PIR/Microwave Flash MCU

BA45F6622

Revision: V1.10 Date: November 04, 2019

www.holtek.com

Table of Contents

Features

CPU Features

- Operating voltage
  - f_{SYS}=8MHz: 2.2V~5.5V
- Up to 0.5µs instruction cycle with 8MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 8MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 1K×14
- RAM Data Memory: 64×8
- Emulated EEPROM Memory: 32×14
- Watchdog Timer function
- Up to 6 bidirectional I/O lines
- One external interrupt pin shared with I/O pin
- One Timer Module for time measurement, input capture, compare match output or PWM output or single pulse output function
- Dual Time Base functions for generation of fixed time interrupt signals
- 2 external channel 10-bit resolution A/D converter with internal reference voltage V_{BG}
- Auto conversion function
  - A/D conversion data lower/upper limit preset
- Dual operational amplifiers including a programmable gain amplifier (OPA1)
- LDO function
- Low voltage reset function
- Package types: 16-pin NSOP, 16-pin QFN

General Description

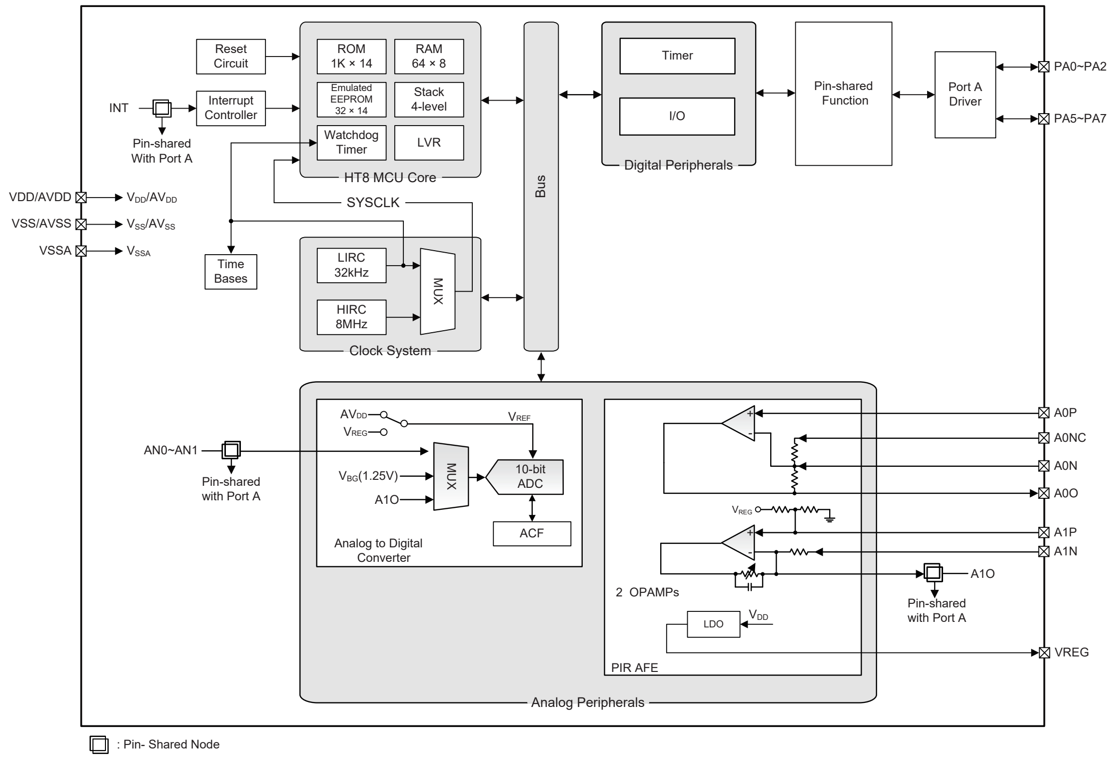
The BA45F6622 is an 8-bit high performance RISC architecture microcontroller, designed especially for PIR/Microwave Sensor applications. Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of Emulated EEPROM memory for storage of non-volatile data such as serial numbers, calibration data, etc.

Analog feature includes a multi-channel 10-bit A/D converter, LDO and internal operational amplifiers. An extremely flexible Timer Module provides timing, pulse generation and PWM generation functions. Protective features such as an internal Watchdog Timer and Low Voltage Reset function coupled with excellent noise immunity and ESD protection ensures that reliable operation is maintained in hostile electrical environments.

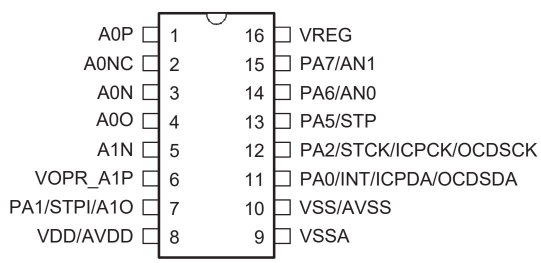
A full choice of internal high and low speed oscillators are provided and the two fully integrated system oscillators require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimize microcontroller operation and minimize power consumption.

The inclusion of flexible I/O programming features and Time Base funcitons along with other features ensure that only a minimum of external components is required for application implementation, resulting in reduced component cost, low power consumption and high performance. In addition to the above features, the integrated A/D converter and ACF (Auto Conversion Function), cooperated with external PIR/Microwave Sensor applications, provide the device with versatility for a wide range of PIR/Microwave Sensor products, such as PIR bulb, Microwave sensor switch, visitor notification, security monitoring, yard lamp, intelligent appliance to name but a few.

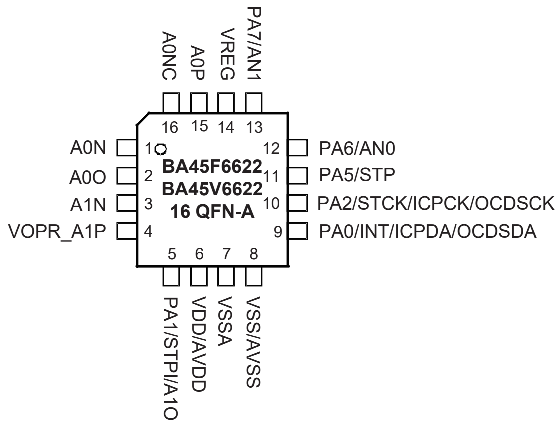
Block Diagram



Pin Assignment



BA45F6622/BA45V6622
16 NSOP-A



Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.
2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and are only available for the BA45V6622 device which is the OCDS EV chip for the BA45F6622 device.

Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet. As each Pin Description table shows the situation for the package with the most pins, not all pins in the tables will be available on smaller package sizes.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT/ICPDA/OCDSDA	PA0	PAWU PAPU PAPD	ST	CMOS	General purpose I/O. Register enabled pull-high, pull-low and wake-up
	INT	INTEG INTC0	ST	—	External interrupt input
	ICPDA	—	ST	CMOS	ICP data/address pin
	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
PA1/STPI/A1O	PA1	PAWU PAPU PAPD PASR	ST	CMOS	General purpose I/O. Register enabled pull-high, pull-low and wake-up
	STPI	PASR	ST	—	STM capture input
	A1O	PASR	—	AN	OPA1 output
PA2/STCK/ICPCK/OCDSCK	PA2	PAWU PAPU PAPD	ST	CMOS	General purpose I/O. Register enabled pull-high, pull-low and wake-up
	STCK	—	ST	—	STM clock input
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA5/STP	PA5	PAWU PAPU PAPD PASR	ST	CMOS	General purpose I/O. Register enabled pull-high, pull-low and wake-up
	STP	PASR	—	CMOS	STM output

Pin Name	Function	OPT	I/T	O/T	Description
PA6/AN0	PA6	PAWU PAPU PAPD PASR	ST	CMOS	General purpose I/O. Register enabled pull-high, pull-low and wake-up
	AN0	PASR	AN	—	A/D converter external input channel 0
PA7/AN1	PA7	PAWU PAPU PAPD PASR	ST	CMOS	General purpose I/O. Register enabled pull-high, pull-low and wake-up
	AN1	PASR	AN	—	A/D converter external input channel 1
A0O	A0O	—	—	AN	OPA0 output
A0P	A0P	—	AN	—	OPA0 non-inverting input
A0N	A0N	—	AN	—	OPA0 inverting input
A0NC	A0NC	—	AN	—	OPA0 inverting input via capacitor
VOPR_A1P	VOPR_A1P	—	AN	—	OPA1 non-inverting input
A1N	A1N	—	AN	—	OPA1 inverting input
VREG	VREG	—	—	PWR	LDO output voltage
VDD/AVDD	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
VSS/AVSS	VSS	—	PWR	—	Digital negative power supply
	AVSS	—	PWR	—	Analog negative power supply
VSSA	VSSA	—	PWR	—	PIR AFE negative power supply

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

Absolute Maximum Ratings

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

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

Storage Temperature............ -50°C to 125°C

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

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

I_{OL} Total 80mA

Total Power Dissipation 500mW

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

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	f_{SYS}=8MHz	2.2	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32kHz	2.2	—	5.5	V

Operating Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Max.@ 85°C	Unit
		V_{DD}	Conditions					
I_{DD}	SLOW Mode – LIRC	2.2V	f_{SYS}=32kHz	—	8	16	16	μA
		3V		—	10	20	20	
		5V		—	30	50	50	
	FAST Mode – HIRC	2.2V	f_{SYS}=8MHz	—	0.6	1.0	1.0	mA
		3V		—	0.8	1.2	1.2	
		5V		—	1.6	2.4	2.4	

Note: When using the characteristic table data, the following notes should be taken into consideration:
1. Any digital inputs are set 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_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.2V	WDT off	—	0.08	0.12	1.40	μA
		3V		—	0.08	0.12	1.40	
		5V		—	0.15	0.29	2.20	
		2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3	5	6	
	IDLE0 Mode – LIRC	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE1 Mode – HIRC	2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	

Note: When using the characteristic table data, the following notes should be taken into consideration:
1. Any digital inputs are set in a non-floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

A.C. Characteristics

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

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 3V or 5V.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	

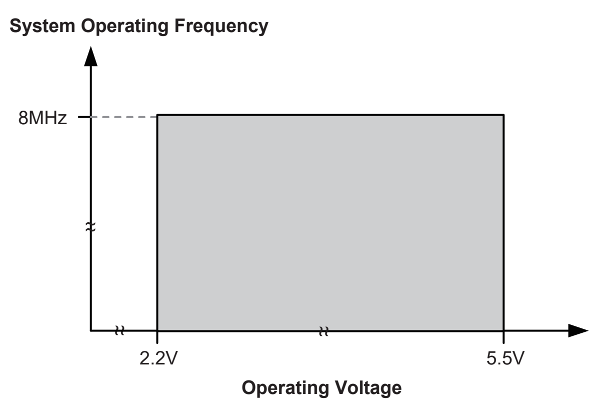
Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

2.The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

Low Speed Internal Oscillator – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency (Writer Trim)	3V/5V	25°C	-1%	32	+1%	kHz
	LIRC Frequency	2.2V~3.6V (trim @ 3V)	-10°C~50°C	-4.0%	32	+4.0%	
		3.3V~5.5V (trim @ 5V)	-10°C~50°C	-4.0%	32	+4.0%	
			-40°C~85°C	-6.0%	32	+6.0%	
		2.2V~5.5V (trim @ 3V)	-40°C~85°C	-6.0%	32	+6.0%	
t_{START}	LIRC Start-up Time	—	-40°C~85°C	—	—	100	μs

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time Wake-up from Condition where f_{SYS} is Off	—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time Wake-up from Condition where f_{SYS} is On	—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode	—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
t_{RSTD}	System Reset Delay Time Reset Source from Power-on Reset or LVR Hardware Reset	—	RR_{POR}=5V/ms	42	48	54	ms
	System Reset Delay Time LVRC/WDTC Software Reset	—	—				
	System Reset Delay Time Reset Source from WDT Overflow	—	—	14	16	18	ms
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

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

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

3. 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_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} 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

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Pins	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Pins	3V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	
R_{PH}	Pull-high Resistance for I/O Ports (1)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
R_{PL}	Pull-low Resistance for I/O Ports (2)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I_{LEAK}	Input Leakage Current for I/O Ports	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{TPI}	STPI Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TCK}	STCK Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs

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

2. The R_{PL} internal pull-low resistance value is calculated by connecting to V_{DD} and enabling the input pin with a pull-low resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R_{PL} value.

Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Flash Program / Emulated EEPROM Memory							
V_{DD}	V_{DD} for Read – Emulated EEPROM Memory	—	—	2.2	—	5.5	V
	V_{DD} for Erase/Write – Emulated EEPROM Memory	—	—	4.5	5.0	5.5	
t_{DEW}	Erase/Write Time – Flash Program Memory	5.0V	Ta=25°C	—	2	3	ms
	Erase/Write Cycle Time – Emulated EEPROM Memory	—	EWRTS[1:0]=00B	—	2	3	ms
		—	EWRTS[1:0]=01B	—	4	6	
		—	EWRTS[1:0]=10B	—	8	12	
		—	EWRTS[1:0]=11B	—	16	24	
E_P	Cell Endurance	—	—	10K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DD}	Operating Voltage for Read/Write	—	—	V_{DDmin}	—	V_{DDmax}	V
V_{DR}	RAM Data Retention Voltage	—	Device in SLEEP Mode	1.0	—	—	V

Note: The Emulated EEPROM erase/write operation can only be executed when the f_{SYS} clock frequency is equal to or greater than 2MHz.

LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	-5%	2.1	+5%	V
I_{LVRBG}	Operating Current	3V	LVR enable, VBGEN=0	—	—	18	μA
		5V		—	20	25	
		3V	LVR enable, VBGEN=1	—	—	150	μA
		5V		—	180	200	
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
I_{LVR}	Additional Current for LVR Enable	—	VBGEN=0	—	—	25	μA

Internal Reference Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{BG}	Bandgap Reference Voltage	—	—	-5%	1.25	+5%	V
I_{BG}	Additional Current for Bandgap Reference Enable	—	LVR disable	—	—	180	μA

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

A/D Converter Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	2	—	AV_{DD}	V
N_R	Resolution	—	—	—	—	10	Bit
DNL	Differential Non-linearity	—	$V_{REF}=AV_{DD}$, t_{ADCK}=0.5μs	-1.5	—	1.5	LSB
INL	Integral Non-linearity	—	$V_{REF}=AV_{DD}$, t_{ADCK}=0.5μs	-2	—	2	LSB
I_{ADC}	Additional Current for ADC Enable	2.2V	No load (t_{ADCK}=0.5μs)	—	300	420	μA
		3V		—	340	500	
		5V		—	500	700	
t_{ADCK}	Clock Period	—	—	0.5	—	10	μs
t_{ON2ST}	A/D Converter On-to-start Time	—	—	4	—	—	μs
t_{ADS}	Sampling Time	—	—	—	4	—	t_{ADCK}
t_{ADC}	Conversion Time (Include ADC Sample and Hold Time)	—	—	—	14	—	t_{ADCK}
I_{buf}	Pre-buffer Operating Current	5V	No load	—	270	450	μA
V_{OS}	Pre-buffer Input Offset Voltage	5V	—	-15	—	15	mV
GBW	Pre-buffer Gain Band Bandwidth	5V	R_L=1MΩ, C_L=100pF	0.5	—	—	MHz
PSRR	Power Supply Rejection Ratio	5V	—	60	80	—	dB
CMRR	Common Mode Rejection Ratio	5V	—	60	80	—	dB
A_{OL}	Open Loop Gain	5V	No load	60	80	—	dB
SR	Slew Rate+, Slew Rate-	5V	No load	1.8	2.5	—	V/μs
V_{CM}	Common Mode Voltage Range	5V	—	V_{SS}	—	V_{DD}	V

OP Amplifier Electrical Characteristics

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OPA}	Operating Current	5V	OPAnBW[1:0]=00B, no load	—	3	5	μA
			OPAnBW[1:0]=01B, no load	—	10	16	
			OPAnBW[1:0]=10B, no load	—	80	128	
			OPAnBW[1:0]=11B, no load	—	200	320	
V_{OS}	Input Offset Voltage	5V	Without calibration (OnOF[5:0]=100000B)	-15	—	15	mV
			With calibration	-2	—	2	mV
I_{OS}	Input Offset Current	5V	$V_{IN}=1/2V_{CM}$	—	1	10	nA
V_{CM}	Common Mode Voltage Range	5V	OPAnBW[1:0]=00B, 01B, 10B, 11B	V_{SS}	—	V_{DD} -1.4	V
PSRR	Power Supply Rejection Ratio	5V	OPAnBW[1:0]=00B, 01B, 10B, 11B	50	70	—	dB
CMRR	Common Mode Rejection Ratio	5V	OPAnBW[1:0]=00B, 01B, 10B, 11B	50	80	—	dB
A_{OL}	Open Loop Gain	5V	OPAnBW[1:0]=00B, 01B, 10B, 11B	60	80	—	dB

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
SR	Slew Rate	5V	R_{LOAD}=1MΩ, C_{LOAD}=60pF, OPAnBW[1:0]=00B	0.5	1.5	—	V/ms
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, OPAnBW[1:0]=01B	5	15	—	
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, OPAnBW[1:0]=10B	180	500	—	
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, OPAnBW[1:0]=11B	600	1800	—	
GBW	Gain Bandwidth	5V	R_{LOAD}=1MΩ, C_{LOAD}=60pF, OPAnBW[1:0]=00B	2.5	5.0	—	kHz
		5V	R_{LOAD}=1MΩ, C_{LOAD}=60pF, OPAnBW[1:0]=01B	20	40	—	
		5V	R_{LOAD}=1MΩ, C_{LOAD}=60pF, OPAnBW[1:0]=10B	400	600	—	
		5V	R_{LOAD}=1MΩ, C_{LOAD}=60pF, OPAnBW[1:0]=11B	1300	2000	—	
V_{OR}	Maximum Output Voltage Range of A0O	5V	OPAnBW[1:0]=00B, 01B R_{LOAD}=5kΩ to V_{DD}/2	V_{SS} +80	—	V_{DD} -120	mV
			OPAnBW[1:0]=10B, 11B R_{LOAD}=5kΩ to V_{DD}/2	V_{SS} +40	—	V_{DD} -60	
	Maximum Output Voltage Range of A1O Which is Pin-shared with PA1	5V	OPAnBW[1:0]=00B, 01B R_{LOAD}=5kΩ to V_{DD}/2	V_{SS} +140	—	V_{DD} -160	mV
			OPAnBW[1:0]=10B, 11B R_{LOAD}=5kΩ to V_{DD}/2	V_{SS} +120	—	V_{DD} -140	
I_{SC}	Output Short Circuit Current	5V	R_{LOAD}=5.1Ω, OPAnBW[1:0]=00B, 01B	±6	±12	—	mA
			R_{LOAD}=5.1Ω, OPAnBW[1:0]=10B, 11B	±10	±20	—	

LDO Characteristics

V_{IN}=V_{OUT}+1V, C_{LOAD}=10μF+0.1μF, Ta=-40°C~85°C, unless otherwise specified

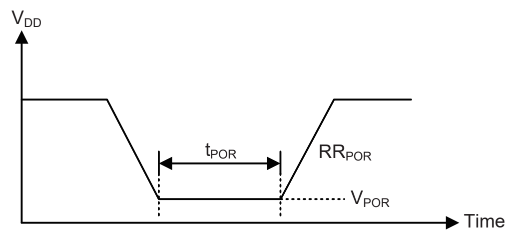
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IN}	Supply Voltage	—	—	2.7	3.3	5.5	V
V_{OUT}	Output Voltage	—	V_{REG} output decided by VSEL fields	-3%	V_{REG}	3%	V
I_{OUT}	Driving Current	5V	V_{IN}=5V, C_{VREG}=0.1μF, ΔV_{OUT}=-3%	1	—	—	mA
I_Q	Quiescent Current	—	After startup, no load, include bandgap consumption	—	—	8	μA

Note: 1. This LDO can provide stable power supply for PIR sensor with a 10μF capacitor.
2. The VREG pin should be connected to 0.1μF capacitor for ADC reference voltage and 10μF capacitor for PIR sensor.

Power-on Reset Characteristics

Ta=-40°C~85°C

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



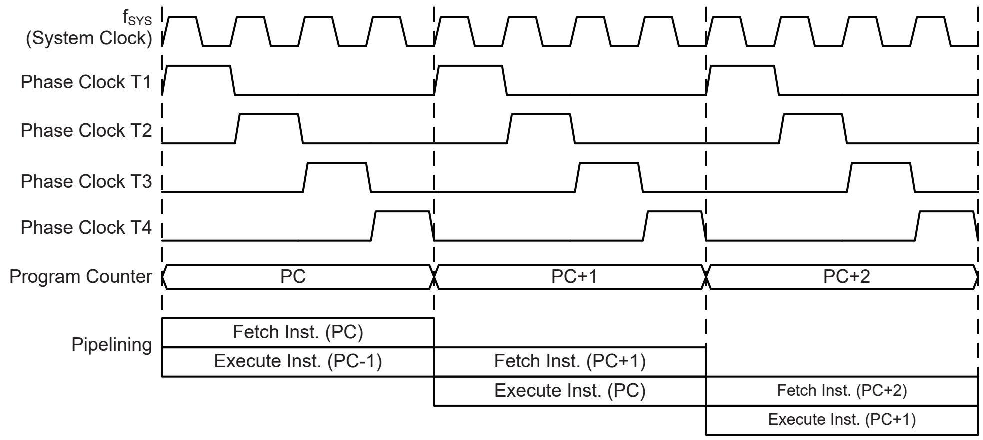
System Architecture

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

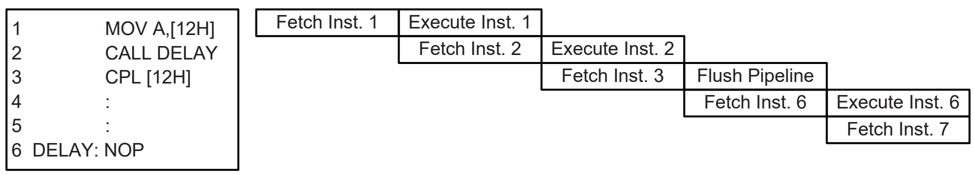
Clocking and Pipelining

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

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

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

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

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

Program Counter

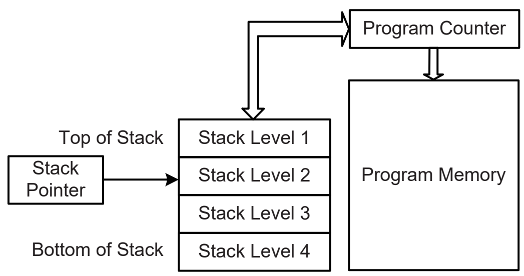
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack, organized into 4 levels, is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



Arithmetic and Logic Unit – ALU

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

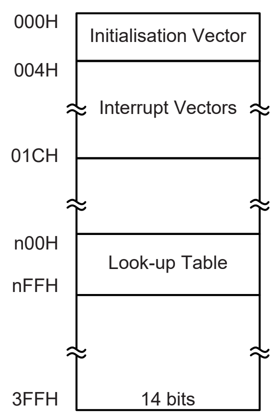
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement: INCA, INC, DECA, DEC
- Branch decision: JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET,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

Flash Program Memory

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

Structure

The Program Memory has a capacity of 1K×14 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be set in any location within the Program Memory, is addressed by a separate table pointer register.



Program Memory Structure

Special Vectors

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

Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the "TABRD [m]" or "TABRDL [m]" instructions respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as "0".

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

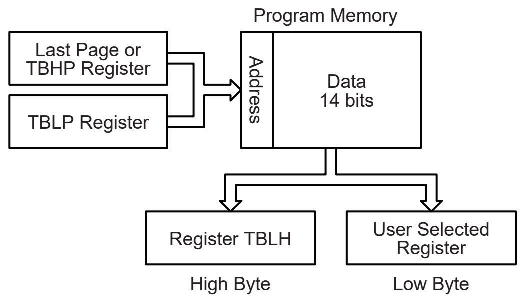


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontrollers. 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 "300H" which refers to the start address of the last page within the 1K Program Memory of the device. The table pointer low byte register is set 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 "306H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by the TBLP and TBHP registers if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

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

Table Read Program Example

```
tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
:
mov a,06h          ; initialise low table pointer - note that this address is referenced
mov tblp,a         ; to the last page or the page that tbhp pointed
mov a,03h          ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1     ; transfers value in table referenced by table pointer,
                   ; data at program memory address "306H" 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 "305H" transferred to tempreg2 and TBLH
                   ; in this example the data "1AH" is transferred to tempreg1 and data "0FH"
                   ; to register tempreg2
                   ; the value "00H" will be transferred to the high byte register TBLH
```

```
:
:
org 300h          ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
```

In Circuit Programming – ICP

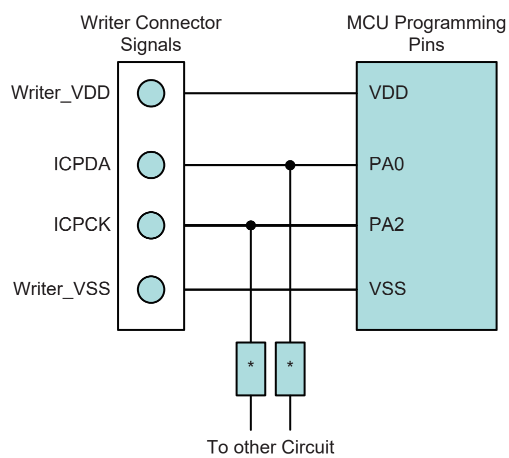
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

As an additional convenience, Holtek has provided a means of programming the microcontrollers 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/AVDD	Power supply
VSS	VSS/AVSS	Ground

The program memory can be programmed serially in-circuit using this 4-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply. The technical details regarding the in-circuit programming of the device 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 BA45V6622 which is used to emulate the BA45F6622 device. The EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

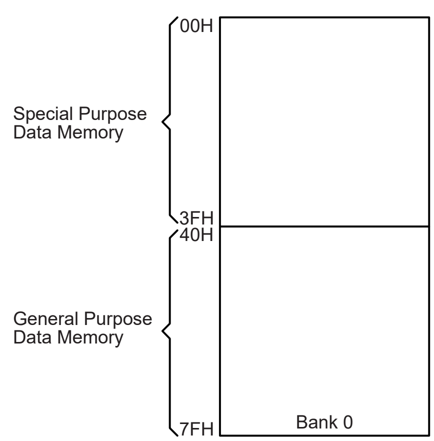
Data Memory

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

Structure

Categorised into two types, the first of these is an area of RAM, known as the Special Function Data Memory. 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 known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The start address of the Data Memory for the device is 00H. The address range of the Special Purpose Data Memory for the device is from 00H to 3FH while the address range of the General Purpose Data Memory is from 40H to 7FH.



Data Memory Structure

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".

	Bank 0		Bank 0
00H	IAR0	20H	EDH
01H	MP0	21H	STMC0
02H	(unused)	22H	STMC1
03H	(unused)	23H	STMDL
04H	(unused)	24H	STMDH
05H	ACC	25H	STMAL
06H	PCL	26H	STMAH
07H	TBLP	27H	SADOL
08H	TBLH	28H	SADOH
09H	TBHP	29H	SADC0
0AH	STATUS	2AH	SADC1
0BH	SCC	2BH	OPAC0
0CH	HIRCC	2CH	OPAC1
0DH	WDTC	2DH	ACFC0
0EH	LVRC	2EH	ACFC1
0FH	RSTFC	2FH	LULV
10H	INTEG	30H	HULV
11H	INTC0	31H	LLLV
12H	INTC1	32H	HLLV
13H	VBGC	33H	OPAC2
14H	PA	34H	OPA0VOS
15H	PAC	35H	OPA1VOS
16H	PAPU	36H	LDOC
17H	PAWU	37H	(unused)
18H	PAPD	38H	(unused)
19H	PASR	39H	(unused)
1AH	TB0C	3AH	(unused)
1BH	TB1C	3BH	(unused)
1CH	PSCR	3CH	(unused)
1DH	ECR	3DH	(unused)
1EH	EAR	3EH	(unused)
1FH	EDL	3FH	(unused)

(shaded): Unused, read as 00H

Special Purpose Data Memory

Special Function Register Description

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

Indirect Addressing Register – IAR0

The Indirect Addressing Register, IAR0, although having its location in normal RAM register space, does not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses this Indirect Addressing Register and Memory Pointer, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 register will result in no actual read or write operation to this register but rather to the memory location specified by the corresponding Memory Pointer, MP0. Acting as a pair, IAR0 and MP0 can together access data from Bank 0. As the Indirect Addressing Register is not physically implemented, reading the Indirect Addressing Register will return a result of “00H” and writing to the register will result in no operation.

