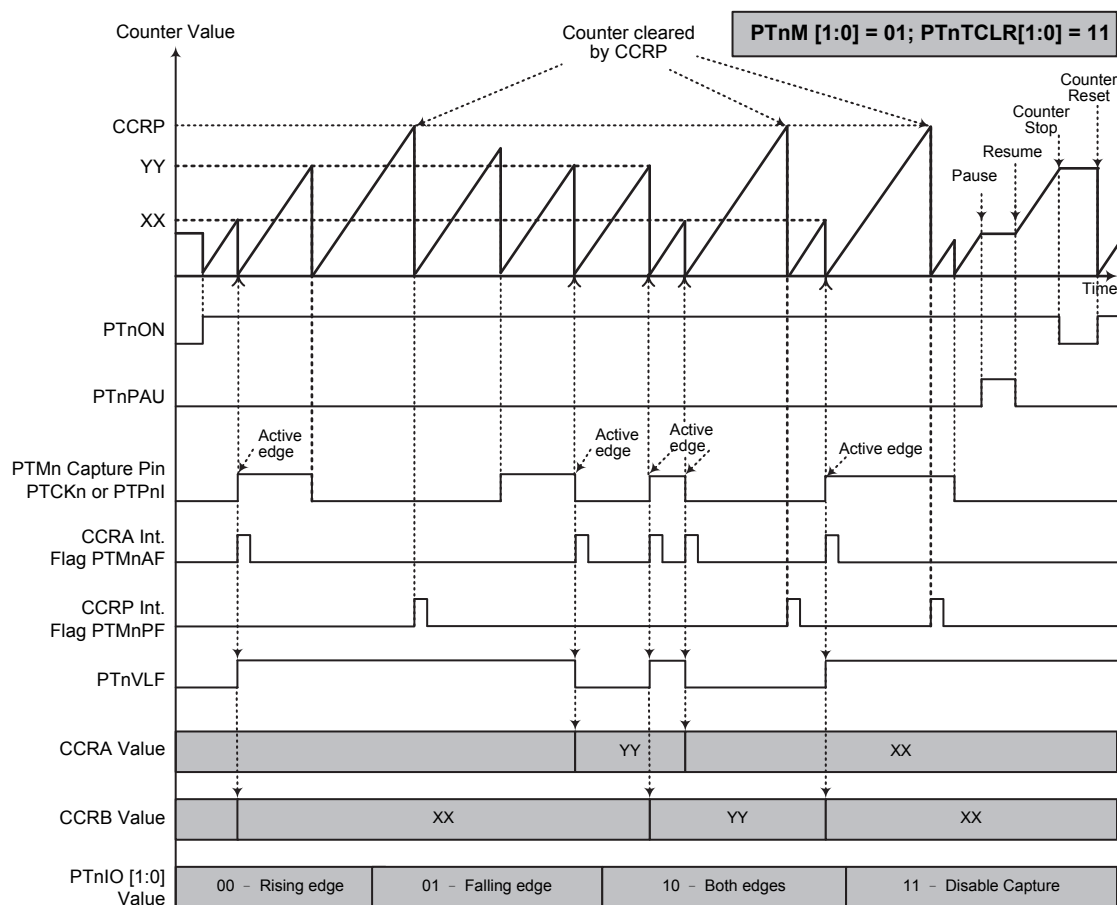
**Capture Input Mode – PTnTCLR[1:0]=00 (n=0~1)**

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





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



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

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

## Analog to Digital Converter

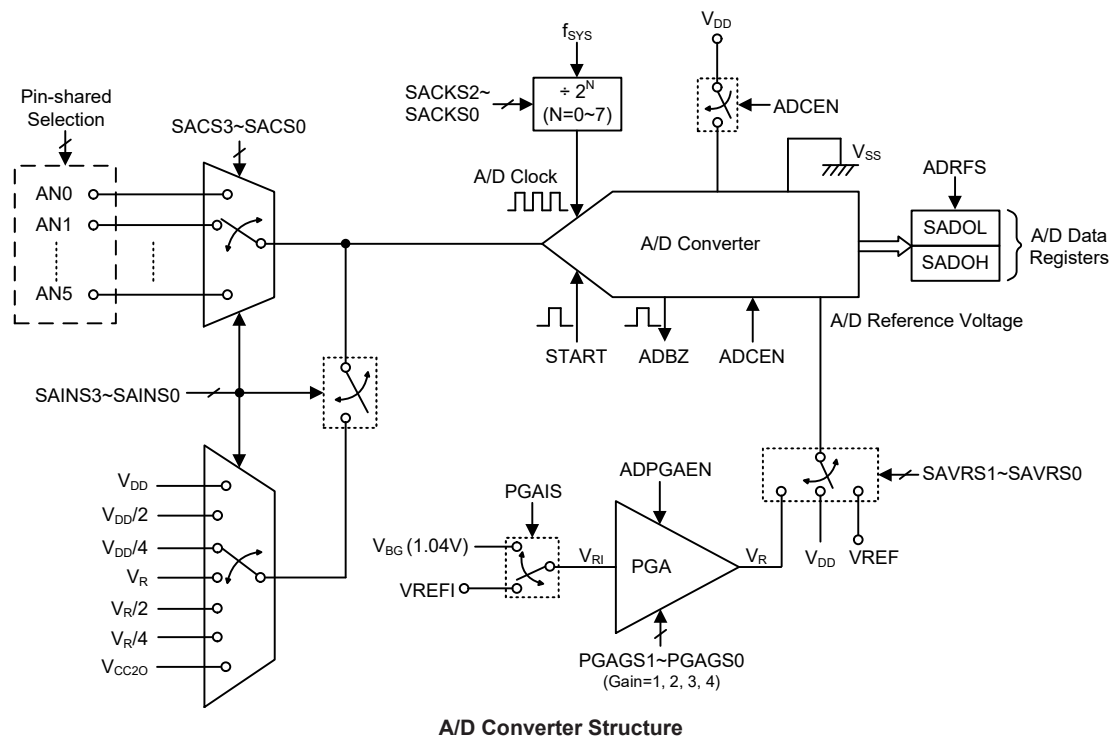
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

### A/D Converter Overview

This device contains a multi-channel analog to digital converter which can directly interface to external analog signals (such as that from sensors or other control signals) or internal analog signals and convert these signals directly into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS3~SAINS0 bits together with the SACS3~SACS0 bits. Note that when the internal analog signal is to be converted, the selected external input channel will be automatically disconnected to avoid malfunction. More detailed information about the A/D input signal is described in the “A/D Converter Control Registers” and “A/D Converter Input Signals” sections respectively.

External Input Channels	Internal Input Signals	A/D Channel Select Bits
AN0~AN5	$V_{DD}$ , $V_{DD}/2$ , $V_{DD}/4$ , $V_R$ , $V_R/2$ , $V_R/4$ , $V_{CC2O}$	SAINS3~SAINS0, SACS3~SACS0

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



## A/D Converter Register Description

Overall operation of the A/D converter is controlled using five registers. A read only register pair exists to store the A/D converter data 12-bit value. The remaining three 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 (ADRF5=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRF5=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRF5=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRF5=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRF5	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
SADC2	ADPGAEN	—	D5	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0

**A/D Converter Register List**

## A/D Converter Data Registers – SADOL, SADOH

As the device contains an internal 12-bit A/D converter, it requires two data registers to store the converted value. These are a high byte register, known as SADOH, and a low byte register, known as SADOL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 12 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRF5 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 unchanged if the A/D converter is disabled.

ADRF5	SADOH								SADOL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

**A/D Data Registers**

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

To control the function and operation of the A/D converter, several control registers known as SADC0, SADC1 and SADC2 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. As the device contains only one actual analog to digital converter hardware circuit, each of the external or internal analog signal inputs must be routed to the converter. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS3~SAINS0 bits in the SADC1 register are used to determine if the analog signal to be converted comes from an internal analog signal or from an external analog channel input.



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

• **SADC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	START	ADBZ	ADCEN	ADRF5	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 A/D conversion  
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: A/D conversion ended or no conversion  
1: A/D is busy  
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 as 0 after the A/D conversion is complete.
- Bit 5      **ADCEN**: A/D converter enable/disable control  
0: Disable  
1: Enable  
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is 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 known as SADOH and SADOL will be unchanged.
- Bit 4      **ADRF5**: A/D output data format selection bit  
0: ADC output data format → SADOH=D[11:4]; SADOL=D[3:0]  
1: ADC output data format → SADOH=D[11:8]; SADOL=D[7:0]  
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D converter data register section.
- Bit 3~0    **SACS3~SACS0**: A/D converter external analog input channel selection  
0000: AN0  
0001: AN1  
0010: AN2  
0011: AN3  
0100: AN4  
0101: AN5  
0110~1111: Non-existed channel, the input will be floating if selected

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	0	0	0	0	—	0	0	0

Bit 7~4     **SAINS3~SAINS0**: Internal A/D converter input channel selection bit  
 0000: External input – External analog channel input  
 0001: Internal input – Internal A/D converter power supply voltage  $V_{DD}$   
 0010: Internal input – Internal A/D converter power supply voltage  $V_{DD}/2$   
 0011: Internal input – Internal A/D converter power supply voltage  $V_{DD}/4$   
 0100: External input – External analog channel input  
 0101: Internal input – Internal A/D converter PGA output voltage  $V_R$   
 0110: Internal input – Internal A/D converter PGA output voltage  $V_R/2$   
 0111: Internal input – Internal A/D converter PGA output voltage  $V_R/4$   
 1000: Internal input – Internal LDO resistor divider output voltage  $V_{CC20}$   
 1001~1011: Reserved, connected to ground  
 1100~1111: External signal – External analog channel input

When an internal analog signal is selected to be converted, the external channel input signal will automatically be switched off regardless of the SACKS3~SACKS0 bit field value.

