



---

**UART Digital Servo Flash MCU**

**HT45F4842**

Revision: V1.00 Date: January 21, 2020

[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>7</b>
<b>Pin Assignment</b> .....	<b>8</b>
<b>Pin Description</b> .....	<b>9</b>
H-Bridge Driver Pin Description .....	10
Interconnection Signal Description.....	10
<b>Absolute Maximum Ratings</b> .....	<b>11</b>
<b>D.C. Electrical Characteristics</b> .....	<b>11</b>
Operating Voltage Characteristics.....	11
Operating Current Characteristics.....	11
Standby Current Characteristics .....	12
<b>A.C. Characteristics</b> .....	<b>12</b>
High Speed Internal Oscillator – HIRC – Frequency Accuracy .....	12
Low Speed Oscillators Characteristics – LIRC .....	13
System Start Up Time Characteristics .....	13
<b>Input/Output Characteristics</b> .....	<b>13</b>
<b>LVR Electrical Characteristics</b> .....	<b>14</b>
<b>A/D Converter Electrical Characteristics</b> .....	<b>14</b>
<b>DNF Electrical Characteristics</b> .....	<b>15</b>
<b>LDO Electrical Characteristics</b> .....	<b>15</b>
<b>Level Shifter Electrical Characteristics</b> .....	<b>16</b>
<b>Power-on Reset Characteristics</b> .....	<b>16</b>
<b>H-Bridge Driver Electrical Characteristics</b> .....	<b>16</b>
<b>System Architecture</b> .....	<b>17</b>
Clocking and Pipelining.....	18
Program Counter – PC.....	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.....	21
Table Program Example.....	21
In Circuit Programming – ICP .....	22
On-Chip Debug Support – OCDS .....	23
In Application Programming – IAP .....	23

<b>Data Memory .....</b>	<b>36</b>
Structure.....	36
Data Memory Addressing.....	37
General Purpose Data Memory .....	37
Special Purpose Data Memory .....	37
<b>Special Function Register Description.....</b>	<b>39</b>
Indirect Addressing Registers – IAR0, IAR1, IAR2 .....	39
Memory Pointers – MP0, MP1L, MP1H, MP2L, MP2H.....	39
Accumulator – ACC.....	40
Program Counter Low Register – PCL.....	41
Look-up Table Registers – TBLP, TBHP, TBLH.....	41
Status Register – STATUS.....	41
<b>Oscillators .....</b>	<b>43</b>
Oscillator Overview .....	43
System Clock Configurations.....	43
Internal High Speed RC Oscillator – HIRC .....	43
Internal 32kHz Oscillator – LIRC.....	44
<b>Operating Modes and System Clocks .....</b>	<b>44</b>
System Clocks .....	44
System Operation Modes.....	45
Control Registers .....	46
Operating Mode Switching.....	47
Standby Current Considerations .....	50
Wake-up.....	50
<b>Watchdog Timer.....</b>	<b>51</b>
Watchdog Timer Clock Source.....	51
Watchdog Timer Control Register .....	51
Watchdog Timer Operation .....	52
<b>Reset and Initialisation.....</b>	<b>53</b>
Reset Functions .....	53
Reset Initial Conditions .....	55
<b>Input/Output Ports .....</b>	<b>58</b>
Pull-high Resistors .....	58
Port A Wake-up .....	59
I/O Port Control Registers .....	59
Pin-shared Functions .....	60
I/O Pin Structures.....	62
Programming Considerations .....	62
<b>Timer Modules – TM .....</b>	<b>63</b>
Introduction .....	63
TM Operation .....	63
TM Clock Source.....	63
TM Interrupts.....	64
TM External Pins.....	64
Programming Considerations.....	65

<b>Compact Type TM – CTM .....</b>	<b>66</b>
Compact Type TM Operation .....	66
Compact Type TM Register Description.....	66
Compact Type TM Operating Modes .....	70
<b>Periodic Type TM – PTM.....</b>	<b>76</b>
Periodic Type TM Operation.....	76
Periodic Type TM Register Description.....	76
Periodic Type TM Operation Modes.....	81
<b>Standard Type TM – STM .....</b>	<b>90</b>
Standard Type TM Operation.....	90
Standard Type TM Register Description .....	90
Standard Type TM Operation Modes .....	94
<b>Analog to Digital Converter .....</b>	<b>104</b>
A/D Converter Overview .....	104
A/D Converter Register Description.....	104
A/D Converter Reference Voltage.....	107
A/D Converter Input Signal .....	107
A/D Converter Operation.....	108
Conversion Rate and Timing Diagram .....	109
Summary of A/D Conversion Steps.....	109
Programming Considerations.....	110
A/D Conversion Function .....	110
A/D Conversion Programming Examples.....	111
<b>UART Interface.....</b>	<b>113</b>
UART External Pins .....	114
UART Data Transfer Scheme.....	114
UART Status and Control Registers.....	114
Baud Rate Generator .....	121
UART Setup and Control.....	122
UART Transmitter.....	124
UART Receiver .....	125
Managing Receiver Errors .....	126
UART Interrupt Structure.....	127
UART Power Down and Wake-up.....	129
<b>Digital Noise Filter .....</b>	<b>129</b>
Noise Filter Register Description.....	130
<b>LDO Divider Circuit.....</b>	<b>131</b>
<b>Interrupts .....</b>	<b>131</b>
Interrupt Registers.....	131
Interrupt Operation .....	135
External Interrupt.....	136
UART Interrupt .....	137
Noise Filter Input Interrupt.....	137
Multi-function Interrupts.....	137
Timer Module Interrupts .....	138

A/D Converter Interrupt .....	138
Time Base Interrupts .....	138
Interrupt Wake-up Function .....	140
Programming Considerations .....	140
<b>Configuration Options .....</b>	<b>141</b>
<b>H-Bridge Driver .....</b>	<b>141</b>
H-Bridge Driver Control Registers .....	141
H-Bridge Output Control .....	143
Active Period and Sleep Period .....	143
V <sub>DD,H</sub> Under Voltage Lock-Out .....	144
Over Current Protection – OCP .....	144
Output Short-Circuit Protection – OSP .....	145
Over Temperature Protection – OTP .....	145
Component/Motor Selection Guide .....	146
<b>Application Circuits .....</b>	<b>147</b>
<b>Instruction Set .....</b>	<b>148</b>
Introduction .....	148
Instruction Timing .....	148
Moving and Transferring Data .....	148
Arithmetic Operations .....	148
Logical and Rotate Operation .....	149
Branches and Control Transfer .....	149
Bit Operations .....	149
Table Read Operations .....	149
Other Operations .....	149
<b>Instruction Set Summary .....</b>	<b>150</b>
Table Conventions .....	150
Extended Instruction Set .....	152
<b>Instruction Definition .....</b>	<b>154</b>
Extended Instruction Definition .....	163
<b>Package Information .....</b>	<b>170</b>
10-pin SOP-EP Outline Dimensions .....	171
SAW Type 24-pin QFN (4mm×4mm×0.75mm) Outline Dimensions .....	172

## Features

### CPU Features

- Operating Voltage
  - ♦  $V_{DD}=3.0V\sim 5.5V$
  - ♦  $V_{CC2}=3.1V\sim 12V$
- Up to 0.25 $\mu$ s instruction cycle with 16MHz system clock at  $V_{DD}=5V$
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - ♦ Internal 16MHz High Speed Oscillator – HIRC
  - ♦ Internal 32kHz Low Speed Oscillator – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: 4K $\times$ 16
- RAM Data Memory: 256 $\times$ 8
- Watchdog Timer function
- 6 bidirectional I/O lines
- One pin-shared external interrupt
- 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
- Fully-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- 3 external channels 12-bit resolution A/D converter
- Digital Noise Filter with interrupt
- Internal LDO outputs: 3.3V, 5V
- Low Voltage Reset function
- Flash program memory can be re-programmed up to 10,000 times
- Flash program memory data retention > 10 years
- 1-channel H-Bridge Driver
- Package types: 10-pin SOP-EP, 24-pin QFN

## General Description

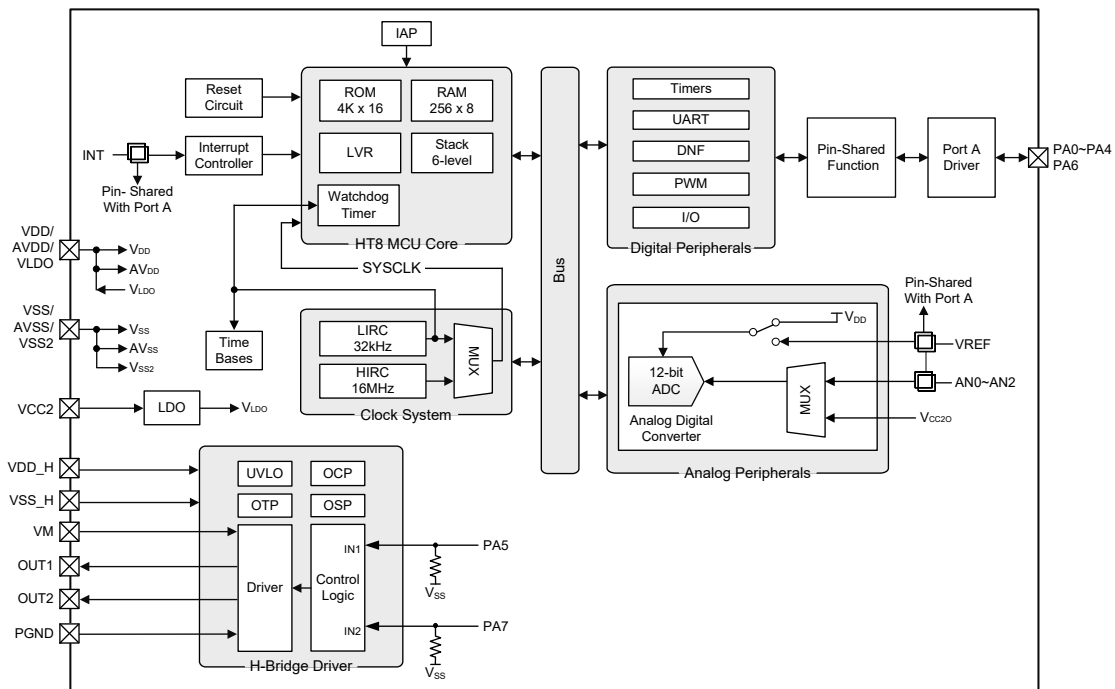
The HT45F4842 device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller, specifically designed for products using up to two lithium batteries or digital servo applications. Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and features. Other memory includes an area of RAM Data Memory.

Analog features include a multi-channel 12-bit A/D converter function and an internal LDO for power supply. Multiple extremely flexible Timer Modules provide timing, pulse generation, capture input, compare match output and PWM generation functions. Communication with the outside world is catered for by including fully integrated UART interface function, the popular interface which provides designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

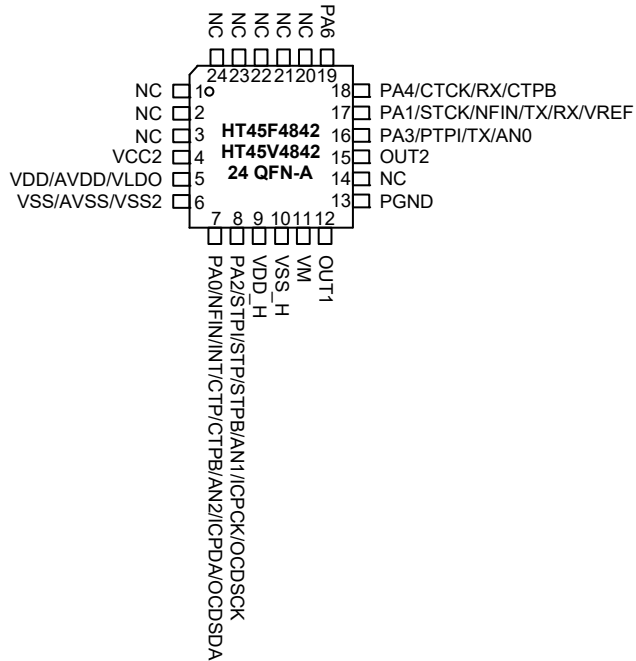
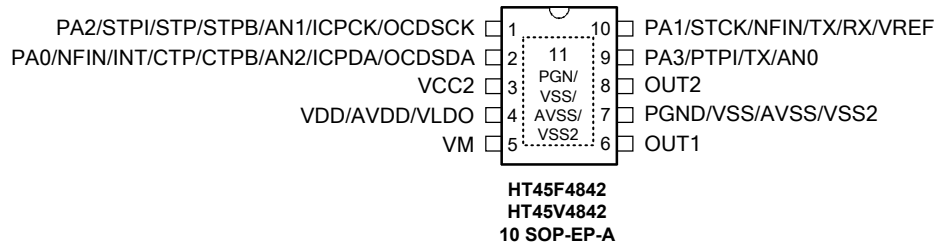
The device also includes fully integrated high and low speed oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation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

The inclusion of 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 digital servo applications although the device will also lend itself for use in a range of other related applications.

## Block Diagram



## Pin Assignment



- 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 supplied for the OCDS dedicated pins and as such only available for the HT45V4842 device which is the OCDS EV chip for the HT45F4842 device.
3. For the less pin count package type there will be unbounded pins which should be properly configured to avoid unwanted power consumption resulting from floating input conditions. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.



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

Pin Name	Function	OPT	I/T	O/T	Description
PA0/NFIN/INT/ CTP/CTPB/ AN2/ICPDA/ OCDSDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	NFIN	PAS0 IFS	ST	—	Noise Filter external input
	INT	PAS0	ST	—	External interrupt input
	CTP	PAS0	—	CMOS	CTM output
	CTPB	PAS0	—	CMOS	CTM inverted output
	AN2	PAS0	AN	—	A/D Converter external input channel 2
	ICPDA	—	ST	CMOS	ICP data/address pin
OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only	
PA1/STCK/NFIN/ TX/RX/VREF	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	STCK	PAS0	ST	—	STM clock input
	NFIN	PAS0 IFS	ST	—	Noise Filter external input
	TX	PAS0	—	CMOS	UART transmitter pin
	RX	PAS0 IFS	ST	—	UART receiver pin
	VREF	PAS0	AN	—	A/D Converter external reference voltage input
PA2/STPI/STP/ STPB/AN1/ ICPCK/OCDSCK	PA2	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	STPI	PAS0	ST	—	STM capture input
	STP	PAS0	—	CMOS	STM output
	STPB	PAS0	—	CMOS	STM inverted output
	AN1	PAS0	AN	—	A/D Converter external input channel 1
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/PTPI/TX/ AN0	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTPI	PAS0	ST	—	PTM capture input
	TX	PAS0	—	CMOS	UART transmitter pin
	AN0	PAS0	AN	—	A/D Converter external input channel 0
PA4/CTCK/RX/ CTPB	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK	PAS1	ST	—	CTM clock input
	RX	PAS1 IFS	ST	—	UART receiver pin
	CTPB	PAS1	—	CMOS	CTM inverted output
PA6	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up

Pin Name	Function	OPT	I/T	O/T	Description
VCC2	VCC2	—	PWR	—	High voltage positive power for LDO input voltage
VDD/AVDD/ VLDO	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
	VLDO	—	—	PWR	LDO output voltage
VSS/AVSS/VSS2	VSS	—	PWR	—	Digital negative power supply
	AVSS	—	PWR	—	Analog negative power supply
	VSS2	—	PWR	—	High voltage negative power supply
NC	NC	—	—	—	No connection

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

### H-Bridge Driver Pin Description

Name	Type	Description
VM	P	H-Bridge Driver Motor Power supply
OUT1	O	H-Bridge Output 1
OUT2	O	H-Bridge Output 2
PGND	G	Motor Current Sensing Terminal Connect via a sensing resistor to GND. If it is not necessary to sense the motor current, the PGND line should be directly connected to GND.
VDD_H	P	H-Bridge Driver Power supply
VSS_H	P	H-Bridge negative power supply

Note: I: Input; O: Output; P: Power; G: Ground

### Interconnection Signal Description

Several signals are not connected to external package pins. These signals are interconnection lines between the MCU and the H-Bridge driver and are listed in the following table.

MCU Signal	H-Bridge Signal	Function	Description
PA5	IN1	PA5	General purpose I/O Register enabled pull-high and always enabled pull-low. Internally connected to the H-bridge driver input IN1.
		IN1	H-bridge Driver input 1 Internally connected to the MCU I/O line PA5.
PA7	IN2	PA7	General purpose I/O Register enabled pull-high and always enabled pull-low. Internally connected to the H-bridge driver input IN2.
		IN2	H-bridge Driver input 2 Internally connected to the MCU I/O line PA7.

Note: 1. The internal signals, PA5 and PA7, are internally connected to the H-Bridge driver inputs, IN1 and IN2, respectively which should be properly configured to control the H-Bridge driver. Refer to the "H-Bridge Driver" chapter for more details.

2. The internal signal, PA6 is unbounded line in package 10-pin SOP\_EP which should be properly configured to avoid unwanted power consumption resulting from floating input conditions. Refer to the "Standby Current Considerations" and "Input/Output Ports" sections.

## Absolute Maximum Ratings

Supply Voltage ( $V_{CC2}$ ).....	$V_{SS}-0.3V$ to $12.0V$
Supply Voltage ( $V_{DD}$ ) .....	$V_{SS}-0.3V$ to $6.0V$
Input Voltage .....	$V_{SS}-0.3V$ to $V_{DD}+0.3V$
Storage Temperature.....	$-50^{\circ}C$ to $125^{\circ}C$
Operating Temperature.....	$-40^{\circ}C$ to $85^{\circ}C$
$I_{OH}$ Total .....	$-80mA$
$I_{OL}$ Total .....	$80mA$
Total Power Dissipation .....	$500mW$
$V_M$ , OUT1, OUT2 .....	$-0.3V$ to $20V$
Maximum Junction Temperature.....	$+150^{\circ}C$
Junction-to-Ambient Thermal Resistance, $\theta_{JA}$ .....	$-0.3V$ to $19V$

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

## D.C. Electrical Characteristics

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

### Operating Voltage Characteristics

$T_a=-40^{\circ}C\sim 85^{\circ}C$

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$V_{DD}$	Operating Voltage – HIRC	$f_{SYS}=f_{HIRC}=16MHz$	3.0	—	5.5	V
	Operating Voltage – LIRC	$f_{SYS}=f_{LIRC}=32kHz$	3.0	—	5.5	V

### Operating Current Characteristics

$T_a=-40^{\circ}C\sim 85^{\circ}C$

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		$V_{DD}$	Conditions				
$I_{DD}$	SLOW Mode – LIRC	3.3V	$f_{SYS}=f_{LIRC}=32kHz$	—	10	20	$\mu A$
		5V		—	30	50	
	FAST Mode – HIRC	3.3V	$f_{SYS}=f_{HIRC}=16MHz$	—	1.2	1.8	mA
		5V		—	3.6	4.8	

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

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

### Standby Current Characteristics

Ta=25°C, unless otherwise specified.

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V <sub>DD</sub>	Conditions					
I <sub>STB</sub>	SLEEP Mode	3.3V	WDT off	—	0.2	0.8	1.0	μA
		5V		—	0.5	1.0	1.2	
		3.3V	WDT on	—	3	5	6	μA
		5V		—	5	10	12	
	IDLE0 Mode – LIRC	3.3V	f <sub>SUB</sub> on	—	3	5	6	μA
		5V		—	5	10	12	
	IDLE1 Mode – HIRC	3.3V	f <sub>SUB</sub> on, f <sub>SYS</sub> =16MHz	—	1.1	1.6	1.6	mA
		5V		—	1.4	2.0	2.0	

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

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

### A.C. Characteristics

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

#### High Speed Internal Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at a user selected HIRC frequency and user selected voltage of either 3.3V or 5V.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Temp.				
f <sub>HIRC</sub>	16MHz Writer Trimmed HIRC Frequency	3.3V	25°C	-1%	16	+1%	MHz
			-40°C~85°C	-3%	16	+3%	
		3.0V~ 3.6V	25°C	-2.5%	16	+2.5%	
			-40°C~85°C	-4%	16	+4%	
	16MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	16	+1%	MHz
			-40°C~85°C	-3%	16	+3%	
		3.3V~ 5.5V	25°C	-2.5%	16	+2.5%	
			-40°C~85°C	-4%	16	+4%	

Note: 1. The 3.3V/5V values for V<sub>DD</sub> are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

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

### Low Speed Oscillators Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Temp.				
f <sub>LIRC</sub>	LIRC Frequency	5V	25°C	25.6	32.0	38.4	kHz
		3.0V~5.5V	25°C	12.8	32.0	41.6	
			-40°C~85°C	8	32	60	
t <sub>START</sub>	LIRC Start Up Time	—	25°C	—	—	100	μs

### System Start Up Time Characteristics

T<sub>a</sub>=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
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>
	System Speed Switch Time (FAST to SLOW Mode or SLOW to FAST Mode)	—	f <sub>HIRC</sub> switches from off → on	—	16	—	t <sub>HIRC</sub>
t <sub>RSTD</sub>	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR <sub>POR</sub> =5V/ms	25	50	150	ms
	System Reset Delay Time (WDTC Software Reset or IAP Reset)	—	—	—	—	—	—
	System Reset Delay Time (Reset Source from WDT Overflow)	—	—	8.3	16.7	50.0	ms
t <sub>SRESET</sub>	Minimum Software Reset Width to Reset	—	—	45	90	375	μs

- 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>, as provided in the LIRC frequency table, 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.

### Input/Output Characteristics

T<sub>a</sub>=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
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	—	5.0	V
		—		0.8V <sub>DD</sub>	—	V <sub>DD</sub>	
I <sub>OL</sub>	Sink Current for I/O Ports	3.3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	15.5	31.0	—	mA
		5V		31	62	—	
I <sub>OH</sub>	Source Current for I/O Ports	3.3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-3.5	-7.0	—	mA
		5V		-7.2	-14.5	—	
R <sub>PH</sub>	Pull-high Resistance for I/O Ports <sup>(Note)</sup>	3.3V	—	20	60	100	kΩ
		5V		10	30	50	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
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
t <sub>TCK</sub>	CTM/STM Clock Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>TPI</sub>	PTM/STM Capture Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>INT</sub>	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs

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

## LVR Electrical Characteristics

T<sub>a</sub>=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	—	3.0	—	5.5	V
V <sub>LVR</sub>	Low Voltage Reset Voltage	—	LVR enable	-5%	2.8	+5%	V
t <sub>LVR</sub>	Minimum Low Voltage Width to Reset	—	—	140	600	1000	μs

## A/D Converter Electrical Characteristics

T<sub>a</sub>=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	—	3.0	—	5.5	V
V <sub>ADI</sub>	Input Voltage	—	—	0	—	V <sub>REF</sub>	V
V <sub>REF</sub>	Reference Voltage	—	—	2	—	V <sub>DD</sub>	V
DNL	Differential Non-linearity	3.3V	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs	-3	—	+3	LSB
		5V					
		3.3V	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs				
		5V					
INL	Integral Non-linearity	3.3V	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs	-4	—	+4	LSB
		5V					
		3.3V	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs				
		5V					
I <sub>ADC</sub>	Additional Current Consumption for A/D Converter Enable	3.3V	No load, t <sub>ADCK</sub> =0.5μs	—	1	2	mA
		5V		—	1.5	3.3	
t <sub>ADCK</sub>	Clock Period	—	—	0.5	—	10.0	μs
t <sub>ON2ST</sub>	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t <sub>ADS</sub>	Sampling Time	—	—	—	4	—	t <sub>ADCK</sub>
t <sub>ADC</sub>	Conversion Time (Including A/D Sample and Hold Time)	—	—	—	16	—	t <sub>ADCK</sub>
GERR	A/D Conversion Gain Error	3.3V	V <sub>REF</sub> =V <sub>DD</sub>	-4	—	4	LSB
		5V					
OSRR	A/D Conversion Offset Error	3.3V	V <sub>REF</sub> =V <sub>DD</sub>	-4	—	4	LSB
		5V					

## DNF Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>NFIN</sub>	DNF Capture Pulse Width	—	f <sub>SYS</sub> =f <sub>HIRC</sub>	4	—	128	1/f <sub>SYS</sub>

## LDO Electrical Characteristics

V<sub>IN</sub>=V<sub>OUT</sub>+1V, C<sub>LOAD</sub>=10μF+0.1μF, Ta=-40°C~85°C, unless otherwise specified.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>IN</sub>	Conditions				
V <sub>IN</sub>	Input Voltage	—	—	V <sub>OUT</sub> +0.1	6	12	V
V <sub>OUT</sub>	Output Voltage	—	Ta=25°C, I <sub>LOAD</sub> =1mA, V <sub>OUT</sub> =3.3V	-2%	3.3	+2%	V
			Ta=-40°C~85°C, I <sub>LOAD</sub> =1mA, V <sub>OUT</sub> =3.3V	-5%	3.3	+5%	
			Ta=25°C, I <sub>LOAD</sub> =1mA, V <sub>OUT</sub> =5V	-2%	5	+2%	
			Ta=-40°C~85°C, I <sub>LOAD</sub> =1mA, V <sub>OUT</sub> =5V	-5%	5	+5%	
ΔV <sub>LOAD</sub>	Load Regulation <sup>(1)</sup>	—	1mA ≤ I <sub>LOAD</sub> ≤ 70mA	—	0.015	0.033	%/mA
V <sub>DROP</sub>	Dropout Voltage <sup>(2)</sup>	—	ΔV <sub>OUT</sub> =2%, I <sub>LOAD</sub> =1mA	—	20	—	mV
			ΔV <sub>OUT</sub> =2%, I <sub>LOAD</sub> =10mA	—	100	—	
			V <sub>IN</sub> =V <sub>OUT</sub> +1.5V, ΔV <sub>OUT</sub> =2%, I <sub>LOAD</sub> =70mA	—	600	1000	
I <sub>OUT</sub>	Output Current	—	V <sub>IN</sub> =V <sub>OUT</sub> +1V, V <sub>OUT</sub> =5V, ΔV <sub>OUT</sub> =-3%	70	—	—	mA
			V <sub>IN</sub> =V <sub>OUT</sub> +2V, V <sub>OUT</sub> =5V, ΔV <sub>OUT</sub> =-3%	150	—	—	
			V <sub>IN</sub> =V <sub>OUT</sub> +1V, V <sub>OUT</sub> =3.3V, ΔV <sub>OUT</sub> =-3%	45	—	—	mA
			V <sub>IN</sub> =V <sub>OUT</sub> +2V, V <sub>OUT</sub> =3.3V, ΔV <sub>OUT</sub> =-3%	90	—	—	
I <sub>Q</sub>	Quiescent Current	12V	No load	—	5	7	μA
ΔV <sub>LINE</sub>	Line Regulation	—	V <sub>OUT</sub> +1V ≤ V <sub>IN</sub> ≤ 12V, I <sub>LOAD</sub> =1mA	—	—	0.2	%/V
TC	Temperature Coefficient	—	Ta=-40°C~85°C, I <sub>LOAD</sub> =10mA	—	±1.5	±2	mV/°C
ΔV <sub>OUT_RIPPLE</sub>	Output Voltage Ripple <sup>(3)</sup>	6V	I <sub>LOAD</sub> =10mA	—	—	40	mV
RR	Ripple Rejection	—	V <sub>IN</sub> =10V <sub>DC</sub> +2V <sub>P-P(AC)</sub> , I <sub>LOAD</sub> ≤ 150mA, f=120Hz	35	—	—	dB
I <sub>LIMIT</sub>	Current Limit	6V	ΔV <sub>OUT</sub> =-10%	180	380	—	mA
t <sub>START</sub>	LDO Start-up Time	6V	I <sub>LOAD</sub> =1mA, V <sub>OUT</sub> settle to ±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<sub>D</sub>=(T<sub>J(MAX)</sub>-Ta)/θ<sub>JA</sub>.

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<sub>IN</sub>.

3. Ripple rejection ratio measurement circuit. RR=20×log(ΔV<sub>IN</sub>/ΔV<sub>OUT</sub>).

## Level Shifter Electrical Characteristics

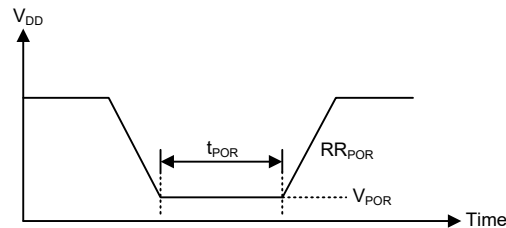
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>IN</sub>	Input Voltage	3.3V	—	3.1	—	12	V
		5V		5.1	—	12	
V <sub>CC20</sub>	V <sub>CC20</sub> Accuracy	—	V <sub>IN</sub> =3.1V~12V, A/D clock=f <sub>sys</sub> /8	-5%	0.2V <sub>IN</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



## H-Bridge Driver Electrical Characteristics

 V<sub>DD,H</sub>=5V, V<sub>M</sub>=15V and Ta=+25 °C, unless otherwise specified

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
<b>Power Supply</b>						
V <sub>DD,H</sub>	Supply Voltage	—	3.0	—	5.5	V
I <sub>DD</sub>	Supply Operation Current	PWM=25kHz, OUT1 and OUT2 open	—	0.45	1.00	mA
I <sub>DD(STB)</sub>	Supply Standby Current	PA5=PA7="0", Standby mode	—	550	800	μA
I <sub>DD(SLP)</sub>	Supply Sleep Current	PA5=PA7="0" or "1", Sleep Period	—	—	0.1	μA
V <sub>M</sub>	Motor Power Supply	—	2.5	—	15	V
I <sub>M</sub>	V <sub>M</sub> Operation Current	PWM=25kHz, no load	—	0.85	1.00	mA
I <sub>M(STB)</sub>	V <sub>M</sub> Standby Current	PA5=PA7="0", Standby mode	—	950	1100	μA
<b>H-Bridge Circuit</b>						
R <sub>ON</sub>	HS+LS FET On-resistance <sup>(1)</sup>	V <sub>DD,H</sub> =3.3V, I <sub>OUT</sub> =800mA	—	0.3	0.4	Ω
V <sub>CLAMP</sub>	Clamp Diode Voltage	I=500mA (HS and LS)	—	0.8	—	V
I <sub>HS(OFF)</sub>	HS MOSFET Leakage Current	PA5=PA7="0", V <sub>M</sub> =15V, V <sub>OUT</sub> =0V, measure I (V <sub>M</sub> )	—	—	0.1	μA
t <sub>r(OUT)</sub>	Output Rise Time	R <sub>L</sub> =20Ω, 10% to 90% (Figure1)	—	100	—	ns
t <sub>f(OUT)</sub>	Output Fall Time	R <sub>L</sub> =20Ω, 90% to 10% (Figure1)	—	100	—	ns



Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
<b>Control Logic</b>						
t <sub>P1</sub>	IN-to-OUT Propagation Delay (Figure 1)	R <sub>L</sub> =20Ω, PAn to OUTx (high-Z to high/low)	—	100	—	ns
t <sub>P2</sub>		R <sub>L</sub> =20Ω, PAn to OUTx (high/low to high-Z)	—	100	—	ns
t <sub>P3</sub>		R <sub>L</sub> =20Ω, PAn to OUTx	—	100	—	ns
t <sub>P4</sub>		R <sub>L</sub> =20Ω, PAn to OUTx	—	100	—	ns
t <sub>SLPEN</sub>	Sleep Period Entry Time	PA5=PA7="0" or "1" until charge pump switches off (Figure 2)	—	10	—	ms
f <sub>PWM</sub>	Input PWM Frequency	Internal charge pump activates	—	—	200	kHz
R <sub>PD</sub>	Input Pull Down Resistance	PA5 and PA7	—	135	—	kΩ
<b>Charge Pump</b>						
t <sub>CP_ON</sub>	Charge Pump on Time	Charge pump activates time (Figure 2)	—	10	—	ms
<b>Protection</b>						
V <sub>UVLO+</sub>	V <sub>DD,H</sub> Turn on Level	V <sub>DD,H</sub> rises	—	—	2.5	V
V <sub>UVLO-</sub>	V <sub>DD,H</sub> Turn off Level	V <sub>DD,H</sub> falls	1.8	—	—	V
I <sub>OC</sub>	Over Current Threshold	With deglitch time, t <sub>DEG</sub>	2.5	3.0	3.5	A
t <sub>DEG</sub>	Over Current Deglitch Time	(Figure 3)	—	2	—	μs
t <sub>RETRY</sub>	Over Current Retry Time <sup>(3)</sup>	(Figure 3 & 4)	—	1	—	ms
I <sub>OSP</sub>	Short Circuit Threshold <sup>(2)</sup>	Without deglitch time (Figure 4)	—	4.5	—	A
T <sub>SHD</sub>	Thermal Shutdown Threshold	—	—	155	—	°C
T <sub>REC</sub>	Thermal Recovery Temperature	—	—	120	—	°C

Note: 1. HS means High Side and LS means Low Side.