Memory Pointer – MP0

One Memory Pointer, known as MP0 is provided. This Memory Pointer is 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 Indirect Addressing Register is carried out, the actual address that the microcontroller is directed to is the address specified by the Memory Pointer. MP0, together with the Indirect Addressing Register, IAR0, are used to access data from Bank 0.

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

Indirect Addressing Program Example

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

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

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 set before any table read commands are executed. Their value can be changed, for example using the "INC" or "DEC" instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

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

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

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

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

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

- **STATUS Register**

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

"x": Unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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

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

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

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

The "C" flag is also affected by a rotate through carry instruction.

Emulated EEPROM Data Memory

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

Emulated EEPROM Data Memory Structure

The Emulated EEPROM Data Memory capacity is 32×14 bits for the device. The Emulated EEPROM Erase operation is carried out in a page format while the Write and Read operation is carried out in a word format. The page size is assigned with a capacity of 16 words. Note that the Erase operation should be executed before the Write operation is executed.

Operations	Format
Erase	16 words/page
Write	1 word/time
Read	1 word/time
Note: Page size=16 words.	

Emulated EEPROM Erase/Write/Read Format

Erase Page	EAR4	EAR[3:0]
0	0	xxxx
1	1	xxxx

"x": Don't care

Erase Page Number and Selection

Emulated EEPROM Registers

Four registers control the overall operation of the Emulated EEPROM Data Memory. These are the address register, EAR, the data registers, EDL and EDH, and a single control register, ECR.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EAR	—	—	—	EAR4	EAR3	EAR2	EAR1	EAR0
EDL	D7	D6	D5	D4	D3	D2	D1	D0
EDH	—	—	D13	D12	D11	D10	D9	D8
ECR	EWRTS1	EWRTS0	EEREN	EER	EWREN	EWR	ERDEN	ERD

Emulated EEPROM Register List

• EAR Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	EAR4	EAR3	EAR2	EAR1	EAR0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **EAR4~EAR0**: Emulated EEPROM address bit 4 ~ bit 0

• **EDL 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**: Emulated EEPROM data bit 7 ~ bit 0

• **EDH Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **D13~D8**: Emulated EEPROM data bit 13 ~ bit 8

• **ECR Register**

Bit	7	6	5	4	3	2	1	0
Name	EWRTS1	EWRTS0	EEREN	EER	EWREN	EWR	ERDEN	ERD
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 **EWRTS1~EWRTS0**: Emulated EEPROM erase/write time selection
00: 2ms
01: 4ms
10: 8ms
11: 16ms

Bit 5 **EEREN**: Emulated EEPROM erase enable
0: Disable
1: Enable
This bit is used to enable the Emulated EEPROM erase function and must be set high before erase operations are carried out. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Clearing this bit to zero will inhibit the Emulated EEPROM erase operations.

Bit 4 **EER**: Emulated EEPROM erase control
0: Erase cycle has finished
1: Activate an erase cycle
When this bit is set high by the application program, an erase cycle will be activated. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Setting this bit high will have no effect if the EEREN has not first been set high.

Bit 3 **EWREN**: Emulated EEPROM write enable
0: Disable
1: Enable
This bit is used to enable the Emulated EEPROM write function and must be set high before write operations are carried out. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Clearing this bit to zero will inhibit the Emulated EEPROM write operations.

Bit 2 **EWR**: Emulated EEPROM write control
0: Write cycle has finished
1: Activate a write cycle
When this bit is set high by the application program, a write cycle will be activated. This bit will be automatically reset to zero by the hardware after the write cycle has

finished. Setting this bit high will have no effect if the EWREN has not first been set high.

Bit 1 **ERDEN**: Emulated EEPROM read enable
0: Disable
1: Enable
This bit is used to enable the Emulated EEPROM read function and must be set high before read operations are carried out. Clearing this bit to zero will inhibit the Emulated EEPROM read operations.

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

Note: 1. The EEREN, EER, EWREN, EWR, ERDEN and ERD cannot be set to “1” at the same time in one instruction.
2. Note that the CPU will be stopped when a read, write or erase operation is successfully activated.
3. Ensure that the f_{SYS} clock frequency is equal to or greater than 2MHz and the f_{SUB} clock is stable before executing the erase or write operation.
4. Ensure that the read, write or erase operation is totally complete before executing other operations.

Erasing the Emulated EEPROM

For Emulated EEPROM erase operation the desired erase page address should first be placed in the EAR register. The number of the page erase operation is 16 words per page, therefore, the available page erase address is only specified by the EAR4 bit in the EAR register and the content of EAR3~EAR0 in the EAR register is not used to specify the page address. To erase the Emulated EEPROM page, the EEREN bit in the ECR register must first be set high to enable the erase function. After this the EER bit in the ECR register must be immediately set high to initiate an erase cycle. These two instructions must be executed in two consecutive instruction cycles to activate an erase operation successfully. The global interrupt bit EMI should also first be cleared before implementing any erase operations, and then set high again after the a valid erase activation procedure has completed. Note that the CPU will be stopped when an erase operation is successfully activated. When the erase cycle terminates, the CPU will resume executing the application program. And the EER bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been erased. The Emulated EEPROM erased page content will all be zero after an erase operation.

Writing Data to the Emulated EEPROM

For Emulated EEPROM write operation, the data and the desired write address should first be placed in the EDL/EDH registers and the EAR register respectively. To write data to the Emulated EEPROM, the EWREN bit in the ECR register must first be set high to enable the write function. After this the EWR bit in the ECR register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that the CPU will be stopped when a write operation is successfully activated. When the write cycle terminates, the CPU will resume executing the application program. And the EWR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the Emulated EEPROM.

Reading Data from the Emulated EEPROM

For Emulated EEPROM read operation the desired read address should first be placed in the EAR register. To read data from the Emulated EEPROM, the ERDEN bit in the ECR register must first be set high to enable the read function. After this a read cycle will be initiated if the ERD bit in the ECR register is now set high. Note that the CPU will be stopped when the read operation is successfully activated. When the read cycle terminates, the CPU will resume executing the application program. And the ERD bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been read from the Emulated EEPROM. Then the data can be read from the EDH/EDL data register pair by application program. The data will remain in the data register pair until another read, write or erase operation is executed.

Programming Considerations

Care must be taken that data is not inadvertently written to the Emulated EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process. When writing or erasing data the EWR or EER bit must be set high immediately after the EWREN or EEREN bit has been set high, to ensure the write or erase cycle executes correctly. The global interrupt bit EMI should also be cleared before a write or erase cycle is executed and then set again after a valid write or erase activation procedure has completed. Note that the device should not enter the IDLE or SLEEP mode until Emulated EEPROM read, write or erase operation is totally complete. Otherwise, Emulated EEPROM read, write or erase operation will fail.

Programming Examples

Erasing a Data Page of the Emulated EEPROM – polling method

```
MOV  A, EEPROM_ADRES      ; user-defined page
MOV  EAR, A
MOV  A, 00H               ; Erase time=2ms (40H for 4ms, 80H for 8ms, C0H for 16ms)
MOV  ECR, A
CLR  EMI
SET  EEREN                ; set EEREN bit, enable erase operation
SET  EER                  ; start Erase Cycle - set EER bit - executed immediately after
                          ; setting EEREN bit
SET  EMI
BACK:
SZ   EER                  ; check for erase cycle end
JMP  BACK
:
```

Writing Data to the Emulated EEPROM – polling method

```
MOV  A, EEPROM_ADRES      ; user-defined address
MOV  EAR, A
MOV  A, EEPROM_DATA_L     ; user defined data
MOV  EDL, A
MOV  A, EEPROM_DATA_H
MOV  EDH, A
MOV  A, 00H               ; Write time=2ms (40H for 4ms, 80H for 8ms, C0H for 16ms)
MOV  ECR, A
CLR  EMI
SET  EWREN                ; set EWREN bit, enable write operation
SET  EWR                  ; start Write Cycle - set EWR bit - executed immediately after
                          ; setting EWREN bit
```

```
SET  EMI
BACK:
SZ   EWR                 ; check for write cycle end
JMP  BACK
:
```

Reading Data from the Emulated EEPROM – polling method

```
MOV  A, EEPROM_ADRES     ; user defined address
MOV  EAR, A
SET  ERDEN               ; set ERDEN bit, enable read operation
SET  ERD                 ; start Read Cycle - set ERD bit
BACK:
SZ   ERD                 ; check for read cycle end
JMP  BACK
CLR  ECR                 ; disable Emulated EEPROM read if no more read operations are
                         ; required
MOV  A, EDL              ; move read data to user-defined register
MOV  READ_DATA_L, A
MOV  A, EDH
MOV  READ_DATA_H, A
```

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

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 operations are selected through the relevant control registers.

Oscillator Overview

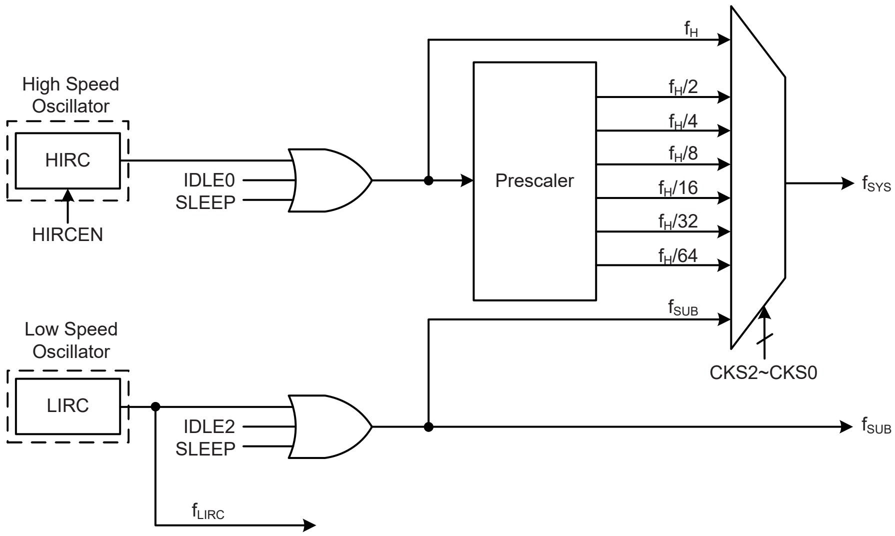
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. The fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators.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

Type	Name	Frequency
Internal High Speed RC	HIRC	8MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two oscillator sources, one high speed oscillator and one low speed oscillator. The high speed system clock is sourced from the internal 8MHz 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 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 high speed RC oscillator has one fixed frequency of 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Internal 32kHz Oscillator – LIRC

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

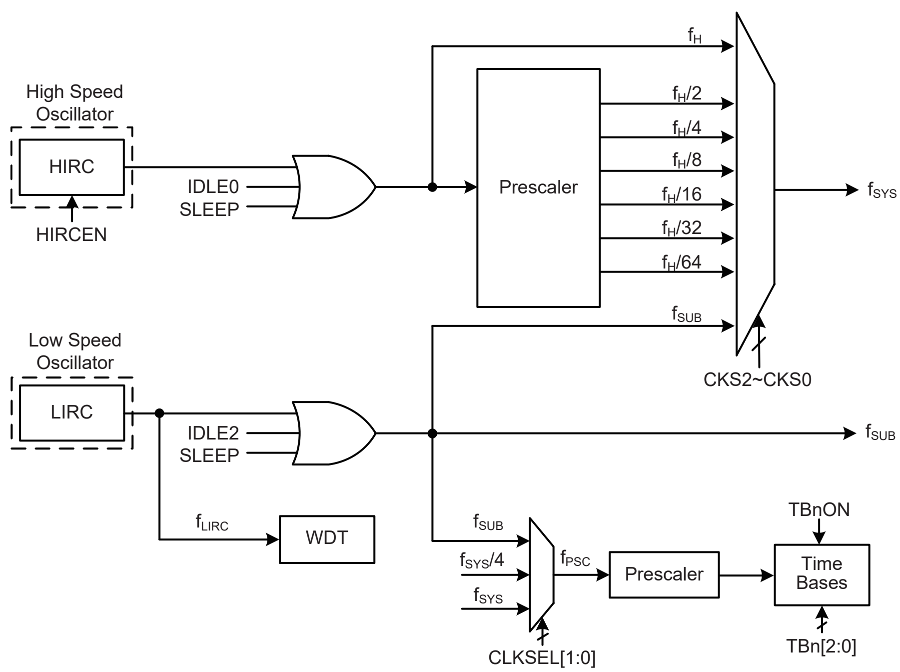
Operating Modes and System Clocks

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

System Clocks

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

The main system clock, can come from 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 $f_H/2$~$f_H/64$.



Device Clock Configurations

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

System Operation Modes

There are six different modes of operation for the microcontrollers, 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	Related Register Value			f_{SYS}	f_H	f_{SUB}	f_{LIRC}	f_{ACF}
		FHIDEN	FSIDEN	CKS[2:0]					
FAST Mode	On	x	x	000~110	f_H~f_H/64	On	On	On	On
SLOW Mode	On	x	x	111	f_{SUB}	On/Off(1)	On	On	On/Off(3)
IDLE0 Mode	Off	0	1	000~110	Off	Off	On	On	On/Off(3)
				111	On				
IDLE1 Mode	Off	1	1	xxx	On	On	On	On	On
IDLE2 Mode	Off	1	0	000~110	On	On	Off	On	On
				111	Off				
SLEEP Mode	Off	0	0	xxx	Off	Off	Off	On/Off(2)	On/Off(3)

"x": Don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.
2. The f_{LIRC} clock can be switched on or off which is controlled by the WDT function or the ACF function being enabled or disabled in the SLEEP mode.
3. The f_{ACF} clock can be switched on or off which is controlled by the ACF function being enabled or disabled in the SLOW/IDLE0/SLEEP mode.