Bit 3     Unimplemented, read as “0”

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

• **SADC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADPGAEN	—	D5	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0
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     **ADPGAEN**: A/D converter PGA enable/disable control  
 0: Disable  
 1: Enable

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

Bit 6     Unimplemented, read as “0”

Bit 5     **D5**: Reserved, cannot be used and must be fixed at 0

Bit 4     **PGAIS**: PGA input ( $V_{RI}$ ) selection  
 0:  $V_{RI}$  comes from VREFI pin  
 1:  $V_{RI}$  comes from  $V_{BG}$

When the internal independent reference voltage  $V_{BG}$  is selected as the PGA input, the external reference voltage on the VREFI pin will be automatically switched off. When this bit is set high to select  $V_{BG}$  as PGA input, the internal bandgap reference  $V_{BG}$  should be enabled by setting the VBGGEN bit in the LVDC register to “1”.

- Bit 3~2     **SAVRS1~SAVRS0**: A/D converter reference voltage selection  
               00: ADC reference voltage comes from  $V_{DD}$   
               01: ADC reference voltage comes from VREF pin  
               1x: ADC reference voltage comes from  $V_R$  (PGA output)  
               When the internal A/D converter power or the internal PGA output voltage is selected as the reference voltage, the hardware will automatically disconnect the external VREF input.
- Bit 1~0     **PGAGS1~PGAGS0**: PGA gain selection bits  
               00: Gain=1  
               01: Gain=2  
               10: Gain=3  
               11: Gain=4

## A/D Converter Operation

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

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in process 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 appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the 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~SACKS0 in the SADC1, there are some limitations on the A/D clock source speed that can be selected. As the recommended value of permissible A/D clock period,  $t_{ADCK}$ , is from 0.5 $\mu$ s to 10 $\mu$ s, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the SACKS2~SACKS0 bits should not be set to 000B or 11xB. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk \* special care must be taken.

$f_{SYS}$	A/D Clock Period ( $t_{ADCK}$ )							
	<b>SACKS2, SACKS1, SACKS0 =000 (<math>f_{SYS}</math>)</b>	<b>SACKS2, SACKS1, SACKS0 =001 (<math>f_{SYS}/2</math>)</b>	<b>SACKS2, SACKS1, SACKS0 =010 (<math>f_{SYS}/4</math>)</b>	<b>SACKS2, SACKS1, SACKS0 =011 (<math>f_{SYS}/8</math>)</b>	<b>SACKS2, SACKS1, SACKS0 =100 (<math>f_{SYS}/16</math>)</b>	<b>SACKS2, SACKS1, SACKS0 =101 (<math>f_{SYS}/32</math>)</b>	<b>SACKS2, SACKS1, SACKS0 =110 (<math>f_{SYS}/64</math>)</b>	<b>SACKS2, SACKS1, SACKS0 =111 (<math>f_{SYS}/128</math>)</b>
1MHz	1 $\mu$ s	2 $\mu$ s	4 $\mu$ s	8 $\mu$ s	16 $\mu$ s*	32 $\mu$ s*	64 $\mu$ s*	128 $\mu$ s*
2MHz	500ns	1 $\mu$ s	2 $\mu$ s	4 $\mu$ s	8 $\mu$ s	16 $\mu$ s*	32 $\mu$ s*	64 $\mu$ s*
4MHz	250ns*	500ns	1 $\mu$ s	2 $\mu$ s	4 $\mu$ s	8 $\mu$ s	16 $\mu$ s*	32 $\mu$ s*
8MHz	125ns*	250ns*	500ns	1 $\mu$ s	2 $\mu$ s	4 $\mu$ s	8 $\mu$ s	16 $\mu$ s*

**A/D Clock Period Examples**

Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs by configuring the corresponding pin-shared control bits, if

the ADCEN bit is high then some power will still be consumed. In power conscious applications it is therefore recommended that the ADCEN is set low to reduce power consumption when the A/D converter function is not being used.

### A/D Converter Reference Voltage

The reference voltage supply to the A/D Converter can be supplied from either the positive power supply,  $V_{DD}$ , or from an external reference sources supplied on pin VREF or from PGA output voltage  $V_R$ . The desired selection is made using the SAVRS1~SAVRS0 bits in the SADC2 register. When the SAVRS bit field is set to “00”, the A/D converter reference voltage will come from  $V_{DD}$ . If the SAVRS bit field is set to “01”, the A/D converter reference voltage will come from the VREF pin. If the SAVRS bit field is set to “1x”, the A/D converter reference voltage will come from  $V_R$ . 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. When  $V_R$  is selected as A/D converter reference voltage, the PGA needs to be enabled by setting the ADPGAEN bit in the SADC2 register to 1. If the program selects the internal reference voltage  $V_{DD}$  or  $V_R$  as the A/D converter reference voltage, the hardware will automatically disconnect the external reference voltage input.

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

SAVRS[1:0]	Reference Source	Description
00	$V_{DD}$	Internal A/D converter power supply
01	VREF Pin	PGA external input pin VREF
10 or 11	$V_R$	Internal A/D converter PGA output voltage

**A/D Converter Reference Voltage Selection**

### A/D Converter Input Signals

All of the A/D analog input pins are pin-shared with the I/O pins on Port A as well as other functions. The corresponding selection bits for each I/O pin in the PxPSn register, determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the pin-shared function control bits configure its corresponding pin as an A/D analog channel input, the pin will be setup to be an A/D converter external channel input and the original pin functions 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 PAC port control register to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

If the SAINS3~SAINS0 bits are set to “0000”, “0100” or “1100~1111”, the external analog channel input is selected to be converted and the SACS3~SACS0 bits can determine which actual external channel is selected to be converted. If the SAINS3~SAINS0 bits are set to “0001~0011”, the  $V_{DD}$  or its divided voltage is selected to be converted. If the SAINS3~SAINS0 bits are set to “0101~0111”, the PGA output  $V_R$  or its divided voltage is selected to be converted. If the SAINS3~SAINS0 bits are set to “1000”, the internal LDO resistor divider output voltage is selected to be converted.

The PGA input can be configured to come from the VREFI pin or come from the internal bandgap reference voltage,  $V_{BG}$ , using the PGAIS bit in the SADC2 register.

Note that if the program selects an internal signal as the A/D converter input, the hardware will automatically disconnect the external analog input. The same hardware control method can be applied to the PGA input selection. If the application program selects the internal voltage  $V_{BG}$  as PGA input, then the hardware will automatically disconnect the external VREFI input.

SAINS[3:0]	SACS[3:0]	Input Signals	Description
0000, 0100, 1100~1111	0000~0101	AN0~AN5	External channel analog input
	0010~1111	—	Floating
0001	xxxx	V <sub>DD</sub>	Internal A/D converter power supply voltage
0010	xxxx	V <sub>DD</sub> /2	Internal A/D converter power supply voltage/2
0011	xxxx	V <sub>DD</sub> /4	Internal A/D converter power supply voltage/4
0101	xxxx	V <sub>R</sub>	Internal A/D converter PGA output voltage
0110	xxxx	V <sub>R</sub> /2	Internal A/D converter PGA output/2
0111	xxxx	V <sub>R</sub> /4	Internal A/D converter PGA output/4
1000	xxxx	V <sub>CC2O</sub>	Internal LDO resistor divider output
1001~1011	xxxx	Ground	Unused

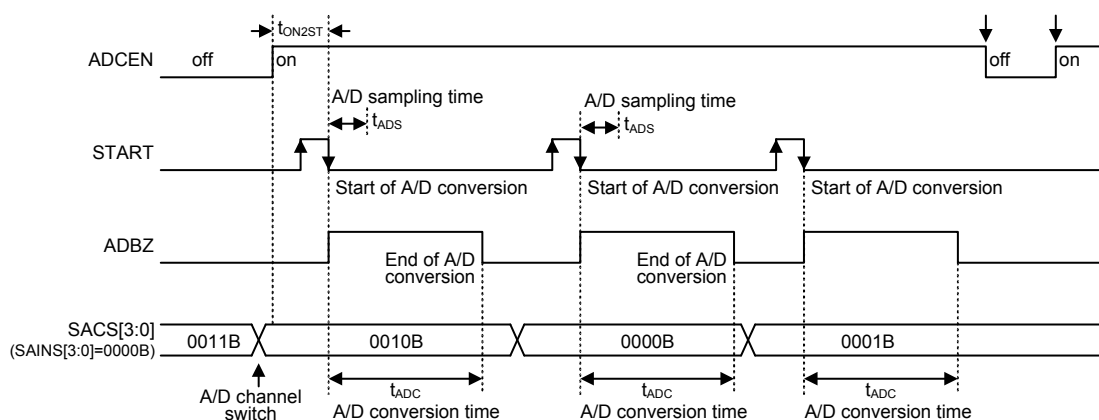
**A/D Converter Input Signal Selection**

## 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.

$$\text{Maximum single A/D conversion rate} = \text{A/D clock period} / 16$$

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 – External Channel Input**

## Summary of A/D Conversion Steps

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

- Step 1  
Select the required A/D conversion clock by properly programming the SACKS2~SACKS0 bits in the SADC1 register.
- Step 2  
Enable the A/D converter by setting the ADCEN bit in the SADC0 register to 1.