- The devices provide full short circuit protection for the OUTx-to-ground, OUTx-to-power or OUT1-to-OUT2 path.
- The retry mechanism is only active in Forward and Reverse mode.

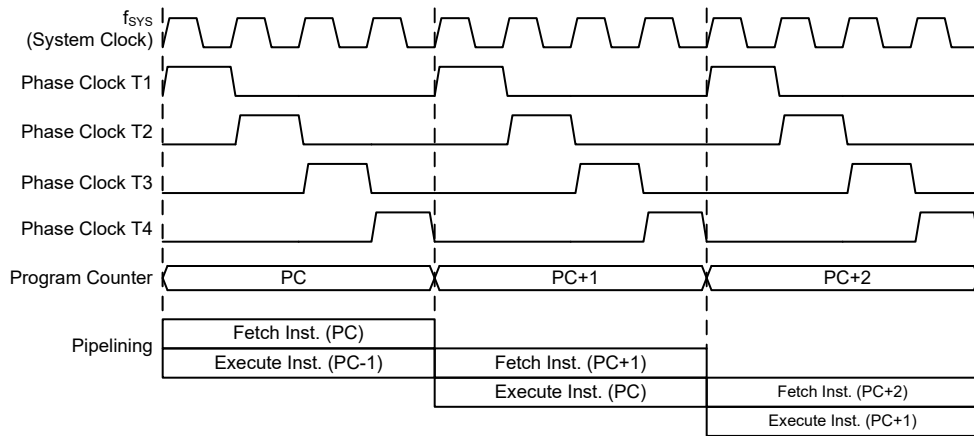
## System Architecture

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

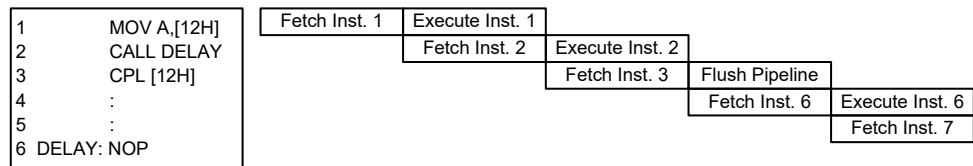
**Clocking and Pipelining**

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

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



**System Clocking and Pipelining**



**Instruction Fetching**

**Program Counter – PC**

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

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

Program Counter	
High Byte	Low Byte (PCL)
PC11~PC8	PCL7~PCL0

**Program Counter**

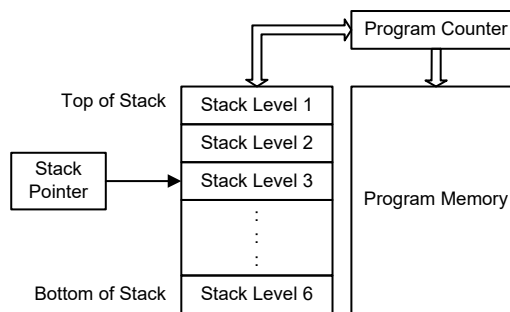
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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. For this device the stack has 6 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



### Arithmetic and Logic Unit – ALU

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

- Arithmetic operations:  
 ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,  
 LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA

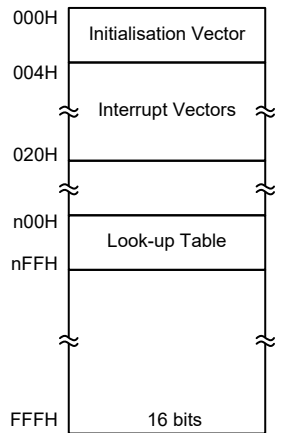
- Logic operations:  
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,  
LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:  
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,  
LRR, LRRCA, LRRCA, LRRC, LRLA, LRL, LRLCA, LRLC
- Increment and Decrement:  
INCA, INC, DECA, DEC,  
LINCA, LINC, LDECA, LDEC
- Branch decision:  
JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI,  
LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

## Flash Program Memory

The Program Memory is the location where the user code or program is stored. For 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 4K×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 pair of data table pointer registers.



**Program Memory Structure**

### Special Vectors

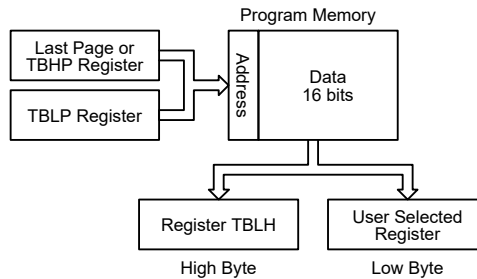
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

## Look-up Table

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

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

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 “0F00H” which refers to the start address of the last page within the 4K words Program Memory of the device. The table pointer low byte register is setup here to have an initial value of “06H”. This will ensure that the first data read from the data table will be at the Program Memory address “0F06H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specified address pointed by the TBLP and TBHP registers if the “TABRD [m]” or “LTABRD [m]” instruction is being used. In this case the high byte of the table data which is equal to zero will be transferred to the TBLH register automatically when the “TABRD [m]” or “LTABRD [m]” instruction is executed.

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

**Table Read Program Example**

```

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

```

**In Circuit Programming – ICP**

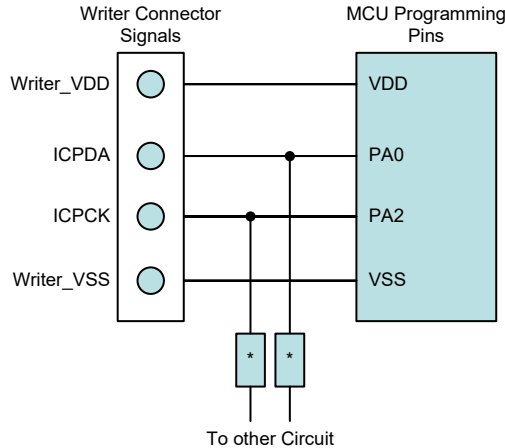
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

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

The Program Memory 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 one line for the reset. The technical details regarding the in-circuit programming of the device is beyond the scope of this document and will be supplied in supplementary literature.

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



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

### On-Chip Debug Support – OCDS

There is an EV chip named HT45V4842 which is used to emulate the HT45F4842 device. This EV chip device also provides an “On-Chip Debug” function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for “On-Chip Debug” function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCSDSA and OCDSCK pins to the Holtek HT-IDE development tools. The OCSDSA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCSDSA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named “Holtek e-Link for 8-bit MCU OCDS User’s Guide”.

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

### In Application Programming – IAP

Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. The provision of the IAP function offers users the convenience of Flash Memory multi-programming features. The convenience of the IAP function is that it can execute the updated program procedure using its internal firmware, without requiring an external Program Writer or PC. In addition, the IAP interface can also be any type of communication protocol, such as UART or USB, using I/O pins. Regarding the internal firmware, the user can select versions provided by Holtek or create their own. The following section illustrates the procedures regarding how to implement the IAP firmware.

#### Flash Memory Read/Write Size

The Flash memory Erase operation is carried out in a block format while the Write operation is carried out in 4-word format and the Read operation is carried out in a word format. The block size is assigned with a capacity of 256 words. Note that the Erase operation should be executed before the Write operation is executed.

When the Flash Memory Erase/Write Function is successfully enabled, the CFWEN bit will be set high. When the CFWEN bit is set high, the data can be written into the data registers. The FWT bit is used to initiate the write process and then indicate the write operation status. This bit is set high by application programs to initiate a write process and will be cleared by hardware if the write process is finished.

The Read operation can be carried out by executing a specific read procedure. The FRDEN bit is used to enable the read function and the FRD bit is used to initiate the read process by application programs and then indicate the read operation status. When the read process is finished, this bit will be cleared by hardware.

Operations	Format
Erase	256 words/block
Write	4 words/time
Read	1 word/time

**IAP Operation Format**

Erase Block	FARH[3:0]	FARL
0	0000	xxxx xxxx
1	0001	xxxx xxxx
2	0010	xxxx xxxx
3	0011	xxxx xxxx
4	0100	xxxx xxxx
5	0101	xxxx xxxx
6	0110	xxxx xxxx
7	0111	xxxx xxxx
8	1000	xxxx xxxx
9	1001	xxxx xxxx
10	1010	xxxx xxxx
11	1011	xxxx xxxx
12	1100	xxxx xxxx
13	1101	xxxx xxxx
14	1110	xxxx xxxx
15	1111	xxxx xxxx

"x": don't care

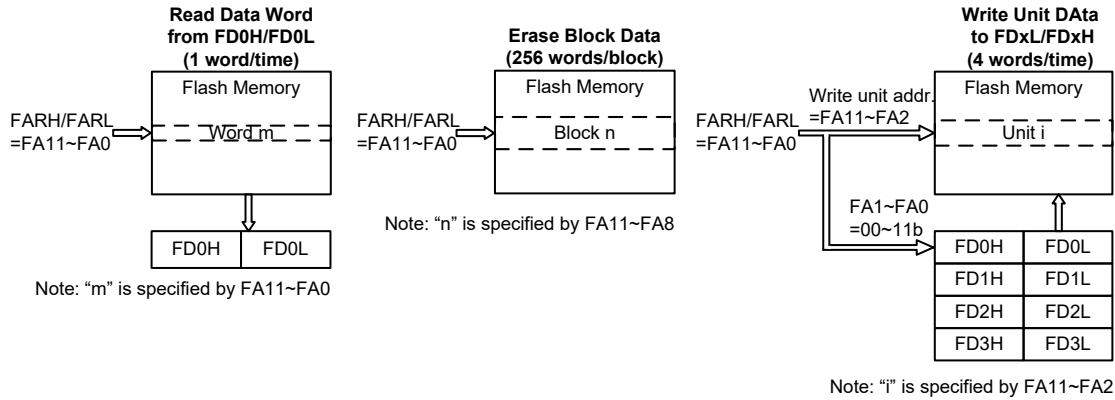
**Erase Block Number and Selection**

Write Unit	FARH[3:0]	FARL[7:2]	FARL[1:0]
0	0000	0000 00	xx
1	0000	0000 01	xx
2	0000	0000 10	xx
3	0000	0000 11	xx
4	0000	0001 00	xx
:	:	:	:
:	:	:	:
63	0000	1111 11	xx
64	0001	0000 00	xx
:	:	:	:
:	:	:	:
1022	1111	1111 10	xx
1023	1111	1111 11	xx

"x": don't care

**Write Unit Number and Selection**





Flash Memory IAP Read/Erase/Write Structure

### IAP Flash Program Memory Registers

There are two address registers, four 16-bit data registers and two control registers. Read and Write operations to the Flash memory are carried out using 16-bit data operations using the address and data registers and the control register. Several registers control the overall operation of the internal Flash Program Memory. The address registers are named FARL and FARH, the data registers are named  $FDnL$  and  $FDnH$  and the control registers are named FC0 and FC1.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	—	—	—	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Register List

#### • FC0 Register

Bit	7	6	5	4	3	2	1	0
Name	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
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 **CFWEN**: Flash Memory Erase/Write function enable control

0: Flash memory erase/write function is disabled

1: Flash memory erase/write function has been successfully enabled

When this bit is cleared to 0 by application program, the Flash memory erase/write function is disabled. Note that this bit cannot be set high by application programs. Writing a "1" into this bit results in no action. This bit is used to indicate the Flash memory erase/write function status. When this bit is set to 1 by the hardware, it means

that the Flash memory erase/write function is enabled successfully. Otherwise, the Flash memory erase/write function is disabled if the bit is zero.

- Bit 6~4    **FMOD2~FMOD0**: Flash memory Mode selection  
           000: Write Mode  
           001: Block Erase Mode  
           010: Reserved  
           011: Read Mode  
           100: Reserved  
           101: Reserved  
           110: Flash memory Erase/Write function Enable Mode  
           111: Reserved

These bits are used to select the Flash Memory operation modes. Note that the “Flash memory Erase/Write function Enable Mode” should first be successfully enabled before the Erase or Write Flash memory operation is executed.

- Bit 3        **FWPEN**: Flash memory Erase/Write function enable procedure Trigger  
           0: Erase/Write function enable procedure is not triggered or procedure timer times out  
           1: Erase/Write function enable procedure is triggered and procedure timer starts to count  
 This bit is used to activate the Flash memory Erase/Write function enable procedure and an internal timer. It is set by the application programs and then cleared by the hardware when the internal timer times out. The correct patterns must be written into the FD1L/FD1H, FD2L/FD2H and FD3L/FD3H register pairs respectively as soon as possible after the FWPEN bit is set high.

- Bit 2        **FWT**: Flash memory write initiate control  
           0: Do not initiate Flash memory write or indicating that a Flash memory write process has completed  
           1: Initiate Flash memory write process  
 This bit is set by software and cleared by the hardware when the Flash memory write process has completed.

- Bit 1        **FRDEN**: Flash memory read enable control  
           0: Flash memory read disable  
           1: Flash memory read enable  
 This is the Flash memory Read Enable Bit which must be set high before any Flash memory read operations are carried out. Clearing this bit to zero will inhibit Flash memory read operations.

- Bit 0        **FRD**: Flash memory read initiate control  
           0: Do not initiate Flash memory read or indicating that a Flash memory read process has completed  
           1: Initiate Flash memory read process  
 This bit is set by software and cleared by the hardware when the Flash memory read process has completed.

- Note: 1. The FWT, FRDEN and FRD bits cannot be set to “1” at the same time with a single instruction.  
 2. Ensure that the  $f_{SUB}$  clock is stable before executing the erase or write operation.  
 3. Note that the CPU will be stopped when a read, write or erase operation is successfully activated.  
 4. Ensure that the read, erase or write operation is totally complete before executing other operations.

• **FC1 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**: Chip Reset Pattern  
When a specific value of “55H” is written into this register, a reset signal will be generated to reset the whole chip.

• **FARL Register**

Bit	7	6	5	4	3	2	1	0
Name	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
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 **FA7~FA0**: Flash Memory Address bit 7 ~ bit 0

• **FARH Register**

Bit	7	6	5	4	3	2	1	0
Name	FA15	FA14	FA13	FA12	FA11	FA10	FA9	FA8
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 **FA15~FA8**: Flash Memory Address bit 15 ~ bit 8

• **FD0L 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**: The first Flash Memory data word bit 7 ~ bit 0  
Note that data written into the low byte data register FD0L will only be stored in the FD0L register and not loaded into the lower 8-bit write buffer.

• **FD0H 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**: The first Flash Memory data word bit 15 ~ bit 8  
Note that when 8-bit data is written into the high byte data register FD0H, the whole 16 bits of data stored in the FD0H and FD0L registers will simultaneously be loaded into the 16-bit write buffer after which the contents of the Flash memory address register pair, FARH and FARL, will be incremented by one.

• **FD1L 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**: The second Flash Memory data word bit 7 ~ bit 0

• **FD1H 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**: The second Flash Memory data word bit 15 ~ bit 8

• **FD2L 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**: The third Flash Memory data word bit 7 ~ bit 0

• **FD2H 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**: The third Flash Memory data word bit 15 ~ bit 8

• **FD3L 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**: The fourth Flash Memory data word bit 7 ~ bit 0

• **FD3H 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

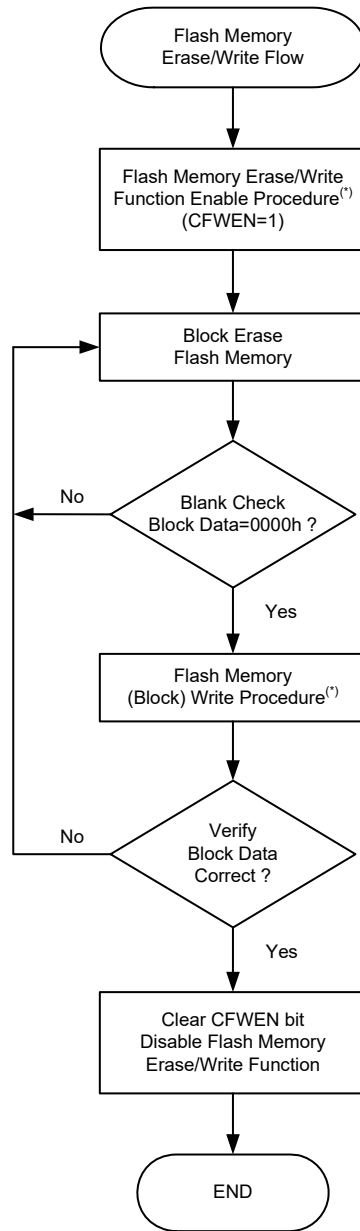
Bit 7~0      **D15~D8**: The fourth Flash Memory data word bit 15 ~ bit 8

### **Flash Memory Erase/Write Flow**

It is important to understand the Flash memory Erase/Write flow before the Flash memory contents are updated. Users can refer to the corresponding operation procedures when developing their IAP program to ensure that the Flash memory contents are correctly updated.

#### **Flash Memory Erase/Write Flow Descriptions**

1. Activate the “Flash Memory Erase/Write function enable procedure” first. When the Flash Memory Erase/Write function is successfully enabled, the CFWEN bit in the FC0 register will automatically be set high by hardware. After this, Erase or Write operations can be executed on the Flash memory. Refer to the “Flash Memory Erase/Write Function Enable Procedure” for details.
2. Configure the Flash memory address to select the desired erase block and then erase this block.
3. Execute a Blank Check operation to ensure whether the block erase operation is successful or not. The “TABRD” instruction should be executed to read the Flash memory contents and to check if the contents is 0000h or not. If the Flash memory block erase operation fails, users should go back to Step 2 and execute the block erase operation again.
4. Write data into the specific block. Refer to the “Flash Memory Write Procedure” for details.
5. Execute the “TABRD” instruction to read the Flash memory contents and check if the written data is correct or not. If the data read from the Flash memory is different from the written data, it means that the block write operation has failed, go back to Step 2 and execute the block erase operation again.
6. Clear the CFWEN bit to disable the Flash Memory Erase/Write function enable mode if the current block Erase and Write operations are complete if no more blocks need to be erased or written.



**Flash Memory Erase/Write Flow**

Note: The Flash Memory Erase/Write Function Enable procedure and Flash Memory Write procedure will be described in the following sections.

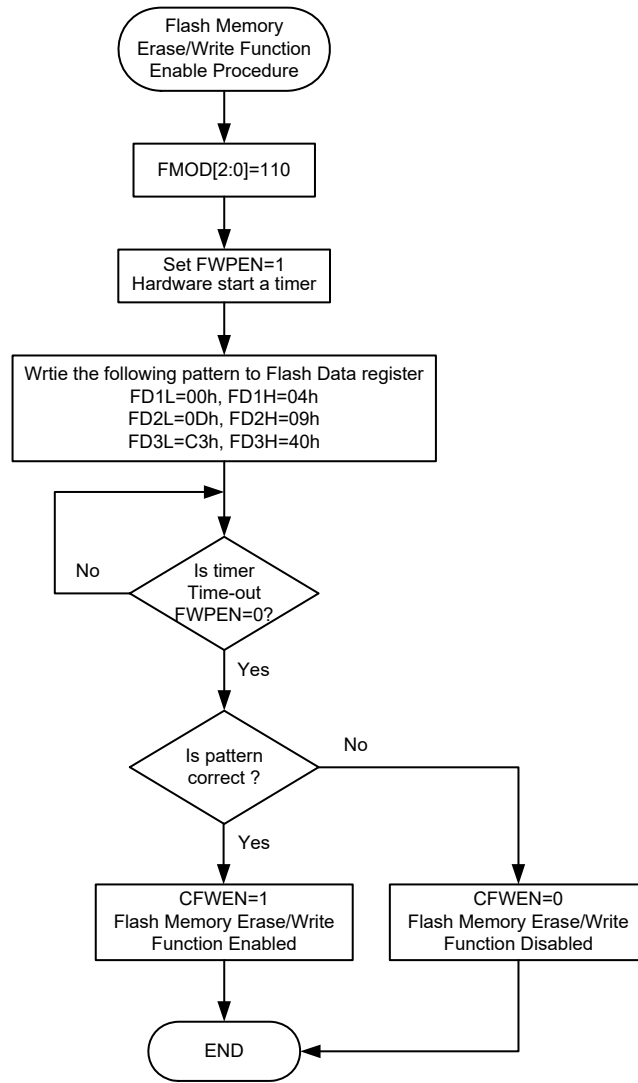
### **Flash Memory Erase/Write Function Enable Procedure**

The Flash Memory Erase/Write Function Enable Mode is specially designed to prevent the Flash memory contents from being wrongly modified. In order to allow users to change the Flash memory data using the IAP control registers, users must first enable the Flash memory Erase/Write function.

#### **Flash Memory Erase/Write Function Enable Procedure Description**

1. Write data “110” to the FMOD [2:0] bits in the FC0 register to select the Flash Memory Erase/Write Function Enable Mode.
2. Set the FWPEN bit in the FC0 register to “1” to activate the Flash Memory Erase/Write Function. This will also activate an internal timer.
3. Write the correct data pattern into the Flash data registers, FD1L~FD3L and FD1H~FD3H, as soon as possible after the FWPEN bit is set high. The enable Flash memory erase/write function data pattern is 00H, 0DH, C3H, 04H, 09H and 40H corresponding to the FD1L~FD3L and FD1H~FD3H registers respectively.
4. Once the timer has timed out, the FWPEN bit will automatically be cleared to 0 by hardware regardless of the input data pattern.
5. If the written data pattern is incorrect, the Flash memory erase/write function will not be enabled successfully and the above steps should be repeated. If the written data pattern is correct, the Flash memory erase/write function will be enabled successfully.
6. Once the Flash memory erase/write function is enabled, the Flash memory contents can be updated by executing the block erase and write operations using the IAP control registers.

To disable the Flash memory erase/write function, the CFWEN bit in the FC0 register can be cleared. There is no need to execute the above procedure.



**Flash Memory Erase/Write Function Enable Procedure**



### **Flash Memory Write Procedure**

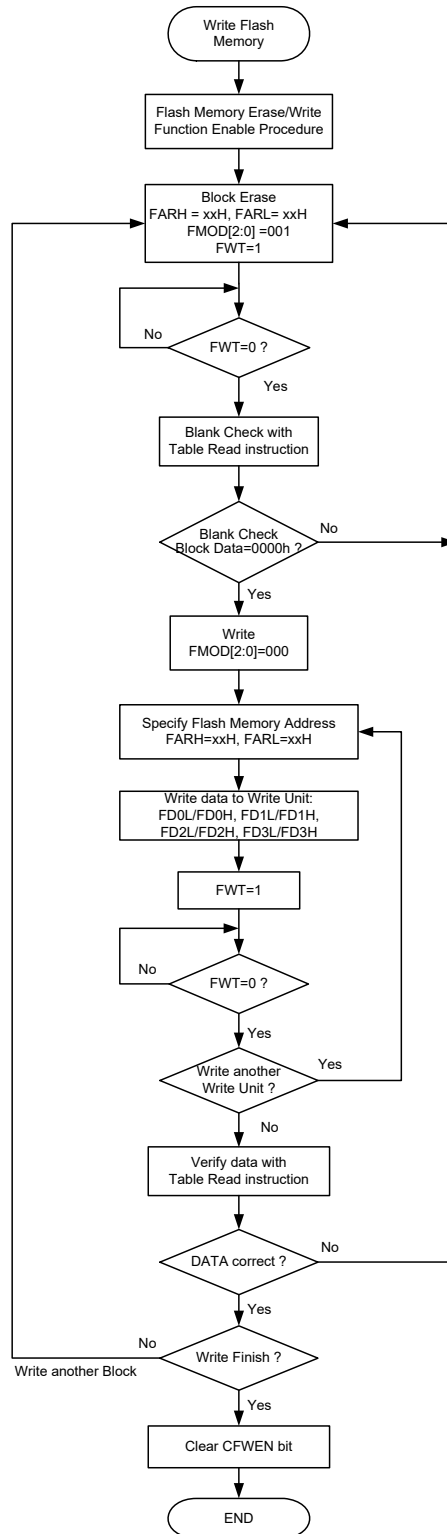
After the Flash memory erase/write function has been successfully enabled as the CFWEN bit is set high, the data to be written into the flash memory can be filled into the data registers. The selected Flash memory block data should be erased by properly configuring the IAP control registers before the data write procedure is executed.

The block size is 256 words, whose address is specified by the memory address bits, FA11~FA8.

### **Flash Memory Write Procedure Description**

For each write operation the desired write unit address should first be placed in the FARL and FARH registers and the data placed in the FD0L/FD0H~FD3L/FD3H registers. The number of the write operation is 4 words each time, therefore, the available write unit address is only specified by the FA11~FA2 bits in the FARH and FARL registers and the content of FA1~FA0 in the FARL register is not used to specify the unit address.

1. Activate the “Flash Memory Erase/Write function enable procedure”. Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the “Flash Memory Erase/Write function enable procedure” for more details.
2. Set the FMOD field to “001” to select the erase operation. Set the FWT bit high to erase the desired block which is specified by the FARH and FARL registers. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.  
Go to step 2 if the erase operation is not successful.  
Go to step 4 if the erase operation is successful.
4. Set the FMOD field to “000” to select the write operation.
5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L/FD0H ~ FD3L/FD3H registers.
6. Set the FWT bit high to write the data words to the Flash memory at four consecutive addresses starting from FARL[1:0]=00b. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.  
If the write operation has not successfully completed, then go to step 2.  
Go to step 8 if the write operation is successful.
8. Clear the CFWEN bit low to disable the Flash memory erase/write function.



**Flash Memory Write Procedure**

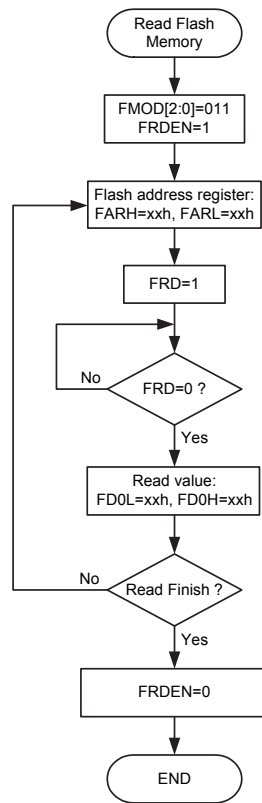
Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.  
2. It will take a typical time of 2.2ms for the FWT bit state changing from high to low.

**Important Points to Note for Flash Memory Write Operations**

1. The “Flash Memory Erase/Write Function Enable Procedure” must be successfully activated before the Flash Memory erase/write operation is executed.
2. The Flash Memory erase operation is executed to erase a whole block.
3. After the data is written into the Flash memory the Flash memory contents must be read out using the table read instruction, TABRD, and checked if it is correct or not. If the data written into the Flash memory is incorrect, erase the block and then activate a write operation on the same Flash memory block. The data check, block erase and data re-write steps should be repeatedly executed until the data written into the Flash memory is correct.
4. The system frequency should be setup to the maximum application frequency when data write and data check operations are executed using the IAP function.

**Flash Memory Read Procedure**

To activate the Flash Memory Read procedure, the FMOD field should be set to “011” to select the Flash memory read mode and the FRDEN bit should be set high to enable the read function. The desired Flash memory address should be written into the FARH and FARL registers and then the FRD bit should be set high. After this the Flash memory read operation will be activated. The data stored in the specified address can be read from the data registers, FD0H and FD0L, when the FRD bit goes low. There is no need to first activate the Flash Memory Erase/Write Function Enable Procedure before the Flash memory read operation is executed.



**Flash Memory Read Procedure**

- Note: 1. When the read operation is successfully activated, all CPU operations will temporarily cease.  
 2. It will take a typical time of three instruction cycles for the FRD bit state changing from high to low.

## Data Memory

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

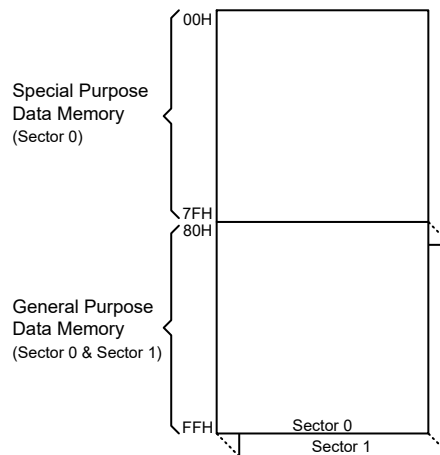
### Structure

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

The Data Memory is subdivided into two sectors, all of which are implemented in 8-bit wide RAM. The Special Purpose Data Memory registers are accessible in Sector 0. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH in Sector 0 while the General Purpose Data Memory address range is from 80H to FFH in Sector 0 and Sector 1. Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value if using indirect addressing method.

Special Purpose Data Memory	General Purpose Data Memory	
Located Sectors	Capacity	Sector: Address
Sector 0	256×8	Sector 0: 80H~FFH Sector 1: 80H~FFH

**Data Memory Summary**



**Data Memory Structure**

### **Data Memory Addressing**

For the device that supports the extended instructions, the desired Data Memory Sector is selected by the MP1H or MP2H register and the certain Data Memory address in the pointed sector is specified by the MP1L or MP2L register when using indirect addressing access.

Direct Addressing can be used in all sectors using the corresponding instructions which can address all available data memory space. For the accessed data memory which is located in any data memory sector except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address “m” in the extended instructions has 9 valid bits for this device, the high byte indicates a sector and the low byte indicates a specific address.