FAST Mode

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

SLEEP Mode

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

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is low and the FSIDEN bit in the SCC register is high. In the IDLE0 Mode the CPU will be switched off but 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 SCC and HIRCC registers 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 a "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 a "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"

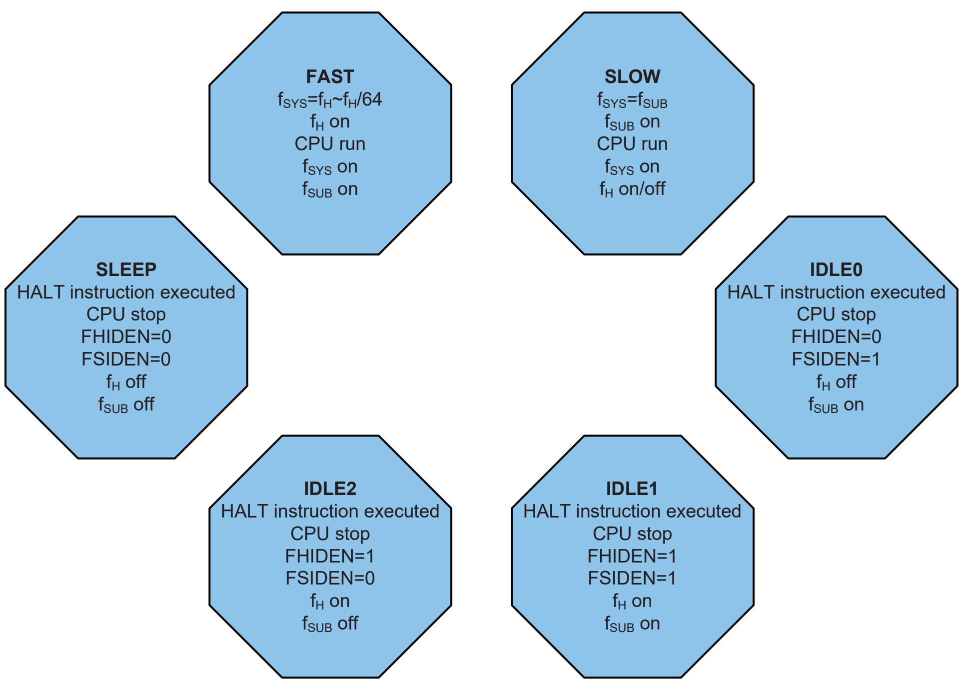
Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is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

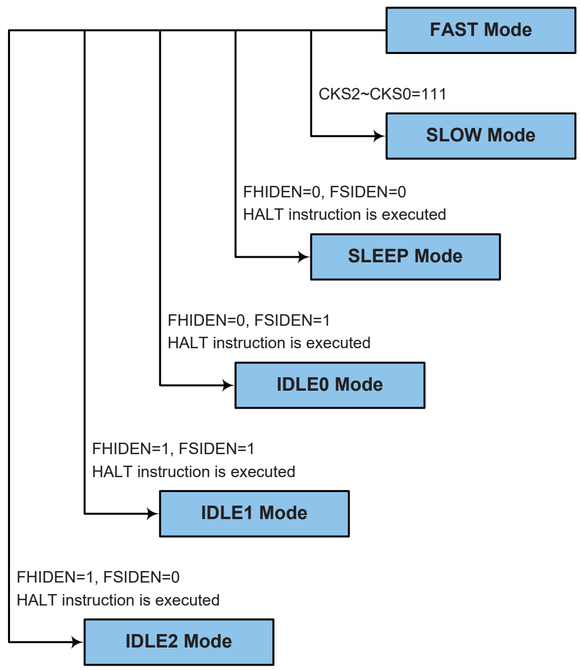
In simple terms, mode switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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

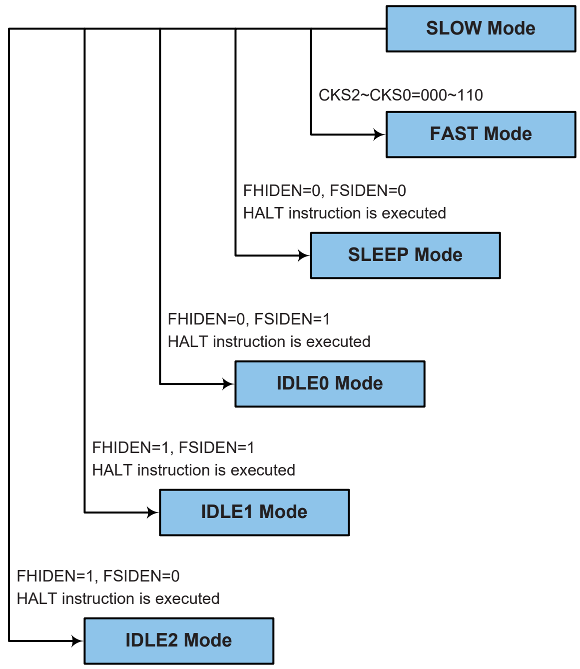
The SLOW Mode system clock is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

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

However, if f_H is not used in the 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 stabilisation is specified in the System Start Up Time Characteristics.



Entering the SLEEP Mode

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

- The system clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting 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 the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The f_H clock will be 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 the Watchdog time-out 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 IDLE1 Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

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

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These must either be set as outputs or if set 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 set 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, it will enter the IDLE or SLEEP mode and 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 set using the PAWU register to permit a negative transition on the pin to wake-up the system. When a pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which wakes 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, the enable/disable operation as the MCU reset operation.

• **WDTC Register**

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

Bit 7~3 **WE4~WE0**: WDT function software control
10101: Disable
01010: Enable
Other values: MCU reset
When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set high.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $[(2^8-2^0)\sim2^8]/f_{LIRC}$
001: $[(2^9-2^1)\sim2^9]/f_{LIRC}$
010: $[(2^{10}-2^2)\sim2^{10}]/f_{LIRC}$
011: $[(2^{11}-2^3)\sim2^{11}]/f_{LIRC}$
100: $[(2^{12}-2^4)\sim2^{12}]/f_{LIRC}$
101: $[(2^{13}-2^5)\sim2^{13}]/f_{LIRC}$
110: $[(2^{14}-2^6)\sim2^{14}]/f_{LIRC}$
111: $[(2^{15}-2^7)\sim2^{15}]/f_{LIRC}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

• **RSTFC Register**

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

"x": Unknown

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.

Bit 1 **LRF**: LVR control register software reset flag
Refer to the Low Voltage Reset section.

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

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to offer the enable/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_{SRESET}. After power on these bits will have a value of 01010B.

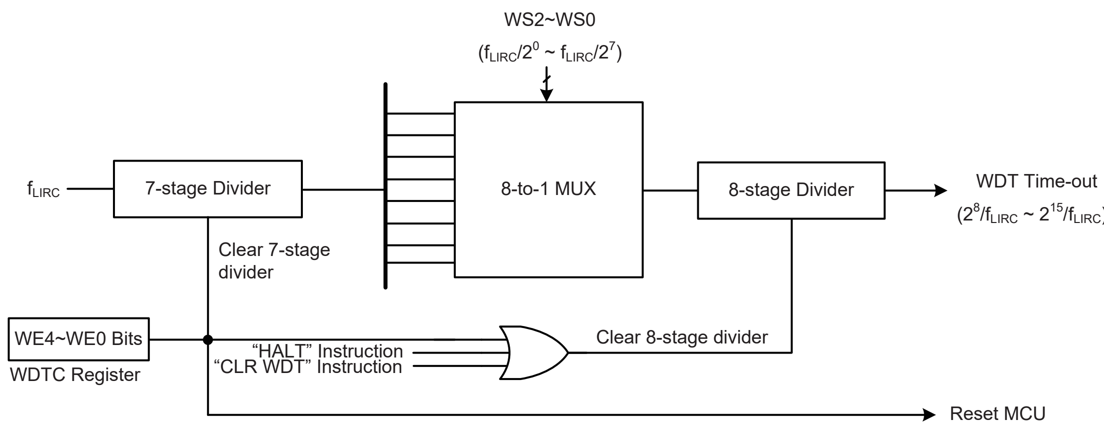
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	MCU reset

Watchdog Timer Function Control

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

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

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



Watchdog Timer

Reset and Initialisation

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

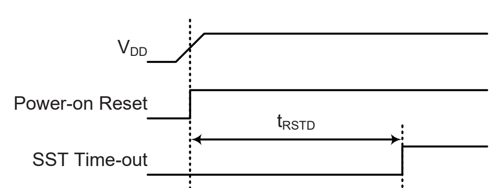
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

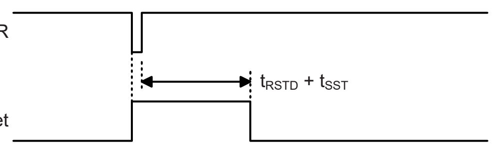
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontrollers. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions.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



Power-on Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level. This function can be enabled or disabled by the LVRC control register. If the LVRC control register is configured to enable the LVR, the LVR function will be always enabled, except in the SLEEP/IDLE mode, with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing 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 high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVR 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. If the LVS7~LVS0 bits are set to 01011010B, the LVR function is enabled with a fixed LVR voltage of 2.1V. If the LVS7~LVS0 bits are set to 10100101B, the LVR function is disabled. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01011010B. Note that the LVR function will be automatically disabled when the device enters the IDLE or SLEEP mode.



Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select control
01011010: 2.1V
10100101: LVR disable
Any other value: MCU reset – register is reset to POR value

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

Any register value, other than 01011010B and 10100101B, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time, t_{SRESET}. However in this situation the register contents will be reset to the POR value.

• **RSTFC Register**

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

"x": Unknown

Bit 7~3 Unimplemented, read as "0"

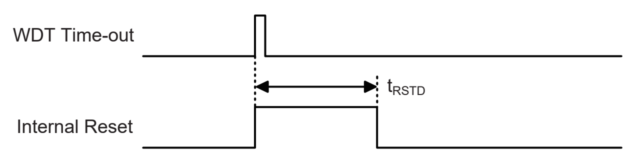
Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set high when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

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

Bit 0 **WRF**: WDT control register software reset flag
Refer to the Watchdog Timer Control Register section.

Watchdog Time-out Reset during Normal Operation

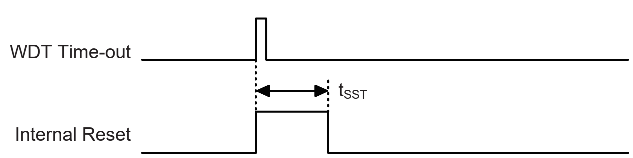
When the Watchdog time-out Reset during normal operations in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

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

"u": Unchanged

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

Item	Condition After Reset
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Bases	Cleared after reset, WDT begins counting
Timer Modules	All Timer Modules will be turned off
Input/Output Ports	I/O ports will be set 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/SLEEP)
IAR0	xxxx xxxx	uuuu uuuu	uuuu uuuu
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu
TBHP	---- --xx	---- --uu	---- --uu
STATUS	--00 xxxx	--1u uuuu	--11 uuuu
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- --01	---- --01	---- --uu
WDTC	0101 0111	0101 0111	uuuu uuuu
LVRC	0101 1010	0101 1010	uuuu uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu
INTEG	---- --00	---- --00	---- --uu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
VBGC	---- 0---	---- 0---	---- u---
PA	111- -111	111- -111	uuu- -uuu
PAC	111- -111	111- -111	uuu- -uuu
PAPU	000- -000	000- -000	uuu- -uuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PAWU	000- -000	000- -000	uuu- -uuu
PAPD	000- -000	000- -000	uuu- -uuu
PASR	000- --0-	000- --0-	uuu- --u-
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
PSCR	---- -000	---- -000	---- -uuu
ECR	0000 0000	0000 0000	uuuu uuuu
EAR	---0 0000	---0 0000	---u uuuu
EDL	0000 0000	0000 0000	uuuu uuuu
EDH	--00 0000	--00 0000	--uu uuuu
STMC0	0000 0000	0000 0000	uuuu uuuu
STMC1	0000 0000	0000 0000	uuuu uuuu
STMDL	0000 0000	0000 0000	uuuu uuuu
STMDH	---- --00	---- --00	---- --uu
STMAL	0000 0000	0000 0000	uuuu uuuu
STMAH	---- --00	---- --00	---- --uu
SADOL	xx-- ----	xx-- ----	uu-- ----
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADC0	0000 ---0	0000 ---0	uuuu ---u
SADC1	-000 0000	-000 0000	-uuu uuuu
OPAC0	0000 -000	0000 -000	uuuu -uuu
OPAC1	---0 0000	---0 0000	---u uuuu
ACFC0	0-0- -000	0-0- -000	u-u- -uuu
ACFC1	-000 --00	-000 --00	-uuu --uu
LULV	00-- ----	00-- ----	uu-- ----
HULV	0000 0000	0000 0000	uuuu uuuu
LLLV	00-- ----	00-- ----	uu-- ----
HLLV	0000 0000	0000 0000	uuuu uuuu
OPAC2	0-00 0-00	0-00 0-00	u-uu u-uu
OPA0VOS	0010 0000	0010 0000	uuuu uuuu
OPA1VOS	0010 0000	0010 0000	uuuu uuuu
LDOC	0000 0000	0000 0000	uuuu uuuu

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	—	—	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	—	—	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	—	—	PAPU2	PAPU1	PAPU0
PAPD	PAPD7	PAPD6	PAPD5	—	—	PAPD2	PAPD1	PAPD0
PAWU	PAWU7	PAWU6	PAWU5	—	—	PAWU2	PAWU1	PAWU0

“—”: Unimplemented, read as “0”

I/O Logic Function Register List

Pull-high and Pull-low Resistors

Many product applications require pull-high resistors or pull-low resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as a digital input have the capability of being connected to an internal pull-high resistor or pull-low resistor. These pull-high resistors are selected using register PAPU and are implemented using weak PMOS transistors. The pull-low resistors in this device are selected using register PAPD and are implemented using weak NMOS 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. For the pull-low resistors, they can be controlled by the relevant pull-low control register only when the pin-shared functional pin is selected as a digital input, otherwise the pull-low resisters cannot be enabled. It should be noted that the pull-high and pull-low resistors on one pin cannot be enabled at the same time.

- **PAPU Register**

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

PAPUn: PAn pin pull-high function control
0: Disable
1: Enable

• **PAPD Register**

Bit	7	6	5	4	3	2	1	0
Name	PAPD7	PAPD6	PAPD5	—	—	PAPD2	PAPD1	PAPD0
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

PAPDn: PAn pin pull-low function control
0: Disable
1: Enable
It should be noted that the pull-high and pull-low resistors on one pin cannot be enabled at the same time.