- Step 3  
Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS3~SAINS0 bits in the SADC1 register.  
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 is selected to come from the external channel input by configuring the SAINS bit field, the corresponding pin should first be configured as A/D input function by configuring the relevant pin-shared 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  
If the A/D input signal comes from the internal analog signal, the SAINS bit field should be properly configured and then the external channel input will automatically be disconnected regardless of the SACS bit field value. After this step, go to Step 6.
- Step 6  
Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC2 register. If the reference voltage is selected to come from the PGA output, then enable the PGA and select the PGA input by configuring the PGAIS bit in the SADC2 register.
- Step 7  
Select A/D converter output data format by configuring the ADRFS bit in the SADC0 register.
- Step 8  
If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, the A/D converter interrupt control bit, ADE, and the related multi-function interrupt control bit, MFnE, must all be set high in advance.
- Step 9  
The A/D convert procedure can now be initialised 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 A/D conversion process is completed, the ADBZ flag will go low and then output data can be read from the SADOH and SADOL registers. If the ADC interrupt is enabled and the stack is not full, data can be acquired by interrupt service program.

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 clearing the ADCEN bit 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 Conversion Function

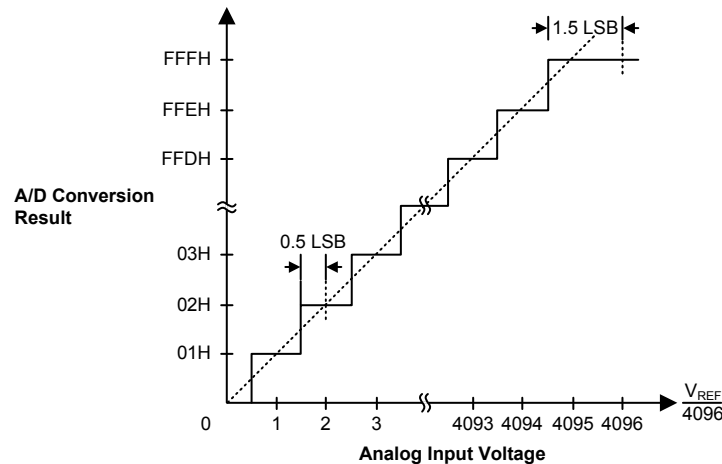
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage,  $V_{REF}$ , this gives a single bit analog input value of  $V_{REF}$  divided by 4096.

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

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

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

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



## A/D Conversion Programming Examples

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the 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 A/D converter interrupt
mov a,03H
mov SADC1,a            ; select fsys/8 as A/D clock
set ADCEN
mov a,20h              ; setup PBPS0 to configure pin AN0
mov PBPS0,a
mov a,20h
mov SADC0,a            ; enable A/D converter 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 SADOL_buffer,a ; save result to user defined register
mov a,SADOH      ; read high byte conversion result value
mov SADOH_buffer,a ; save result to user defined register
:
:
jmp start_conversion ; start next A/D conversion

```

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