### **General Purpose Data Memory**

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

### **Special Purpose Data Memory**

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

Sector 0	
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1L
04H	MP1H
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	
0CH	IAR2
0DH	MP2L
0EH	MP2H
0FH	RSTFC
10H	SCC
11H	HIRCC
12H	LVRC
13H	WDTC
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	PMS
19H	PMC
1AH	
1BH	
1CH	PSCR
1DH	INTEG
1EH	INTC0
1FH	INTC1
20H	INTC2
21H	MFI0
22H	MFI1
23H	MFI2
24H	
25H	SADOL
26H	SADOH
27H	SADC0
28H	SADC1
29H	
2AH	
2BH	
2CH	PTMC0
2DH	PTMC1
2EH	PTMDL
2FH	PTMDH
30H	PTMAL
31H	PTMAH
32H	PTMRPL
33H	PTMRPH
34H	
35H	PAS0
36H	PAS1
37H	IFS
38H	TB0C
39H	TB1C
3AH	CTMC0
3BH	CTMC1
3CH	CTMDL
3DH	CTMDH
3EH	CTMAL
3FH	CTMAH

Sector 0	
40H	CTMRP
41H	
42H	STMC0
43H	STMC1
44H	STMDL
45H	STMDH
46H	STMAL
47H	STMAH
48H	STMRP
49H	
4AH	FC0
4BH	FC1
4CH	FARL
4DH	FARH
4EH	FD0L
4FH	FD0H
50H	FD1L
51H	FD1H
52H	FD2L
53H	FD2H
54H	FD3L
55H	FD3H
56H	
57H	NF_VIH
58H	NF_VIL
59H	
5AH	USR
5BH	UCR1
5CH	UCR2
5DH	BRDH
5EH	BRDL
5FH	UFCR
60H	TXR_RXR
61H	RxCNT
62H	
63H	
64H	
65H	
66H	
67H	
68H	
69H	
6AH	
6BH	
6CH	
6DH	
6EH	
6FH	
70H	
71H	
72H	
73H	
74H	
75H	
76H	
77H	
78H	
79H	
7AH	
7BH	
7CH	
7DH	
7EH	
7FH	

: Unused, read as 00H

: Reserved, cannot be changed

**Special Purpose Data Memory**

## Special Function Register Description

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

### Indirect Addressing Registers – IAR0, IAR1, IAR2

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

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

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

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

#### Indirect Addressing Program Example 1

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

### Indirect Addressing Program Example 2

```

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

```

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

### Direct Addressing Program Example using extended instructions

```

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

```

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

## Accumulator – ACC

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



### **Program Counter Low Register – PCL**

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

### **Look-up Table Registers – TBLP, TBHP, TBLH**

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

### **Status Register – STATUS**

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

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

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

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

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

• **STATUS Register**

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

“x”: unknown

- Bit 7     **SC**: The result of the “XOR” operation which is performed by the OV flag and the MSB of the instruction operation result.
- Bit 6     **CZ**: The operational result of different flags for different instructions.  
 For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.  
 For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the “AND” operation result which is performed by the previous operation CZ flag and current operation Z flag.  
 For other instructions, the CZ flag will not be affected.
- Bit 5     **TO**: Watchdog Time-out flag  
 0: After power up or executing the “CLR WDT” or “HALT” instruction  
 1: A watchdog time-out occurred
- Bit 4     **PDF**: Power down flag  
 0: After power up or executing the “CLR WDT” instruction  
 1: By executing the “HALT” instruction
- Bit 3     **OV**: Overflow flag  
 0: No overflow  
 1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa
- Bit 2     **Z**: Zero flag  
 0: The result of an arithmetic or logical operation is not zero  
 1: The result of an arithmetic or logical operation is zero
- Bit 1     **AC**: Auxiliary flag  
 0: No auxiliary carry  
 1: An operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction
- Bit 0     **C**: Carry flag  
 0: No carry-out  
 1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation  
 The “C” flag is also affected by a rotate through carry instruction.

## Oscillators

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

### Oscillator Overview

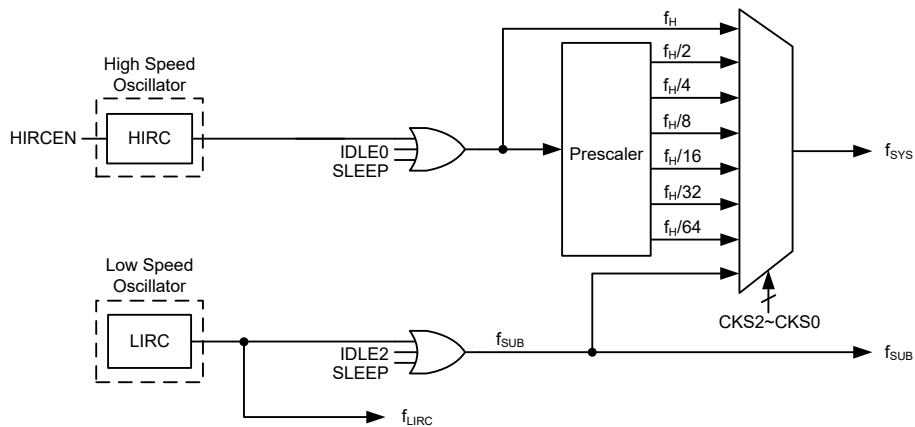
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators.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

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

Oscillator Types

### System Clock Configurations

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



System Clock Configurations

### Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 16MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

### 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.

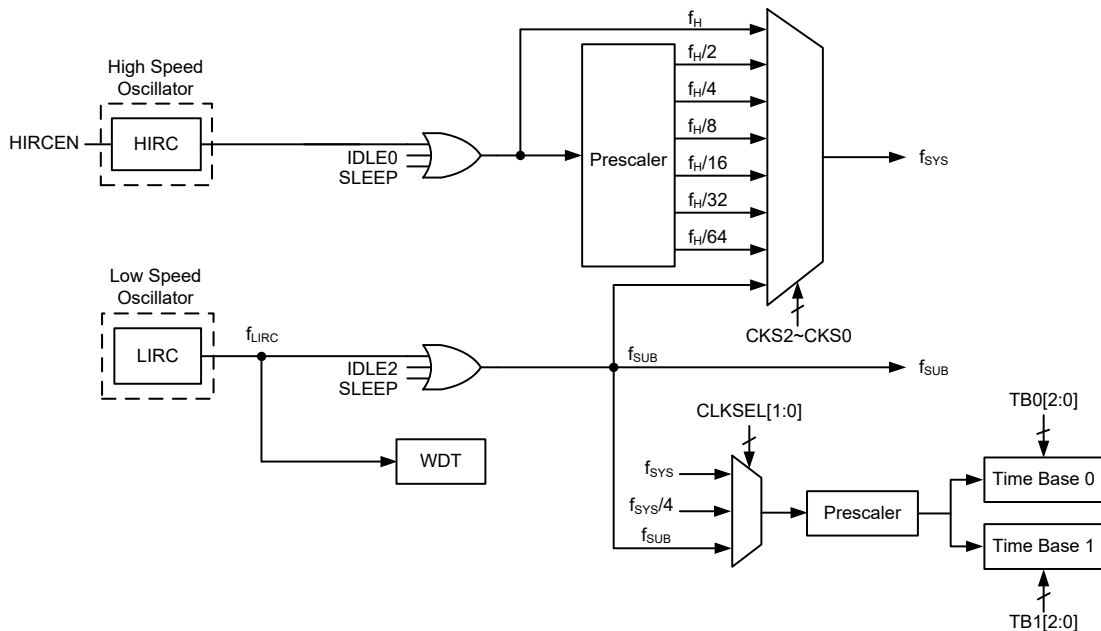
## Operating Modes and System Clocks

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

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



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.

## System Operation Modes

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

Operation Mode	CPU	Register Setting			f <sub>sys</sub>	f <sub>H</sub>	f <sub>sub</sub>	f <sub>LIRC</sub>
		FHIDEN	FSIDEN	CKS2~CKS0				
FAST	On	x	x	000~110	f <sub>H</sub> ~f <sub>H</sub> /64	On	On	On
SLOW	On	x	x	111	f <sub>sub</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<sub>H</sub> clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.

2. The f<sub>LIRC</sub> clock can be on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

### FAST Mode

This is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source will come from the HIRC oscillator. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

### SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f<sub>sub</sub>. The f<sub>sub</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<sub>H</sub> clock will be switched on or off by configuring the corresponding oscillator enable bit HIRCEN.

### SLEEP Mode

The SLEEP Mode is entered when a 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>sub</sub> clock provided to peripheral will be stopped too. However the f<sub>LIRC</sub> clock can still continue to operate if the WDT function is enabled. The f<sub>LIRC</sub> clock will be stopped too, if the Watchdog Timer function is disabled.

### IDLE0 Mode

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

### IDLE1 Mode

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

### IDLE2 Mode

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

## Control Registers

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

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

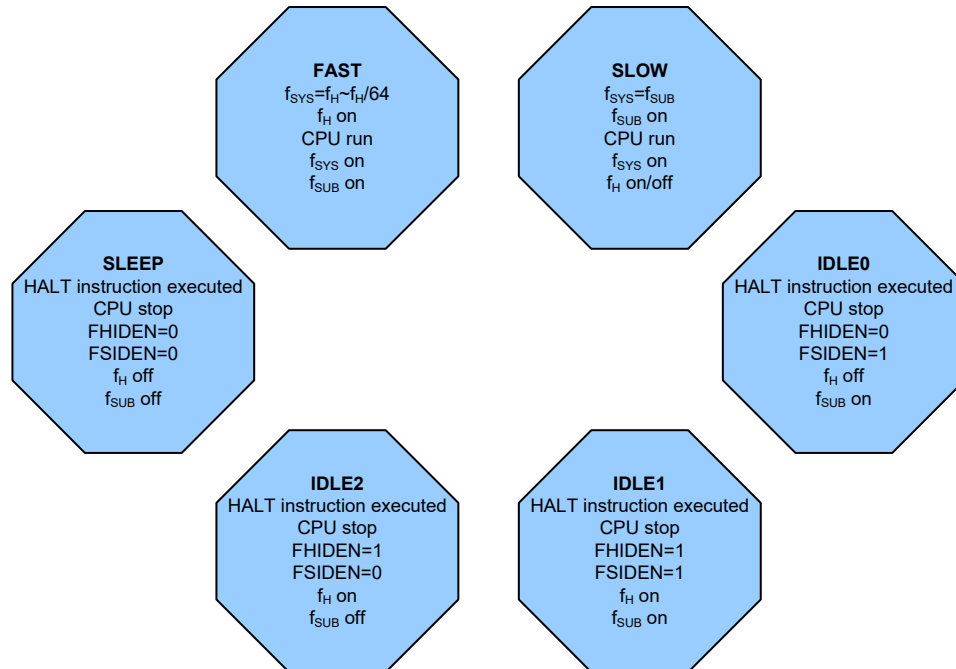
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

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.

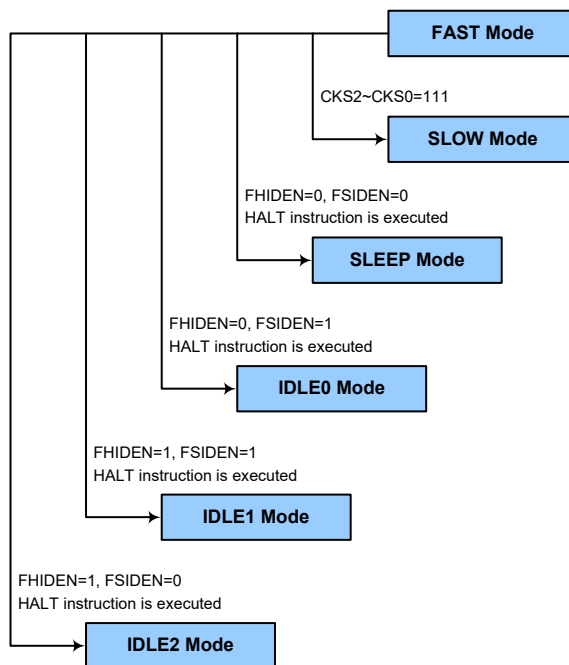
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



### FAST Mode to SLOW Mode Switching

When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by 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.

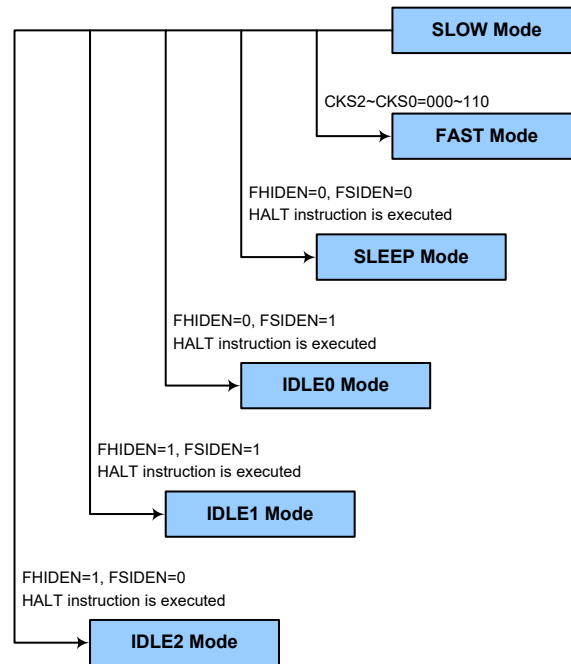


### SLOW Mode to FAST Mode Switching

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

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 FAST mode from the SLOW Mode. This is monitored using the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the System Start Up Time Characteristics.





### Entering the SLEEP Mode

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

- The system clock will be stopped and the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and 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 the 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 devices which have 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,  $t_{\text{RESET}}$ , and the WRF bit in the RSTFC register will be set high.

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

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

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

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

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

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

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

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

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

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,  $t_{\text{RESET}}$ . After power on these bits will have a value of 01010B.

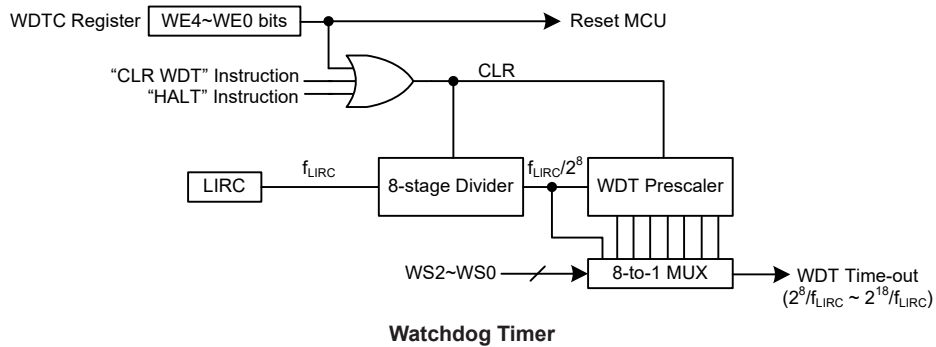
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

**Watchdog Timer Function Control**

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the STATUS register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a 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.

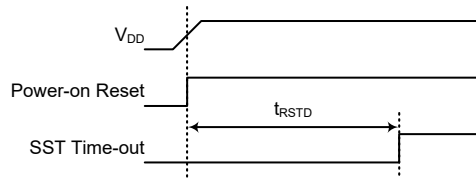
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

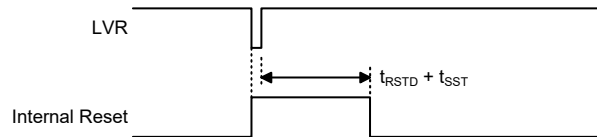


Note:  $t_{RSTD}$  is power-on delay specified in System Start Up Time Characteristics

**Power-On Reset Timing Chart**

**Low Voltage Reset – LVR**

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level. The LVR function is always enabled in the FAST or Slow mode with a specific LVR voltage  $V_{LVR}$ . If the supply voltage of the device drops to within a range of  $0.9V \sim V_{LVR}$  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 2.8V. Note that the LVR function will be automatically disabled when the device enters the IDLE or SLEEP mode.



Note:  $t_{RSTD}$  is power-on delay specified in System Start Up Time Characteristics

**Low Voltage Reset Timing Chart**

• **LVRC Register**

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

Bit 7~4 Unimplemented, read as “0”

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

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

Bit 2~0 Unimplemented, read as “0”

• **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”

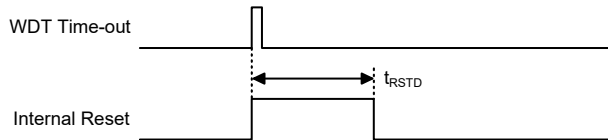
- Bit 2      **LVRF**: LVR function reset flag  
             0: Not occur  
             1: Occurred  
             This bit is set high when a specific Low Voltage Reset situation occurs. This bit can only be cleared to zero by the application program.
- Bit 1      Unimplemented, read as “0”
- Bit 0      **WRF**: WDT Control register software reset flag  
             Described in the Watchdog Timer Control Register section

**IAP Reset**

When a specific value of “55H” is written into the FC1 register, a reset signal will be generated to reset the whole device. Refer to the IAP section for more associated details.

**Watchdog Time-out Reset during Normal Operation**

When the Watchdog time-out Reset during normal operation occurs, the Watchdog time-out flag TO will be set to “1”.

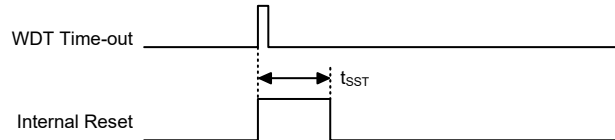


Note:  $t_{RSTD}$  is power-on delay specified in System Start Up Time Characteristics

**WDT Time-out Reset during Normal Operation Timing Chart**

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

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



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

**Reset Initial Conditions**

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

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

“u”: unchanged

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

Item	Condition after Reset
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time 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	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	uuuu uuuu
ACC	x xxx x xxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	x xxx x xxx	uuuu uuuu	uuuu uuuu
TBLH	x xxx x xxx	uuuu uuuu	uuuu uuuu
TBHP	- - - - x xxx	- - - - uuuu	- - - - uuuu
STATUS	xx00 x xxx	uu1u uuuu	uu11 uuuu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	- - - - -x-0	- - - - -u-u	- - - - -u-u
SCC	000- --00	000- --00	uuu- --uu
HIRCC	- - - - --01	- - - - --01	- - - - --uu
LVRC	- - - - 0 - - -	- - - - 0 - - -	- - - - u - - -
WDTC	0101 0011	0101 0011	uuuu 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
PMC	- - - - 0000	- - - - 0000	- - - - uuuu
PSCR	- - - - --00	- - - - --00	- - - - --uu
INTEG	- - - - --00	- - - - --00	- - - - --uu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	- - - - ---0	- - - - ---0	- - - - ---u
MFI0	--00 --00	--00 --00	--uu --uu
MFI1	--00 --00	--00 --00	--uu --uu



Register	Reset (Power On)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE or SLEEP)
MF12	-000 -000	-000 -000	-uuu -uuu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRF5=0)
			uuuu uuuu (ADRF5=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRF5=0)
			---- uuuu (ADRF5=1)
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	uuuu uuuu
PTMC0	0000 0---	0000 0---	uuuu u---
PTMC1	0000 0000	0000 0000	uuuu uuuu
PTMDL	0000 0000	0000 0000	uuuu uuuu
PTMDH	---- --00	---- --00	---- --uu
PTMAL	0000 0000	0000 0000	uuuu uuuu
PTMAH	---- --00	---- --00	---- --uu
PTMRPL	0000 0000	0000 0000	uuuu uuuu
PTMRPH	---- --00	---- --00	---- --uu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
IFS0	---- --00	---- --00	---- --uu
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
CTMC0	0000 0---	0000 0---	uuuu u---
CTMC1	0000 0000	0000 0000	uuuu uuuu
CTMDL	0000 0000	0000 0000	uuuu uuuu
CTMDH	0000 0000	0000 0000	uuuu uuuu
CTMAL	0000 0000	0000 0000	uuuu uuuu
CTMAH	0000 0000	0000 0000	uuuu uuuu
CTMRP	0000 0000	0000 0000	uuuu uuuu
STMC0	0000 0---	0000 0---	uuuu u---
STMC1	0000 0000	0000 0000	uuuu uuuu
STMDL	0000 0000	0000 0000	uuuu uuuu
STMDH	0000 0000	0000 0000	uuuu uuuu
STMAL	0000 0000	0000 0000	uuuu uuuu
STMAH	0000 0000	0000 0000	uuuu uuuu
STMRP	0000 0000	0000 0000	uuuu uuuu
FC0	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	uuuu uuuu
FARL	0000 0000	0000 0000	uuuu uuuu
FARH	---- 0000	---- 0000	---- uuuu
FD0L	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	uuuu uuuu
NF_VIH	00-1 1001	00-1 1001	uu-u uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE or SLEEP)
NF_VIL	00-0 1010	00-0 1010	uu-u uuuuu
USR	0000 1011	0000 1011	uuuu uuuuu
UCR1	0000 00x0	0000 00x0	uuuu uuuuu
UCR2	0000 0000	0000 0000	uuuu uuuuu
BRDH	0000 0000	0000 0000	uuuu uuuuu
BRDL	0000 0000	0000 0000	uuuu uuuuu
UFCR	--00 0000	--00 0000	--uu uuuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuuu
RxCNT	---- -000	---- -000	---- -uuu

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 name PA. 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

**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, and are implemented using weak PMOS transistors.

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

• **PAPU Register**

Bit	7	6	5	4	3	2	1	0
Name	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0      **PAPU7~PAPU0**: PA7~PA0 pull-high function control  
 0: Disable  
 1: Enable

For the PA5 and PA7 lines which include an internal pull-low resistor, if their pull-high functions are enabled, additional power consumption will be required. These two lines are internally connected to the H-bridge driver inputs, IN1 and IN2, respectively which should be properly configured to control the H-bridge driver.

Special attention has to be paid to the unbonded line in package 10-pin SOP-EP, PA6, which should either be set as output or if set as input must have pull-high resistor connected to avoid unwanted power consumption result from floating input conditions.

**Port A Wake-up**

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

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

• **PAWU Register**

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

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

**I/O Port Control Registers**

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

• **PAC Register**

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

Bit 7~0     **PAC7~PAC0:** PA7~PA0 pin type selection  
 0: Output  
 1: Input

For the PA5 and PA7 lines which include an internal pull-low resistor, if their pull-high functions are enabled, additional power consumption will be required. These two lines are internally connected to the H-bridge driver inputs, IN1 and IN2, respectively which should be properly configured to control the H-bridge driver.

Special attention has to be paid to the unbonded line in package 10-pin SOP-EP, PA6, which should either be set as output or if set as input must have pull-high resistor connected to avoid unwanted power consumption result from floating input conditions.

**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 Output Function Selection register “n”, labeled as PASn, and Input Function Selection register, labeled as IFS, which can select the desired functions of the multi-function pin-shared pins.

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
IFS	—	—	—	—	—	—	IFS1	IFS0

**Pin-shared Function Selection Register List**

• **PAS0 Register**

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

- Bit 7~6     **PAS07~PAS06:** PA3 pin-shared function selection  
 00: PA3/PTPI  
 01: Reserved  
 10: TX  
 11: AN0
- Bit 5~4     **PAS05~PAS04:** PA2 pin-shared function selection  
 00: PA2/STPI  
 01: STP  
 10: STPB  
 11: AN1
- Bit 3~2     **PAS03~PAS02:** PA1 pin-shared function selection  
 00: PA1/STCK/NFIN  
 01: TX  
 10: RX  
 11: VREF
- Bit 1~0     **PAS01~PAS00:** PA0 pin-shared function selection  
 00: PA0/NFIN/INT  
 01: CTP  
 10: CTPB  
 11: AN2

• **PAS1 Register**

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

- Bit 7~6     **PAS17~PAS16:** PA7 pin-shared function selection  
 00: PA7  
 01: PA7  
 10: PA7  
 11: GT0
- Bit 5~4     **PAS15~PAS14:** PA6 pin-shared function selection  
 00: PA6  
 01: PA6  
 10: PA6  
 11: Reserved
- Bit 3~2     **PAS13~PAS12:** PA5 pin-shared function selection  
 00: PA5  
 01: PA5  
 10: PA5  
 11: GT1
- Bit 1~0     **PAS11~PAS10:** PA4 pin-shared function selection  
 00: PA4/CTCK  
 01: RX  
 10: CTPB  
 11: Reserved

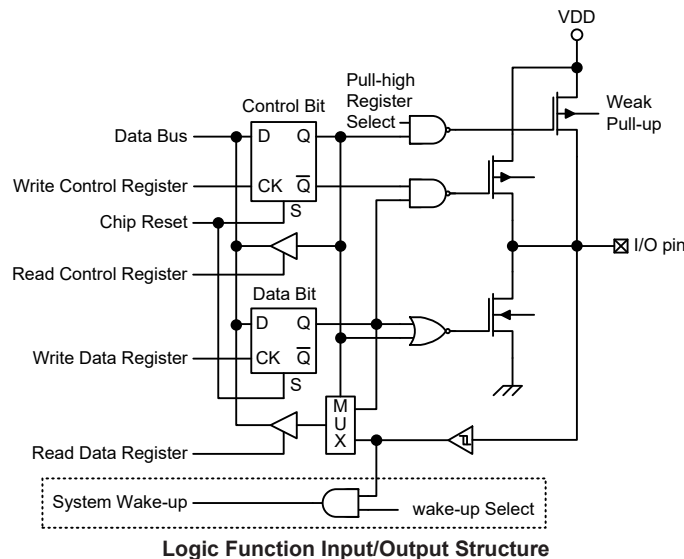
• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	IFS1	IFS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

- Bit 7~2      Unimplemented, read as “0”
- Bit 1        **IFS1**: RX input source pin selection  
              0: PA1  
              1: PA4
- Bit 0        **IFS0**: NFIN input source pin selection  
              0: PA1  
              1: PA0

**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 Modules – TM

One of the most fundamental functions in any microcontroller devices is the ability to control and measure time. To implement time related functions the device includes several Timer Modules, generally abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

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

### Introduction

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

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

**TM Function Summary**

### TM Operation

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

### TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTCK2~xTCK0 bits in the xTM control registers, where “x” stands for C, S or P type TM. The clock source can be a ratio of the system clock  $f_{SYS}$  or the internal high clock  $f_{H}$ , the  $f_{SUB}$  clock source or the external xTCK pin. The xTCK pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

## TM Interrupts

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

## TM External Pins

Each of the TMs, irrespective of what type, has one or two TM input pins, with the label xTCK and xTPI. One of the TM input pins, CTCK or STCK, is essentially a clock source for the CTM or STM and is selected using the CTCK2~CTCK0 or STCK2~STCK0 bits in the CTMC0 or STMC0 register respectively. This external TM input pin allows an external clock source to drive the internal TM. The CTCK or STCK input pin can be chosen to have either a rising or falling active edge. The STCK pin is also used as the external trigger input pin in single pulse output mode for the STM.

The other TM input pin, STPI or PTPI, is the capture input for the STM or PTM 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 STIO1~STIO0 or PTIO1~PTIO0 bits in the STMC1 or PTMC1 register respectively.

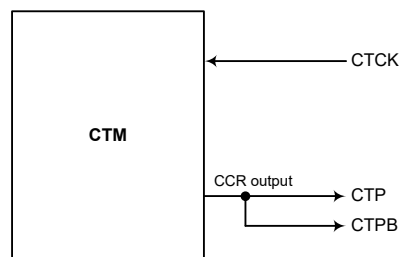
The Compact and Standard type TMs each has two output pins with the label CTP/STP and CTPB/STPB. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external output pins are also the pins where the TM generates the PWM output waveform. Note that the Periodic type TM only has one output signal, PTP.

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 register. The details of the pin-shared function selection are described in the pin-shared function section.

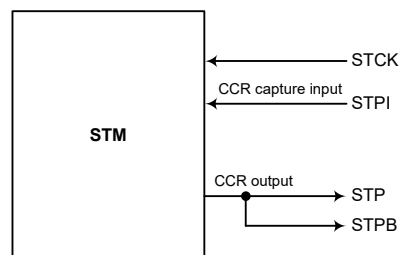
CTM		STM		PTM	
Input	Output	Input	Output	Input	Output
CTCK	CTP, CTPB	STCK, STPI	STP, STPB	PTPI	—

Note: The PTM output signal PTP is not bounded to the external package pin.

### TM External Pins

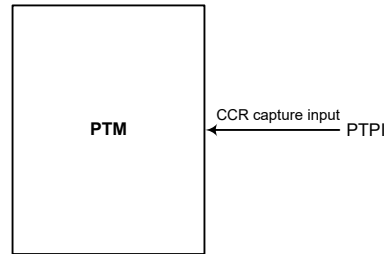


**CTM Function Pin Block Diagram**



**STM Function Pin Block Diagram**



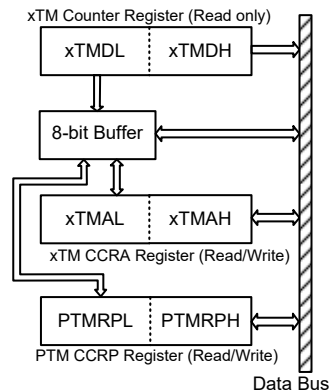


PTM Function Pin Block Diagram

### Programming Considerations

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

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

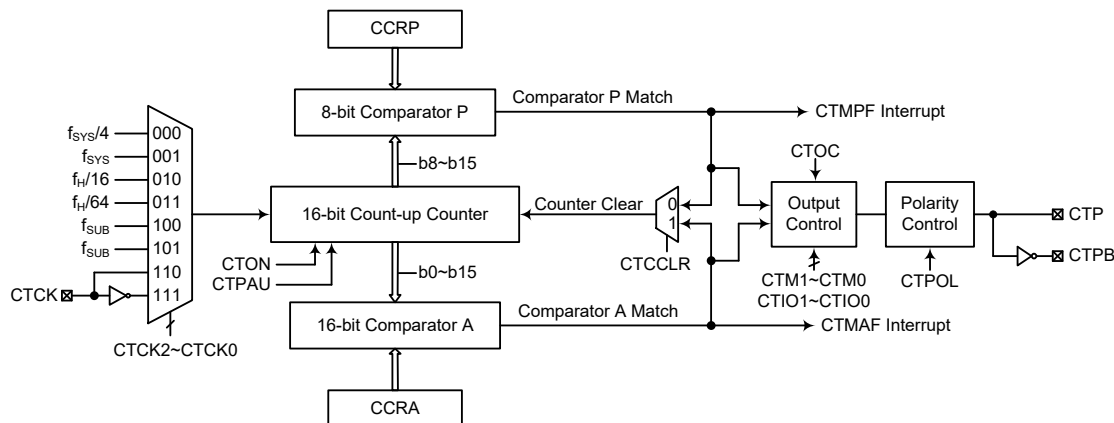


The following steps show the read and write procedures:

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

## Compact Type TM – CTM

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



Note: The CTM external pins are pin-shared with other functions, so before using the CTM function, the pin-shared function registers must be set properly to enable the CTM pin function.

**Compact Type TM Block Diagram**

### Compact Type TM Operation

The Compact Type TM core is a 16-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is 8-bit wide whose value is compared with the highest eight bits in the counter while the CCRA is 16-bit wide and therefore compares with all counter bits.

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

### Compact Type TM Register Description

Overall operation of the Compact Type TM is controlled using a series of registers. A read only register pair exists to store the internal counter 16-bit value, while a read/write register pair exists to store the internal 16-bit CCRA value. The CTMRP register is used to store the 8-bit CCRP value. The remaining two registers are control registers which setup the different operating and control modes.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CTMC0	CTPAU	CTCK2	CTCK1	CTCK0	CTON	—	—	—
CTMC1	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
CTMDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMDH	D15	D14	D13	D12	D11	D10	D9	D8
CTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMAH	D15	D14	D13	D12	D11	D10	D9	D8
CTMRP	CTRP7	CTRP6	CTRP5	CTRP4	CTRP3	CTRP2	CTRP1	CTRP0

16-bit Compact Type TM Register List

• CTMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	CTPAU	CTCK2	CTCK1	CTCK0	CTON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **CTPAU**: CTM Counter Pause control

0: Run  
1: Pause

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

Bit 6~4 **CTCK2~CTCK0**: Select CTM Counter clock

000:  $f_{SYS}/4$   
001:  $f_{SYS}$   
010:  $f_H/16$   
011:  $f_H/64$   
100:  $f_{SUB}$   
101:  $f_{SUB}$   
110: CTCK rising edge clock  
111: CTCK falling edge clock

These three bits are used to select the clock source for the CTM. 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 **CTON**: CTM Counter On/Off control

0: Off  
1: On

This bit controls the overall on/off function of the CTM. Setting the bit high enables the counter to run while clearing the bit disables the CTM. Clearing this bit to zero will stop the counter from counting and turn off the CTM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the CTM is in the Compare Match Output Mode or the PWM Output Mode then the CTM output pin will be reset to its initial condition, as specified by the CTOC bit, when the CTON bit changes from low to high.