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	—	—	PAWU2	PAWU1	PAWU0
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

PAWUn: PAn pin wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

Each I/O port has its own control register known as PAC, 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 set as a CMOS output. If the pin is currently set 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	—	—	PAC2	PAC1	PAC0
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	1	1	1	—	—	1	1	1

PACn: PAn pin type selection
0: Output
1: Input

Pin-shared Functions

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

Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The device includes a Port A Output Function Selection register, labeled as PASR, 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 digital input pins, such as STCK, STPI and INT, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bits. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be set as an input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

- **PASR Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS7	PAS6	PAS5	—	—	—	PAS1	—
R/W	R/W	R/W	R/W	—	—	—	R/W	—
POR	0	0	0	—	—	—	0	—

Bit 7 **PAS7**: PA7 pin-shared function selection
0: PA7
1: AN1

Bit 6 **PAS6**: PA6 pin-shared function selection
0: PA6
1: AN0

Bit 5 **PAS5**: PA5 pin-shared function selection
0: PA5
1: STP

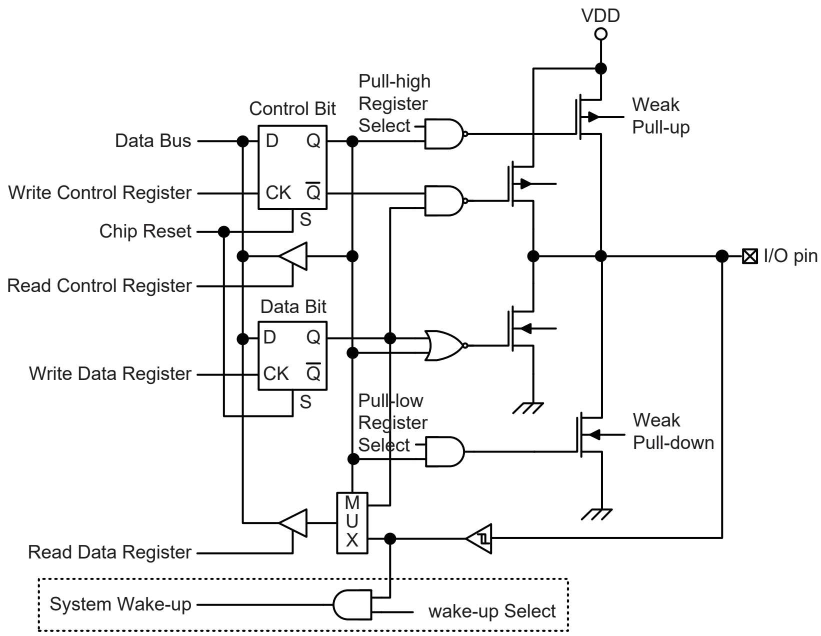
Bit 4~2 Unimplemented, read as “0”

Bit 1 **PAS1**: PA1 pin-shared function selection
0: PA1/STPI
1: A1O

Bit 0 Unimplemented, read as “0”

I/O Pin Structure

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.



Logic Function Input/Output Structure

Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to set 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 function. 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 set 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 a Timer Module, generally abbreviated to the name TM. The TM is a multi-purpose timing unit and serves 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. The TM has two interrupts. The addition of input and output pins for the TM ensures that users are provided with timing units with a wide and flexible range of features.

Introduction

The device contains a Timer Module, namely the Standard Type TM. The main features of TM is summarised in the accompanying table.

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

TM Function Summary

TM Operation

The TM offers 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 STCK2~STCK0 bits in the STM control register. 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 STCK pin. The STCK pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

TM Interrupts

The Standard 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

The TM has two TM input pins, with the label STCK and STPI. One of the STM input pin, STCK, is essentially a clock source for the STM and is selected using the STCK2~STCK0 bits in the STMC0 register. This external TM input pin allows an external clock source to drive the internal TM. The 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.

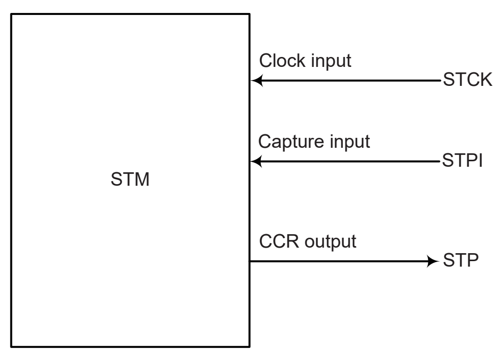
Another STM input pin, STPI, which is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the STIO1~STIO0 bits in the STMC1 register.

The TM has one output pin, STP. When the TM is in the Compare Match Output Mode, the pin 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 STP output pin is also the pin where the TM generates the PWM output waveform.

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

STM	
Input	Output
STCK, STPI	STP

TM External Pins

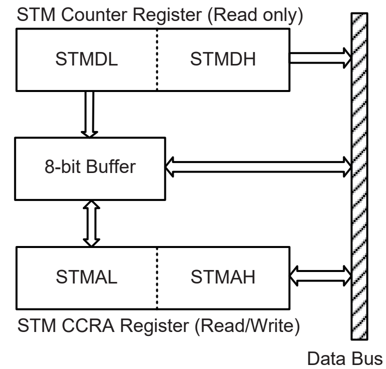


STM Function Pin Block Diagram

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA 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 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 low byte register, named STMAL, using the following access procedures. Accessing the CCRA low byte register without following these access procedures will result in unpredictable values.

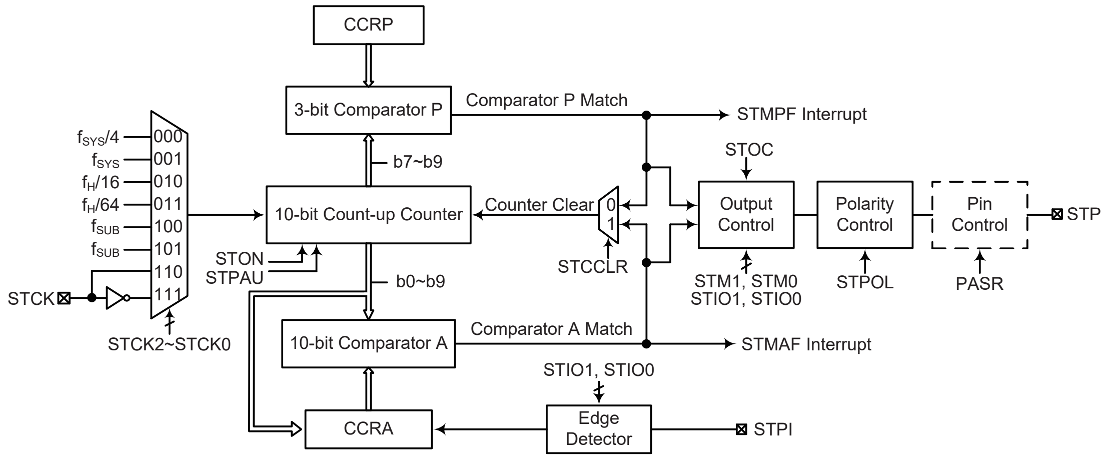


The following steps show the read and write procedures:

- Writing Data to CCRA
  - Step 1. Write data to Low Byte STMAL
    – Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte STMAH
    – 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
  - Step 1. Read data from the High Byte STMDH, STMAH
    – 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 STMDL, STMAL
    – This step reads data from the 8-bit buffer.

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 also be controlled with two external input pins and can drive one external output pin.



10-bit Standard Type TM Block Diagram

Standard Type TM Operation

The size of Standard type TM is 10-bit wide and its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP comparator is 3-bit wide whose value is compared with the highest 3 bits in the counter while the CCRA is the 10-bit and therefore compares 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 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, an 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 one output pin. 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 10-bit value, while a read/write register pair exists to store the internal 10-bit CCRA value. The remaining two registers are control registers which setup the different operating and control modes and the 3-bit CCRP value.

Register Name	Bit							
	7	6	5	4	3	2	1	0
STMC0	STPAU	STCK2	STCK1	STCK0	STON	STRP2	STRP1	STRP0
STMC1	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
STMDL	D7	D6	D5	D4	D3	D2	D1	D0
STMDH	—	—	—	—	—	—	D9	D8
STMAL	D7	D6	D5	D4	D3	D2	D1	D0
STMAH	—	—	—	—	—	—	D9	D8

10-bit Standard Type TM Register List

• STMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	STPAU	STCK2	STCK1	STCK0	STON	STRP2	STRP1	STRP0
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 **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 and 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 **STRP2~STRP0**: STM CCRP 3-bit register, compared with the STM counter bit 9~bit 7

Comparator P Match Period
000: 1024 STM clocks
001: 128 STM clocks
010: 256 STM clocks
011: 384 STM clocks
100: 512 STM clocks
101: 640 STM clocks
110: 768 STM clocks
111: 896 STM clocks

These three bits are used to setup the value on the internal CCRP 3-bit register, which are then compared with the internal counter's highest three bits. The result of this comparison can be selected to clear the internal counter if the STCCLR bit is cleared to zero. Clearing 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 three counter bits, the compare values exist in 128 clock cycle multiples. Clearing all three bits to zero is in effect allowing the counter to overflow at its maximum value.

• 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 STPI
01: Input capture at falling edge of STPI
10: Input capture at rising/falling edge of STPI
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-invert
1: Invert

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 period/duty control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

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

Bit 0 **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 Mode, Single Pulse Output Mode 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 10-bit counter bit 7 ~ bit 0

• **STMDH 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**: STM counter high byte register bit 1 ~ bit 0
STM 10-bit counter bit 9 ~ 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 10-bit CCRA bit 7 ~ bit 0

• **STMAH 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**: STM CCRA high byte register bit 1 ~ bit 0
STM 10-bit counter bit 9 ~ bit 8

Standard Type TM Operation Modes

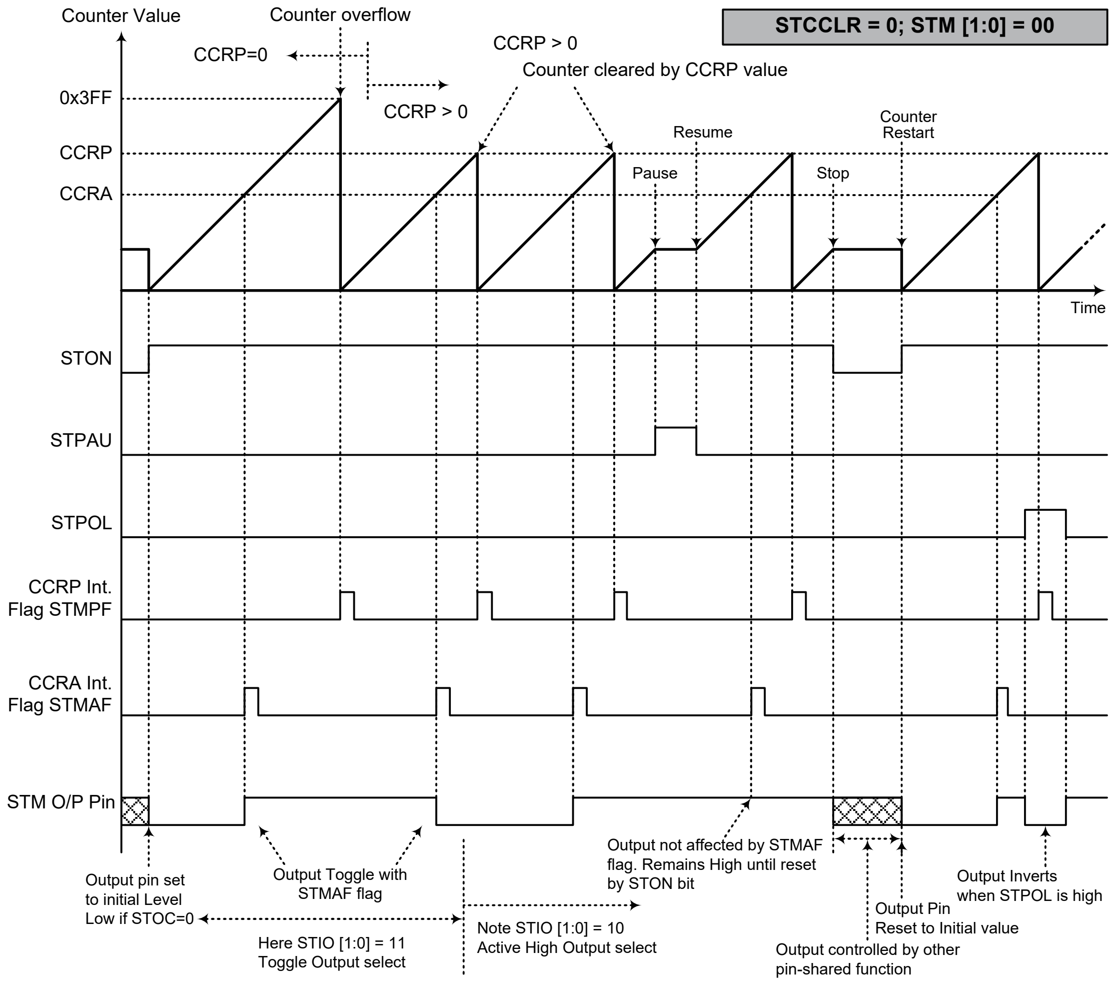
The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

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.

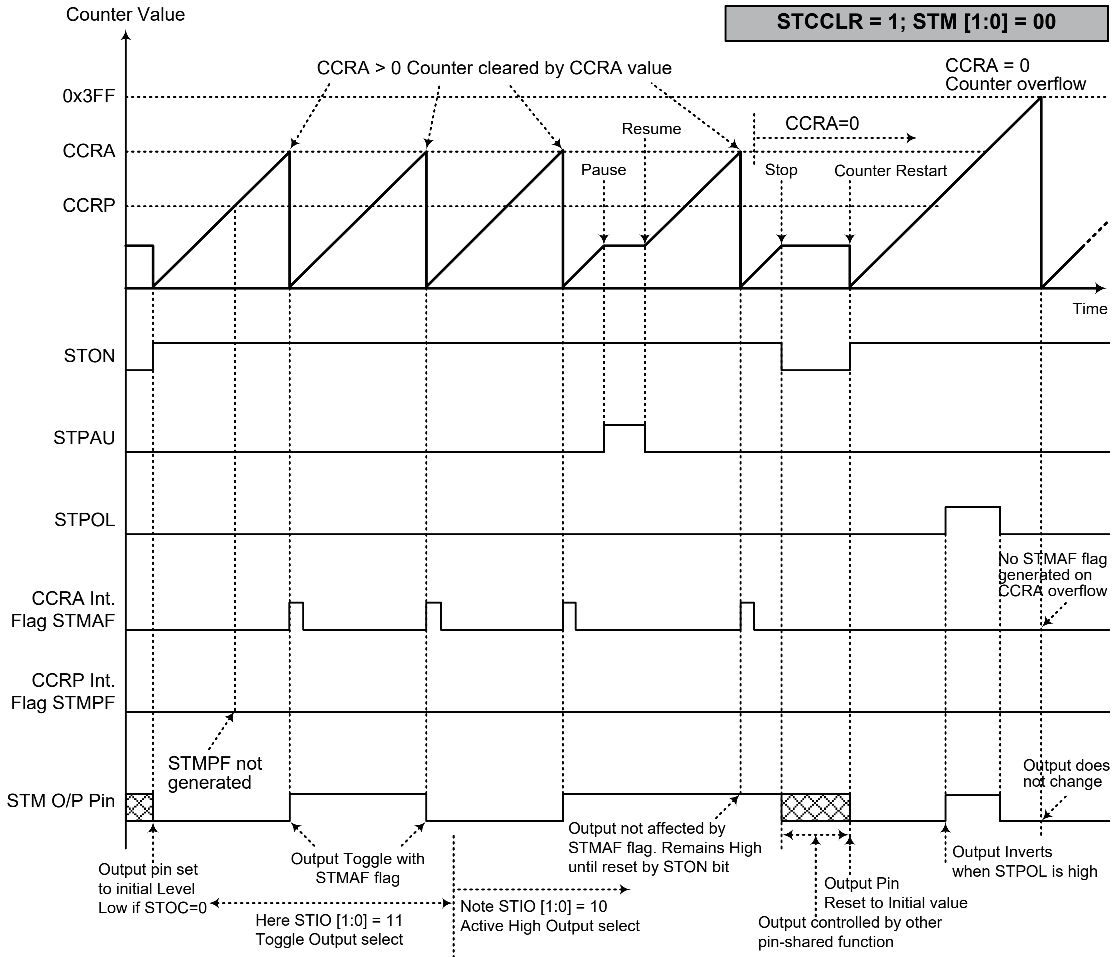
If the STCCLR bit in the STMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when STCCLR is high no STMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be cleared to “0”. If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the STMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the STM output pin, will change state. The STM output pin condition however only changes state when an STMAF interrupt request flag is generated after a compare match occurs from Comparator A. The STMPF 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.