```

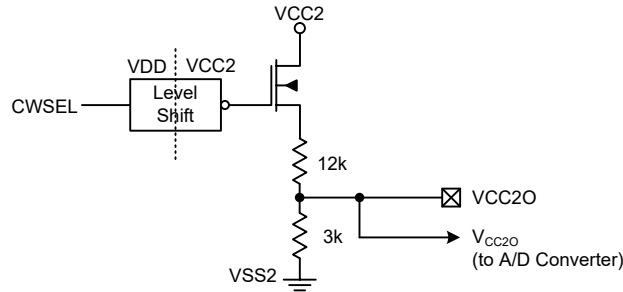
clr ADE          ; disable A/D converter interrupt
mov a,03H
mov SADC1,a      ; select fsys/8 as A/D clock
set ADCEN
mov a,20h        ; setup PBPS0 to configure pin AN0
mov PBPS0,a
mov a,20h
mov SADC0,a      ; enable A/D converter and connect AN0 channel to A/D converter
Start_conversion:
clr START        ; 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
clr MF1F         ; clear multi-function interrupt 1 request flag
set ADE          ; enable A/D converter interrupt
set MF1E         ; enable multi-function interrupt 1 interrupt
set EMI          ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov acc_stack,a  ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a ; save STATUS to user defined memory
:
:
mov a,SADOL      ; read low byte conversion result value
mov SADOL_buffer,a ; save result to user defined register
mov a,SADOH      ; read high byte conversion result value
mov SADOH_buffer,a ; save result to user defined register
:
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a     ; restore STATUS from user defined memory
mov a,acc_stack  ; restore ACC from user defined memory
reti

```



## LDO Divider Circuit

The device contains an internal LDO which outputs 5V voltage for internal or external power supply. Its internal resistor divider output value can be read by connecting it to the A/D converter internal input channel.

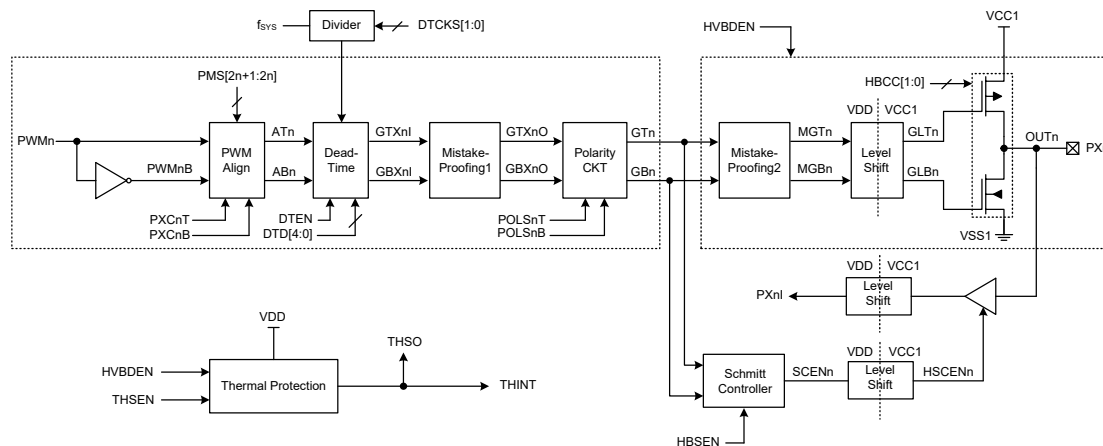


Note: When the A/D converter selects the  $V_{CC2O}$  signal as its internal input,  $CWSEL=0$ , otherwise  $CWSEL=1$ .

**LDO Divider Circuit**

## H-Bridge Driver

The device provides an H-bridge driver which is mainly composed of two sections, complementary output control circuits and high voltage output driver circuits.

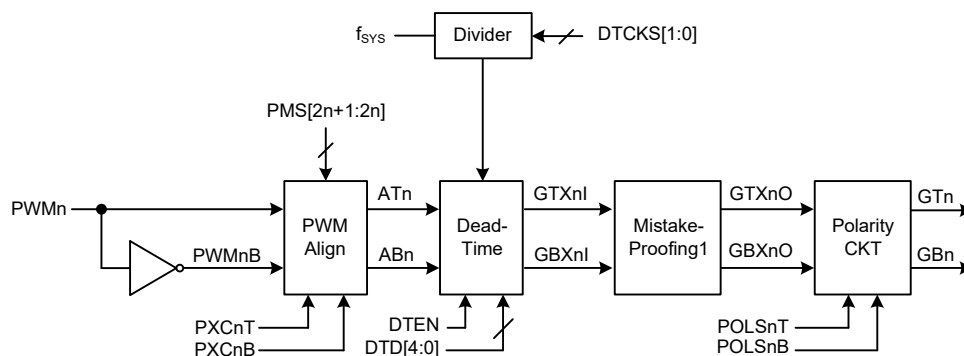


Note: PWMn stand for the PWM output signals from PTMn. PWMnB is the inversion of PWMn.

**H-Bridge Driver Structure (n=0~1)**

## Complementary Output Control

There are two groups of complementary output control circuits. Each group is composed of four circuits, PWM align circuit, dead-time circuit, mistake-proofing circuit and polarity control circuit.



Note: PWMn stand for the PWM output signals from PTMn. PWMnB is the inversion of PWMn.

### Complementary Output Control (n=0~1)

## Complementary Output Control Registers

The complementary output control is implemented using several registers. These registers are used to select the PWM align mode, setup the dead-time and control the output polarity, etc.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PMS	—	—	—	—	PMS3	PMS2	PMS1	PMS0
PXC	—	—	—	—	PXC1T	PXC1B	PXC0T	PXC0B
DTC	DTCKS1	DTCKS0	DTEN	DTD4	DTD3	DTD2	DTD1	DTD0
POLS	—	—	—	—	POLS1T	POLS1B	POLS0T	POLS0B

Complementary Output Control Register List

### • PMS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PMS3	PMS2	PMS1	PMS0
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 **PMS3~PMS2**: Select the PWM align mode of OUT1 high voltage level shift driver  
 00: Complementary PWM signal pair  
 01: High side non-complementary PWM signal  
 10: Low side non-complementary PWM signal  
 11: OUT1 control (PXC1T and PXC1B control High and Low sides respectively)

Bit 1~0 **PMS1~PMS0**: Select the PWM align mode of OUT0 high voltage level shift driver  
 00: Complementary PWM signal pair  
 01: High side non-complementary PWM signal  
 10: Low side non-complementary PWM signal  
 11: OUT0 control (PXC0T and PXC0B control High and Low sides respectively)

• PXC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PXC1T	PXC1B	PXC0T	PXC0B
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 **PXC1T~PXC1B**: Control the High/Low outputs of the OUT1 high voltage level shift driver  
 00: High/Low sides both turn off  
 01: High side turn off/Low side turn on, output low  
 10: High side turn on/Low side turn off, output high  
 11: High/Low sides both turn off (High/Low sides being both turned on is forbidden)  
 When the PMS3~PMS2 bits in the PMS register are set to “11”, the High/Low outputs of the OUT1 high voltage level shift driver are controlled by the PXC1T~PXC1B bits.

Bit 1~0 **PXC0T~PXC0B**: Control the High/Low outputs of the OUT0 high voltage level shift driver  
 00: High/Low sides both turn off  
 01: High side turn off/Low side turn on, output low  
 10: High side turn on/Low side turn off, output high  
 11: High/Low sides both turn off (High/Low sides being both turned on is forbidden)  
 When the PMS1~PMS0 bits in the PMS register are set to “11”, the High/Low outputs of the OUT0 high voltage level shift driver are controlled by the PXC0T~PXC0B bits.

• DTC Register

Bit	7	6	5	4	3	2	1	0
Name	DTCKS1	DTCKS0	DTEN	DTD4	DTD3	DTD2	DTD1	DTD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **DTCKS1~DTCKS0**: Dead-time clock source ( $f_{DT}$ ) selection  
 00:  $f_{DT}=f_{SYS}$   
 01:  $f_{DT}=f_{SYS}/2$   
 10:  $f_{DT}=f_{SYS}/4$   
 11:  $f_{DT}=f_{SYS}/8$

Bit 5 **DTEN**: Dead-time enable/disable control  
 0: Disable  
 1: Enable  
 If this bit is set high to enable the dead-time insertion, the dead-time is furtherly controlled by the DTD4~DTD0 bits.

Bit 4~0 **DTD4~DTD0**: Dead-time counter  
 $Dead-time=(DTD[4:0]+1)/f_{DT}$

• POLS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	POLS1T	POLS1B	POLS0T	POLS0B
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as “0”

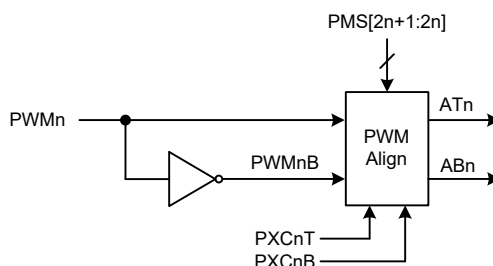
Bit 3 **POLS1T**: Select the polarity of High side of the OUT1 high voltage level shift driver  
 0: Non-invert  
 1: Invert

Bit 2 **POLS1B**: Select the polarity of Low side of the OUT1 high voltage level shift driver  
 0: Non-invert  
 1: Invert

- Bit 1     **POLS0T**: Select the polarity of High side of the OUT0 high voltage level shift driver  
0: Non-invert  
1: Invert
- Bit 0     **POLS0B**: Select the polarity of Low side of the OUT0 high voltage level shift driver  
0: Non-invert  
1: Invert

### PWM Align

For PWM signal generation, users can decide whether to have single side PWM signal or complementary PWM signal pair or use software control bits to drive the H-Bridge driver. The driving signals are selected using the associated bits in the PMS and PXC registers, as shown in the following figure and table.



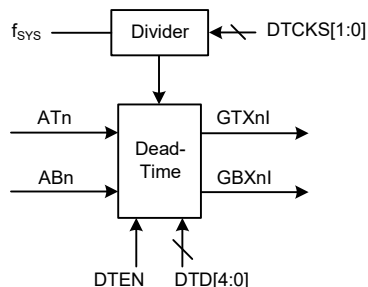
Note:  $PWM_n$  stand for the PWM output signals from  $PTM_n$ .  $PWM_nB$  is the inversion of  $PWM_n$ .

**PWM Align Block Diagram (n=0~1)**

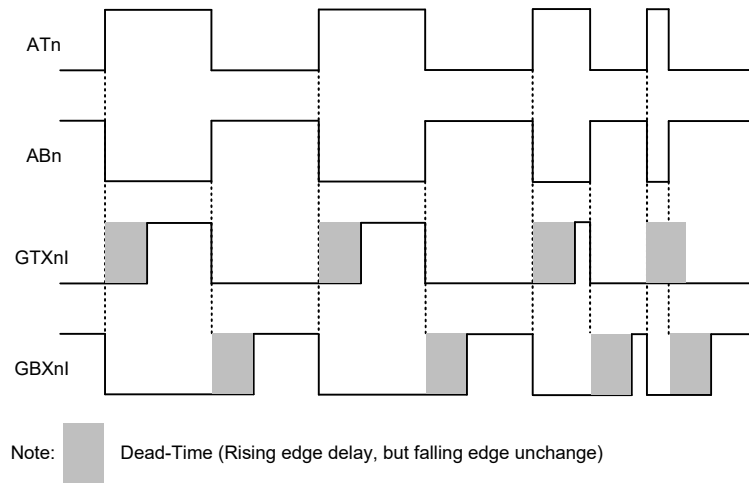
$PMS[2n+1:2n]$	PWM Align Mode	$AT_n$	$AB_n$
00	Complementary PWM signal pair	$PWM_n$	$PWM_nB$
01	High side non-complementary PWM signal	$PWM_n$	0
10	Low side non-complementary PWM signal	0	$PWM_n$
11	PXC register controls High/Low outputs	$PXC_nT$	$PXC_nB$

### Dead-Time

During the transition of the external driving transistors, there may be a moment when both the high and low sides are turned on, which will result in short-circuit. To avoid this situation, a dead-time can be inserted. The enable or disable function of the dead-time is controlled by the DTEN bit of the DTC register. The dead-time should be configured in the range of  $0.3\mu s \sim 5\mu s$ . Its clock source is selected using the DTCKS1~DTCKS0 bits and its duration is determined by the DTD4~DTD0 bits. The following shows the dead-time block diagram and dead-time insertion timing. Note that after the dead-time function is enabled, it is inserted at each rising edge only, the falling edges remain unchanged.



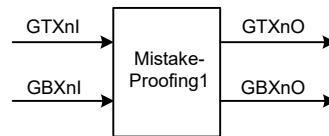
**Dead-Time Block Diagram (n=0~1)**



**Dead-Time Insertion Timing**

### Mistake-Proofing for Complementary Control

Incorrect write operations or external factors such as an ESD condition, may cause incorrect on/off control resulting in the high and low sides of external transistors being both turned on. A mistake-proofing circuit is designed to avoid such situation by forcing both output MOS off to protect the motor.



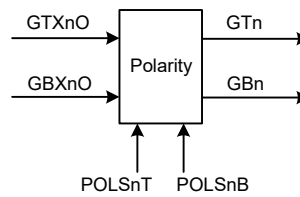
**Mistake-Proofing Circuit (n=0~1)**

GTXnl	GBXnl	GTXnO	GBXnO
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

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

### Polarity Control

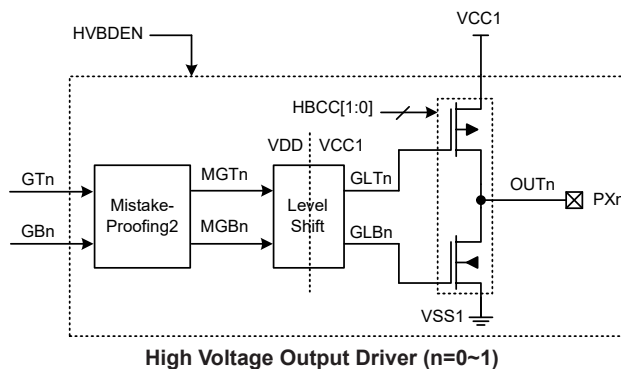
The external MOS gate outputs polarity are controlled by the POLSnT and POLSnB bits in the POLS register.



**Polarity Control (n=0~1)**

## High Voltage Output Driver

There are two groups of high voltage output driver circuits. Each group is mainly composed of a mistake-proofing circuit and a 12V high voltage process level shift circuit. Two high voltage lines are connected in parallel to a final high voltage output to provide DC motor driving signal. Use one or two high voltage lines according to different current requirements to support multiple external driving circuits. The high voltage output driver enable or disable function is controlled by the HVBDEN bit in the HVC register.