Bit 2~0 Unimplemented, read as “0”

• **CTMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6     **CTM1~CTM0**: Select CTM Operating Mode

- 00: Compare Match Output Mode
- 01: Undefined
- 10: PWM Output Mode
- 11: Timer/Counter Mode

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

Bit 5~4     **CTIO1~CTIO0**: Select CTM function

Compare Match Output Mode

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

PWM Output Mode

- 00: PWM output inactive state
- 01: PWM output active state
- 10: PWM output
- 11: Undefined

Timer/Counter Mode

Unused

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

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

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

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

Compare Match Output Mode

- 0: Initial low
- 1: Initial high

PWM Output Mode/Single Pulse Output Mode

- 0: Active low
- 1: Active high

This is the output control bit for the CTM output pin. Its operation depends upon whether CTM is being used in the Compare Match Output Mode or in the PWM Output Mode. It has no effect if the CTM is in the Timer/Counter Mode. In the

Compare Match Output Mode it determines the logic level of the CTM output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low.

Bit 2 **CTPOL**: CTM CTP Output polarity control

- 0: Non-invert
- 1: Invert

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

Bit 1 **CTDPX**: CTM PWM duty/period control

- 0: CCRP – period; CCRA – duty
- 1: CCRP – duty; CCRA – period

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

Bit 0 **CTCCLR**: CTM Counter Clear condition selection

- 0: CTM Comparator P match
- 1: CTM Comparator A match

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

• **CTMDL Register**

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

Bit 7~0 **D7~D0**: CTM Counter Low Byte Register bit 7 ~ bit 0  
 CTM 16-bit Counter bit 7 ~ bit 0

• **CTMDH 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**: CTM Counter High Byte Register bit 7~ bit 0  
 CTM 16-bit Counter bit 15 ~ bit 8

• **CTMAL Register**

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

Bit 7~0 **D7~D0**: CTM CCRA Low Byte Register bit 7 ~ bit 0  
 CTM 16-bit CCRA bit 7 ~ bit 0

• **CTMAH 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:** CTM CCRA High Byte Register bit 7 ~ bit 0  
CTM 16-bit CCRA bit 15 ~ bit 8

• **CTMRP register**

Bit	7	6	5	4	3	2	1	0
Name	CTRP7	CTRP6	CTRP5	CTRP4	CTRP3	CTRP2	CTRP1	CTRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0      **CTRP7~CTRP0:** CTM CCRP 8-bit register, compared with the CTM Counter bit 15 ~ bit 8

Comparator P Match Period=  
0: 65536 CTM clocks  
1~255: 256×(1~255) CTM clocks

These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight bits. The result of this comparison can be selected to clear the internal counter if the CTCCLR bit is set to zero. Setting the CTCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

## Compact Type TM Operating Modes

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

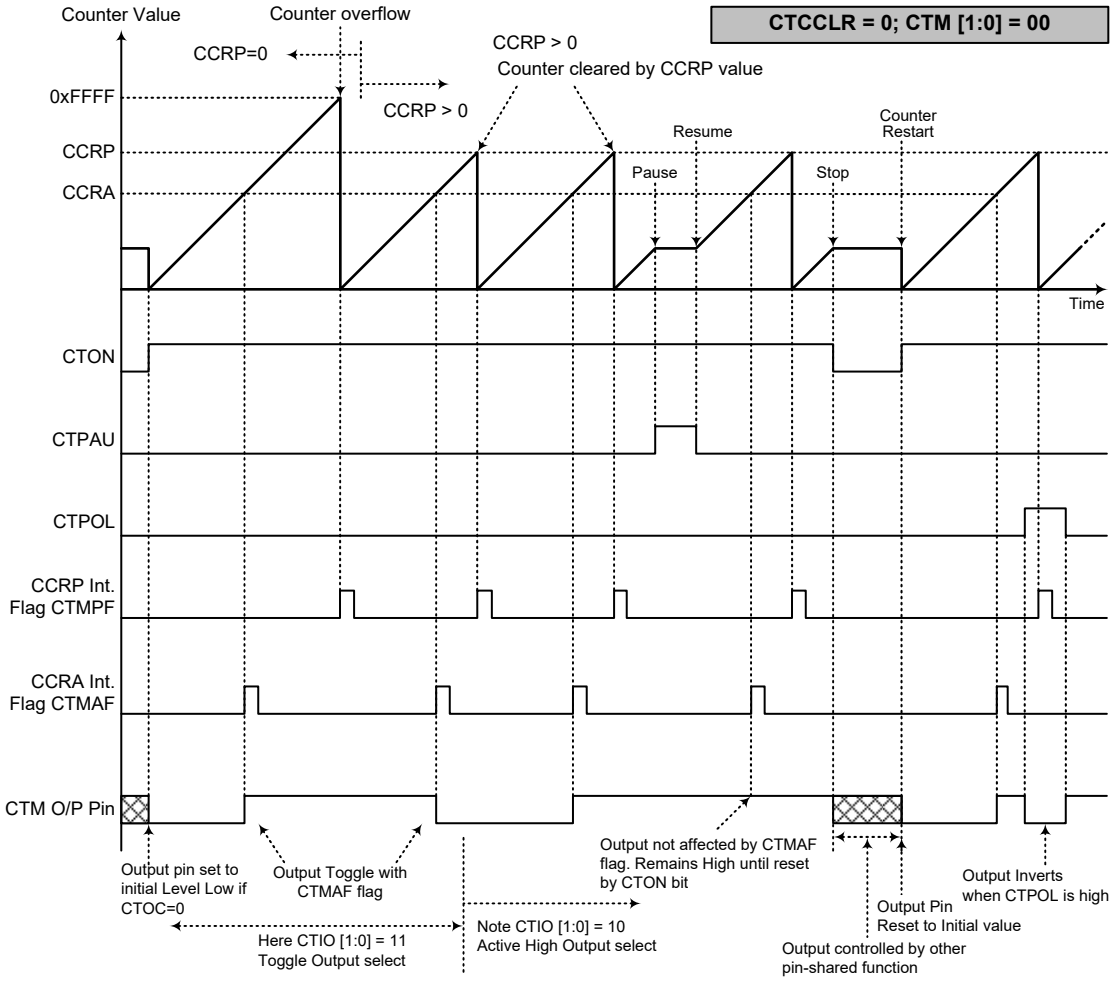
### Compare Match Output Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match occurs from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMAF and CTMPF interrupt request flags for the Comparator A and Comparator P respectively, will both be generated.

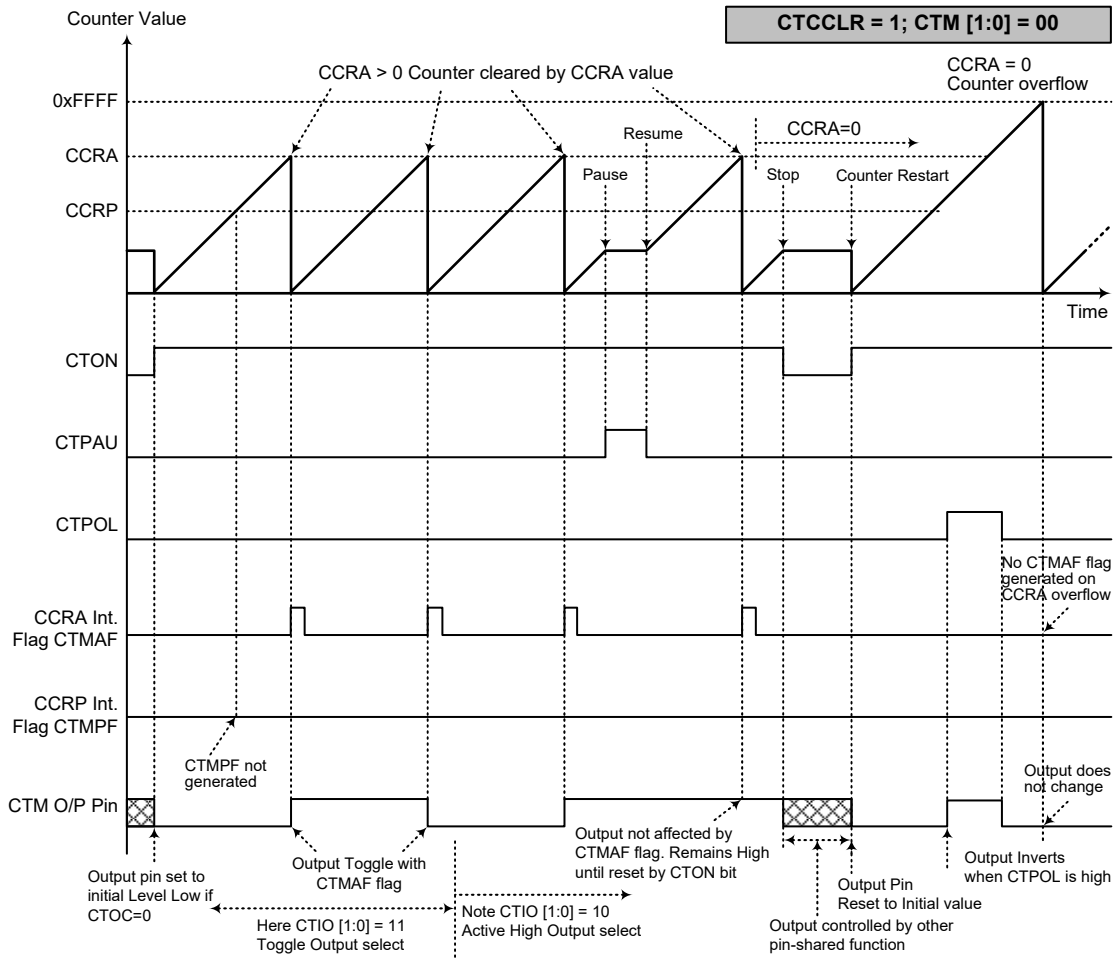
If the CTCCLR bit in the CTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTCCLR is high no CTMPF interrupt request flag will be generated.

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

As the name of the mode suggests, after a comparison is made, the CTM output pin will change state. The CTM output pin condition however only changes state when a CTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTM output pin. The way in which the CTM output pin changes state are determined by the condition of the CTIO1 and CTIO0 bits in the CTMC1 register. The CTM output pin can be selected using the CTIO1 and CTIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the CTM output pin, which is setup after the CTON bit changes from low to high, is setup using the CTCOC bit. Note that if the CTIO1 and CTIO0 bits are zero then no pin change will take place.



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



**Compare Match Output Mode – CTCCLR=1**

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



**Timer/Counter Mode**

To select this mode, bits CTM1 and CTM0 in the CTMC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the CTM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the CTM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared functions.

**PWM Output Mode**

To select this mode, bits CTM1 and CTM0 in the CTMC1 register should be set to 10 respectively. The PWM function within the CTM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the CTM output 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 CTCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTD PX bit in the CTMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

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

• **16-bit CTM, PWM Output Mode, Edge-aligned Mode, CTD PX=0**

CCRP	1~255	0
Period	CCRP×256	65536
Duty	CCRA	

If  $f_{SYS}=16\text{MHz}$ , CTM clock source select  $f_{SYS}/4$ , CCRP=2 and CCRA=128,

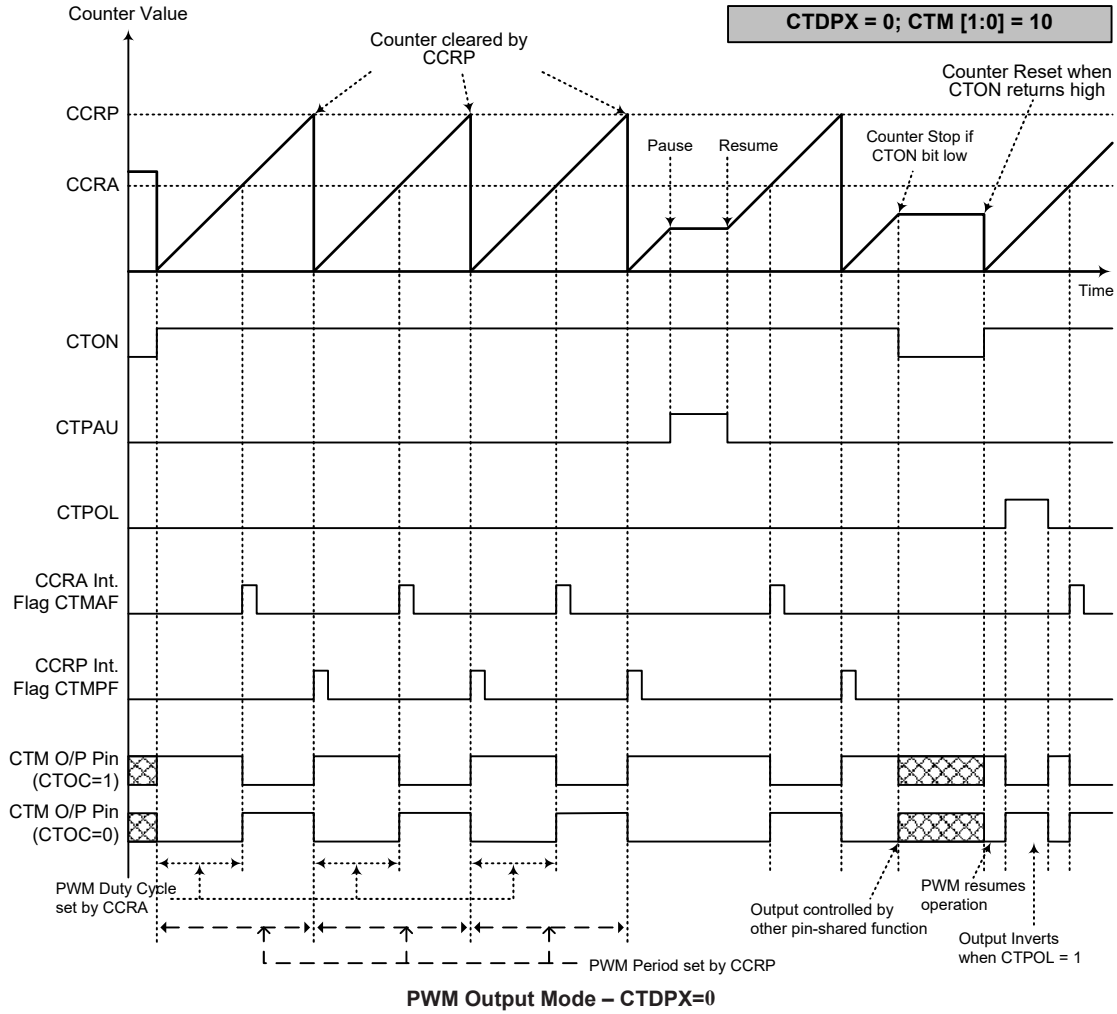
The CTM PWM output frequency= $(f_{SYS}/4)/(2 \times 256)=f_{SYS}/2048=8\text{kHz}$ , duty= $128/(2 \times 256)=25\%$ .

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

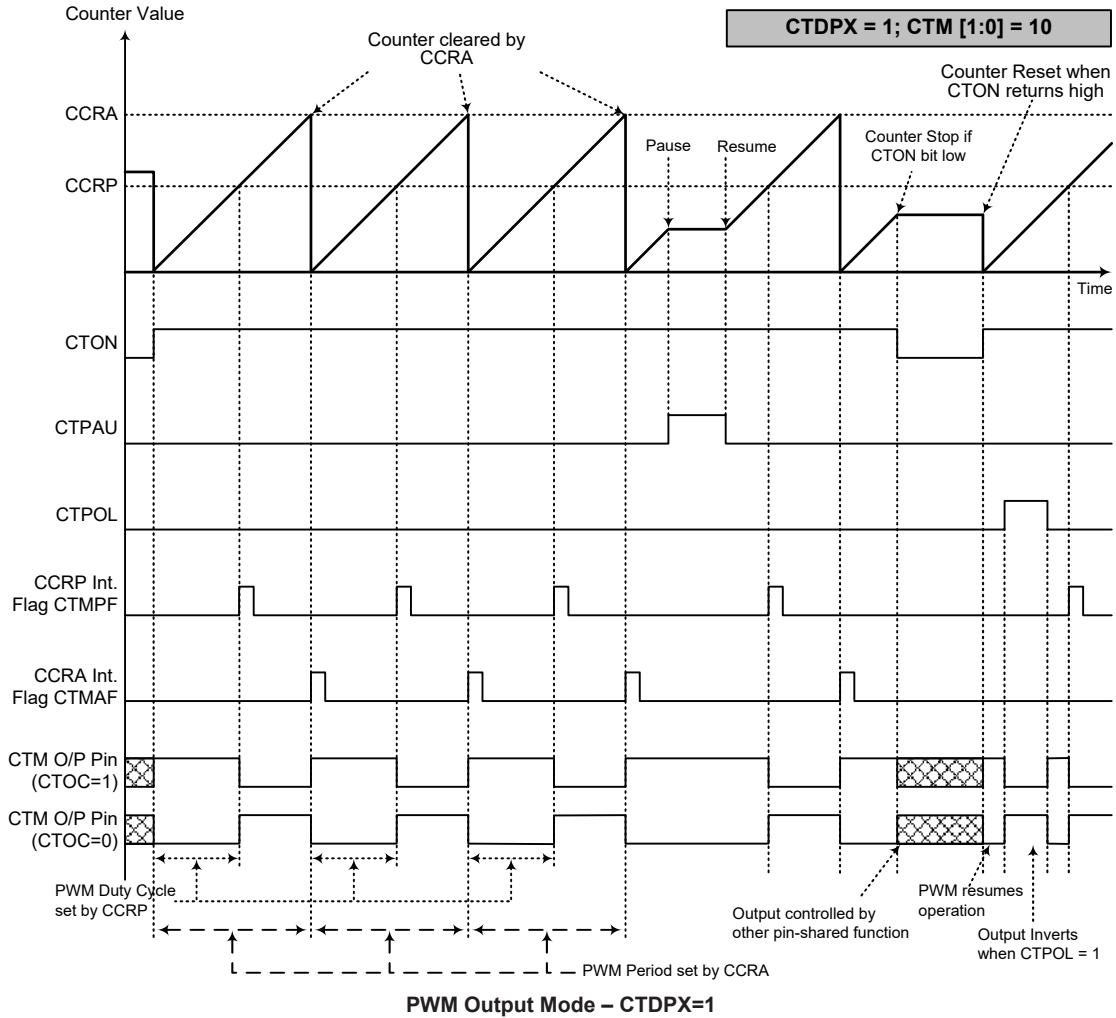
• **16-bit CTM, PWM Output Mode, Edge-aligned Mode, CTD PX=1**

CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

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



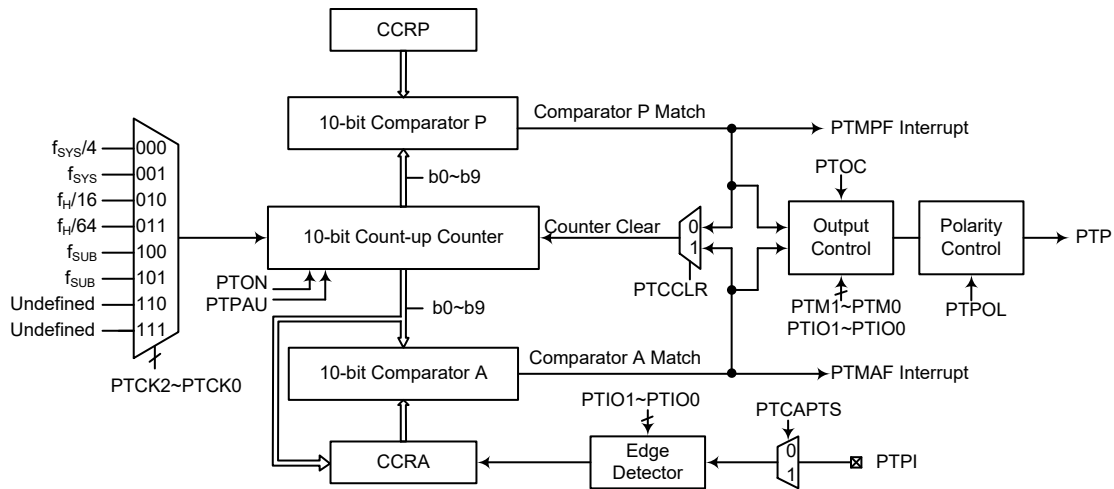
- Note: 1. Here CTDPX=0 – Counter cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues even when CTIO[1:0]=00 or 01  
 4. The CTCCLR bit has no influence on PWM operation



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

## Periodic Type TM – PTM

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



- Note:
1. There is no external clock input for the PTM. The PTM has a PTP output path which is internally used for Complementary PWM circuit input source.
  2. When the PTM operates in the capture input mode, the PTCAPTS bit can only be set to 0 to select the PTPI pin as the capture input source.
  3. The PTM external pin is pin-shared with other functions, so before using the PTM function, the pin-shared function register must be set properly to enable the PTM pin function.

**Periodic Type TM Block Diagram**

### Periodic Type TM Operation

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

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

### Periodic Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	—	—	—	—	—	—	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	—	—	—	—	—	—	D9	D8
PTMRPL	PTRP7	PTRP6	PTRP5	PTRP4	PTRP3	PTRP2	PTRP1	PTRP0
PTMRPH	—	—	—	—	—	—	PTRP9	PTRP8

10-bit Periodic Type TM Register List

• PTMC0 Register

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

Bit 7 **PTPAU**: PTM Counter Pause control

0: Run  
1: Pause

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

Bit 6~4 **PTCK2~PTCK0**: Select PTM Counter clock

000:  $f_{SYS}/4$   
001:  $f_{SYS}$   
010:  $f_H/16$   
011:  $f_H/64$   
100:  $f_{SUB}$   
101:  $f_{SUB}$   
110: Undefined  
111: Undefined

These three bits are used to select the clock source for the PTM. The clock source  $f_{SYS}$  is the system clock, while  $f_H$  and  $f_{SUB}$  are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **PTON**: PTM Counter On/Off control

0: Off  
1: On

This bit controls the overall on/off function of the PTM. Setting the bit high enables the counter to run while clearing the bit disables the PTM. Clearing this bit to zero will stop the counter from counting and turn off the PTM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the PTM is in the Compare Match Output Mode, PWM output Mode or Single Pulse Output Mode then the PTM output will be reset to its initial condition, as specified by the PTOC bit, when the PTON bit changes from low to high.

Bit 2~0 Unimplemented, read as “0”

• **PTMC1 Register**

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

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

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

Bit 5~4     **PTIO1~PTIO0**: Select PTM function  
             Compare Match Output Mode  
                 00: No change  
                 01: Output low  
                 10: Output high  
                 11: Toggle output  
             PWM Output Mode/Single Pulse Output Mode  
                 00: PWM output inactive state  
                 01: PWM output active state  
                 10: PWM output  
                 11: Single Pulse Output  
             Capture Input Mode  
                 00: Input capture at rising edge of PTPI  
                 01: Input capture at falling edge of PTPI  
                 10: Input capture at rising/falling edge of PTPI  
                 11: Input capture disabled  
             Timer/Counter Mode  
                 Unused

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

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

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

Bit 3       **PTOC**: PTM PTP Output control  
             Compare Match Output Mode  
                 0: Initial low  
                 1: Initial high

PWM Output Mode/Single Pulse Output Mode

- 0: Active low
- 1: Active high

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

Bit 2 **PTPOL**: PTM PTP Output polarity control

- 0: Non-invert
- 1: Invert

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

Bit 1 **PTCAPTS**: PTM Capture Trigger Source selection

- 0: From PTPI pin
- 1: Undefined

Note that when the PTM is being used in the capture input mode, this bit must be cleared to zero.

Bit 0 **PTCCLR**: PTM Counter Clear condition selection

- 0: Comparator P match
- 1: Comparator A match

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

• **PTMDL Register**

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

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

• **PTMDH Register**

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

Bit 7~2 Unimplemented, read as “0”

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

• **PTMAL Register**

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

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

• **PTMAH Register**

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

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

• **PTMRPL Register**

Bit	7	6	5	4	3	2	1	0
Name	PTRP7	PTRP6	PTRP5	PTRP4	PTRP3	PTRP2	PTRP1	PTRP0
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      **PTRP7~PTRP0**: PTM CCRP Low Byte Register bit 7 ~ bit 0  
 PTM 10-bit CCRP bit 7 ~ bit 0

• **PTMRPH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PTRP9	PTRP8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2      Unimplemented, read as “0”  
 Bit 1~0      **PTRP9~PTRP8**: PTM CCRP High Byte Register bit 1 ~ bit 0  
 PTM 10-bit CCRP bit 9 ~ bit 8



## Periodic Type TM Operation Modes

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

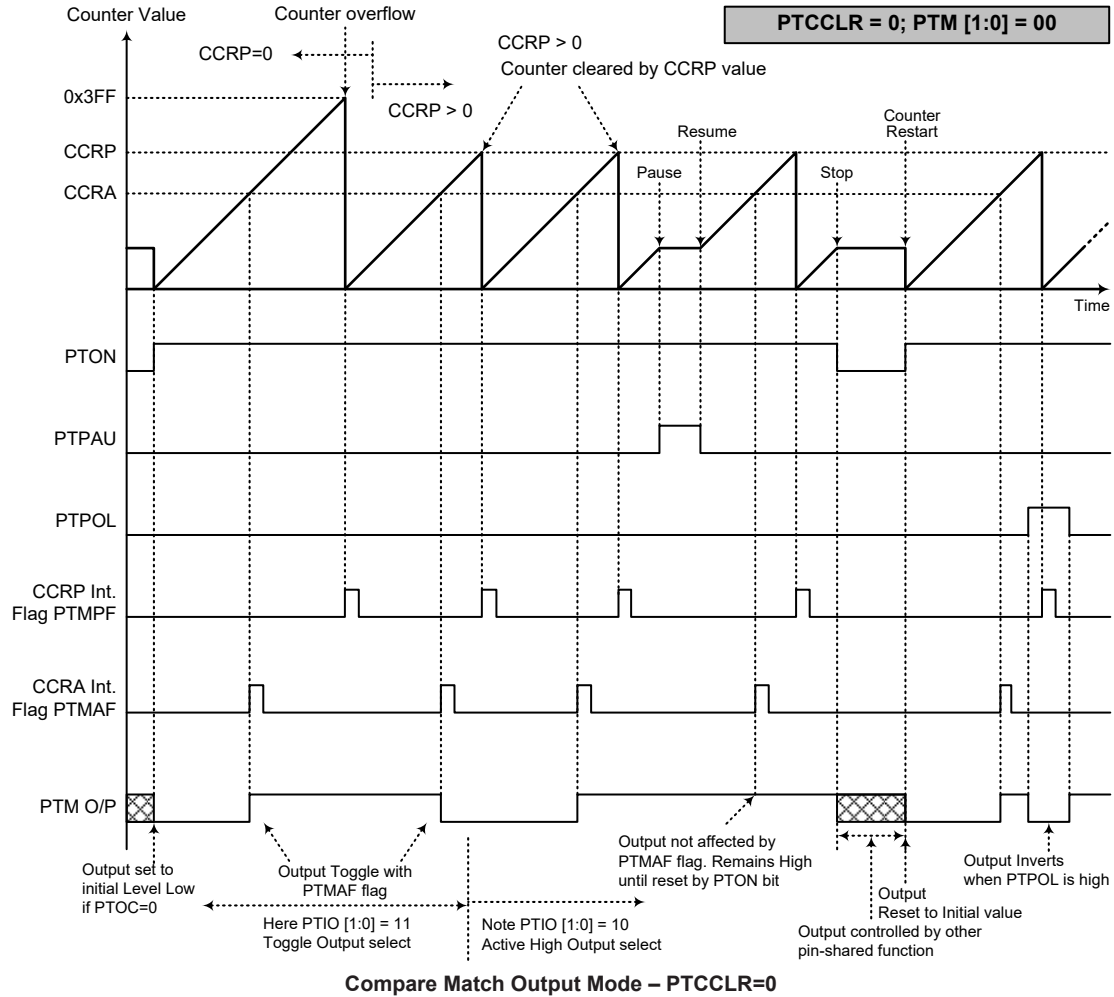
### Compare Match Output Mode

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

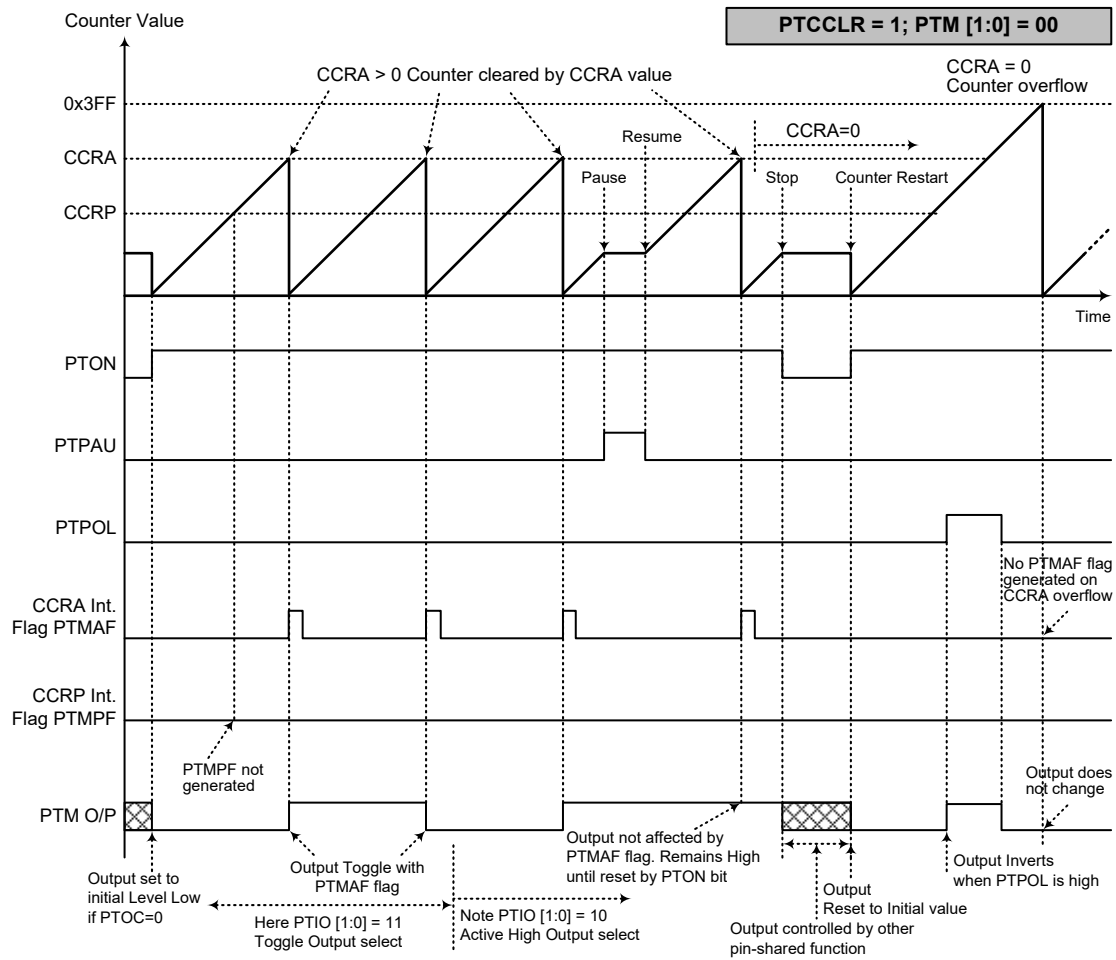
If the PTCCLR bit in the PTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTCCLR is high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

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

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



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



**Compare Match Output Mode – PTCCLR=1**

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

### Timer/Counter Mode

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

### PWM Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 10 respectively. The PWM function within the PTM is useful for applications which require functions such as motor control, heating control, illumination control, etc. By providing a signal of fixed frequency but of varying duty cycle on the PTM output, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the PTCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA registers are used to generate the PWM waveform, 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 PTOC bit in the PTMC1 register is used to select the required polarity of the PWM waveform while the two PTIO1 and PTIO0 bits are used to enable the PWM output or to force the PTM output to a fixed high or low level. The PTPOL bit is used to reverse the polarity of the PWM output waveform.