Compare Match Output Mode – STCCLR=0

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 its initial state by an 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 an STON bit rising edge
4. An 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 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 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.

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

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

If f_{SYS}=8MHz, TM clock source is f_{SYS}/4, CCRP=2 and CCRA=128,

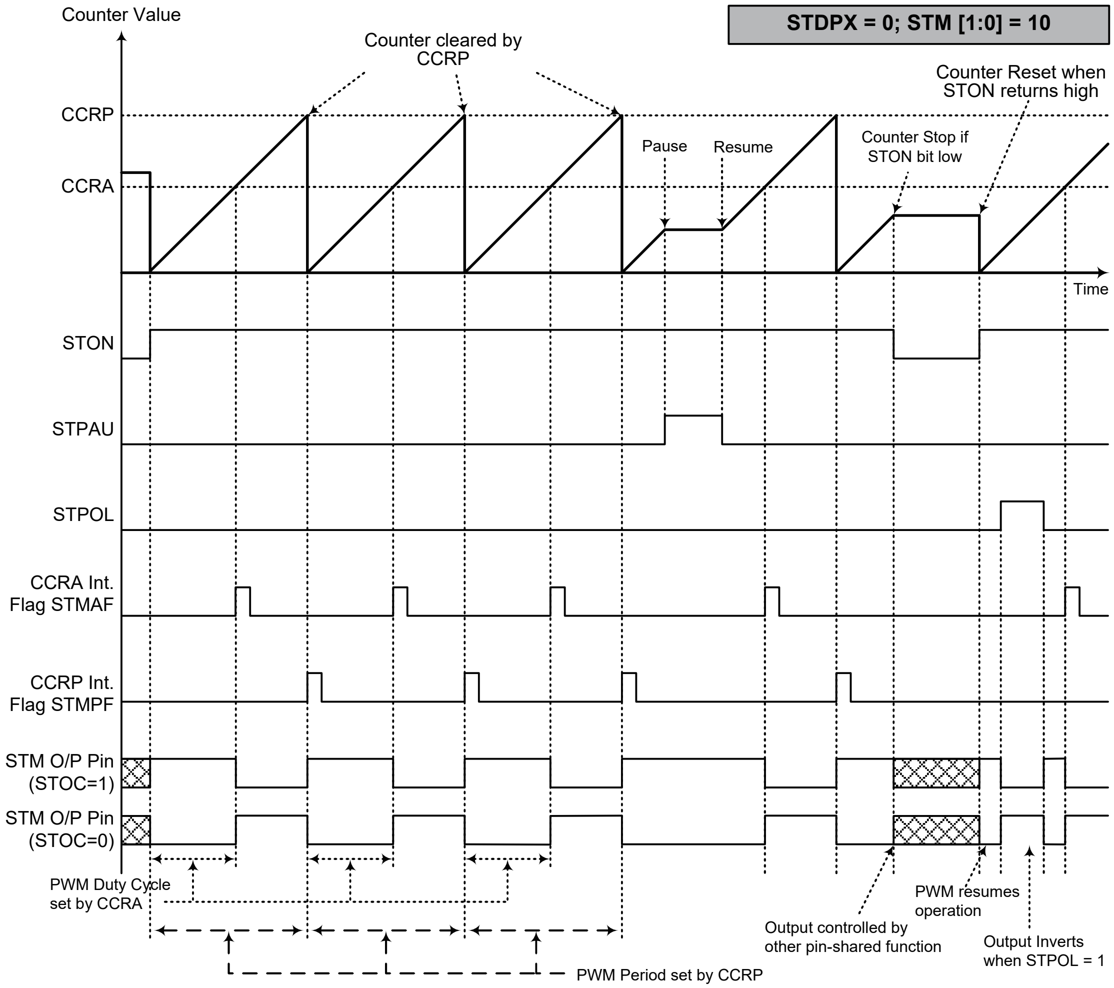
The STM PWM output frequency=(f_{SYS}/4)/(2×128)=f_{SYS}/1024=7.8125 kHz, duty=128/(2×128)=50%.

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

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

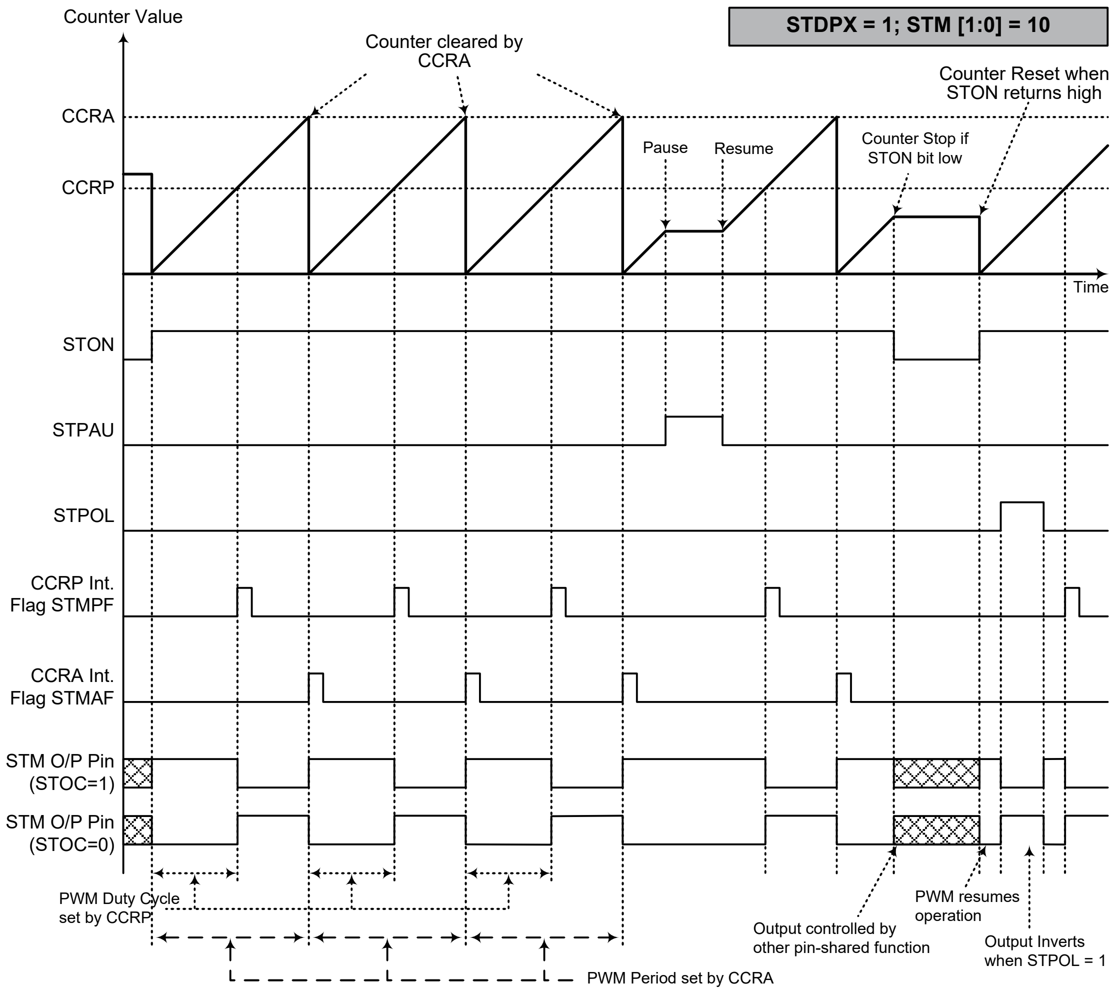
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

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



PWM Output Mode – STDPX=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



PWM Output Mode – STDPX=1

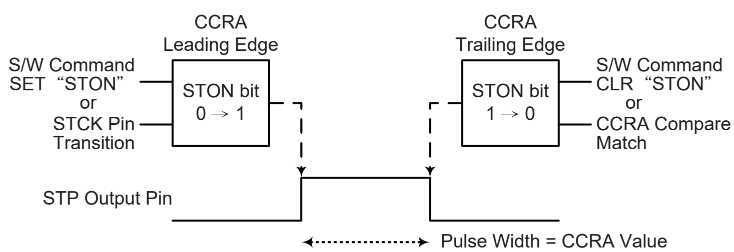
Note: 1. Here STDPX=1 – Counter cleared by CCRA
2. A counter clear sets the PWM Period
3. The internal PWM function continues even when STIO [1:0]=00 or 01
4. The STCCLR bit has no influence on PWM operation

Single Pulse Output Mode

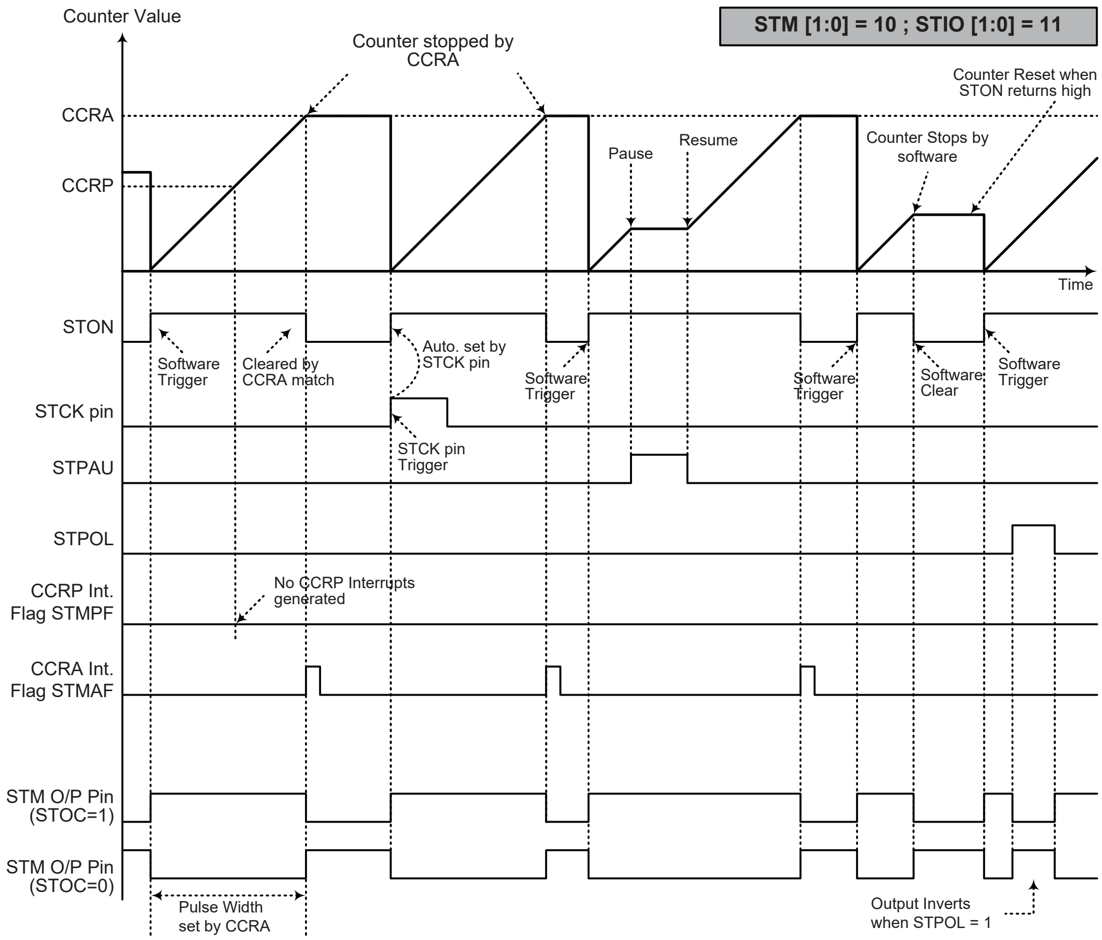
To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

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 Output Mode, the STON bit can also be made to automatically change from low to high using the 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 an 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 Output Mode CCRP is not used. The STCCLR and STDPX bits are not used in this Mode.