## High Voltage Output Driver Register

The overall operation of the high voltage output driver is controlled using the HVC register. The register controls the high voltage output driver enable and disable functions, thermal protection and driving current selection.

### • HVC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	HBSEN	HVBDEN	THSEN	THSO	HBCC1	HBCC0
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 **HBSEN**: Schmitt Controller enable/disable control  
Described in the High Voltage Output Read Back Circuit section

Bit 4 **HVBDEN**: High voltage output driver turn on/off control  
0: Turn off the whole high voltage output driver circuits  
1: Turn on the whole high voltage output driver circuits

Bit 3 **THSEN**: Thermal protection function enable/disable control  
0: Disable  
1: Enable

To disable the thermal protection function, clear this bit to 0. If the high voltage output driver circuits also need to be turned off, clear the THSEN bit first, then clear the HVBDEN bit. To enable the thermal protection function, set the THSEN bit high first, then set the HVBDEN bit high.

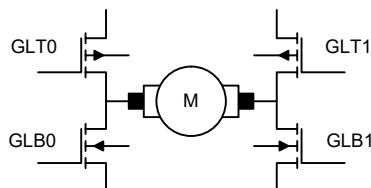
Bit 2 **THSO**: Thermal protection flag  
0: No over thermal condition occurs  
1: Over thermal condition occurs

Bit 1~0 **HBCC1~HBCC0**: H-bridge current control  
00: H-bridge disable  
01: 400mA  
10: 400mA  
11: 400mA + 400mA

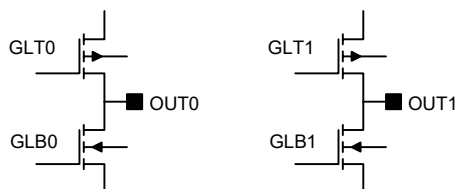
### High Voltage Output Driver Applications

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

- H-Bridge Group: directly drive DC Motor

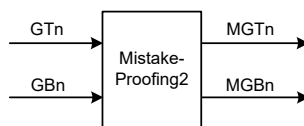


- PMOS/NMOS are used independently



### Mistake-Proofing for High Voltage Output Driver

Incorrect write operations or external factors such as an ESD condition, may cause incorrect on/off control resulting in the high and low sides of external transistor being both turned on. A mistake-proofing circuit is designed to avoid such situation.



**Mistake-Proofing Circuit (n=0~1)**

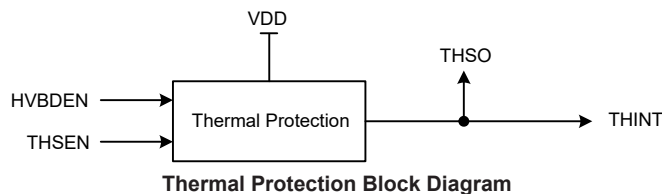
GTn	GBn	MGTn	MGBn
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	1

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

### Thermal Protection for High Voltage Output Driver

There is a thermal protection for the high voltage output driver. The on/off function of the thermal protection is controlled using the THSEN bit. To avoid abnormal thermal protection, when enabling the protect function, first set the THSEN bit to 1, then set the HVB DEN bit to 1. When disabling the high voltage output driver circuits, first clear the THSEN bit, then clear the HVB DEN bit.

If the thermal protection has been enabled, the THSO bit in the HVC register can be used to check whether an over thermal condition has occurred. When the temperature exceeds the preset range, the THSO bit will change from 0 to 1 to indicate an over thermal occurrence. Additionally, the thermal protection flag in the interrupt control register will also be set high, if the corresponding interrupt has been enabled, a thermal protection interrupt will be generated to inform the MCU.



## High Voltage Output Read Back Circuit

The high voltage output read back circuit is composed of a Schmitt trigger and a level shifter. The actual PXn output status can be read back using the PXnI bit in the HBC register to be compared with the GLTn and GLBn driving signals. When GLTn and GLBn are turned off the PXn pin will be floating, which will result in current leakage on the Schmitt trigger and the level shifter. A Schmitt controller is designed to solve this problem. When the Schmitt controller is disabled, the Schmitt trigger will be always enabled, in which case the aforementioned condition may occur. However, by enabling the Schmitt controller, the Schmitt trigger can be turned off to avoid current leakage when GLTn and GLBn are turned off.

### • HVC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	HBS EN	HVBD EN	THSEN	THSO	HBCC1	HBCC0
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 **HBS EN**: Schmitt Controller enable/disable control

0: Disable – Schmitt trigger is always enabled

1: Enable – Schmitt trigger is controlled by hardware logic combination

To avoid the un-desired condition that may occur when the Schmitt trigger is always enabled, it is recommended to enable the Schmitt controller by setting the HBS EN bit high.

Bit 4 **HVBD EN**: High voltage output driver turn on/off control

Described in the High Voltage Output Driver Register section

Bit 3 **THSEN**: Thermal protection function enable/disable control

Described in the High Voltage Output Driver Register section

Bit 2 **THSO**: Thermal protection flag

Described in the High Voltage Output Driver Register section

Bit 1~0 **HBCC1~HBCC0**: H-bridge current control

Described in the High Voltage Output Driver Register section

### • HBC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PX1I	PX0I
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	x	x

“x”: unknown

Bit 7~2 Unimplemented, read as “0”

Bit 1 **PX1I**: PX1 high voltage output status read back signal

0: Low

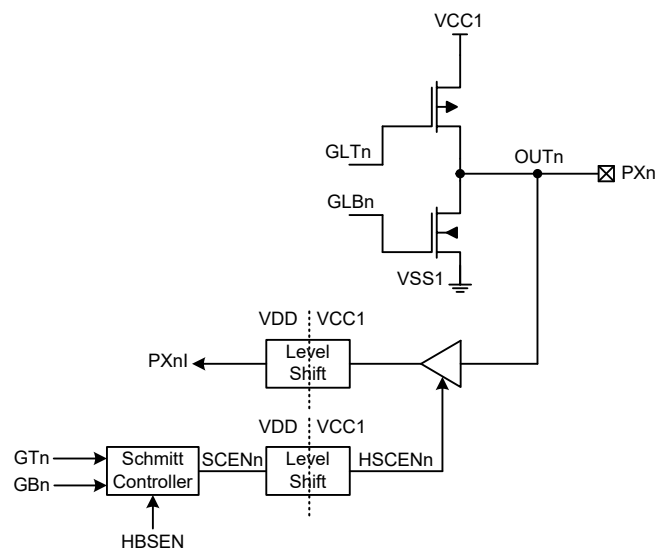
1: High

Bit 0 **PX0I**: PX0 high voltage output status read back signal

0: Low

1: High





**High Voltage Output Read Back Circuit (n=0~1)**

As shown in the above diagram, the Schmitt controller inputs come from the polarity control circuit outputs, GTn and GBn. The Schmitt controller is controlled by the HBSn bit in the HVC register. The SCENn signal enters the level shifter and generates the HSCENn signal, which is used to control the Schmitt trigger. The relationship between the Schmitt controller inputs and output when the Schmitt controller is enabled/disabled is shown in the following table.

HBSn	GTn	GBn	SCENn (HSCENn)
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	x	x	1

"x": Don't care

## Low Voltage Detector – LVD

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

### LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of eight fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V<sub>DD</sub> 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	LV DEN	VBGEN	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5 **LVDO**: LVD Output Flag  
 0: No Low Voltage Detect  
 1: Low Voltage Detect

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

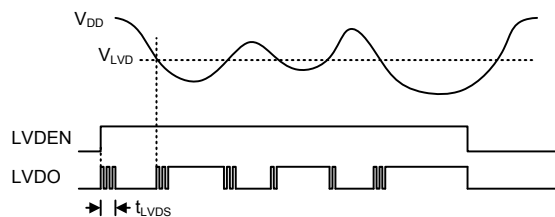
Bit 3 **VBGEN**: Bandgap Buffer Control  
 0: Disable  
 1: Enable

Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VBGEN bit is set to 1.

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. When the device is in the SLEEP mode, the low voltage detector will be disabled even if the LV DEN bit is high. After enabling the Low Voltage Detector, a time delay  $t_{LVDS}$  should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the  $V_{DD}$  voltage may rise and fall rather slowly, at the voltage nears that of  $V_{LVD}$ , there may be multiple bit LVDO transitions.


**LVD Operation**

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

## Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains external and internal interrupts functions. The external interrupt is generated by the action of the external INT0~INT3 pins, while the internal interrupts are generated by various internal functions such as the Timer Modules, Time Bases, EEPROM, A/D converter and High Voltage Output Driver Thermal Protection.

### Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The registers fall into three categories. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0~MFI1 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	—	—
Thermal Protection	THE	THF	—
LVD	LVE	LVF	—
INTn Pin	INTnE	INTnF	n=0~3
Multi-function	MFnE	MFnF	n=0~1
Time Bases	TBnE	TBnF	n=0~1
EEPROM write operation	DEE	DEF	—
A/D Converter	ADE	ADF	—
PTM	PTMnAF	PTMnAE	n=0~1
	PTMnPF	PTMnPE	

**Interrupt Register Bit Naming Conventions**

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	INT0F	LVF	THF	INT0E	LVE	THE	EMI
INTC1	MF0F	INT3F	INT2F	INT1F	MF0E	INT3E	INT2E	INT1E
INTC2	DEF	TB1F	TB0F	MF1F	DEE	TB1E	TB0E	MF1E
MFI0	—	—	PTM0AF	PTM0PF	—	—	PTM0AE	PTM0PE
MFI1	—	ADF	PTM1AF	PTM1PF	—	ADE	PTM1AE	PTM1PE

**Interrupt Register List**

• **INTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
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      **INT3S1~INT3S0**: Interrupt Edge Control for INT3 Pin

00: Disable  
01: Rising edge  
10: Falling edge  
11: Rising and falling edges

Bit 5~4      **INT2S1~INT2S0**: Interrupt Edge Control for INT2 Pin

00: Disable  
01: Rising edge  
10: Falling edge  
11: Rising and falling edges

Bit 3~2      **INT1S1~INT1S0**: Interrupt Edge Control for INT1 Pin

00: Disable  
01: Rising edge  
10: Falling edge  
11: Rising and falling edges

Bit 1~0      **INT0S1~INT0S0**: Interrupt Edge Control for INT0 Pin

00: Disable  
01: Rising edge  