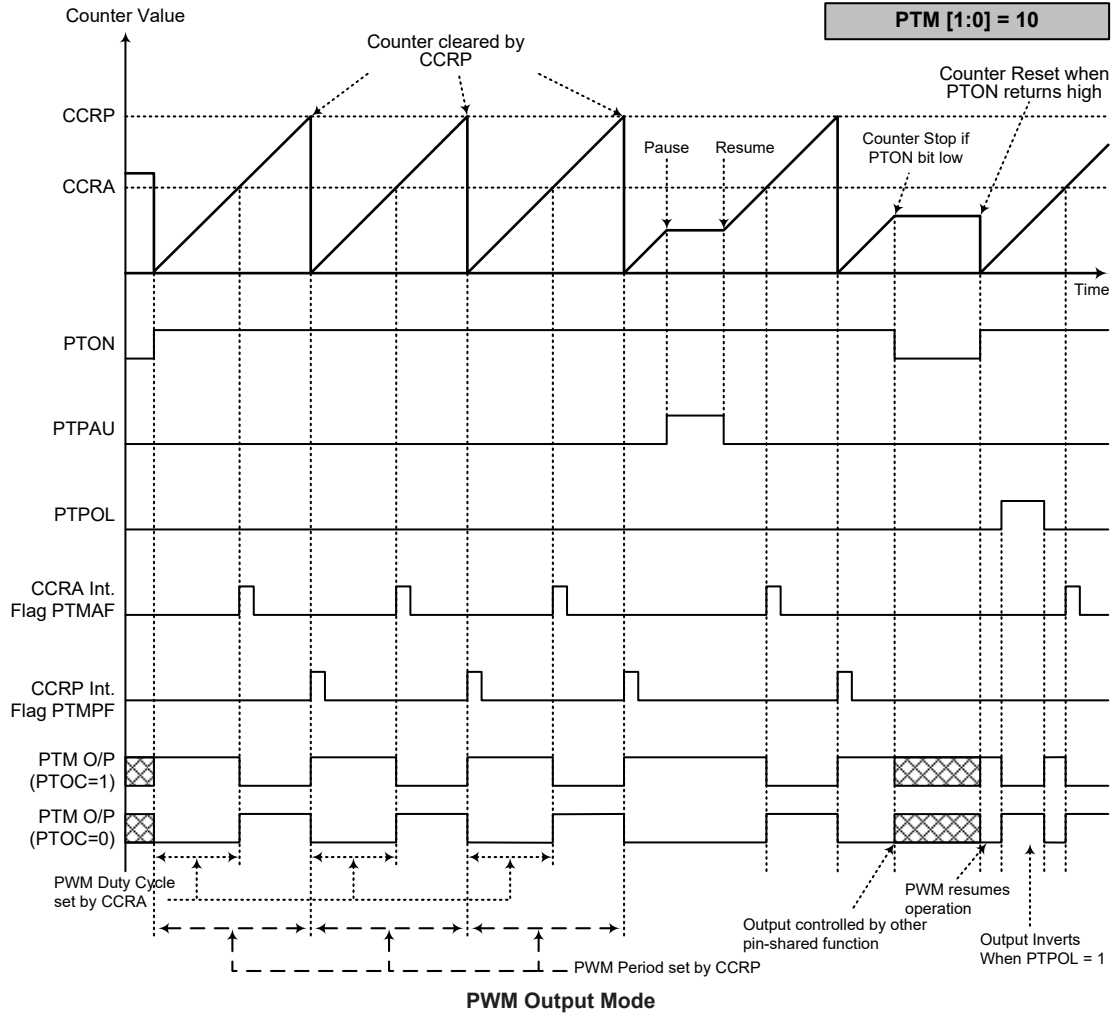
#### • 10-bit PTM, PWM Output Mode, Edge-aligned Mode

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

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

The PTM PWM output frequency= $(f_{SYS}/4)/512=f_{SYS}/2048=8\text{kHz}$ , 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%.



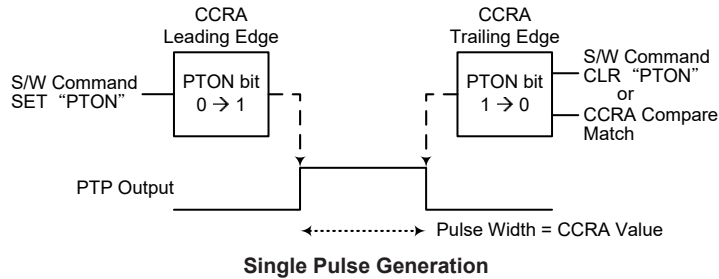
- Note:
1. Counter is cleared by CCRP
  2. A counter clear sets the PWM Period
  3. The internal PWM function continues running even when PTIO[1:0]=00 or 01
  4. The PTCCLR bit has no influence on PWM operation

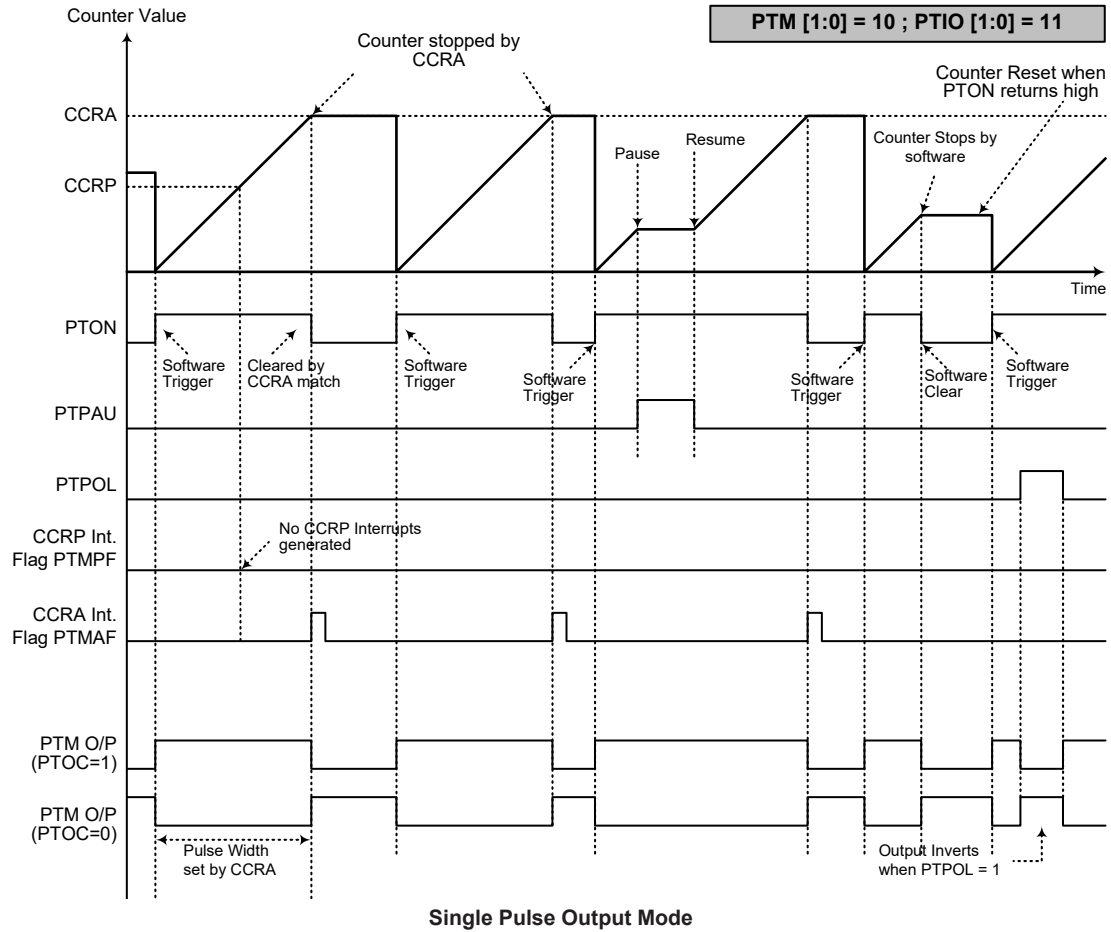
**Single Pulse Output Mode**

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

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

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





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

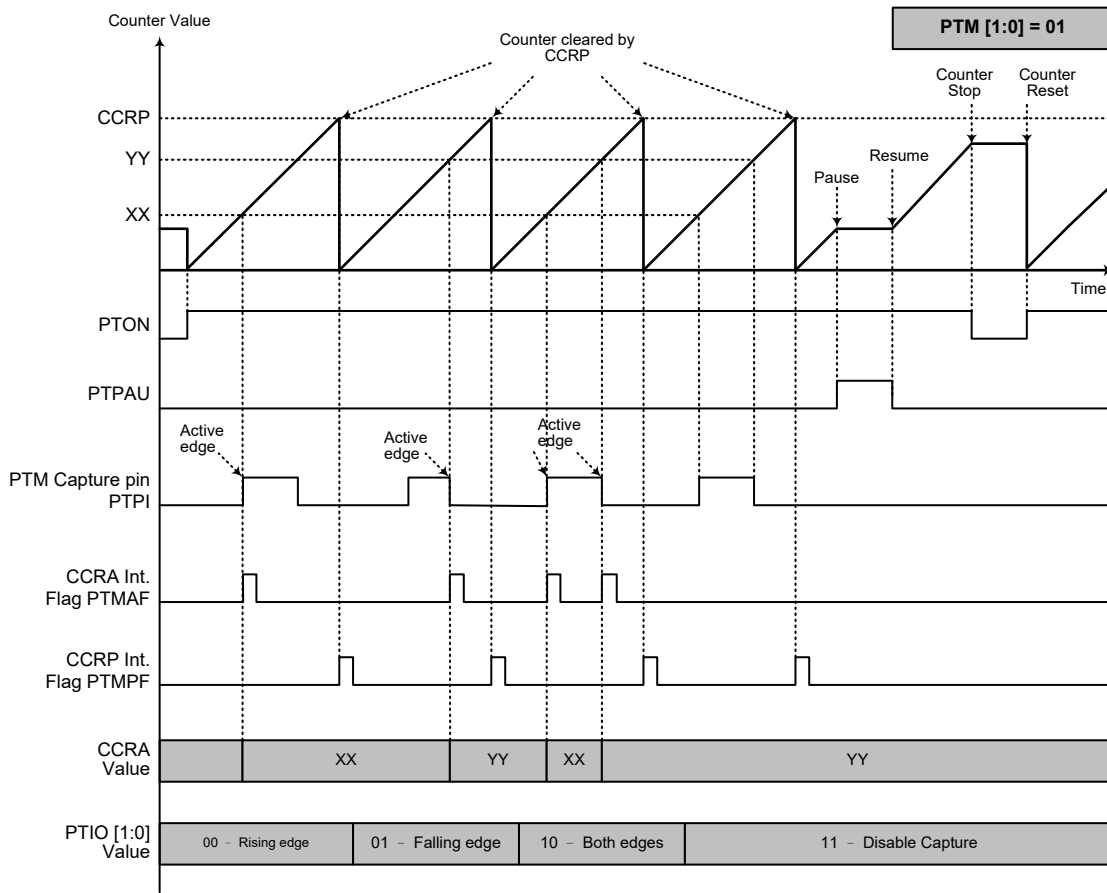
### **Capture Input Mode**

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

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

As the PTPI pin is pin-shared with other functions, care must be taken if the PTM 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 PTCCLR, PTOC and PTPOL bits are not used in this Mode.



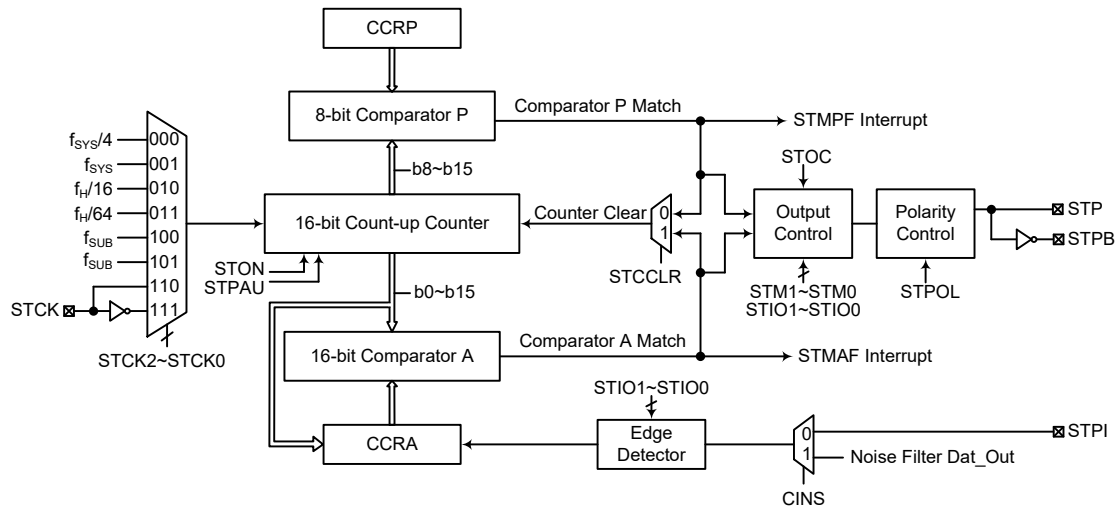


**Capture Input Mode**

- Note: 1. PTM[1:0]=01 and active edge set by the PTIO[1:0] bits  
 2. A PTM Capture input pin active edge transfers the counter value to CCRA  
 3. PTCLLR bit not used  
 4. No output function – PTOC and PTPOL bits are not used  
 5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero

## Standard Type TM – STM

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



Note: 1. The STM capture input can be sourced from the STPI pin or noise filter Dat\_Out signal determined by the CINS bit in the NF\_VIH register. More details can be obtained in the Digital Noise Filter section.

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

Standard Type TM Block Diagram

## Standard Type TM Operation

The Standard Type TM core is a 16-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP comparator is 8-bit wide whose value is compared with the highest 8 bits in the counter while the CCRA is 16 bits and therefore compares all counter bits.

The only way of changing the value of the 16-bit counter using the application program, is to clear the counter by changing the STON 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 STM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

## Standard Type TM Register Description

Overall operation of the Standard Type TM is controlled using a series of registers. A read only register pair exists to store the internal counter 16-bit value, while a read/write register pair exists to store the internal 16-bit CCRA value. The STMRP register is used to store the 8-bit CCRP bits. The remaining two registers are control registers which setup the different operating and control modes.

Register Name	Bit							
	7	6	5	4	3	2	1	0
STMC0	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
STMC1	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
STMDL	D7	D6	D5	D4	D3	D2	D1	D0
STMDH	D15	D14	D13	D12	D11	D10	D9	D8
STMAL	D7	D6	D5	D4	D3	D2	D1	D0
STMAH	D15	D14	D13	D12	D11	D10	D9	D8
STMRP	D7	D6	D5	D4	D3	D2	D1	D0

16-bit Standard Type TM Register List

• **STMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **STPAU**: STM 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 STM 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 **STCK2~STCK0**: Select STM Counter Clock

000:  $f_{SYS}/4$   
001:  $f_{SYS}$   
010:  $f_H/16$   
011:  $f_H/64$   
100:  $f_{SUB}$   
101:  $f_{SUB}$   
110: STCK rising edge clock  
111: STCK falling edge clock

These three bits are used to select the clock source for the STM. 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 **STON**: STM Counter On/Off Control

0: Off  
1: On

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

Bit 2~0 Unimplemented, read as “0”

• **STMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
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     **STM1~STM0**: Select STM 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 STM. To ensure reliable operation the STM should be switched off before any changes are made to the STM1 and STM0 bits. In the Timer/Counter Mode, the STM output pin state is undefined.

Bit 5~4     **STIO1~STIO0**: Select STM function  
             Compare Match Output Mode  
                 00: No change  
                 01: Output low  
                 10: Output high  
                 11: Toggle output  
             PWM Output Mode/Single Pulse Output Mode  
                 00: PWM output inactive state  
                 01: PWM output active state  
                 10: PWM output  
                 11: Single Pulse Output  
             Capture Input Mode  
                 00: Input capture at rising edge of input source (STPI pin or Dat\_Out signal)  
                 01: Input capture at falling edge of input source (STPI pin or Dat\_Out signal)  
                 10: Input capture at rising/falling edge of input (STPI pin or Dat\_Out signal)  
                 11: Input capture disabled

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

In the Compare Match Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a compare match occurs from the Comparator A. The STM 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 STM output pin should be setup using the STOC bit in the STMC1 register. Note that the output level requested by the STIO1 and STIO0 bits must be different from the initial value setup using the STOC bit otherwise no change will occur on the STM output pin when a compare match occurs. After the STM output pin changes state, it can be reset to its initial level by changing the level of the STON bit from low to high.

In the PWM Output Mode, the STIO1 and STIO0 bits determine how the STM 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 STIO1 and STIO0 bits only after the STM has been switched off. Unpredictable PWM outputs will occur if the STIO1 and STIO0 bits are changed when the STM is running.

Bit 3       **STOC**: STM STP Output Control  
             Compare Match Output Mode  
                 0: Initial low  
                 1: Initial high

PWM Output Mode/Single Pulse Output Mode

0: Active low

1: Active high

This is the output control bit for the STM output pin. Its operation depends upon whether STM 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 STM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the STM 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 STM output pin when the STON bit changes from low to high.

Bit 2 **STPOL**: STM STP Output Polarity Control

0: Non-inverted

1: Inverted

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

Bit 1 **STDPX**: STM PWM Duty/Period Control

0: CCRP – period; CCRA – duty

1: CCRP – duty; CCRA – period

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

Bit 0 **STCCLR**: STM Counter Clear Condition Selection

0: Comparator P match

1: Comparator A match

This bit is used to select the method which clears the counter. Remember that the Standard Type TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the STCCLR 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 STCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

• **STMDL Register**

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

Bit 7~0 **D7~D0**: STM Counter Low Byte Register bit 7 ~ bit 0

STM 16-bit Counter bit 7 ~ bit 0

• **STMDH 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**: STM Counter High Byte Register bit 7 ~ bit 0

STM 16-bit Counter bit 15 ~ bit 8

**• STMAL 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:** STM CCRA Low Byte Register bit 7 ~ bit 0  
 STM 16-bit CCRA bit 7 ~ bit 0

**• STMAH 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:** STM CCRA High Byte Register bit 7 ~ bit 0  
 STM 16-bit CCRA bit 15 ~ bit 8

**• STMRP 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:** STM CCRP 8-bit Register, Compared with the STM Counter bit 15 ~ bit 8  
 Comparator P Match period =  
 0: 65536 STM clocks  
 1~255: (1~255)×256 STM clocks

These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight bits. The result of this comparison can be selected to clear the internal counter if the STCCLR bit is set to zero. Setting the STCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

## Standard Type TM Operation Modes

The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

### Compare Match Output Mode

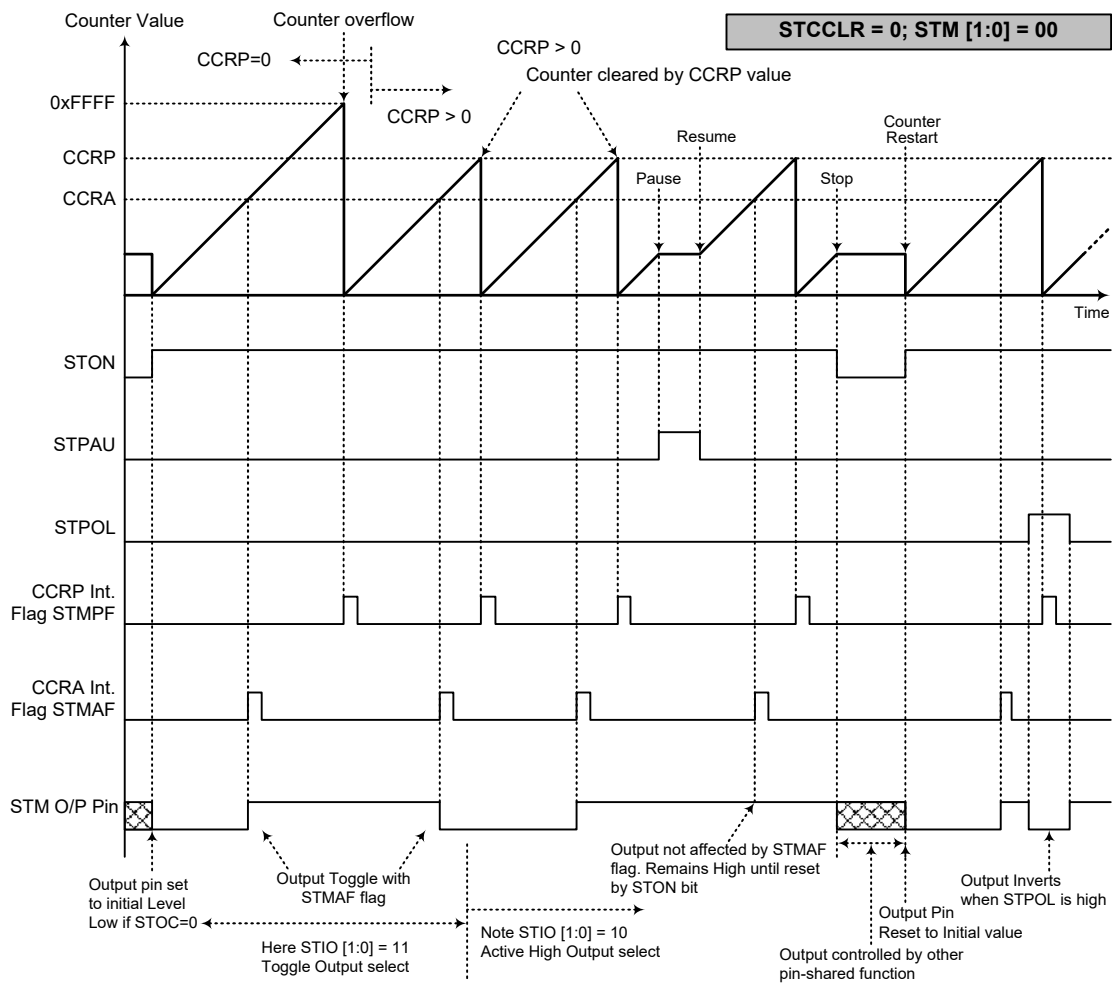
To select this mode, bits STM1 and STM0 in the STMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the STCCLR 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 STMAF and STMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the STCCLR bit in the STMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when

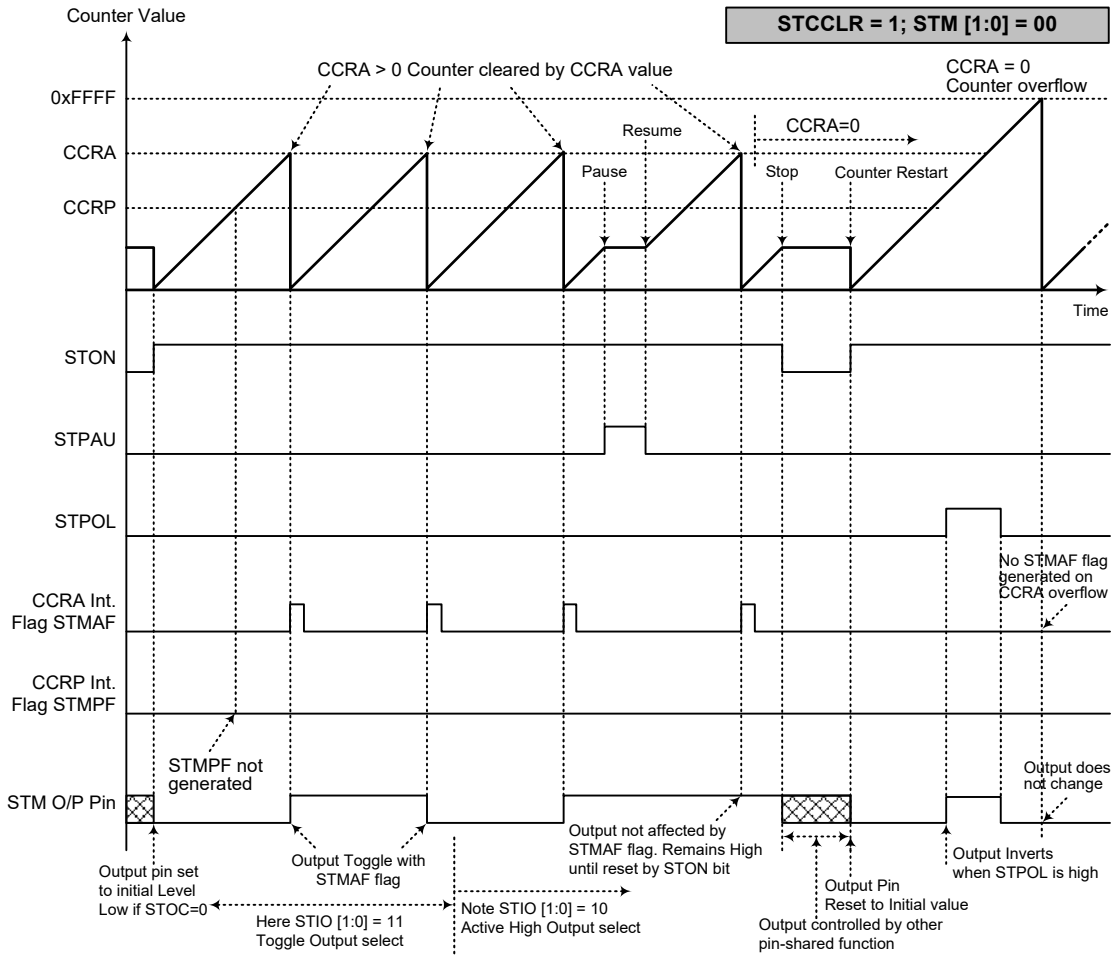
STCCLR is high no STMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be set to “0”.

If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 16-bit, FFFF Hex, value, however here the STMAF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the STM output pin.

The way in which the STM output pin changes state are determined by the condition of the STIO1 and STIO0 bits in the STMC1 register. The STM output pin can be selected using the STIO1 and STIO0 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 STM output pin, which is setup after the STON bit changes from low to high, is setup using the STOC bit. Note that if the STIO1 and STIO0 bits are zero then no pin change will take place.



- Note: 1. With STCCLR=0 a Comparator P match will clear the counter
- 2. The STM output pin is controlled only by the STMAF flag
- 3. The output pin is reset to initial state by a STON bit rising edge



**Compare Match Output Mode – STCCLR=1**

- Note: 1. With STCCLR=1 a Comparator A match will clear the counter
2. The STM output pin is controlled only by the STMAF flag
3. The output pin is reset to its initial state by a STON bit rising edge
4. A STMPF flag is not generated when STCCLR=1



**Timer/Counter Mode**

To select this mode, bits STM1 and STM0 in the STMC1 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 STM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the STM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

**PWM Output Mode**

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 10 respectively. The PWM function within the STM 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 STM output pin, a square waveform 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 STCCLR bit has no effect as the PWM period. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the STDPX bit in the STMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The STOC bit in the STMC1 register is used to select the required polarity of the PWM waveform while the two STIO1 and STIO0 bits are used to enable the PWM output or to force the STM output pin to a fixed high or low level. The STPOL bit is used to reverse the polarity of the PWM output waveform.

**• 16-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=0**

CCRP	1~255	0
Period	CCRP×256	65536
Duty	CCRA	

If  $f_{SYS}=16\text{MHz}$ , STM clock source is  $f_{SYS}/4$ , CCRP=2 and CCRA=128,

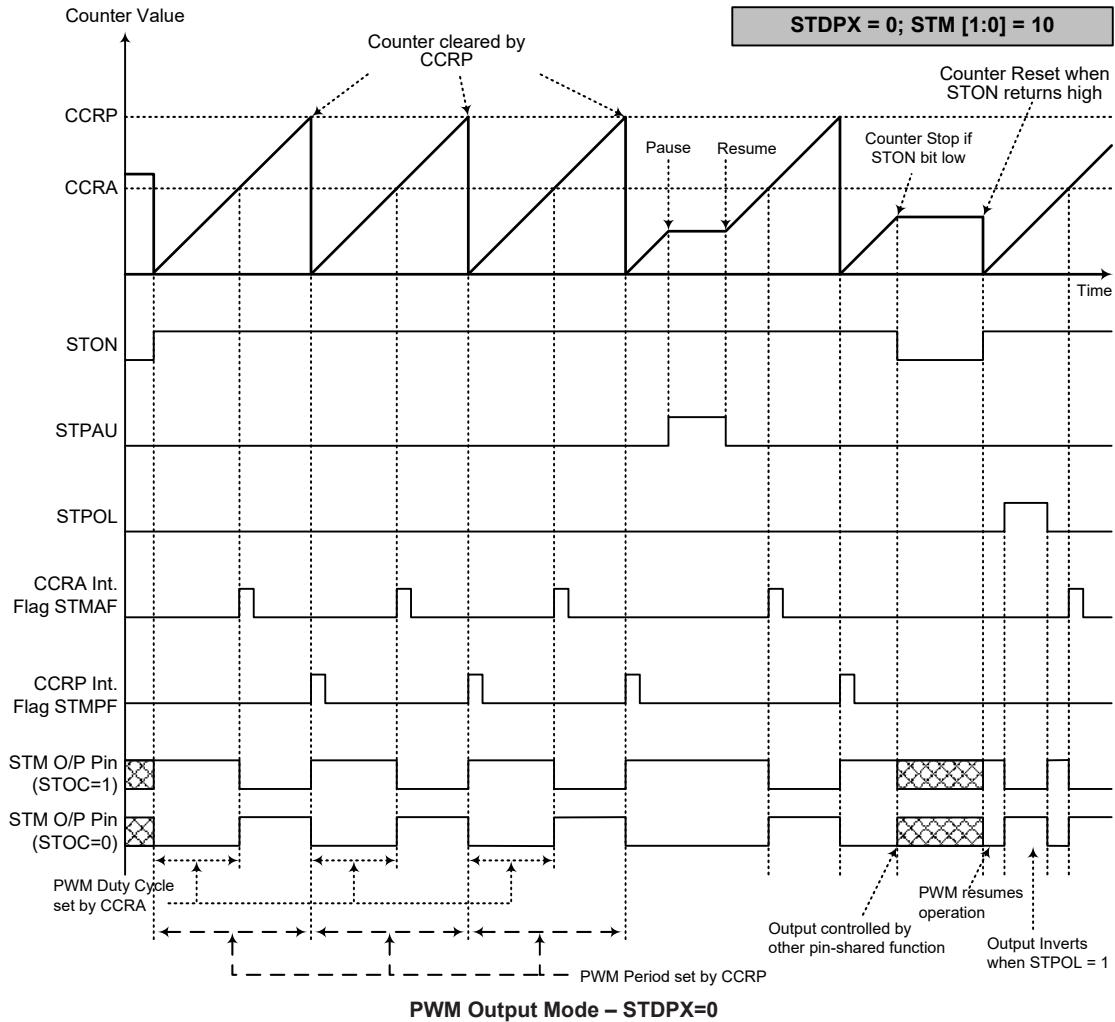
The STM PWM output frequency= $(f_{SYS}/4)/(2 \times 256)=f_{SYS}/2048=8\text{kHz}$ , duty= $128/(2 \times 256)=25\%$ .