Single Pulse Generation



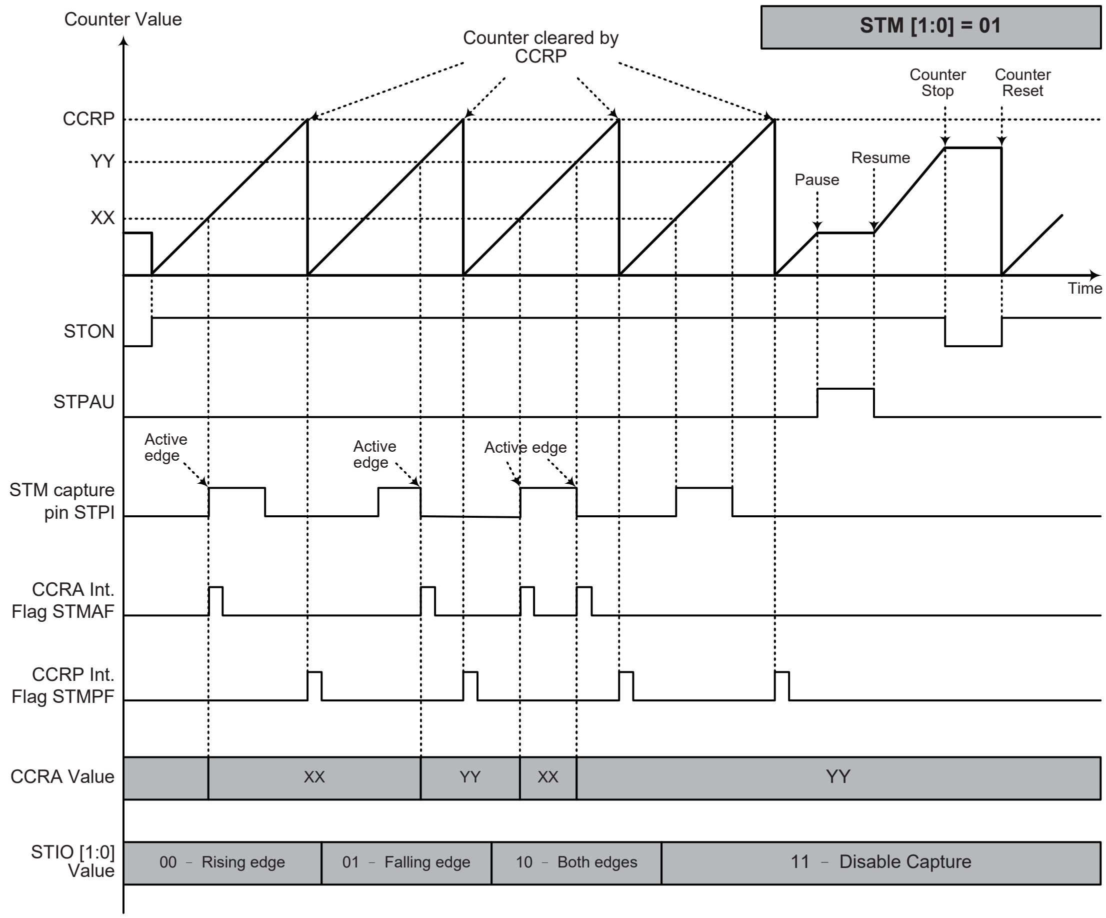
Single Pulse Output Mode

Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse triggered by the STCK pin or by setting the STON bit high
4. An STCK pin active edge will automatically set the STON bit high
5. In the Single Pulse Output Mode, STIO [1:0] must be set to “11” and can not 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 external signal is supplied on the STPI pin, whose 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 the present value in the counter will be latched into the CCRA registers and an STM interrupt generated. Irrespective of what events occur on the STPI pin 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, an 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 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, however it must be noted that the counter will continue to run. 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. An 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 – ADC

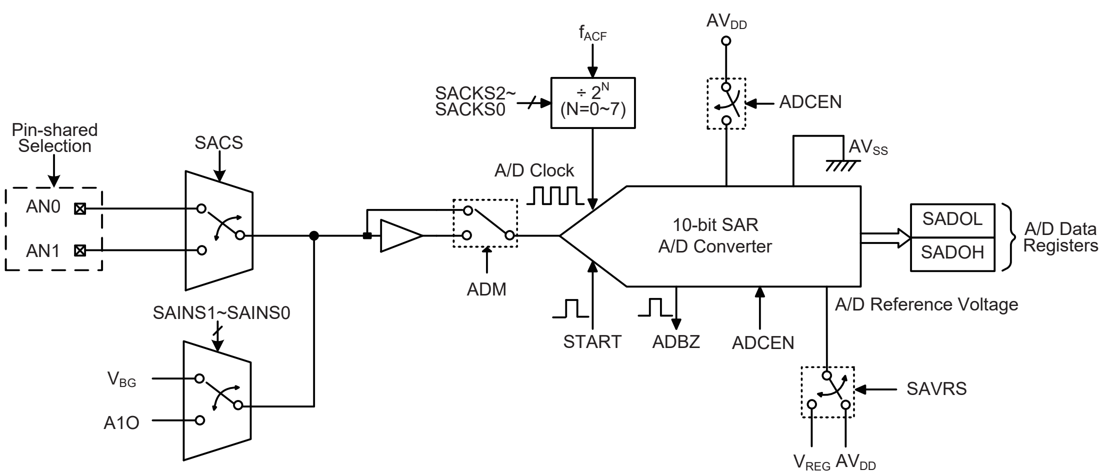
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

A/D Overview

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 10-bit digital value.

Input Channels	A/D Channel Select Bits	External Input Pins	Internal Input Signals
2+2	SAINS1~SAINS0, SACS	AN0~AN1	V_{BG}, A1O

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



Note: f_{ACF} is sourced from HIRC.

A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using five registers. A read only register pair exists to store the ADC data 10-bit value. The remaining two registers are control registers which setup the operating and control function of the A/D converter. The VBGC register contains the VBGEN bit to control the bandgap reference voltage, which can be used as the A/D converter internal input signal.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL	D1	D0	—	—	—	—	—	—
SADOH	D9	D8	D7	D6	D5	D4	D3	D2
SADC0	START	ADBZ	ADCEN	ADM	—	—	—	SACS
SADC1	—	SAINS1	SAINS0	SAVRS	OPA1V	SACKS2	SACKS1	SACKS0
VBGC	—	—	—	—	VBGEN	—	—	—

A/D Converter Register List

A/D Converter Data Registers – SADOL, SADOH

As the device contains an internal 10-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. D0~D9 are the A/D conversion result data bits. Any unused bits will be read as zero. Note that the A/D converter data register contents will be cleared to zero if the A/D converter is disabled.

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

A/D Data Registers

• SADOL Register

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

Bit 7~6 Lower byte of ADC conversion data

Bit 5~0 Unimplemented, read as “0”

• SADOH Register

Bit	7	6	5	4	3	2	1	0
Name	D9	D8	D7	D6	D5	D4	D3	D2
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 Higher byte of ADC conversion data

A/D Converter Control Registers – SADC0, SADC1

To control the function and operation of the A/D converter, two control registers, known as SADC0 and SADC1, are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SAINS1~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 SACS bit in the SADC0 register is used to determine which external channel input is selected to be converted.

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

• **SADC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	START	ADBZ	ADCEN	ADM	—	—	—	SACS
R/W	R/W	R	R/W	R/W	—	—	—	R/W
POR	0	0	0	0	—	—	—	0

Bit 7 **START**: Start the A/D conversion
0→1→0: Start A/D conversion
This bit is used to initiate an A/D conversion process.

Bit 6 **ADBZ**: A/D conversion 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 1 to enable the A/D converter. If the bit is cleared to zero, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOH and SADOL will be cleared to zero.

Bit 4 **ADM**: A/D converter mode selection
0: Normal mode (analog input bypass pre-buffer, direct to ADC)
1: High drive mode (analog input through pre-buffer to ADC)
When ADM=1 and ADCEN=1, the analog signal will be through buffer to A/D convertor.

Bit 3~1 Unimplemented, read as “0”

Bit 0 **SACS**: A/D converter external analog channel input selection
0: AN0
1: AN1

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	SAINS1	SAINS0	SAVRS	OPA1V	SACKS2	SACKS1	SACKS0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as “0”

Bit 6~5 **SAINS1~SAINS0**: A/D converter input signal selection
00: External source – External analog channel input, ANn
01: Internal source – Internal signal derived from V_{BG}
10: Internal source – Internal signal derived from OPA1 output, A1O
11: External source – External analog channel input, ANn
Care must be taken if the SAINS1~SAINS0 bits are set to “01B”, “10B” 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 converter input channel by properly setting the pin-shared function control bits. 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 **SAVRS**: A/D converter reference voltage selection
0: AV_{DD}
1: V_{REG}

Bit 3 **OPA1V**: OPA1 power voltage selection
0: V_{REG}
1: AV_{DD}

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

- **VBGC 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**: V_{BG} bandgap reference control
0: Disable
1: Enable

Bit 2~0 Unimplemented, read as “0”

A/D Converter Reference Voltage

The reference voltage supply to the A/D converter can be supplied from the positive power supply, AV_{DD} or the LDO output voltage V_{REG}. The desired selection is made using the SAVRS bit in the SADC1 register

A/D Converter Input Signals

All of the A/D analog input pins are pin-shared with the I/O pins. The corresponding pin-shared function selection bit in the PASR register, determines whether the input pin is setup as A/D converter analog input or whether it has other functions. If the corresponding pin is setup to be an A/D converter analog channel input, the original pin function 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 control bits enable an A/D input, the status of the port control register will be overridden.

As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SAINS1~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the external channel input or internal analog signal. The SACS bit in the SADC0 register is used to determine which external channel input is selected to be converted. If the SAINS1~SAINS0 bits are set to “00” or “11”, the external channel input will be selected to be converted and the SACS bit can determine which external channel is selected.

When the SAINS field is set to the value except for "00" and "11", the internal analog signal will be selected. If the internal analog signal is selected to be converted, the external input pin must never be selected as the A/D converter input channel by properly setting the pin-shared function control bits. This can prevent the external channel input from being connected together with the internal analog signal.

SAINS[1:0]	SACS	Input Signals	Description
00, 11	0~1	AN0~AN1	External channel analog input ANn
01	x	V_{BG}	Internal signal derived from V_{BG}
10	x	A1O	Internal signal derived from OPA1 output, A1O

A/D Converter Input Signal Selection

A/D 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 associated 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 f_{ACF} clock, can be chosen to be either f_{ACF} or a subdivided version of f_{ACF}. 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 f_{ACF} clock and by bits SACKS2~SACKS0, there are some limitations on the A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken for system clock frequencies. For example, as the f_{ACF} 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, special care must be taken to values marked with an asterisk *, as these values may be less or greater than the specified A/D clock period.

f_{ACF}	A/D Clock Period (t_{ADCK})							
	SACKS[2:0] =000 (f_{ACF})	SACKS[2:0] =001 (f_{ACF}/2)	SACKS[2:0] =010 (f_{ACF}/4)	SACKS[2:0] =011 (f_{ACF}/8)	SACKS[2:0] =100 (f_{ACF}/16)	SACKS[2:0] =101 (f_{ACF}/32)	SACKS[2:0] =110 (f_{ACF}/64)	SACKS[2:0] =111 (f_{ACF}/128)
8MHz	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *

A/D Clock Period Examples

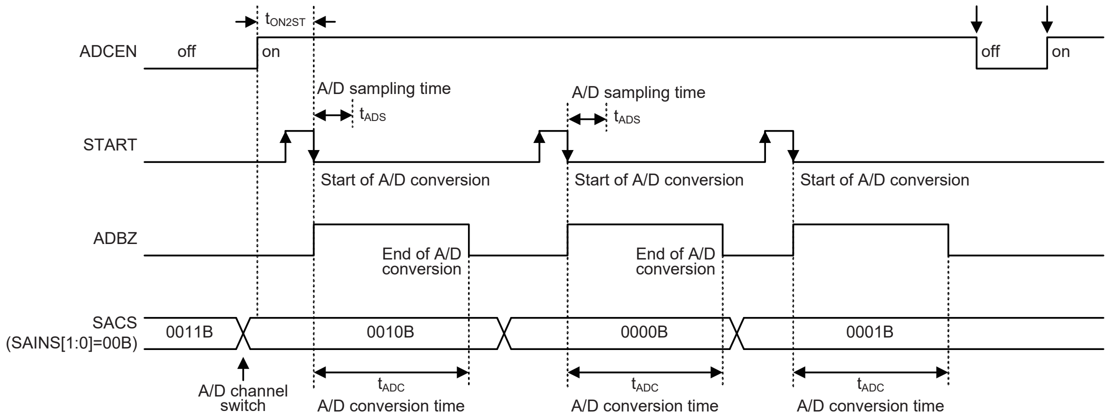
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the 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 10 A/D clock cycles. Therefore a total of 14 A/D clock cycles for an A/D conversion which is defined as t_{ADC} are necessary.

Maximum single A/D conversion rate=A/D clock period/14

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



A/D Conversion Timing – External Channel Input

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by 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
  Set the SAINS[1:0] and SACS bit to select A/D converter input channel.
- Step 4
  Select reference voltage comes from AV_{DD} or V_{REG} by configuring the SAVRS bit.
- Step 5
  If the A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt control bit, ADE, must both be set high in advance.
- Step 6
  The A/D conversion procedure can now be initiated by setting the START bit from low to high and then low again.

• Step 7
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 that the A/D conversion data register contents will be cleared to zero if the A/D converter is disabled.

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/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Transfer Function

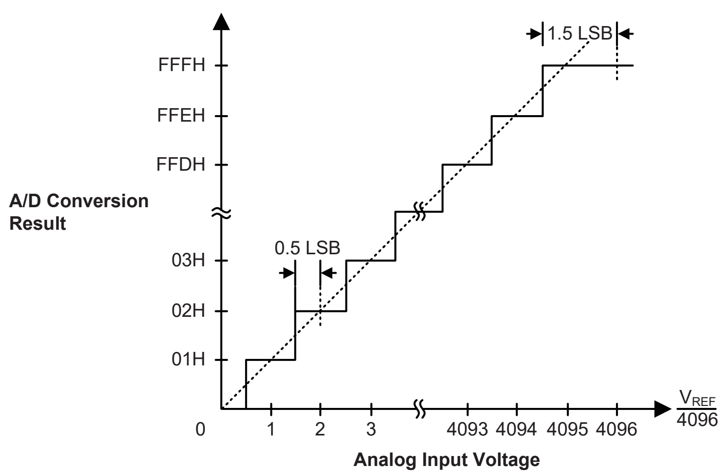
As the device contains a 10-bit A/D converter, its full-scale converted digitised value is equal to 3FFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage, V_{REF}, this gives a single bit analog input value of reference voltage divided by 1024.

$$1\ LSB=V_{REF}/1024$$

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

$$A/D\ input\ voltage=A/D\ output\ digital\ value\times V_{REF}/1024$$

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 bit.