10: Falling edge  
11: Rising and falling edges

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	INT0F	LVF	THF	INT0E	LVE	THE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7      Unimplemented, read as “0”

Bit 6      **INT0F**: External Interrupt 0 Request Flag

0: No request  
1: Interrupt request

Bit 5      **LVF**: LVD Interrupt Request Flag

0: No request  
1: Interrupt request

Bit 4      **THF**: Thermal Protection Interrupt Request Flag

0: No request  
1: Interrupt request

Bit 3      **INT0E**: External Interrupt 0 Control

0: Disable  
1: Enable

Bit 2      **LVE**: LVD Interrupt Control

0: Disable  
1: Enable

Bit 1      **THE**: Thermal Protection Interrupt Control

0: Disable  
1: Enable

Bit 0      **EMI**: Global Interrupt Control

0: Disable  
1: Enable

• **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	MF0F	INT3F	INT2F	INT1F	MF0E	INT3E	INT2E	INT1E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **MF0F**: Multi-function 0 Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 6      **INT3F**: External Interrupt 3 Request Flag  
0: No request  
1: Interrupt request
- Bit 5      **INT2F**: External Interrupt 2 Request Flag  
0: No request  
1: Interrupt request
- Bit 4      **INT1F**: External Interrupt 1 Request Flag  
0: No request  
1: Interrupt request
- Bit 3      **MF0E**: Multi-function 0 Interrupt Control  
0: Disable  
1: Enable
- Bit 2      **INT3E**: External Interrupt 3 Control  
0: Disable  
1: Enable
- Bit 1      **INT2E**: External Interrupt 2 Control  
0: Disable  
1: Enable
- Bit 0      **INT1E**: External Interrupt 1 Control  
0: Disable  
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	DEF	TB1F	TB0F	MF1F	DEE	TB1E	TB0E	MF1E
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      **DEF**: Data EEPROM Interrupt 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      **MF1F**: Multi-function 1 Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 3      **DEE**: Data EEPROM Interrupt 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      **MF1E**: Multi-function 1 Interrupt Control  
0: Disable  
1: Enable

• **MF10 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM0AF	PTM0PF	—	—	PTM0AE	PTM0PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6      Unimplemented, read as “0”
- Bit 5      **PTM0AF**: PTM0 CCRA Comparator Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 4      **PTM0PF**: PTM0 CCRP Comparator Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 3~2      Unimplemented, read as “0”
- Bit 1      **PTM0AE**: PTM0 CCRA Comparator Interrupt Control  
0: Disable  
1: Enable
- Bit 0      **PTM0PE**: PTM0 CCRP Comparator Interrupt Control  
0: Disable  
1: Enable

• **MF11 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	ADF	PTM1AF	PTM1PF	—	ADE	PTM1AE	PTM1PE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

- Bit 7      Unimplemented, read as “0”
- Bit 6      **ADF**: A/D Converter Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 5      **PTM1AF**: PTM1 Comparator A Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 4      **PTM1PF**: PTM1 Comparator P Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 3      Unimplemented, read as “0”
- Bit 2      **ADE**: A/D Converter Interrupt Control  
0: Disable  
1: Enable

Bit 1	<b>PTM1AE</b> : PTM1 Comparator A Interrupt Control 0: Disable 1: Enable
Bit 0	<b>PTM1PE</b> : PTM1 Comparator P Interrupt Control 0: Disable 1: Enable

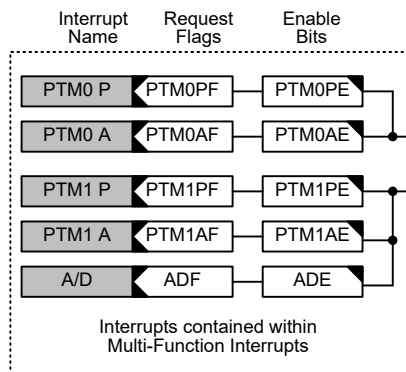
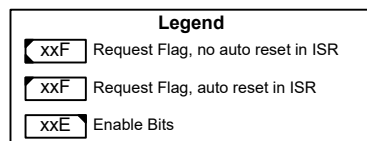
### Interrupt Operation

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 Name	Request Flags	Enable Bits	Master Enable	Vector	Priority
Thermal Protection	THF	THE	EMI	04H	High
LVD	LVF	LVE	EMI	08H	
INT0 Pin	INT0F	INT0E	EMI	0CH	
INT1 Pin	INT1F	INT1E	EMI	10H	
INT2 Pin	INT2F	INT2E	EMI	14H	
INT3 Pin	INT3F	INT3E	EMI	18H	
M. Funct. 0	MF0F	MF0E	EMI	1CH	
M. Funct. 1	MF1F	MF1E	EMI	20H	
Time Base 0	TB0F	TB0E	EMI	24H	
Time Base 1	TB1F	TB1E	EMI	28H	
EEPROM	DEF	DEE	EMI	2CH	Low

**Interrupt Structure**

### Thermal Protection Interrupt

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

### LVD 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, 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 LVD Interface Interrupt is serviced, the interrupt request flag, LVF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

### External Interrupts

The external interrupts are controlled by signal transitions on the pins INT0~INT3. An external interrupt request will take place when the external interrupt request flags, INT0F~INT3F, 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~INT3E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register.

When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT3F, 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.

### 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. The clock source that generates  $f_{TB}$ , which in turn controls the Time Base interrupt period, can originate from several different sources which are selected using the TBCK bit in the TBC register. 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.

#### • 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**: Time Base 0 and Time Base 1 Control bit  
0: Disable  
1: Enable

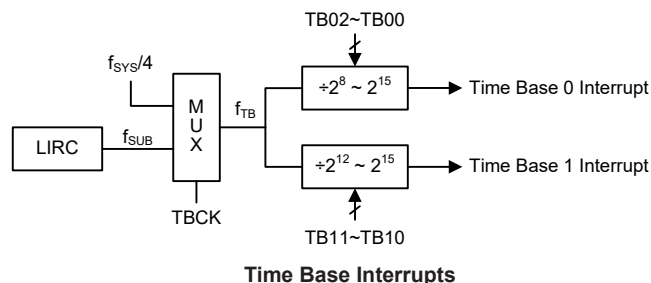
Bit 6      **TBCK**: Select  $f_{TB}$  Clock Source  
0:  $f_{SUB}$   
1:  $f_{SYS}/4$

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

Bit 3      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:  $2^{15}/f_{TB}$



### EEPROM Write Interrupt

An EEPROM Write 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, 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, and the EEPROM interrupt request flag, DEF, will also be automatically cleared.

### Multi-function Interrupts

Within the device there are two Multi-function interrupts. Unlike the other independent interrupts, this interrupt has no independent source, but rather are formed from other existing interrupt sources, namely the PTM interrupts and A/D converter interrupt.

A Multi-function interrupt request will take place when the Multi-function interrupt request flag, MFnF, is set. The Multi-function interrupt flag 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 the Multi-function interrupt occurs, a subroutine call to the Multi-function interrupt vector will take place. When the interrupt is serviced, the Multi-Function request flag, MFnF, 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 flag will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupt, namely the PTM Interrupts and the A/D converter interrupt, will not be automatically reset and must be manually reset by the application program.

### A/D Converter Interrupt

The device contains an A/D converter which is contained within the Multi-function 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, A/D Interrupt enable bit, ADE, and the associated Multi-function interrupt enable bit, MFnF, 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 relevant Multi-function Interrupt vector, will take place. When the A/D converter 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 A/D converter interrupt request flag will not be automatically cleared, it have to be cleared by the application program.

### **Timer Module Interrupts**

Each PTM has two interrupts. All of the PTM interrupts are contained within the Multi-function Interrupt. For each of the PTM there are two interrupt request flags PTMnPF and PTMnAF and two enable bits PTMnPE and PTMnAE. A PTM interrupt request will take place when any of the PTM request flags is set, a situation which occurs when a PTM 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 PTM Interrupt enable bit, and the associated Multi-function interrupt enable bit, MFnF, must first be set. When the interrupt is enabled, the stack is not full and a PTM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector locations, will take place. When the PTM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MFnF flag will be automatically cleared. As the PTM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

### **Interrupt Wake-up Function**

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pin 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 flag, MFnF, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

It is recommended that programs do not use the “CALL” instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a CALL subroutine is executed in the interrupt subroutine.

Every interrupt has the capability of waking up the microcontroller when it is in SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