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

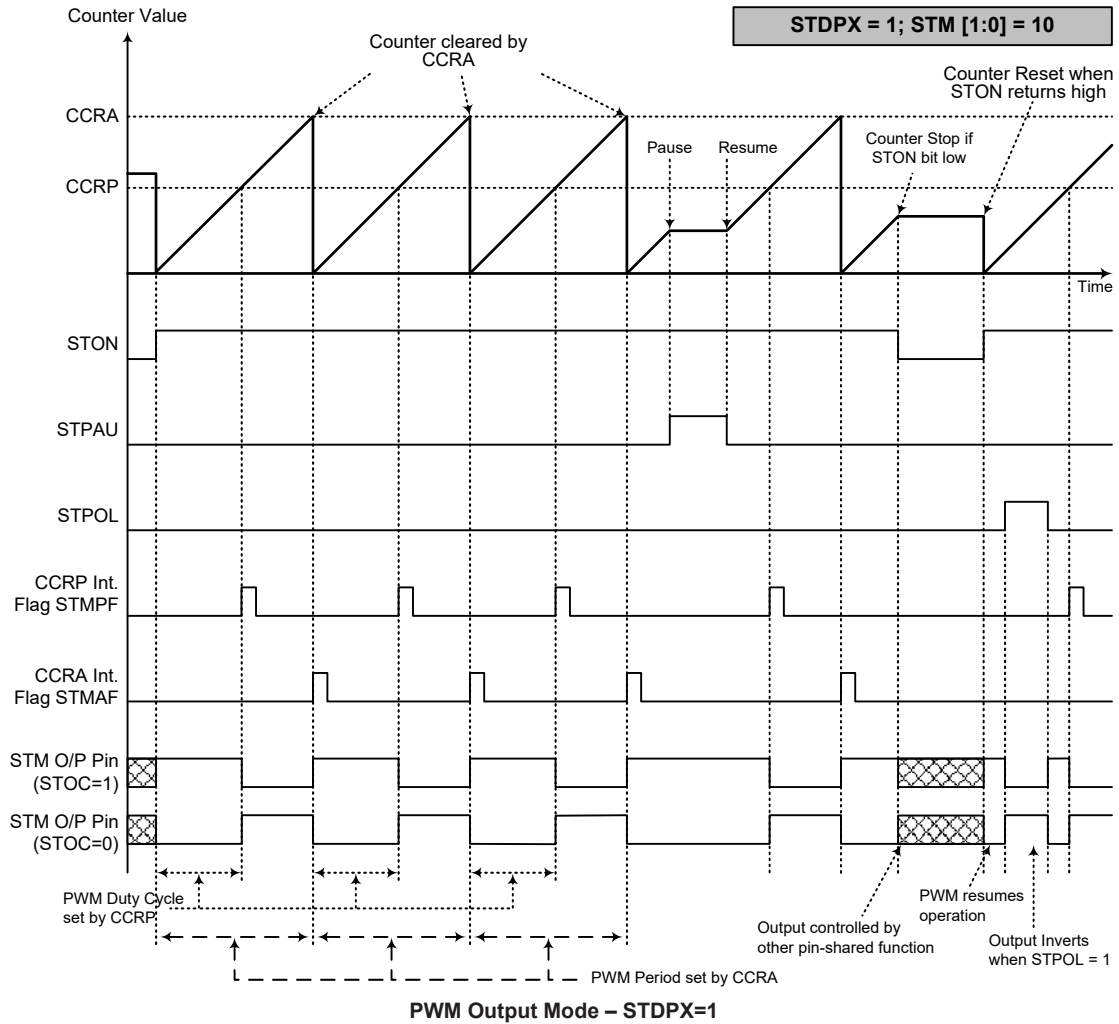
**• 16-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=1**

CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

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



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



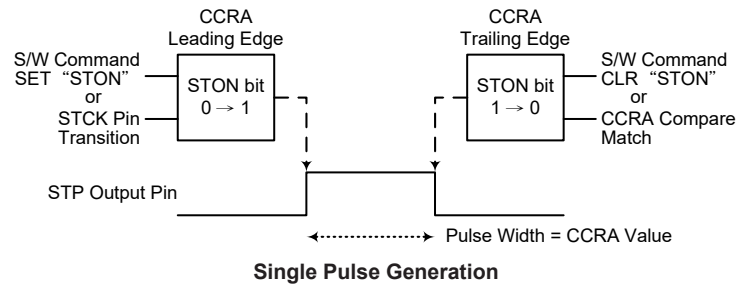
- Note: 1. Here STDPX=1 – Counter cleared by CCRA  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues even when STIO[1:0]=00 or 01  
 4. The STCCLR bit has no influence on PWM operation

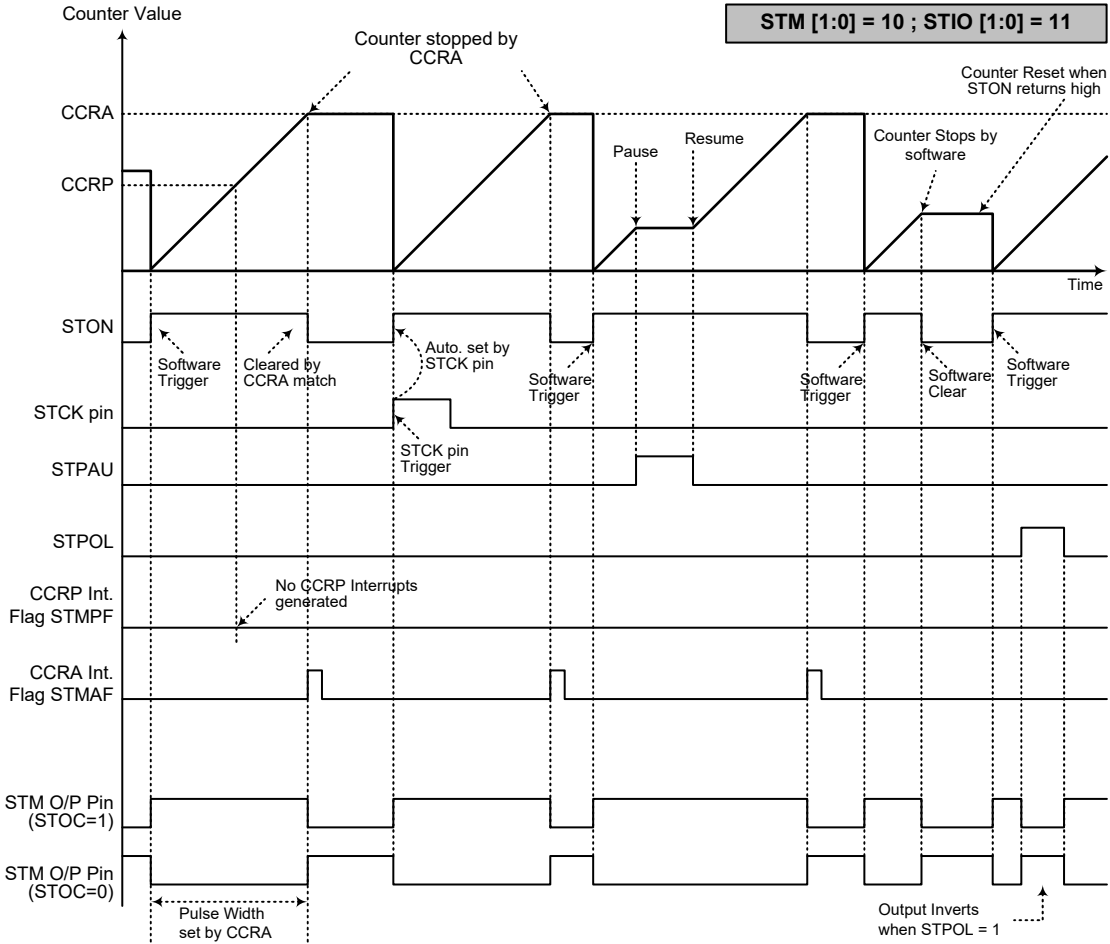
**Single Pulse Output Mode**

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

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





**Single Pulse Output Mode**

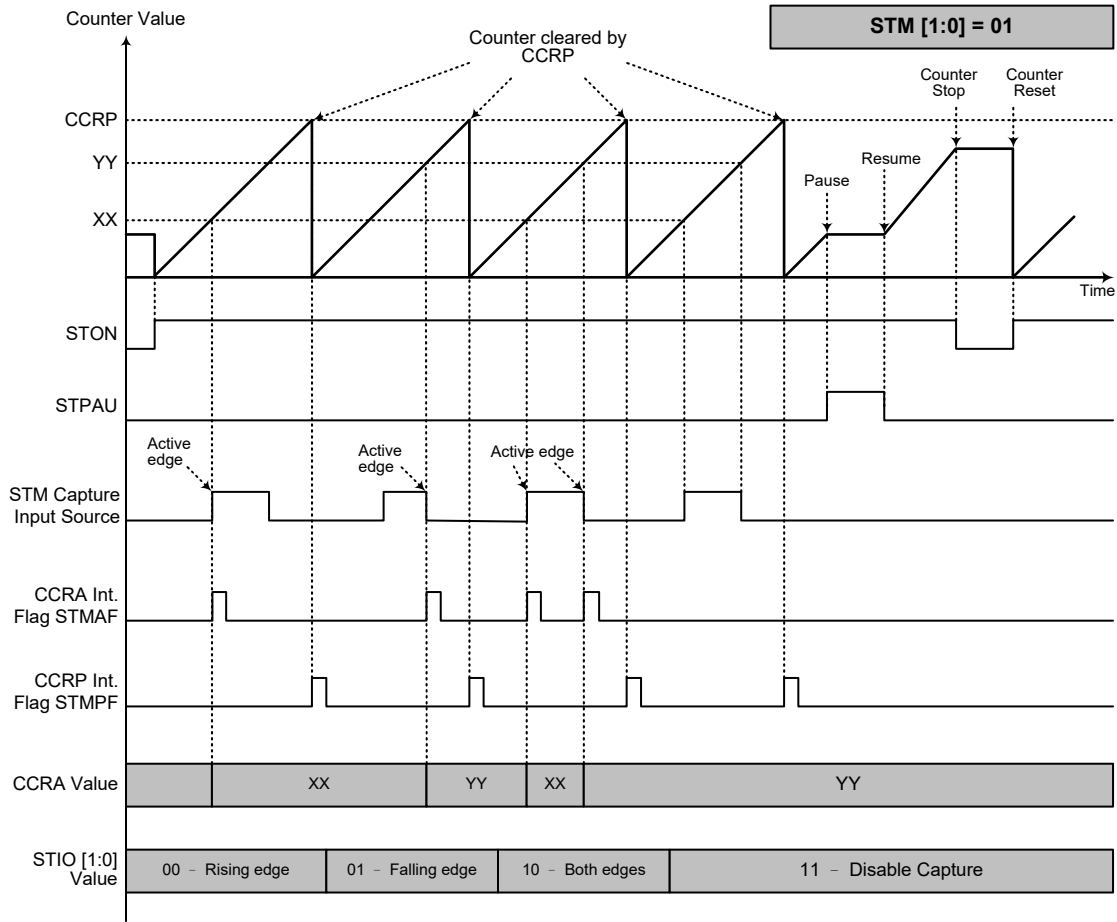
- Note:
1. Counter stopped by CCRA
  2. CCRP is not used
  3. The pulse triggered by the STCK pin or by setting the STON bit high
  4. A STCK pin active edge will automatically set the STON bit high
  5. In the Single Pulse Mode, STIO[1:0] must be set to "11" and cannot be changed

### **Capture Input Mode**

To select this mode bits STM1 and STM0 in the STMC1 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 capture input signal can be sourced from the external signal supplied on the STPI pin or the Noise Filter Dat\_Out which is a filtered signal of the NFIN pin input. The input signal active edge can be a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the STIO1 and STIO0 bits in the STMC1 register. The counter is started when the STON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the STPI pin or Dat\_Out signal the present value in the counter will be latched into the CCRA registers and a STM interrupt generated. Irrespective of what events occur on the STPI pin or Dat\_Out signal the counter will continue to free run until the STON 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 STM 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 STIO1 and STIO0 bits can select the active trigger edge on the STPI pin or Dat\_Out signal to be a rising edge, falling edge or both edge types. If the STIO1 and STIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the STPI pin or Dat\_Out signal, however it must be noted that the counter will continue to run.

As the STPI pin is pin-shared with other functions, care must be taken if the STM 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 STCCLR and STDPX bits are not used in this Mode.



**Capture Input Mode**

- Note: 1. STM[1:0]=01 and active edge set by the STIO[1:0] bits  
 2. A STM Capture input pin active edge transfers the counter value to CCRA  
 3. STCCLR bit not used  
 4. No output function – STOC and STPOL bits are not used  
 5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero

## Analog to Digital Converter

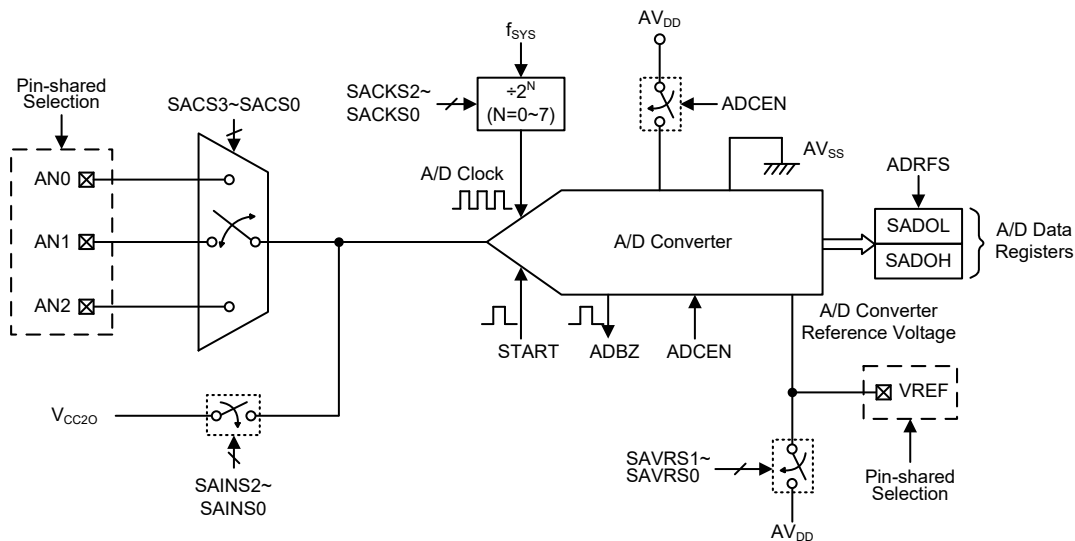
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

### A/D Converter Overview

This device contains a multi-channel analog to digital converter which can directly interface to external analog signals (such as that from sensors or other control signals) or internal LDO resistor divider output voltage,  $V_{CC20}$ , and convert these signals directly into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 bits together with the SACS3~SACS0 bits. 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 Input Select Bits
AN0~AN2	$V_{CC20}$	SAINS2~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 Structure

### A/D Converter Register Description

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



Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (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	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

**A/D Converter Register List**

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

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

To control the function and operation of the A/D converter, several control registers known as SADC0 and SADC1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter 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 SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input.

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  
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process.
- Bit 6**     **ADBZ:** A/D converter busy flag  
0: No A/D conversion is in progress  
1: A/D conversion is in progress  
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.
- Bit 5**     **ADCEN:** A/D converter function enable control  
0: Disable  
1: Enable  
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is 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 converter data format selection  
0: A/D converter data format → SADOH=D[11:4]; SADOL=D[3:0]  
1: A/D converter data format → SADOH=D[11:8]; SADOL=D[7:0]  
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.
- Bit 3~0**   **SACS3~SACS0:** A/D converter external analog input channel selection  
0000: AN0  
0001: AN1  
0010: AN2  
0011~1111: Undefined, input floating

• **SADC1 Register**

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

- Bit 7~5**   **SAINS2~SAINS0:** A/D converter input signal selection  
000: External source – External analog channel input, ANn  
001: Internal source – Internal LDO resistor divider output voltage, V<sub>CC20</sub>  
010~100: Unused, connected to ground  
101~111: External source – External analog channel input, ANn  
Care must be taken if the SAINS2~SAINS0 bits are set to “001” to select the internal analog signal to be converted. When the internal analog signal is selected to be converted, the external input pin must never be selected as the A/D input signal by properly setting the SACS3~SACS0 bits with a value from 0011 to 1111. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

- Bit 4~3     **SAVRS1~SAVRS0:** A/D converter reference voltage selection  
00: External VREF pin  
01: Internal A/D converter power, AV<sub>DD</sub>  
1x: External VREF pin

These bits are used to select the A/D converter reference voltage source. Care must be taken if the SAVRS1~SAVRS0 bits are set to “01” to select the internal A/D converter power as the reference voltage source. When the internal A/D converter power is selected as the reference voltage, the VREF pin cannot be configured as the reference voltage input by properly configuring the corresponding pin-shared function control bits. Otherwise, the external input voltage on VREF pin will be connected to the internal A/D converter power. This will result in unpredictable situations.

- Bit 2~0     **SACKS2~SACKS0:** A/D converter clock rate selection bit  
000: f<sub>SYS</sub>  
001: f<sub>SYS</sub>/2  
010: f<sub>SYS</sub>/4  
011: f<sub>SYS</sub>/8  
100: f<sub>SYS</sub>/16  
101: f<sub>SYS</sub>/32  
110: f<sub>SYS</sub>/64  
111: f<sub>SYS</sub>/128

### A/D Converter Reference Voltage

The reference voltage supply to the A/D converter can be supplied from the A/D converter power, AV<sub>DD</sub>, or from an external reference source supplied on pin VREF. The desired selection is made using the SAVRS1 and SAVRS0 bits. When the SAVRS bit field is set to “01”, the A/D converter reference voltage will come from the AV<sub>DD</sub>. Otherwise, if the SAVRS bit field is set to any other value except “01”, the A/D converter reference voltage will come from the VREF pin. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bit should be properly configured to disable other pin functions. However, if the internal A/D converter power is selected as the reference voltage, the VREF pin must not be configured as the reference voltage input function to avoid the internal connection between the VREF pin and A/D converter power AV<sub>DD</sub>. The analog input values must not be allowed to exceed the value of the selected A/D reference voltage.

### A/D Converter Input Signal

All the external A/D analog channel input pins are pin-shared with the I/O pins as well as other functions. The corresponding control bits for each A/D external input pin in the PAS0 register determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the pin is setup to be as an A/D analog channel input, the original pin functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

There is an internal analog signal derived from the V<sub>CC20</sub>, which can be connected to the A/D converter as the analog input signal by configuring the SAINS2~SAINS0 bits. If the external channel input is selected to be converted, the SAINS2~SAINS0 bits should be set to “000” or “101~111” and the SACS3~SACS0 bits can determine which external channel is selected. If the internal analog signal is selected to be converted, the SACS3~SACS0 bits must be configured with a value from 0011 to 1111 to switch off the external analog channel input. Otherwise, the internal analog signal will be connected together with the external channel input. This will result in unpredictable situations.

SAINS[2:0]	SACS[3:0]	Input Signals	Description
000, 101~111	0000~0010	AN0~AN2	External pin analog input
	0011~1111	—	Floating, no external channel is selected
001	0011~1111	V <sub>CC20</sub>	Internal LDO resistor divider output voltage
010~100	0011~1111	—	Unused, connected to ground

**A/D Converter Input Signal Selection**

## A/D Converter Operation

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

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

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

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

**A/D Clock Period Examples**

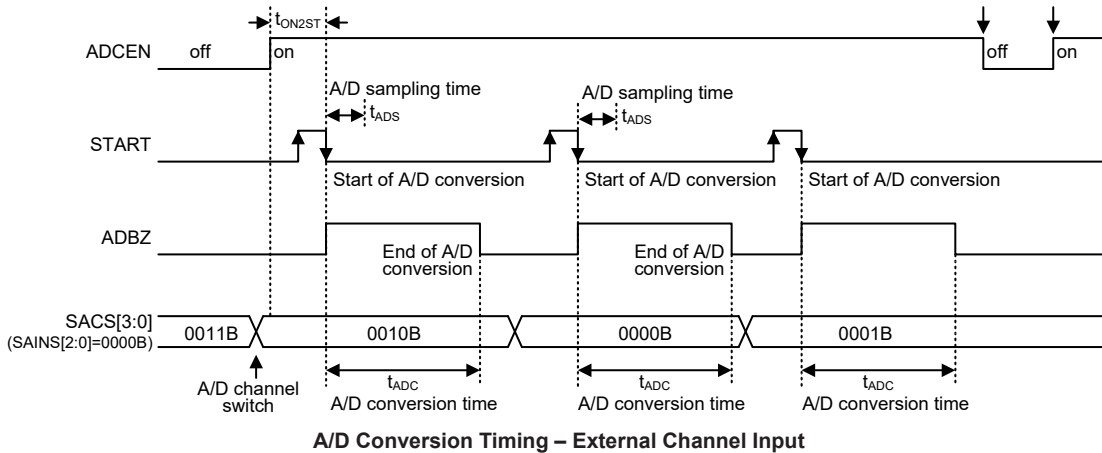
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the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

### Conversion Rate and Timing Diagram

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

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

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.



### Summary of A/D Conversion Steps

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

- Step 1  
 Select the required A/D conversion clock by correctly programming bits SACKS2~SACKS0 in the SADC1 register.
- Step 2  
 Enable the A/D converter by setting the ADCEN bit in the SADC0 register to 1.
- Step 3  
 Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS2~SAINS0 bits.  
 Select the external channel input to be converted, go to Step 4.  
 Select the internal analog signal to be converted, go to Step 5.

- Step 4  
If the A/D converter input signal comes from the external channel input selecting by configuring the SAINS bit field, the corresponding pins should be configured as A/D converter input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS bit field. After this step, go to Step 6.
- Step 5  
Before the A/D converter input signal is selected to come from the internal analog signal by configuring the SAINS bit field, the corresponding external input pin must be switched to a non-existent channel input by setting the SACS3~SACS0 bits with a value from 0011 to 1111. The desired internal analog signal then can be selected by configuring the SAINS bit field. After this step, go to Step 6.
- Step 6  
Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register.
- Step 7  
Select A/D converter output data format by setting 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 associated Multi-function interrupt enable bit, MFnF, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 9  
The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10  
If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

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

### Programming Considerations

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

### A/D Conversion Function

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

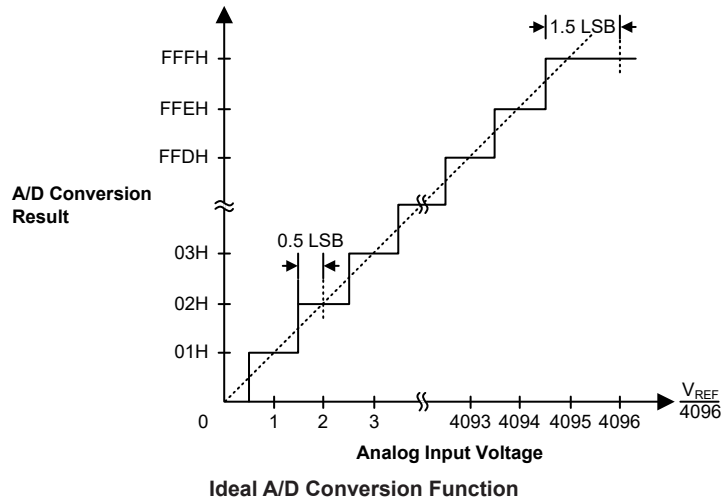
$$1 \text{ LSB} = V_{REF} \div 4096$$

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

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

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

Note that here the  $V_{REF}$  voltage is the actual A/D converter reference voltage determined by the SAVRS field.