## **Application Description**

### **Introduction**

Automatic reclosers are used to actively determine whether the power grid voltage is normal or abnormal and to automatically switch the circuit breaker on or off. When a power system fault occurs, the relay protection device trips the circuit breaker and switches off the power supply. If the fault is temporary, the circuit breaker will auto-reclose after a preset time period. In such situations, if the fault has been automatically removed, the power supply will be resumed. If the fault is continuous, the circuit breaker will be switched off again and will no longer reclose.

The HT45F4833 can be used to implement the automatic recloser applications. Aimed at automatic recloser related products, the device includes multiple external interrupts and various I/O configurations, which detect the knife switch position in real time. Together with LEDs, the current product status can be displayed. By using a multi-channel 12-bit A/D converter, the AC voltage and other important parameters can be monitored. In addition, the device is an all-in-one SOC chip and includes a set of complementary H-bridge outputs and an LDO to provide the internal MCU with a stable voltage, which effectively reduces the cost requirements of peripheral components.

### **Functional Description**

#### **Low Voltage/Over Voltage Protection**

When the voltage is less than or greater than the preset value, the circuit breaker will be automatically switched off. If the voltage is normal, then it will be switched on automatically.

#### **Overload/Short Circuit Protection**

When a short circuit condition occurs or when the current is greater than the rated current, the circuit breaker will be automatically switched off to provide circuit protection.

#### **Automatic Switch-off during Power Off**

When the circuit is powered off, the automatic switch-off operation will disconnect the input and output lines. When the circuit is re-powered on, if the power supply is normal, the circuit breaker will be automatically switched on.

### Automatic Switch-on

When the switch-off operation occurs, the circuit breaker will be re-switched on after a certain time interval. This function can reduce the manual switch-on operation, ensuring electrical continuity, improving the operating efficiency, and offering more user convenience.

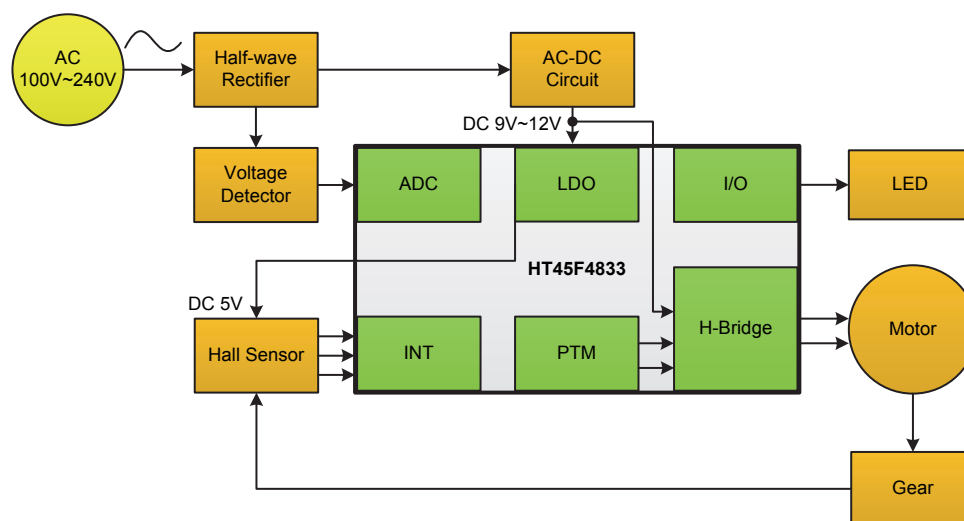
### Manual Switch-on

If the automatic switch-on function is not required, the manual switch-on function can be selected. Its operation is not controlled by electrical power, like normal circuit breakers.

### Fault Alarm

When the above protection functions trip the circuit breaker, different indicators will be turned on to indicate the corresponding fault type.

### Hardware Block Diagram



1. AC 100V~240V is converted into DC 9V~12V through a half-wave rectifier and a AC-DC circuit, providing the operating voltage for the HT45F4833. After this, the DC 9V~12V is reduced to 5V using an internal LDO to provide an internal operating voltage for the hall sensor and the MCU.
2. By using a hall sensor, external interrupt functions and a gear set can be used to determine the current knife switch position for automatic reclosers.
3. The voltage detector circuit outputs a voltage to the MCU internal A/D converter. When the detected voltage exceeds the preset range, the MCU will enter the protection mode and switch off the knife switch. If the voltage is restored to the preset range, switch on the knife switch or remain unchanged according to the current mode.
4. The MCU can control the H-bridge to output a signal through the internal PWM to control the motor operation modes (forward/reverse, soft start, etc.)
5. Use the I/O ports to control LEDs to display the current automatic recloser states.

For example: Normal state: constant on

Over voltage state: blink once every 1 second

Under voltage state: blink twice with a 0.5 second interval every 2 seconds

The schematic diagram illustrates a power supply system for a 100W LED lighting system. The system is powered by an AC input (220V) connected to a transformer (T1). The secondary winding of the transformer is connected to a bridge rectifier (D1-D4) and a filter capacitor (C2). The output of the rectifier is connected to a voltage doubler (D5, D6) and a DC-DC converter (U1). The DC-DC converter (U1) is a switching regulator that converts the input voltage to a regulated 5V output. The 5V output is connected to a 5V regulator (U2) and a 3.3V regulator (U3). The 5V regulator (U2) is connected to the 5V input of the LED array (LED1-LED4) and the 5V input of the power MOSFET (M1). The 3.3V regulator (U3) is connected to the 3.3V input of the power MOSFET (M2). The power MOSFET (M1) is connected to the 100W LED array and the 100W power MOSFET (M2). The power MOSFET (M2) is connected to the 100W LED array and the 100W power MOSFET (M2).

## **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 0.5 $\mu$ s and branch or call instructions would be implemented within 1 $\mu$ s. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

### **Moving and Transferring Data**

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

### **Arithmetic Operations**

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

### Logical and Rotate Operation

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

### Branches and Control Transfer

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

### Bit Operations

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

### Table Read Operations

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

### Other Operations

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



## Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

### Table Conventions

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

Mnemonic	Description	Cycles	Flag Affected
<b>Arithmetic</b>			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV, SC
ADDM A,[m]	Add ACC to Data Memory	1 <sup>Note</sup>	Z, C, AC, OV, SC
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV, SC
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV, SC
ADCM A,[m]	Add ACC to Data memory with Carry	1 <sup>Note</sup>	Z, C, AC, OV, SC
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV, SC, CZ
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV, SC, CZ
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1 <sup>Note</sup>	Z, C, AC, OV, SC, CZ
SBC A,x	Subtract immediate data from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1 <sup>Note</sup>	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1 <sup>Note</sup>	C
<b>Logic Operation</b>			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1 <sup>Note</sup>	Z
ORM A,[m]	Logical OR ACC to Data Memory	1 <sup>Note</sup>	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1 <sup>Note</sup>	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1 <sup>Note</sup>	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
<b>Increment &amp; Decrement</b>			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1 <sup>Note</sup>	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1 <sup>Note</sup>	Z
<b>Rotate</b>			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1 <sup>Note</sup>	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1 <sup>Note</sup>	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1 <sup>Note</sup>	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1 <sup>Note</sup>	C

Mnemonic	Description	Cycles	Flag Affected
<b>Data Move</b>			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1 <sup>Note</sup>	None
MOV A,x	Move immediate data to ACC	1	None
<b>Bit Operation</b>			
CLR [m].i	Clear bit of Data Memory	1 <sup>Note</sup>	None
SET [m].i	Set bit of Data Memory	1 <sup>Note</sup>	None
<b>Branch Operation</b>			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1 <sup>Note</sup>	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1 <sup>Note</sup>	None
SZ [m].i	Skip if bit i of Data Memory is zero	1 <sup>Note</sup>	None
SNZ [m]	Skip if Data Memory is not zero	1 <sup>Note</sup>	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1 <sup>Note</sup>	None
SIZ [m]	Skip if increment Data Memory is zero	1 <sup>Note</sup>	None
SDZ [m]	Skip if decrement Data Memory is zero	1 <sup>Note</sup>	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1 <sup>Note</sup>	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1 <sup>Note</sup>	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
<b>Table Read Operation</b>			
TABRD [m]	Read table (specific page) to TBLH and Data Memory	2 <sup>Note</sup>	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2 <sup>Note</sup>	None
ITABRD [m]	Increment table pointer TBLP first and Read table to TBLH and Data Memory	2 <sup>Note</sup>	None
ITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	2 <sup>Note</sup>	None
<b>Miscellaneous</b>			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1 <sup>Note</sup>	None
SET [m]	Set Data Memory	1 <sup>Note</sup>	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1 <sup>Note</sup>	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then up to three cycles are required, if no skip takes place only one cycle is required.

- Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
- For the “CLR WDT” instruction the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after the “CLR WDT” instructions is executed. Otherwise the TO and PDF flags remain unchanged.

## Instruction Definition

<b>ADC A,[m]</b>	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	$ACC \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C, SC
<b>ADCM A,[m]</b>	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	$[m] \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C, SC
<b>ADD A,[m]</b>	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	$ACC \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C, SC
<b>ADD A,x</b>	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	$ACC \leftarrow ACC + x$
Affected flag(s)	OV, Z, AC, C, SC
<b>ADDM A,[m]</b>	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	$[m] \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C, SC
<b>AND A,[m]</b>	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	$ACC \leftarrow ACC \text{ "AND" } [m]$
Affected flag(s)	Z
<b>AND A,x</b>	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	$ACC \leftarrow ACC \text{ "AND" } x$
Affected flag(s)	Z
<b>ANDM A,[m]</b>	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	$[m] \leftarrow ACC \text{ "AND" } [m]$
Affected flag(s)	Z

<b>CALL addr</b>	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 $\leftarrow$ Program Counter + 1 Program Counter $\leftarrow$ addr
Affected flag(s)	None
<b>CLR [m]</b>	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] $\leftarrow$ 00H
Affected flag(s)	None
<b>CLR [m].i</b>	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i $\leftarrow$ 0
Affected flag(s)	None
<b>CLR WDT</b>	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO $\leftarrow$ 0 PDF $\leftarrow$ 0
Affected flag(s)	TO, PDF
<b>CPL [m]</b>	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	[m] $\leftarrow$ $\overline{[m]}$
Affected flag(s)	Z
<b>CPLA [m]</b>	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
<b>DAA [m]</b>	Decimal-Adjust ACC for addition with result in Data Memory
Description	Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.
Operation	[m] $\leftarrow$ ACC + 00H or [m] $\leftarrow$ ACC + 06H or [m] $\leftarrow$ ACC + 60H or [m] $\leftarrow$ ACC + 66H
Affected flag(s)	C

<b>DEC [m]</b>	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	$[m] \leftarrow [m] - 1$
Affected flag(s)	Z
<b>DECA [m]</b>	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow [m] - 1$
Affected flag(s)	Z
<b>HALT</b>	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	$TO \leftarrow 0$ $PDF \leftarrow 1$
Affected flag(s)	TO, PDF
<b>INC [m]</b>	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	$[m] \leftarrow [m] + 1$
Affected flag(s)	Z
<b>INCA [m]</b>	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow [m] + 1$
Affected flag(s)	Z
<b>JMP addr</b>	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 $\leftarrow$ addr
Affected flag(s)	None
<b>MOV A,[m]</b>	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	$ACC \leftarrow [m]$
Affected flag(s)	None
<b>MOV A,x</b>	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	$ACC \leftarrow x$
Affected flag(s)	None
<b>MOV [m],A</b>	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	$[m] \leftarrow ACC$
Affected flag(s)	None

<b>NOP</b>	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
<b>OR A,[m]</b>	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	$ACC \leftarrow ACC \text{ "OR" } [m]$
Affected flag(s)	Z
<b>OR A,x</b>	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	$ACC \leftarrow ACC \text{ "OR" } x$
Affected flag(s)	Z
<b>ORM A,[m]</b>	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	$[m] \leftarrow ACC \text{ "OR" } [m]$
Affected flag(s)	Z
<b>RET</b>	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	$\text{Program Counter} \leftarrow \text{Stack}$
Affected flag(s)	None
<b>RET A,x</b>	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	$\text{Program Counter} \leftarrow \text{Stack}$ $ACC \leftarrow x$
Affected flag(s)	None
<b>RETI</b>	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	$\text{Program Counter} \leftarrow \text{Stack}$ $EMI \leftarrow 1$
Affected flag(s)	None
<b>RL [m]</b>	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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow [m].7$
Affected flag(s)	None

<b>RLA [m]</b>	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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
Affected flag(s)	None
<b>RLC [m]</b>	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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C
<b>RLCA [m]</b>	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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C
<b>RR [m]</b>	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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow [m].0$
Affected flag(s)	None
<b>RRA [m]</b>	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
Affected flag(s)	None
<b>RRC [m]</b>	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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C

<b>RRCA [m]</b>	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	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
<b>SBC A,[m]</b>	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>SBC A, x</b>	Subtract immediate data from ACC with Carry
Description	The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>SBCM A,[m]</b>	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>SDZ [m]</b>	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] - 1$ Skip if $[m]=0$
Affected flag(s)	None
<b>SDZA [m]</b>	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m] - 1$ Skip if $ACC=0$
Affected flag(s)	None



<b>SET [m]</b>	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	$[m] \leftarrow FFH$
Affected flag(s)	None
<b>SET [m].i</b>	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	$[m].i \leftarrow 1$
Affected flag(s)	None
<b>SIZ [m]</b>	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] + 1$ Skip if $[m]=0$
Affected flag(s)	None
<b>SIZA [m]</b>	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	$ACC \leftarrow [m] + 1$ Skip if $ACC=0$
Affected flag(s)	None
<b>SNZ [m].i</b>	Skip if Data Memory is not 0
Description	If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if $[m].i \neq 0$
Affected flag(s)	None
<b>SNZ [m]</b>	Skip if Data Memory is not 0
Description	If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if $[m] \neq 0$
Affected flag(s)	None
<b>SUB A,[m]</b>	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	$ACC \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C, SC, CZ

<b>SUBM A,[m]</b>	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	$[m] \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>SUB A,x</b>	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	$ACC \leftarrow ACC - x$
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>SWAP [m]</b>	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
Affected flag(s)	None
<b>SWAPA [m]</b>	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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
Affected flag(s)	None
<b>SZ [m]</b>	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
<b>SZA [m]</b>	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	$ACC \leftarrow [m]$ Skip if $[m]=0$
Affected flag(s)	None
<b>SZ [m].i</b>	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

<b>TABRD [m]</b>	Read table (specific page) to TBLH and Data Memory
Description	The low byte of the program code (specific page) addressed by the table pointer pair (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
<b>TABRDL [m]</b>	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
<b>ITABRD [m]</b>	Increment table pointer low byte first and read table to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
<b>ITABRDL [m]</b>	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
<b>XOR A,[m]</b>	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
<b>XORM A,[m]</b>	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
<b>XOR A,x</b>	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

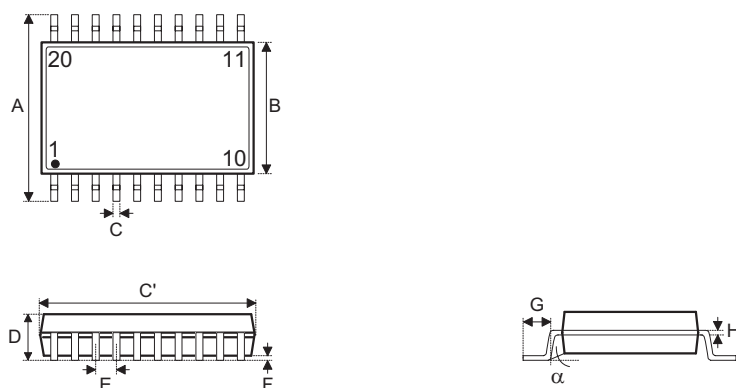
## Package Information

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

**20-pin NSOP (150mil) Outline Dimensions**



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	0.236	0.244
B	0.146	0.154	0.161
C	0.009	—	0.012
C'	0.382	0.390	0.398
D	—	—	0.069
E	—	0.032 BSC	—
F	0.002	—	0.009
G	0.020	—	0.031
H	0.008	—	0.010
$\alpha$	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.80	6.00	6.20
B	3.70	3.90	4.10
C	0.23	—	0.30
C'	9.70	9.90	10.10
D	—	—	1.75
E	—	0.80 BSC	—
F	0.05	—	0.23
G	0.50	—	0.80
H	0.21	—	0.25
$\alpha$	0°	—	8°

Copyright© 2018 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at <http://www.holtek.com/en/>.