### 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         ; select fsys/8 as A/D clock and
mov SADC1,a       ; select external channel input and external reference input
mov a,0CCh        ; setup PAS0 to configure pin AN0 and pin VREF
mov PAS0,a
mov a,20h
mov SADC0,a       ; enable A/D 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 SADOI_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        ; select fsys/8 as A/D clock and
mov SADC1,a      ; select external channel input and external reference input
mov a,0CCh       ; setup PAS0 to configure pin AN0 and pin VREF
mov PAS0,a
mov a,20h
mov SADC0,a      ; enable A/D and connect AN0 channel to A/D converter
Start_conversion:
clr START        ; high pulse on START bit to initiate conversion
set START        ; reset A/D
clr START        ; start A/D
clr ADF          ; clear ADC interrupt request flag
set ADE          ; enable A/D converter 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
```

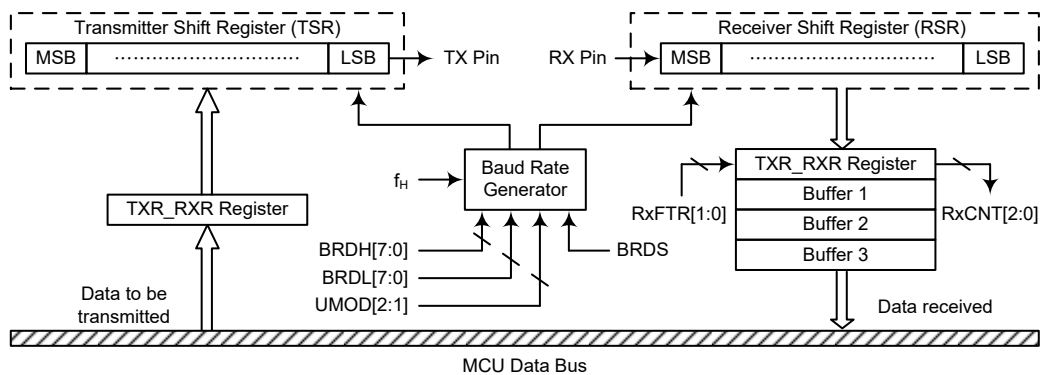


## UART Interface

The device contains an integrated full-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UART function contains the following features:

- Full-duplex, asynchronous communication
- 8 or 9 bits character length
- Even, odd, mark, space or no parity options
- One or two stop bits
- Baud rate generator with 16-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 4-byte Deep FIFO Receive Data Buffer
- 1-byte Deep FIFO Transmit Data Buffer
- RX pin wake-up function
- Transmit and receive interrupts
- Interrupts can be initialized by the following conditions:
  - ♦ Transmitter Empty
  - ♦ Transmitter Idle
  - ♦ Receiver Full
  - ♦ Receiver Overrun
  - ♦ Address Mode Detect



UART Data Transfer Block Diagram

## UART External Pins

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

## UART Data Transfer Scheme

The above block diagram shows the overall data transfer structure arrangement for the UART. The actual data to be transmitted from the MCU is first transferred to the TXR\_RXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate Generator. Only the TXR\_RXR register is mapped onto the MCU Data Memory, the Transmit Shift Register is not mapped and is therefore inaccessible to the application program.

Data to be received by the UART is accepted on the external RX pin, from where it is shifted in, LSB first, to the Receiver Shift Register at a rate controlled by the Baud Rate Generator. When the shift register is full, the data will then be transferred from the shift register to the internal TXR\_RXR register, where it is buffered and can be manipulated by the application program. Only the TXR\_RXR register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.

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

## UART Status and Control Registers

There are eight control registers associated with the UART function. The USR, UCR1, UCR2, UFCR and RxCNT registers control the overall function of the UART, while the BRDH and BRDL registers control the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR\_RXR data register.

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

**UART Register List**

• **USR Register**

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

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

- Bit 7 PERR:** Parity error flag  
 0: No parity error is detected  
 1: Parity error is detected  
 The PERR flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if the parity is enabled and the parity type (odd, even, mark or space) is selected. The flag can also be cleared by a software sequence which involves a read to the status register USR followed by an access to the TXR\_RXR data register.
- Bit 6 NF:** Noise flag  
 0: No noise is detected  
 1: Noise is detected  
 The NF flag is the noise flag. When this read only flag is “0”, it indicates no noise condition. When the flag is “1”, it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of an overrun. The NF flag can be cleared by a software sequence which will involve a read to the status register USR followed by an access to the TXR\_RXR data register.
- Bit 5 FERR:** Framing error flag  
 0: No framing error is detected  
 1: Framing error is detected  
 The FERR flag is the framing error flag. When this read only flag is “0”, it indicates that there is no framing error. When the flag is “1”, it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register USR followed by an access to the TXR\_RXR data register.
- Bit 4 OERR:** Overrun error flag  
 0: No overrun error is detected  
 1: Overrun error is detected  
 The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is “0”, it indicates that there is no overrun error. When the flag is “1”, it indicates that an overrun error occurs which will inhibit further transfers to the TXR\_RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register USR followed by an access to the TXR\_RXR data register.
- Bit 3 RIDLE:** Receiver status  
 0: Data reception is in progress (Data being received)  
 1: No data reception is in progress (Receiver is idle)  
 The RIDLE flag is the receiver status flag. When this read only flag is “0”, it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is “1”, it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is “1” indicating that the UART receiver is idle and the RX pin stays in logic high condition.

- Bit 2**     **RXIF:** Receive TXR\_RXR data register status  
           0: TXR\_RXR data register is empty  
           1: TXR\_RXR data register has available data and reach receiver FIFO trigger level  
 The RXIF flag is the receive data register status flag. When this read only flag is “0”, it indicates that the TXR\_RXR read data register is empty. When the flag is “1”, it indicates that the TXR\_RXR read data register contains new data. When the contents of the shift register are transferred to the TXR\_RXR register and reach receiver FIFO trigger level, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag is cleared when the USR register is read with RXIF set, followed by a read from the TXR\_RXR register, and if the TXR\_RXR register has no data available.
- Bit 1**     **TIDLE:** Transmission idle  
           0: Data transmission is in progress (Data being transmitted)  
           1: No data transmission is in progress (Transmitter is idle)  
 The TIDLE flag is known as the transmission complete flag. When this read only flag is “0”, it indicates that a transmission is in progress. This flag will be set high when the TXIF flag is “1” and when there is no transmit data or break character being transmitted. When TIDLE is equal to “1”, the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared by reading the USR register with TIDLE set and then writing to the TXR\_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.
- Bit 0**     **TXIF:** Transmit TXR\_RXR data register status  
           0: Character is not transferred to the transmit shift register  
           1: Character has transferred to the transmit shift register (TXR\_RXR data register is empty)  
 The TXIF flag is the transmit data register empty flag. When this read only flag is “0”, it indicates that the character is not transferred to the transmitter shift register. When the flag is “1”, it indicates that the transmitter shift register has received a character from the TXR\_RXR data register. The TXIF flag is cleared by reading the UART status register (USR) with TXIF set and then writing to the TXR\_RXR data register. Note that when the TXEN bit is set, the TXIF flag bit will also be set since the transmit data register is not yet full.

• **UCR1 Register**

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

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

“x”: unknown

- Bit 7**     **UARTEN:** UART function enable control  
           0: Disable UART. TX and RX pins are in a floating state  
           1: Enable UART. TX and RX pins function as UART pins  
 The UARTEN bit is the UART enable bit. When this bit is equal to “0”, the UART will be disabled and the RX pin as well as the TX pin will be set in a floating state. When the bit is equal to “1”, the UART will be enabled and the TX and RX pins will function as defined by the TXEN and RXEN enable control bits.  
 When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits will be cleared, while the TIDLE,

TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2, BRDH and BRDL registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6 **BNO**: Number of data transfer bits selection

0: 8-bit data transfer

1: 9-bit data transfer

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

Note that the 9th bit of data if BNO=1, or the 8th bit of data if BNO=0, which is used as the parity bit, does not transfer to RX8 or RX7 respectively when the parity function is enabled.

Bit 5 **PREN**: Parity function enable control

0: Parity function is disabled

1: Parity function is enabled

This is the parity enable bit. When this bit is equal to “1”, the parity function will be enabled. If the bit is equal to “0”, then the parity function will be disabled. Replace the most significant bit position with a parity bit.

Bit 4~3 **PRT1~PRT0**: Parity type selection bits

00: Even parity for parity generator

01: Odd parity for parity generator

10: Mark parity for parity generator

11: Space parity for parity generator

These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.

Bit 2 **TXBRK**: Transmit break character

0: No break character is transmitted

1: Break characters transmit

The TXBRK bit is the Transmit Break Character bit. When this bit is “0”, there are no break characters and the TX pin operates normally. When the bit is “1”, there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to “1”, after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.

Bit 1 **RX8**: Receive data bit 8 for 9-bit data transfer format (read only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **TX8**: Transmit data bit 8 for 9-bit data transfer format (write only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UCR2 Register**

The UCR2 register is the second of the two UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UART Transmitter and Receiver as well as enabling the various UART interrupt sources. The register also serves to control the STOP bit number selection, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

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

**Bit 7 TXEN:** UART Transmitter enabled control

- 0: UART transmitter is disabled
- 1: UART transmitter is enabled

The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to “0”, the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TX pin will be set in a floating state.

If the TXEN bit is equal to “1” and the UARTEN bit is also equal to “1”, the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be set in a floating state.

**Bit 6 RXEN:** UART Receiver enabled control

- 0: UART receiver is disabled
- 1: UART receiver is enabled

The bit named RXEN is the Receiver Enable Bit. When this bit is equal to “0”, the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RX pin will be set in a floating state. If the RXEN bit is equal to “1” and the UARTEN bit is also equal to “1”, the receiver will be enabled and the RX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX pin will be set in a floating state.

**Bit 5 STOPS:** Number of Stop bits selection for transmitter

- 0: One stop bit format is used
- 1: Two stop bits format is used

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

**Bit 4 ADDEN:** Address detect function enable control

- 0: Address detect function is disabled
- 1: Address detect function is enabled

The bit named ADDEN is the address detect function enable control bit. When this bit is equal to “1”, the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to RX7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of “1”, then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the received word is “0” with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

**Bit 3 WAKE:** RX pin wake-up UART function enable control

- 0: RX pin wake-up UART function is disabled
- 1: RX pin wake-up UART function is enabled

This bit is used to control the wake-up UART function when a falling edge on the RX pin occurs. Note that this bit is only available when the UART clock ( $f_{H}$ ) is switched off. There will be no RX pin wake-up UART function if the UART clock ( $f_{H}$ ) exists.

If the WAKE bit is set to 1 as the UART clock ( $f_{H}$ ) is switched off, a UART wake-up request will be initiated when a falling edge on the RX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock ( $f_{H}$ ) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the RX pin when the WAKE bit is cleared to 0.

- Bit 2 RIE:** Receiver interrupt enable control  
 0: Receiver related interrupt is disabled  
 1: Receiver related interrupt is enabled  
 This bit enables or disables the receiver interrupt. If this bit is equal to “1” and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.
- Bit 1 TIE:** Transmitter Idle interrupt enable control  
 0: Transmitter idle interrupt is disabled  
 1: Transmitter idle interrupt is enabled  
 This bit enables or disables the transmitter idle interrupt. If this bit is equal to “1” and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.
- Bit 0 TEIE:** Transmitter Empty interrupt enable control  
 0: Transmitter empty interrupt is disabled  
 1: Transmitter empty interrupt is enabled  
 This bit enables or disables the transmitter empty interrupt. If this bit is equal to “1” and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

• **TXR\_RXR Register**

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

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

“x”: unknown

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

• **BRDH Register**

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

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

• **BRDL Register**

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

- Bit 7~0     **D7~D0:** Baud rate divider low byte  
 The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.  
 $Baud\ Rate = f_{ih} / (BRD + UMOD/8)$   
 $BRD = 16 \sim 65535$  or  $8 \sim 65535$  depending on BRDS  
 Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.  
 2. The BRDL must be written first and then BRDH, otherwise errors may occur.

• **UFCR Register**

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

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

- Bit 7~6     Unimplemented, read as “0”
- Bit 5~3     **UMOD2~UMOD0:** UART Modulation Control bits  
 The modulation control bits are used to correct the baud rate of the received or transmitted UART signal. These bits determine if the extra UART clock cycle should be added in a UART bit time. The UMOD2~UMOD0 will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle.
- Bit 2     **BRDS:** BRD range selection  
 0: BRD range is from 16 to 65535  
 1: BRD range is from 8 to 65535  
 The BRDS is used to control the sampling point in a UART bit time. If the BRDS bit is cleared to zero, the sampling point will be  $BRD/2$ ,  $BRD/2 + 1 \times f_{ih}$ , and  $BRD/2 + 2 \times f_{ih}$  in a UART bit time. If the BRDS bit is set high, the sampling point will be  $BRD/2 - 1 \times f_{ih}$ ,  $BRD/2$ , and  $BRD/2 + 2 \times f_{ih}$  in a UART bit time.
- Bit 1~0     **RxFTR1~RxFTR0:** Receiver FIFO trigger level (bytes)  
 00: 4 bytes in RX FIFO  
 01: 1 or more bytes in RX FIFO  
 10: 2 or more bytes in RX FIFO  
 11: 3 or more bytes in RX FIFO  
 For the receiver these bits define the number of received data bytes in the RX FIFO that will trigger the RXIF bit being set high, an interrupt will also be generated if the RIE bit is enabled. After the reset the receiver FIFO is empty.



• **RxCNT Register**

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

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

Bit 7~3 Unimplemented, read as “0”

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

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

**Baud Rate Generator**

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRDH/BRDL register and the second is the UART modulation control bits UMOD2~UMOD0. If a baud rate BR is required with UART clock  $f_H$ .

$$f_H/BR = \text{Integer Part} + \text{Fractional Part}$$

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

$$BRD = \text{TRUNC}(f_H/BR)$$

$$UMOD = \text{ROUND}[\text{MOD}(f_H/BR) \times 8]$$

Therefore, the actual baud rate is as following:

$$\text{Baud rate} = f_H / [BRD + (UMOD/8)]$$

**Calculating the Baud Rate and Error Values**

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

From the above formula, the  $BRD = \text{TRUNC}(f_H/BR) = \text{TRUNC}(17.36111) = 17$

The  $UMOD = \text{ROUND}[\text{MOD}(f_H/BR) \times 8] = \text{ROUND}(0.36111 \times 8) = \text{ROUND}(2.88888) = 3$

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

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

**Modulation Control Example**

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

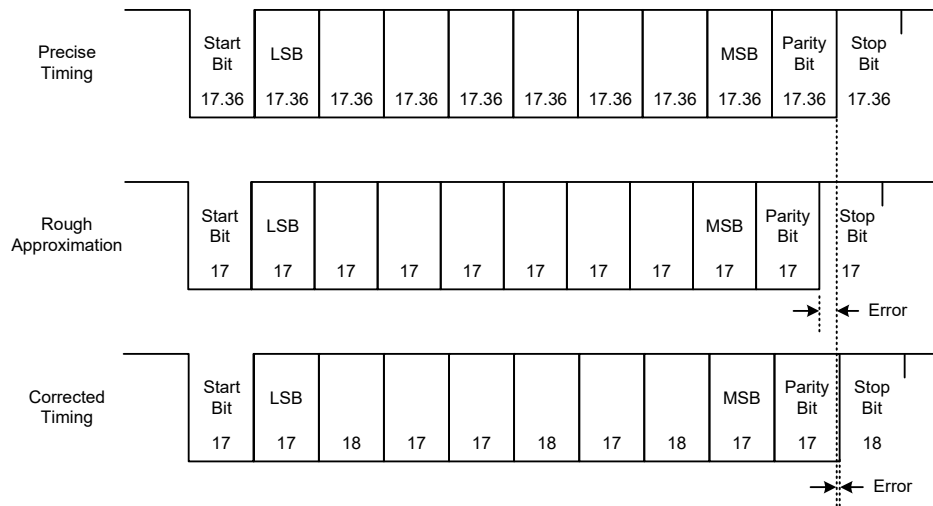
Fraction Addition	Carry to Bit 3	UART Bit Time Sequence	Extra UART Clock Cycle
0000b+0011b=0011b	No	Start bit	No
0011b+0011b=0110b	No	D0	No
0110b+0011b=1001b	Yes	D1	Yes
1001b+0011b=1100b	No	D2	No
1100b+0011b=1111b	No	D3	No
1111b+0011b=0010b	Yes	D4	Yes
0010b+0011b=0101b	No	D5	No
0101b+0011b=1000b	Yes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

### Baud Rate Correction Example

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

The following figure shows three different frames:

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



### UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd, mark, space or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding BNO, PRT1~PRT0, PREN, and STOPS bits. The baud rate used to transmit and receive data is setup using the internal 16-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

### Enabling/Disabling the UART Interface

The basic on/off function of the internal UART function is controlled using the UARTEN bit in the UCR1 register. If the UARTEN, TXEN and RXEN bits are set, then these two UART pins will act as normal TX output pin and RX input pin respectively. If no data is being transmitted on the TX pin, then it will default to a logic high value.

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

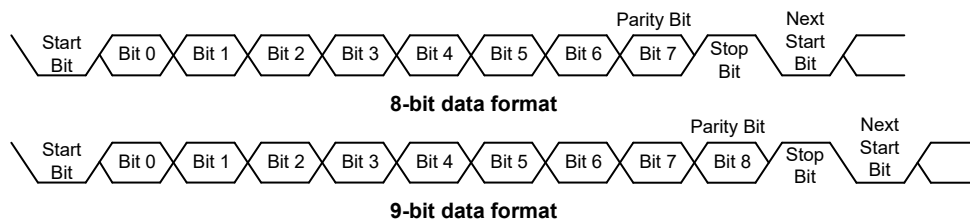
### Data, Parity and Stop Bit Selection

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

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
<b>Example of 8-bit Data Formats</b>				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
<b>Example of 9-bit Data Formats</b>				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

**Transmitter Receiver Data Format**

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



## UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR\_RXR register. The data to be transmitted is loaded into this TXR\_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the TXR\_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN bit is set, but the data will not be transmitted until the TXR\_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR\_RXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR\_RXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared functions by configuring the corresponding pin-shared control bits.

### Transmitting Data

When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant bit first. In the transmit mode, the TXR\_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8 bit in the UCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT1~PRT0, PREN and STOPS bits to define the required word length, parity type and the number of stop bits.
- Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.
- Set the TXEN bit to ensure that the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR\_RXR register. Note that this step will clear the TXIF bit.

This sequence of events can now be repeated to send additional data.

It should be noted that when TXIF=0, data will be inhibited from being written to the TXR\_RXR register. Clearing the TXIF flag is always achieved using the following software sequence:

1. A USR register access
2. A TXR\_RXR register write execution

The read-only TXIF flag is set by the UART hardware and if set indicates that the TXR\_RXR register is empty and that other data can now be written into the TXR\_RXR register without overwriting the previous data. If the TEIE bit is set then the TXIF flag will generate an interrupt.

During a data transmission, a write instruction to the TXR\_RXR register will place the data into the TXR\_RXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR\_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLE bit the following software sequence is used:

1. A USR register access
2. A TXR\_RXR register write execution

Note that both the TXIF and TIDLE bits are cleared by the same software sequence.

### Transmitting Break

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

### UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the BNO bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8 bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

### Receiving Data

When the UART receiver is receiving data, the data is serially shifted in on the external RX input pin, LSB first. In the read mode, the TXR\_RXR register forms a buffer between the internal bus and the receiver shift register. The TXR\_RXR register is a four byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR\_RXR before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

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

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

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

- The RXIF bit in the USR register will be set when the TXR\_RXR register has data available, the number of the available data bytes can be checked by polling the RxCNT register content.
- When the contents of the shift register have been transferred to the TXR\_RXR register and reach receiver FIFO trigger level if the RIEn bit is set, then an interrupt will be generated.

- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

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

1. A USR register access
2. A TXR\_RXR register read execution

### Receiving Break

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

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

### Idle Status

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

### Receiver Interrupt

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

### Managing Receiver Errors

Several types of reception errors can occur within the UART module, the following section describes the various types and how they are managed by the UART.

#### Overrun Error – OERR

The TXR\_RXR register is composed of a four byte deep FIFO data buffer, where four bytes can be held in the FIFO register, while a fifth byte can continue to be received. Before this fifth byte has been entirely shifted in, the data should be read from the TXR\_RXR register. If this is not done, the overrun error flag OERR will be consequently indicated.

In the event of an overrun error occurring, the following will happen:

- The OERR flag in the USR register will be set.
- The TXR\_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIE bit is set.

The OERR flag can be cleared by an access to the USR register followed by a read to the TXR\_RXR register.

#### **Noise Error – NF**

Over-sampling is used for data recovery to identify valid incoming data and noise. If noise is detected within a frame the following will occur:

- The read only noise flag, NF, in the USR register will be set on the rising edge of the RXIF bit.
- Data will be transferred from the Shift register to the TXR\_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the RXIF bit which itself generates an interrupt.

Note that the NF flag is reset by a USR register read operation followed by a TXR\_RXR register read operation.

#### **Framing Error – FERR**

The read only framing error flag, FERR, in the USR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, both stop bits must be high; otherwise the FERR flag will be set. The FERR flag and the received data will be recorded in the USR and TXR\_RXR registers respectively, and the flag is cleared in any reset.

#### **Parity Error – PERR**

The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd, even, mark or space, is selected. The read only PERR flag and the received data will be recorded in the USR and TXR\_RXR registers respectively. It is cleared on any reset, it should be noted that the flags, FERR and PERR, in the USR register should first be read by the application program before reading the data word.

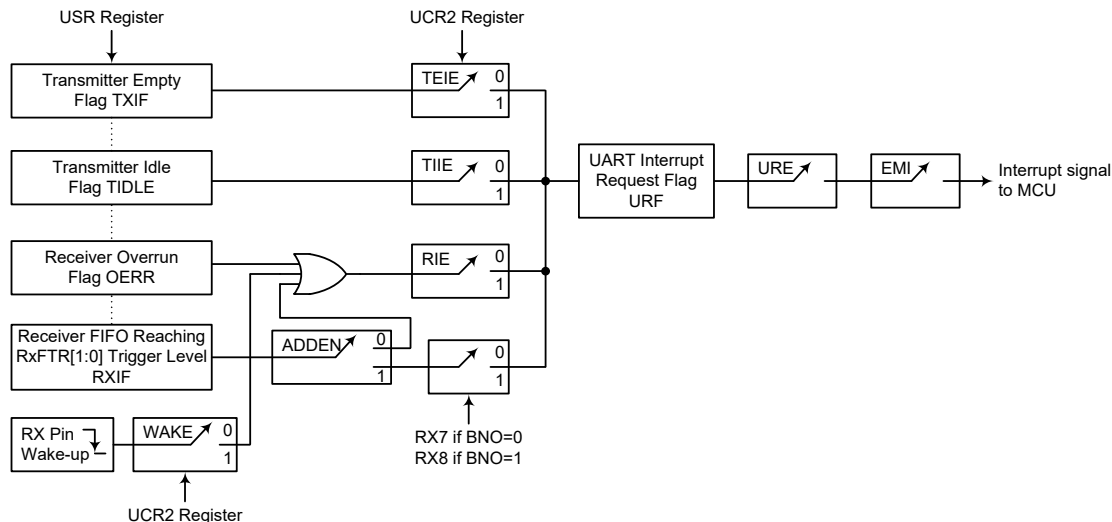
### **UART Interrupt Structure**

Several individual UART conditions can generate a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and its corresponding interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding USR register flags which will generate a UART interrupt if its associated interrupt enable control bit in the UCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the three receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual UART interrupt sources.

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX pin wake-up, which is also a UART

interrupt source, does not have an associated flag, but will generate a UART interrupt if the UART clock (f<sub>ih</sub>) source is switched off and the WAKE and RIE bits in the UCR2 register are set when a falling edge on the RX pin occurs.

Note that the USR register flags are read only and cannot be cleared or set by the application program, neither will they be cleared when the program jumps to the corresponding interrupt servicing routine, as is the case for some of the other interrupts. The flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART register section. The overall UART interrupt can be disabled or enabled by the related interrupt enable control bits in the interrupt control registers of the microcontroller to decide whether the interrupt requested by the UART module is masked out or allowed.



UART Interrupt Structure

**Address Detect Mode**

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the URE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to zero.

ADDEN	Bit 9 if BNO=1, Bit 8 if BNO=0	UART Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

ADDEN Bit Function



### UART Power Down and Wake-up

When the UART clock ( $f_{H}$ ) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock ( $f_{H}$ ) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the IDLE or SLEEP mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the USR, UCR1, UCR2, transmit and receive registers, as well as the BRDH and BRDL registers will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the UART clock ( $f_{H}$ ) is off, then a falling edge on the RX pin will trigger an RX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX pin will be ignored.

For a UART wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URE, must be set. If the EMI and URE bits are not set then only a wake up event will occur and no interrupt will be generated. Note also that as it takes certain system clock cycles after a wake-up before normal microcontroller resumes, the UART interrupt will not be generated until after this time has elapsed.

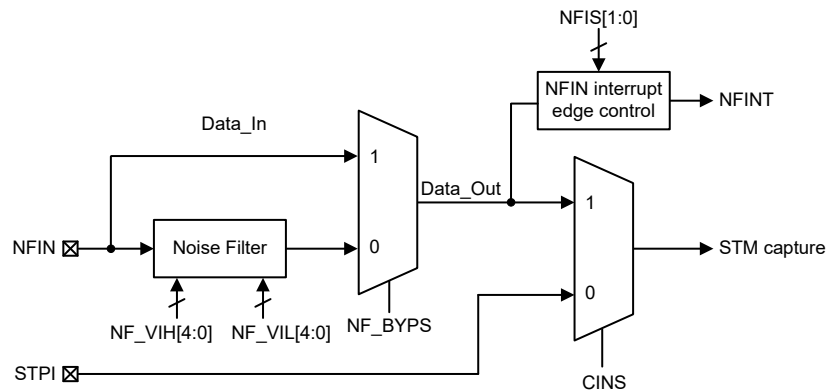
### Digital Noise Filter

The external NFIN pin is connected to an internal digital noise filter to reduce the possibility of unwanted event counting events or inaccurate pulse width measurements due to adverse noise or spikes on the NFIN input signal and then outputs to the 16-bit STM capture circuit in order to ensure that the motor control circuit works normally.

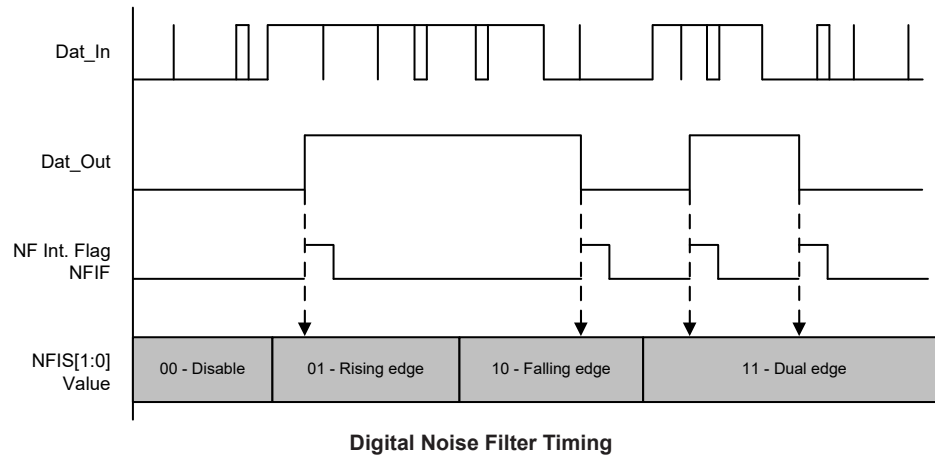
The noise filter circuit is an I/O surge filtering analog circuit which can filter micro-second grade sharp-noise.

Anti-noise pulse width maximum:

$$(NF\_VIH[4:0]-NF\_VIL[4:0]) \times (1/f_{SYS}) \times 4, \text{ where } (NF\_VIH[4:0]-NF\_VIL[4:0]) > 1$$



**Digital Noise Filter Block Diagram**



### Noise Filter Register Description

#### • NF\_VIH Register

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

- Bit 7      **NF\_BYPS**: Bypass Noise Filter control  
           0: Noise Filter used  
           1: Noise Filter Bypass, Dat\_Out=Dat\_In
- Bit 6      **CINS**: STM capture input source selection  
           0: Noise Filter Dat\_Out not selected (remains original STM capture path)  
           1: Noise Filter Dat\_Out selected
- Bit 5      Unimplemented, read as “0”
- Bit 4~0    **D4~D0**: NF\_VIH[4:0] data

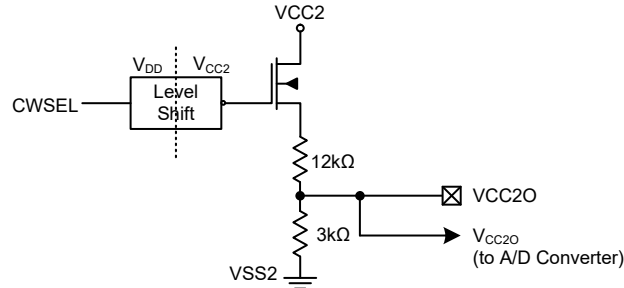
#### • NF\_VIL Register

Bit	7	6	5	4	3	2	1	0
Name	NFIS1	NFIS0	—	D4	D3	D2	D1	D0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	1	0	1	0

- Bit 7~6    **NFIS1~NFIS0**: NFIN interrupt edge control  
           00: Disable  
           01: Rising edge trigger  
           10: Falling edge trigger  
           11: Dual edge trigger
- Bit 5      Unimplemented, read as “0”
- Bit 4~0    **D4~D0**: NF\_VIL[4:0] data

## LDO Divider Circuit

The device contains an internal LDO which outputs 3.3V or 5V voltage for internal or external power supply. The LDO voltage selection is made using a configuration option. 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**

## Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external and internal interrupts functions. The external interrupts are generated by the action of the external INT and NFIN pins, while the internal interrupts are generated by various internal functions such as the Timer Modules, Time Bases, A/D converter and UART.

### 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~MFI2 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	—	—
INT Pin	INTE	INTF	—
UART	URE	URF	—
Noise Filter Input	NFIE	NFIF	—
Multi-function	MFnE	MFnF	n=0~2
Time Bases	TBnE	TBnF	n=0~1
A/D Converter	ADE	ADF	—
CTM	CTMAF	CTMAE	—
	CTMPF	CTMPE	
STM	STMAF	STMAE	—
	STMPF	STMPE	
PTM	PTMAF	PTMAE	—
	PTMPF	PTMPE	

**Interrupt Register Bit Naming Conventions**

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	NFIF	URF	INTF	NFIE	URE	INTE	EMI
INTC1	TB0F	MF2F	MF1F	MF0F	TB0E	MF2E	MF1E	MF0E
INTC2	—	—	—	TB1F	—	—	—	TB1E
MF10	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE
MF11	—	—	STMAF	STMPF	—	—	STMAE	STMPE
MF12	—	ADF	PTMAF	PTMPF	—	ADE	PTMAE	PTMPE

**Interrupt Register List**

• **INTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INTS1	INTS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **INTS1~INTS0**: Interrupt Edge Control for INT 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	—	NFIF	URF	INTF	NFIE	URE	INTE	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 **NFIF**: Noise Filter Input Interrupt Request Flag  
 0: No request  
 1: Interrupt request

- Bit 5      **URF**: UART Interrupt Request Flag  
             0: No request  
             1: Interrupt request
- Bit 4      **INTF**: External Interrupt Request Flag  
             0: No request  
             1: Interrupt request
- Bit 3      **NFIE**: Noise Filter Input Interrupt Control  
             0: Disable  
             1: Enable
- Bit 2      **URE**: UART Interrupt Control  
             0: Disable  
             1: Enable
- Bit 1      **INTE**: External 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	TB0F	MF2F	MF1F	MF0F	TB0E	MF2E	MF1E	MF0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **TB0F**: Time Base 0 Interrupt Request Flag  
             0: No request  
             1: Interrupt request
- Bit 6      **MF2F**: Multi-function 2 Interrupt Request Flag  
             0: No request  
             1: Interrupt request
- Bit 5      **MF1F**: Multi-function 1 Interrupt Request Flag  
             0: No request  
             1: Interrupt request
- Bit 4      **MF0F**: Multi-function 0 Interrupt Request Flag  
             0: No request  
             1: Interrupt request
- Bit 3      **TB0E**: Time Base 0 Interrupt Control  
             0: Disable  
             1: Enable
- Bit 2      **MF2E**: Multi-function 2 Interrupt Control  
             0: Disable  
             1: Enable
- Bit 1      **MF1E**: Multi-function 1 Interrupt Control  
             0: Disable  
             1: Enable
- Bit 0      **MF0E**: Multi-function 0 Interrupt Control  
             0: Disable  
             1: Enable

• **INTC2 Register**

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

- Bit 7~5 Unimplemented, read as “0”
- Bit 4 **TB1F**: Time Base 1 Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 3~2 Unimplemented, read as “0”
- Bit 0 **TB1E**: Time Base 1 Interrupt Control  
0: Disable  
1: Enable

• **MFI0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6 Unimplemented, read as “0”
- Bit 5 **CTMAF**: CTM CCRA Comparator Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 4 **CTMPF**: CTM CCRP Comparator Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 3~2 Unimplemented, read as “0”
- Bit 1 **CTMAE**: CTM CCRA Comparator Interrupt Control  
0: Disable  
1: Enable
- Bit 0 **CTMPE**: CTM CCRP Comparator Interrupt Control  
0: Disable  
1: Enable

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	STMAF	STMPF	—	—	STMAE	STMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6 Unimplemented, read as “0”
- Bit 5 **STMAF**: STM CCRA Comparator Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 4 **STMPF**: STM CCRP Comparator Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 3~2 Unimplemented, read as “0”
- Bit 1 **STMAE**: STM CCRA Comparator Interrupt Control  
0: Disable  
1: Enable
- Bit 0 **STMPE**: STM CCRP Comparator Interrupt Control  
0: Disable  
1: Enable

• **MFI2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	ADF	PTMAF	PTMPF	—	ADE	PTMAE	PTMPE
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      **PTMAF**: PTM Comparator A Interrupt Request Flag  
             0: No request  
             1: Interrupt request
- Bit 4      **PTMPF**: PTM 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      **PTMAE**: PTM Comparator A Interrupt Control  
             0: Disable  
             1: Enable
- Bit 0      **PTMPE**: PTM Comparator P Interrupt Control  
             0: Disable  
             1: Enable

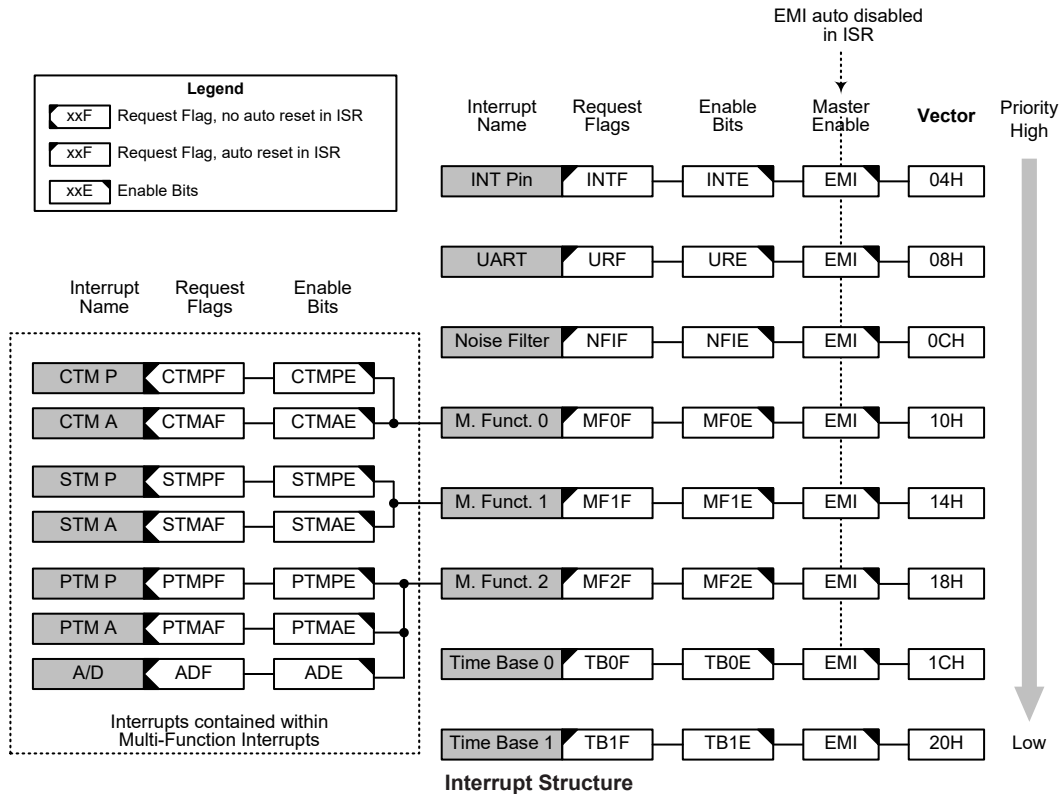
**Interrupt Operation**

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.



### External Interrupt

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

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

### **UART Interrupt**

Several individual UART conditions can generate a UART interrupt. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. To allow the program to branch to the respective interrupt vector addresses, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of these conditions are created, a subroutine call to the UART Interrupt vector, will take place. When the UART Interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

### **Noise Filter Input Interrupt**

The noise filter input external interrupt is controlled by signal transitions on the NFIN pin. An external noise filtered input interrupt request will take place when the noise filter input interrupt request flag, NFIF, is set, which will occur when a transition appears on the external Noise Filter input pin. Additionally the correct interrupt edge type must be selected using the NFIS1 and NFIS0 bits in the NF\_VIL register to enable the external interrupt function and to choose the trigger edge type. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective NFIN input interrupt enable bit, NFIE, must first be set. As the external noise filter interrupt pin NFIN is pin-shared with I/O pin, it should be configured as the noise filter input pin by the corresponding pin-shared function selection bits before the Noise Filter external interrupt function is enabled. 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 noise filter interrupt pin, a subroutine call to the noise filter interrupt vector will take place. When the interrupt is serviced, the noise filter input interrupt request flag, NFIF, 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 NFIN pin will remain valid even if the pin is used as a noise filter interrupt input.

The NF\_VIL register is used to select the type of active edge that will trigger the noise filter interrupt. A choice of either rising or falling or both edge types can be chosen to trigger a noise filter interrupt. Note that the NF\_VIL register can also be used to disable the noise filter interrupt function.

### **Multi-function Interrupts**

Within the device there are three 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 TM 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 will not be automatically reset and must be manually reset by the application program.

### **Timer Module Interrupts**

The Compact, Standard and Periodic Type TM have two interrupts. All of the TM interrupts are contained within the Multi-function Interrupt. For each of the TMs there are two interrupt request flags xTMPF and xTMAF and two enable bits xTMPE and xTMAE. A TM interrupt request will take place when any of the TM request flags is set, a situation which occurs when a TM comparator P or comparator A match situation happens.

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

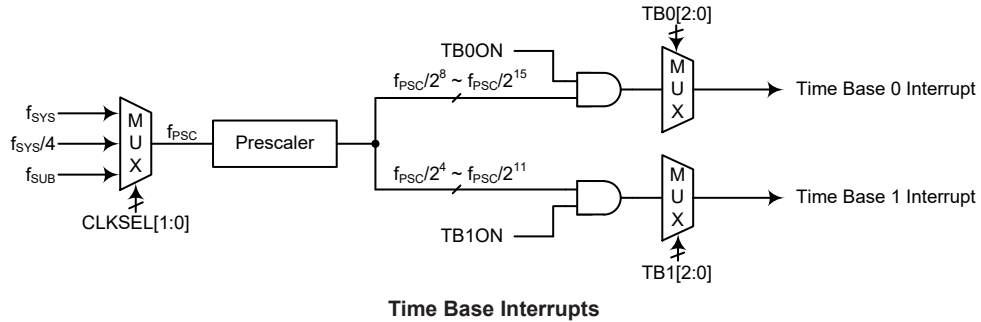
### **A/D Converter Interrupt**

The device contains an A/D Converter interrupt 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.

### **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. Its clock source,  $f_{PSC}$ , originates from the internal clock source  $f_{SYS}$ ,  $f_{SYS}/4$  or  $f_{SUB}$  and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL1~CLKSEL0 bits in the PSCR register.



• **PSCR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL1	CLKSEL0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source selection  
 00:  $f_{SYS}$   
 01:  $f_{SYS}/4$   
 1x:  $f_{SUB}$

• **TB0C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB0ON**: Time Base 0 Control  
 0: Disable  
 1: Enable

Bit 6~3 Unimplemented, read as “0”

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

• **TB1C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	—	—	—	—	TB12	TB11	TB10
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

- Bit 7      **TB1ON**: Time Base 1 Control  
0: Disable  
1: Enable
- Bit 6~3    Unimplemented, read as “0”
- Bit 2~0    **TB12~TB10**: Select Time Base 1 Time-out Period  
000:  $2^4/f_{PSC}$   
001:  $2^5/f_{PSC}$   
010:  $2^6/f_{PSC}$   
011:  $2^7/f_{PSC}$   
100:  $2^8/f_{PSC}$   
101:  $2^9/f_{PSC}$   
110:  $2^{10}/f_{PSC}$   
111:  $2^{11}/f_{PSC}$

**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.

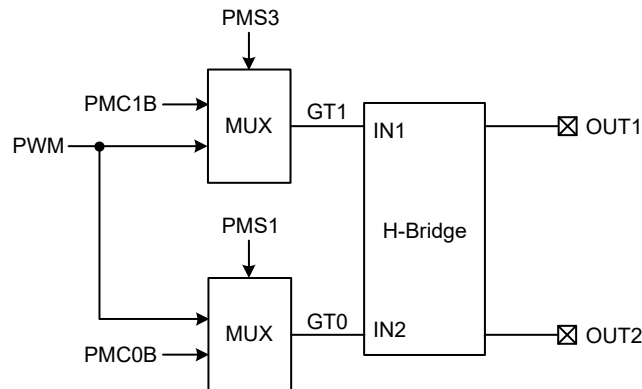
## Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
<b>LDO Option</b>	
1	LDO voltage selection: 3.3V or 5V

## H-Bridge Driver

The device provides H-bridge driver function. A full range of protection functions are integrated including OCP, OSP and OTP to prevent the driver damage even if the motor stalls or experiences a short circuit in critical operating environments.



Note: PWM stand for the PWM output signals from PTM.

**H-Bridge Driver Control Block Diagram**

## H-Bridge Driver Control Registers

The H-Bridge driver is implemented using several registers. These registers are used to select the PWM output.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PMS	—	—	—	—	PMS3	D2	PMS1	D0
PMC	—	—	—	—	PMC1T	D2	PMC0T	D0

**H-Bridge Driver Control Register List**

• **PMS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PMS3	D2	PMS1	D0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

- Bit 7 ~ 4      Unimplemented, read as “0”
- Bit 3          **PMS3**: Select the signal source of GT1 output  
0: from PTP  
1: from PMC1B control bit
- Bit 2          **D2**: Reserved bit, should be fixed as “1”
- Bit 1          **PMS1**: Select the signal source of GT0 output  
0: from PWM Output  
1: from PMC0B control bit
- Bit 0          **D0**: Reserved bit, should be fixed as “1”

• **PMC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PMC1T	D2	PMC0T	D0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

- Bit 7 ~ 4      Unimplemented, read as “0”
- Bit 3          **PMC1T**: Control the GT1 output  
0: GT1 output low  
1: GT1 output high
- Bit 2          **D2**: Reserved bit, should be fixed as “1”
- Bit 1          **PMC0T**: Control the GT0 output  
0: GT0 output low  
1: GT0 output high
- Bit 0          **D0**: Reserved bit, should be fixed as “1”

### H-Bridge Output Control

PA5	PA7	OUT1	OUT2	Operation Mode	H-Bridge Status			
					M1	M2	M3	M4
0	0	Z	Z	Standby	OFF	OFF	OFF	OFF
0	1	L	H	Reverse	OFF	ON	ON	OFF
1	0	H	L	Forward	ON	OFF	OFF	ON
1	1	L	L	Brake	OFF	ON	OFF	ON

Table 1. Operation True Table in Active Period

PA5	PA7	OUT1	OUT2	Operation Mode	H-Bridge Status			
					M1	M2	M3	M4
0	0	Z	Z	Standby	OFF	OFF	OFF	OFF
1	1	L	L	Brake	OFF	ON	OFF	ON

Table 2. Operation True Table in Sleep Period

Note: \* When PA5=PA7="0" for over 10ms, the H-Bridge will enter Sleep mode.

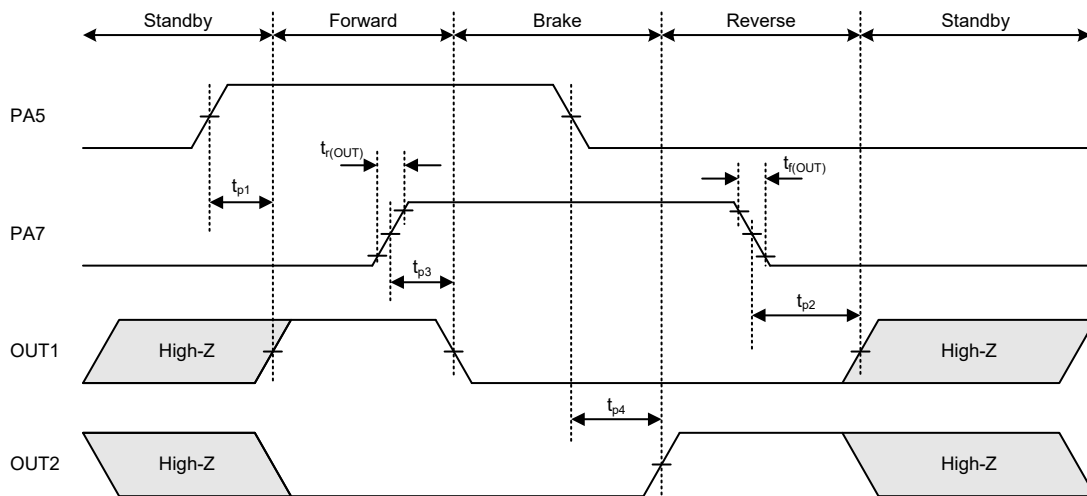


Figure 1. Control Logic and Sleep Mode Timing Diagram

### Active Period and Sleep Period

When the Standby or Brake mode continuously exceeds over 10ms, the H-bridge driver will enter the Sleep Period. At this time, the Standby or Brake mode still works in the Sleep Period as shown in Table 2. Change the operation mode to Forward or Reverse will go back to the Active Period.

In the Sleep Period, all functional blocks are turned off to reduce the current consumption to an ultra-low value of less than 0.1μA (max). At this time, switching PA5/PA7 to Brake or Standby configuration only affects the output – OUT1/OUT2, the driver remains in the Sleep Period (Figure 2). Since all functional blocks are turned off, the Standby and Brake mode outputs are not protected. When an PA5 or PA7 pin is set to “High”, the H-bridge driver will exit from the Sleep Period.

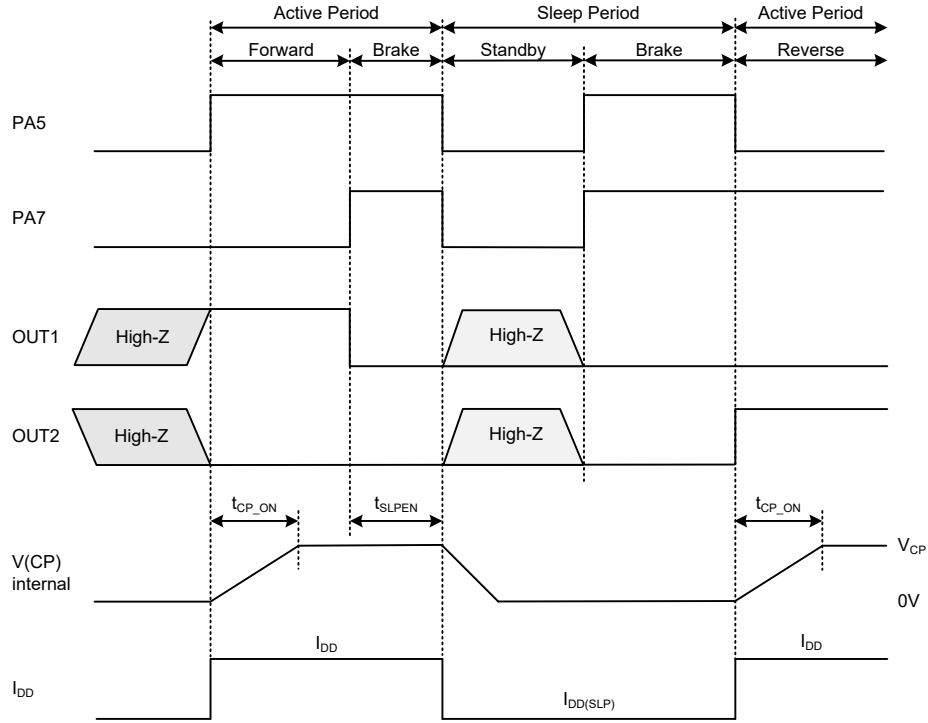


Figure 2. Operation Mode Control Timing Diagram

### V<sub>DD,H</sub> Under Voltage Lock-Out

In order to avoid an H-bridge metastable output condition when powered-on or with a low battery voltage, an under voltage lockout function is integrated within the device. During the power-on period, the H-bridge outputs will remain in high impedance states and the control inputs are ignored when V<sub>DD,H</sub> is lower than V<sub>UVLO+</sub>. The H-bridge outputs are only controlled by inputs when V<sub>DD,H</sub> is higher than V<sub>UVLO+</sub>. The device will be locked again when V<sub>DD,H</sub> falls to a voltage level lower than V<sub>UVLO-</sub>.

### Over Current Protection – OCP

The H-Bridge driver includes a fully integrated over current protection function within each of the internal power MOSFETs. When the motor current exceeds the over current protection threshold, I<sub>OCP</sub>, exceeding a de-glitch time, t<sub>DEG</sub>, all power MOSFETs will be turned off immediately. After the retry time times out, the driver will release the protection activation and allow normal operation to resume. The retry mechanism is only available in Forward and Reverse modes.

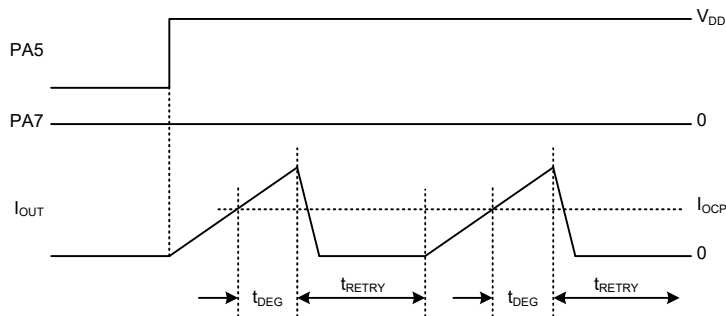


Figure 3. H-Bridge Driver OCP Reaction



### Output Short-Circuit Protection – OSP

The H-Bridge driver provides full output protection for conditions such as an output pin short to ground, to the motor supply or to each other. The driver detects the current through each power MOSFETs and compares it with the output short circuit protection threshold,  $I_{OSP}$ , without a de-glitch time. The current threshold  $I_{OSP}$  is internally set to 1.5 times the  $I_{OCP}$ . When an OSP condition occurs, the driver will turn off all power MOSFETs and keep checking the output status every retry time,  $t_{RETRY}$ , until the fault is removed. The retry mechanism is only available in Forward and Reverse modes.

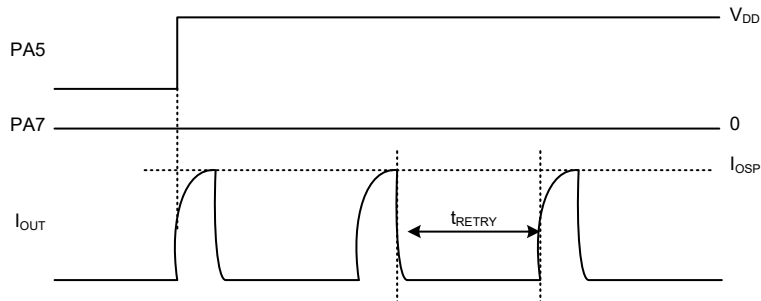


Figure 4. H-Bridge Driver OSP Reaction

### Over Temperature Protection – OTP

If the die temperature exceeds the internal limit threshold,  $T_{SHD}$ , the H-bridge driver will turn off all power MOSFETs until the temperature decreases to a specific level less than the recovery temperature,  $T_{REC}$ .

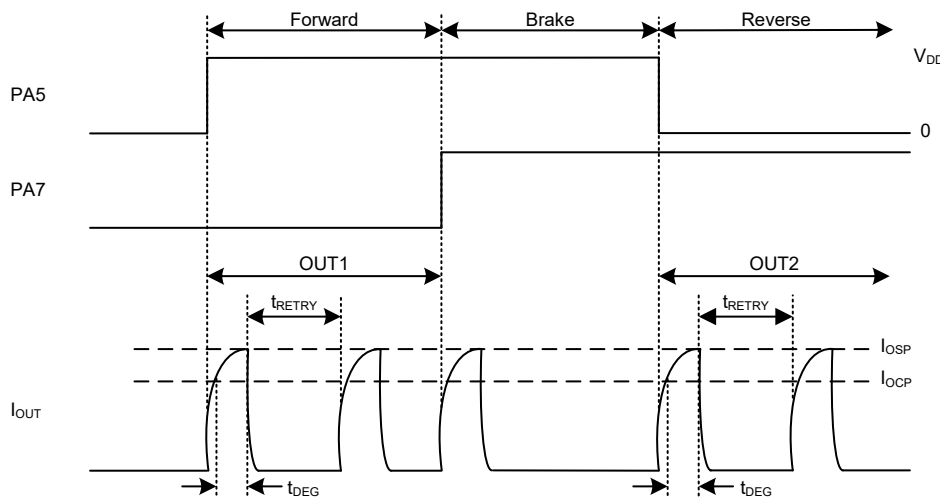


Figure 5. Retry Mechanism

The retry mechanism entry and release conditions are shown as follows.

Protection Type	Retry Entry Condition	Functional Mode				Retry Release Condition
		Forward/Reverse	Brake	Standby	Sleep	
OCP	$I_{OCP} > 3.0A$	O	—	—	—	$I_{OCP} < 3.0A$
OSP	OUTx-to-ground, OUTx-to-power or OUT1-to-OUT2 path	O	—	—	—	Short circuit fault is removed

Table 3. Retry Mechanism Conditions

The protection function entry and release conditions are shown as follows.

Protection Type	Protection Entry Condition	Functional Mode				Protection Release Condition
		Forward/Reverse	Brake	Standby	Sleep	
UVLO	$V_{IN} < 1.8V$	—	O	—	—	$V_{IN} > 2.5V$
OCP	$I_{OCP} > 3.0A$	O	O	—	—	$I_{OCP} < 3.0A$
OSP	OUTx-to-ground, OUTx-to-power or OUT1-to-OUT2 path	O	O	—	—	Short circuit fault is removed
OTP	$T_J > 155^{\circ}C$	O	O	O	—	$T_J < 120^{\circ}C$

**Table 4. Protection Function Conditions**

## Component/Motor Selection Guide

### Motor Consideration

The appropriate motor voltage depends upon the desired RPM and power supply source. Higher motor voltages also increase the motor current rate. Note that the motor stall current must be less than the internal limit output current,  $I_{OCP}$ , to avoid failures when the motor starts up.

### Controller Supply Capacitor

It is suggested to use at least a  $10\mu F$  value capacitor for C1. This provides the necessary power stability for the device excluding the H-Bridge.

### Motor Supply Capacitor

It is suggested to use at least a  $10\mu F$  value capacitor for C2. There are two main functions for this capacitor. Firstly, it absorbs the energy released by the motor to reduce any overshoot voltage damage. Secondly, it provides a transient power source to the motor to compensate for the battery response time or for long connecting wire effects when the motor starts up or for fast control switching between forward and reverse modes.

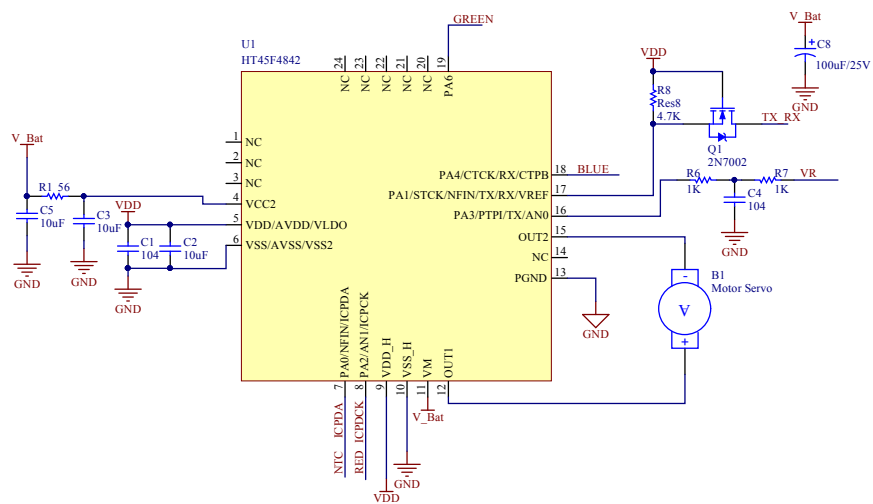
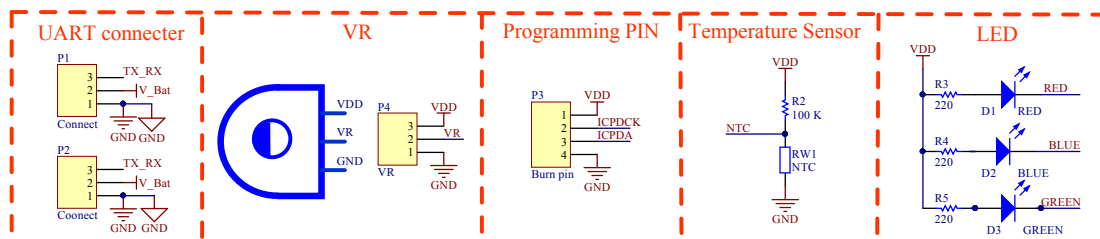
### Motor Bypass Capacitor

The bypass capacitor, C3, provides the fast flywheel path to release the inductive energy of the motor. In most applications, the capacitance value is set to a value of  $0.01\mu F$  to  $0.1\mu F$ . Usually this capacitor is internally contained within the motor and not required externally. In some applications, especially in low speed motors, the large internal motor resistor connected with the bypass capacitor in parallel may result in an instantaneous large current when the motor starts up. It may however trigger a faulty OCP/OSP reaction which will fail to start up the motor. There are two ways to solve this phenomenon: decrease the bypass capacitor value or add a  $47\Omega$  to  $100\Omega$  resistor in series with the bypass capacitor.

### Motor Voltage Zener Diode

The Zener Diode, ZD1, is located at the input of the VM pin to prevent the motor's back EMF voltage from flowing into the VM pin. This back EMF voltage may exceed the IC's rated voltage and cause damage. The ZD1 is set to a value of 19V.

### Application Circuits



## Instruction Set

### Introduction

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

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

### Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5 $\mu$ 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.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

3. 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.

### Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

Mnemonic	Description	Cycles	Flag Affected
<b>Arithmetic</b>			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	2 <sup>Note</sup>	Z, C, AC, OV, SC
LADC A,[m]	Add Data Memory to ACC with Carry	2	Z, C, AC, OV, SC
LADCM A,[m]	Add ACC to Data memory with Carry	2 <sup>Note</sup>	Z, C, AC, OV, SC
LSUB A,[m]	Subtract Data Memory from ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	2 <sup>Note</sup>	Z, C, AC, OV, SC, CZ
LSBC A,[m]	Subtract Data Memory from ACC with Carry	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	2 <sup>Note</sup>	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2 <sup>Note</sup>	C
<b>Logic Operation</b>			
LAND A,[m]	Logical AND Data Memory to ACC	2	Z
LOR A,[m]	Logical OR Data Memory to ACC	2	Z
LXOR A,[m]	Logical XOR Data Memory to ACC	2	Z
LANDM A,[m]	Logical AND ACC to Data Memory	2 <sup>Note</sup>	Z
LORM A,[m]	Logical OR ACC to Data Memory	2 <sup>Note</sup>	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2 <sup>Note</sup>	Z
LCPL [m]	Complement Data Memory	2 <sup>Note</sup>	Z
LCPLA [m]	Complement Data Memory with result in ACC	2	Z
<b>Increment &amp; Decrement</b>			
LINCA [m]	Increment Data Memory with result in ACC	2	Z
LINC [m]	Increment Data Memory	2 <sup>Note</sup>	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2 <sup>Note</sup>	Z
<b>Rotate</b>			
LRRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2 <sup>Note</sup>	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2 <sup>Note</sup>	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2 <sup>Note</sup>	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2 <sup>Note</sup>	C
<b>Data Move</b>			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2 <sup>Note</sup>	None
<b>Bit Operation</b>			
LCLR [m].i	Clear bit of Data Memory	2 <sup>Note</sup>	None
LSET [m].i	Set bit of Data Memory	2 <sup>Note</sup>	None



Mnemonic	Description	Cycles	Flag Affected
<b>Branch</b>			
LSZ [m]	Skip if Data Memory is zero	2 <sup>Note</sup>	None
LSZA [m]	Skip if Data Memory is zero with data movement to ACC	2 <sup>Note</sup>	None
LSNZ [m]	Skip if Data Memory is not zero	2 <sup>Note</sup>	None
LSZ [m].i	Skip if bit i of Data Memory is zero	2 <sup>Note</sup>	None
LSNZ [m].i	Skip if bit i of Data Memory is not zero	2 <sup>Note</sup>	None
LSIZ [m]	Skip if increment Data Memory is zero	2 <sup>Note</sup>	None
LSDZ [m]	Skip if decrement Data Memory is zero	2 <sup>Note</sup>	None
LSIZA [m]	Skip if increment Data Memory is zero with result in ACC	2 <sup>Note</sup>	None
LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC	2 <sup>Note</sup>	None
<b>Table Read</b>			
LTABRD [m]	Read table to TBLH and Data Memory	3 <sup>Note</sup>	None
LTABRDL [m]	Read table (last page) to TBLH and Data Memory	3 <sup>Note</sup>	None
LITABRD [m]	Increment table pointer TBLP first and Read table to TBLH and Data Memory	3 <sup>Note</sup>	None
LITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	3 <sup>Note</sup>	None
<b>Miscellaneous</b>			
LCLR [m]	Clear Data Memory	2 <sup>Note</sup>	None
LSET [m]	Set Data Memory	2 <sup>Note</sup>	None
LSWAP [m]	Swap nibbles of Data Memory	2 <sup>Note</sup>	None
LSWAPA [m]	Swap nibbles of Data Memory with result in ACC	2	None

Note: 1. For these extended skip instructions, if the result of the comparison involves a skip then up to four cycles are required, if no skip takes place two cycles is required.

- Any extended instruction which changes the contents of the PCL register will also require three cycles for execution.

## 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 ← Program Counter + 1 Program Counter ← 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] ← 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 ← 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 ← 0 PDF ← 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] ← $\overline{[m]}$
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 ← $\overline{[m]}$
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] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← 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 ← ACC "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 ← ACC "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] ← ACC "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	Program Counter ← 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	Program Counter ← Stack ACC ← 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	Program Counter ← Stack EMI ← 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) ← [m].i; (i=0~6) [m].0 ← [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 ← [m].(i+1); (i=0~6) ACC.7 ← C C ← [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 ← 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 ← 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] ← 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] ← [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 ← [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

## Extended Instruction Definition

The extended instructions are used to directly access the data stored in any data memory sections.

<b>LADC 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>LADCM 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>LADD 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>LADDM 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>LAND 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>LANDM 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>LCLR [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>LCLR [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>LCPL [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>LCPLA [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>LDAA [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>LDEC [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>LDECA [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>LINC [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>LINCA [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>LMOV 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>LMOV [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>LOR 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>LORM 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>LRL [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	$[m].(i+1) \leftarrow [m].i; (i=0\sim 6)$ $[m].0 \leftarrow [m].7$
Affected flag(s)	None
<b>LRLA [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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim 6)$ $ACC.0 \leftarrow [m].7$
Affected flag(s)	None
<b>LRLC [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	$[m].(i+1) \leftarrow [m].i; (i=0\sim 6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C
<b>LRLCA [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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim 6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C

<b>LRR [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	$[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow [m].0$
Affected flag(s)	None
<b>LRRRA [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	$ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow [m].0$
Affected flag(s)	None
<b>LRRC [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	$[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
<b>LRRCA [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	$ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
<b>LSBC 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>LSBCM 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>LSDZ [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>LSDZA [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>LSET [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>LSET [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>LSIZ [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>LSIZA [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>LSNZ [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>LSNZ [m]</b>	Skip if Data Memory is not 0
Description	If the content of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m] ≠ 0
Affected flag(s)	None
<b>LSUB 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 ← ACC – [m]
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>LSUBM 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] ← ACC – [m]
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>LSWAP [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~[m].0 ↔ [m].7~[m].4
Affected flag(s)	None
<b>LSWAPA [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~ACC.0 ← [m].7~[m].4 ACC.7~ACC.4 ← [m].3~[m].0
Affected flag(s)	None
<b>LSZ [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>LSZA [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 ← [m] Skip if [m]=0
Affected flag(s)	None



<b>LSZ [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>LTABRD [m]</b>	Read table (current page) to TBLH and Data Memory
Description	The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
<b>LTABRDL [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>LITABRD [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 (TBLP 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>LITABRDL [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>LXOR 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>LXORM 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

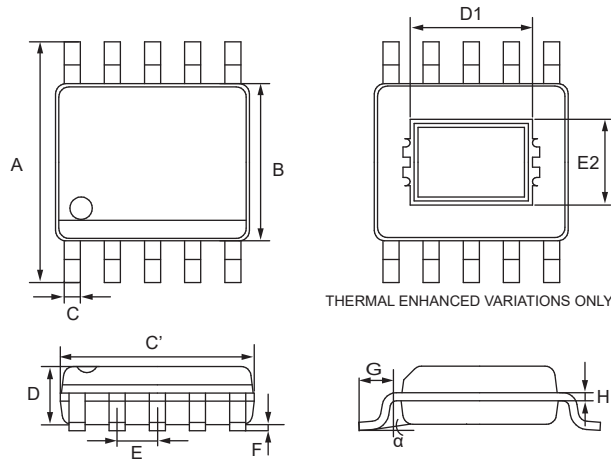
## Package Information

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 [Holttek 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](#)

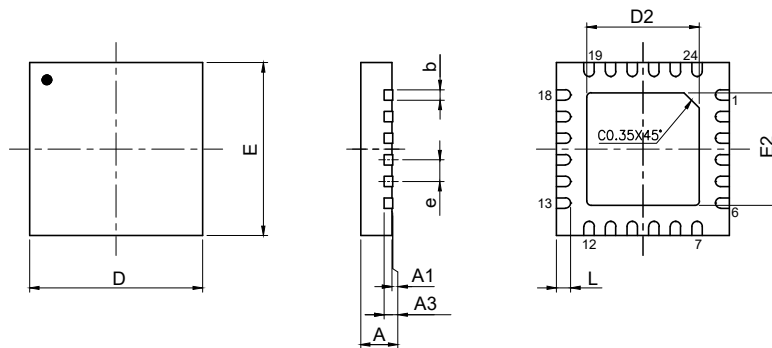
**10-pin SOP-EP Outline Dimensions**



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.018
C'	—	0.193 BSC	—
D	—	—	0.069
D1	0.059	—	—
E	—	0.039 BSC	—
E2	0.039	—	—
F	0.000	—	0.006
G	0.016	—	0.050
H	0.004	—	0.010
$\alpha$	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.0 BSC	—
B	—	3.9 BSC	—
C	0.30	—	0.45
C'	—	4.9 BSC	—
D	—	—	1.75
D1	1.50	—	—
E	—	1.0 BSC	—
E2	1.00	—	—
F	0.00	—	0.15
G	0.40	—	1.27
H	0.10	—	0.25
$\alpha$	0°	—	8°

**SAW Type 24-pin QFN (4mm×4mm×0.75mm) Outline Dimensions**



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	—	0.008 BSC	—
b	0.007	0.010	0.012
D	—	0.157 BSC	—
E	—	0.157 BSC	—
e	—	0.020 BSC	—
D2	0.104	0.106	0.108
E2	0.104	0.106	0.108
L	0.014	0.016	0.018

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	—	0.203 BSC	—
b	0.18	0.25	0.30
D	—	4.00 BSC	—
E	—	4.00 BSC	—
e	—	0.50 BSC	—
D2	2.65	2.70	2.75
E2	2.65	2.70	2.75
L	0.350	0.400	0.450

Copyright© 2020 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>.