Ideal A/D Transfer Function

A/D Programming Examples

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

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

```
clr  ADE              ; disable ADC interrupt
mov  a,03h            ; select fACF/8 as A/D clock and
mov  SADC1,a          ; select external channel input and reference input AVDD
mov  a,40h            ; set PASR to configure pin AN0
mov  PASR,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  SADOL_buffer,a   ; save result to user defined register
mov  a,SADOH          ; read high byte conversion result value
mov  SADOH_buffer,a   ; save result to user defined register
:
:
jmp  start_conversion  ; start next A/D conversion
```

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

```
clr  ADE              ; disable ADC interrupt
mov  a,03h            ; select fACF/8 as A/D clock and
mov  SADC1,a          ; select external channel input and reference input AVDD
mov  a,40h            ; set PASR to configure pin AN0
mov  PASR,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 ADC interrupt
set  EMI              ; enable global interrupt
:
:
ADC_ISR:              ; ADC interrupt service routine
mov  acc_stack,a      ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a   ; save STATUS to user defined memory
:
:
mov  a, SADOL         ; read low byte conversion result value
mov  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
```

Auto Conversion Function

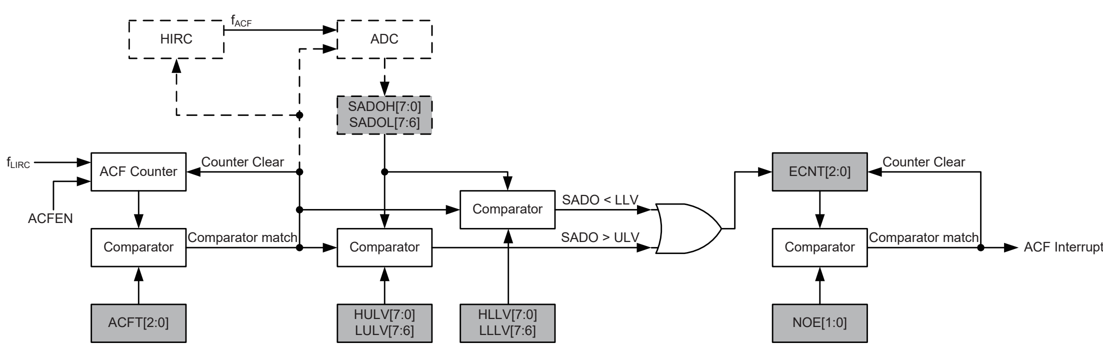
The device contains an auto conversion circuit function which is used to enable automatic A/D conversions. This function can be implemented using the ACF counter.

Auto Conversion Operation

The Auto conversion circuit allows the ACF counter to enable the A/D converter and to compare the A/D conversion value with pre-programmed upper and lower limit values while in the power down state.

The auto A/D conversion time is selected by the user controlled bits, ACFT2~ACFT0, in the ACFC0 register. When the ACF counts up to this period time, it will send a signal to the ADC circuit, to enable the ADC and automatically start a conversion. If the ADC data does not lie between the preset lower and upper limit values, the event counter value ECNT2~ECNT0 will be increased by one. When the ECNT2~ECNT0 value is equal to the number of events as set by bits NOE1~NOE0, the ACF will send an interrupt signal to MCU and the hardware will clear the ECNT2~ECNT0 bits. The CPU will then be woken up and execute the ACF interrupt subroutine.

This function is mainly used when the device is in IDLE or SLEEP mode. With proper configurations, this function can enable the PIR detection with fixed intervals to detect if there is anyone passing by, if so, it will generate an interrupt to wake up the MCU for further actions, thus implementing PIR detection function with the advantage of reducing power consumption.



ACF Block Diagram

Auto Conversion Registers

To control the operation of the Auto Conversion function, several control registers known as ACFC0, ACFC1, LULV, HULV, LLLV and HLLV are provided. The ACFC0 register is used to enable or disable the auto conversion function and set the auto conversion time. When the ACF counts up to a preset time, the hardware circuit will automatically enable the ADC function. The ACFC1 register is the event counter value register. The remaining four data registers LULV, HULV, LLLV and HLLV are used to store the lower and upper limit values.

Register Name	Bit							
	7	6	5	4	3	2	1	0
ACFC0	ACFEN	—	OP2ADB	—	—	ACFT2	ACFT1	ACFT0
ACFC1	—	ECNT2	ECNT1	ECNT0	—	—	NOE1	NOE0
LULV	D1	D0	—	—	—	—	—	—
HULV	D9	D8	D7	D6	D5	D4	D3	D2
LLLV	D1	D0	—	—	—	—	—	—
HLLV	D9	D8	D7	D6	D5	D4	D3	D2

Auto Conversion Function Register List

• ACFC0 Register

Bit	7	6	5	4	3	2	1	0
Name	ACFEN	—	OP2ADB	—	—	ACFT2	ACFT1	ACFT0
R/W	R/W	—	R/W	—	—	R/W	R/W	R/W
POR	0	—	0	—	—	0	0	0

Bit 7 **ACFEN**: Auto conversion function control bit
0: Disable
1: Enable
This bit controls the overall on/off function of the ACF, setting the bit high enables the ACF counter to run and auto conversion when a compare match condition occurs. Clearing this bit will disable the auto conversion function.

Bit 6 Unimplemented, read as "0"

Bit 5 **OP2ADB**: OP and A/D turn-on timing control
0: Enable
1: Disable

Bit 4~3 Unimplemented, read as "0"

Bit 2~0 **ACFT2~ACFT0**: Auto A/D conversion time selection
000: 4ms
001: 8ms
010: 16ms
011: 32ms
100: 64ms
101: 128ms
110: 256ms
111: 512ms
If the ACF counter value is equal to this period, the hardware circuit will enable the ADC function automatically.

• ACFC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	ECNT2	ECNT1	ECNT0	—	—	NOE1	NOE0
R/W	—	R/W	R/W	R/W	—	—	R/W	R/W
POR	—	0	0	0	—	—	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~4 **ECNT2~ ECNT0**: Event counter values
If the ADC data does not lie between the lower limitation value and upper limitation value, the event counter values ECNT2~ECNT0 will be increased by one. When ECNT2~ECNT0 is equal to the number of events values as set by bits NOE1~NOE0, the ACF circuit will send an interrupt signal to the CPU and the hardware will clear the ECNT2~ECNT0 bits to zero.

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **NOE1~NOE0**: The number of events
00: 1 time
01: 2 times
10: 4 times
11: 7 times

- **LULV Register**

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

Bit 7~6 Lower byte of upper limitation value
Bit 5~0 Unimplemented, read as "0"

- **HULV Register**

Bit	7	6	5	4	3	2	1	0
Name	D9	D8	D7	D6	D5	D4	D3	D2
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 Higher byte of upper limitation value

- **LLLV Register**

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

Bit 7~6 Lower byte of Lower limitation value
Bit 5~0 Unimplemented, read as "0"

- **HLLV Register**

Bit	7	6	5	4	3	2	1	0
Name	D9	D8	D7	D6	D5	D4	D3	D2
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 Higher byte of lower limitation value

LDO Function

The device contains a low power voltage regulator implemented in CMOS technology. Using CMOS technology ensures low voltage drop and low quiescent current. The output voltages can range from 2.2V~3.6V, which can be setup by the bits, VSEL3~VSEL0, in the LDOC register. However, the actual LDO output is controlled by a combination of several control bits, as listed below.

VREGS	VSW	LDOEN	VPSW	VREG Output
0	0	0	x	Weak pull low
0	0	1	1	V_{LDO} (Selected by VSEL3~VSEL0)
0	0	1	0	V_{BG} (1.25V)
0	1	x	x	AV_{DD}
1	x	x	x	Floating

LDO Output Status Control Table

- **LDOC Register**

Bit	7	6	5	4	3	2	1	0
Name	LDOEN	LDOM	VREGS	VSW	VSEL3	VSEL2	VSEL1	VSEL0
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 **LDOEN**: LDO control bit
0: Disable
1: Enable

Bit 6 **LDOM**: LDO power mode selection
0: Power-saving mode (2/3 bias current)
1: Normal mode

Bit 5 **VREGS**: VREG status floating control
0: Not floating
1: Floating

Bit 4 **VSW**: V_{REG} voltage source selection
0: V_{LDO}
1: AV_{DD}

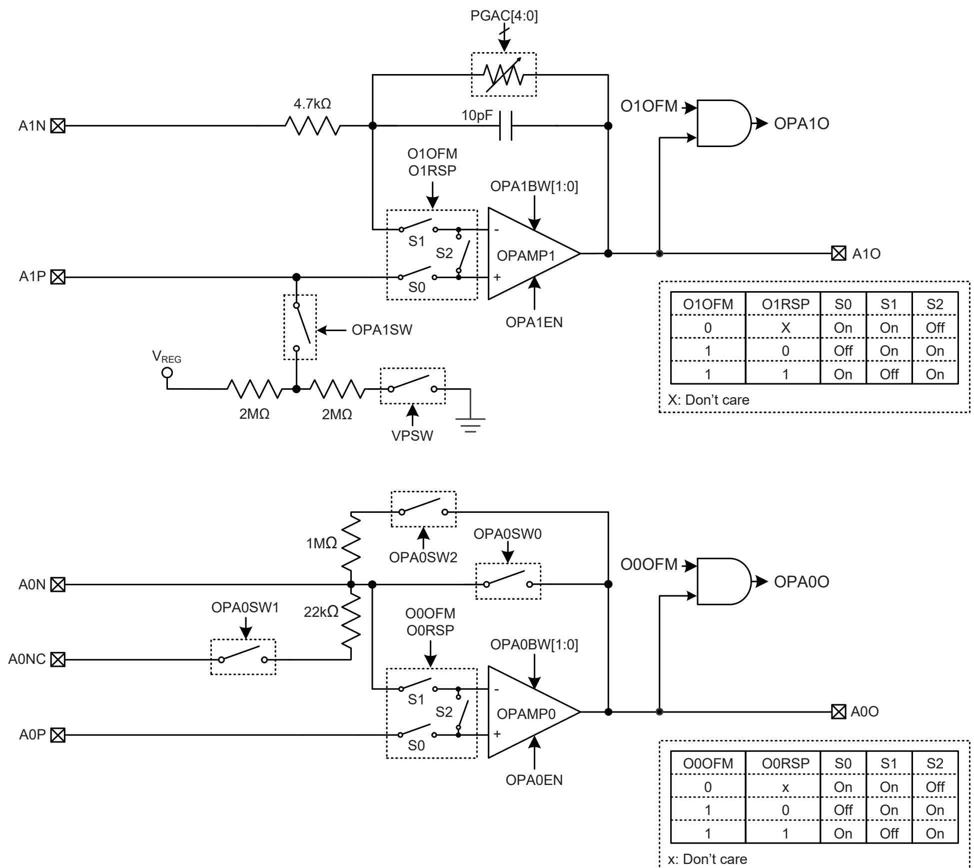
Bit 3~0 **VSEL3~VSEL0**: LDO output voltage selection
0000: 2.2V
0001: 2.3V
0010: 2.4V
0011: 2.5V
0100: 2.6V
0101: 2.7V
0110: 2.8V
0111: 2.9V
1000: 3.0V
1001: 3.1V
1010: 3.2V
1011: 3.3V
1100: 3.4V
1101: 3.5V
1110: 3.6V
1111: 3.6V
These bit only available for LDO output voltage selection when the LDO function is enabled and the VPSW bit is set. If the VPSW bit is 0, the LDO output voltage will not be selected by these four bits, and the voltage will be about 1.25V

Note: 1. When the AV_{DD} voltage is stable and after switching on the LDO, the LDO output voltage will be stable after 20μs.
2. When users change the LDO output voltage, the LDO output voltage will be stable after 12ms.
3. If the LDO output is used as the ADC reference voltage, then the VREG should connect a 0.1μF capacitor to ground.
4. The LDO input voltage (AV_{DD}) must be 0.1V greater than output voltage for obtaining stable output voltage.

Operational Amplifiers

There are two Operational Amplifiers with input offset calibration function and adjustable bandwidth in the device, OPA0 and OPA1. For PIR applications, the OPA0 is setup for a band pass filter application circuit and it is recommended that the bandwidth is setup from 0.3Hz to 8Hz. The OPA1 amplifier is a programmable gain amplifier whose gain can be setup to have a range of 128 to 376. This gain is setup using the application software.

The OPA0 provides internal resistors which can obtain a gain of 46 with the advantage of saving external components. If the gain does not meet the requirements, users can get their desired gain by connecting external resistors together with properly configuring the OPA0SW1 or OPA0SW2 bit to control the corresponding switches. Setting the OPA0SW0 bit high can be used for fast warm-up. It is suggested to connect an external capacitor between the A0O and A0N pins to filter out the DC level of signal. The A1P is the input pin of OPA1. Bits VPSW and OPA1SW are used to control a bias voltage of 1/2V_{REG}, which will be used as the DC level of A1P input signal.



Operational Amplifiers Block Diagram

Operational Amplifier Registers

The Operational Amplifiers are fully under the control of several internal registers.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OPAC0	OPA1EN	OPA0EN	VPSW	OPA1SW	—	OPA0SW2	OPA0SW1	OPA0SW0
OPAC1	—	—	—	PGAC4	PGAC3	PGAC2	PGAC1	PGAC0
OPAC2	OPA1O	—	OPA1BW1	OPA1BW0	OPA0O	—	OPA0BW1	OPA0BW0
OPA0VOS	O0OFM	O0RSP	O0OF5	O0OF4	O0OF3	O0OF2	O0OF1	O0OF0
OPA1VOS	O1OFM	O1RSP	O1OF5	O1OF4	O1OF3	O1OF2	O1OF1	O1OF0

Operational Amplifier Register List

• OPAC0 Register

Bit	7	6	5	4	3	2	1	0
Name	OPA1EN	OPA0EN	VPSW	OPA1SW	—	OPA0SW2	OPA0SW1	OPA0SW0
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 **OPA1EN**: OPA1 enable or disable control bit
0: Disable
1: Enable

Bit 6 **OPA0EN**: OPA0 enable or disable control bit
0: Disable
1: Enable

Bit 5 **VPSW**: Switch control bit
0: Off
1: On
When this bit is set, the corresponding switch will be on. In this case, the internal voltage Vp can be supplied to the OPA1 positive input and if the OPA1SW bit is also set to switch on. It should be noted that the VPSW bit status also affects the LDO output voltage, refer to the “LDO function” section for more detailed information.

Bit 4 **OPA1SW**: Switch control bit
0: Off
1: On

Bit 3 Unimplemented, read as “0”

Bit 2 **OPA0SW2**: Switch control bit
0: Off
1: On

Bit 1 **OPA0SW1**: Switch control bit
0: Off
1: On

Bit 0 **OPA0SW0**: OPA0 short switch between non-inverting input and output
0: Off
1: On

• **OPAC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	PGAC4	PGAC3	PGAC2	PGAC1	PGAC0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **PGAC4~PGAC0**: OPA1 gain control bit
Gain=128+(PGAC×8)

• **OPAC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	OPA1O	—	OPA1BW1	OPA1BW0	OPA0O	—	OPA0BW1	OPA0BW0
R/W	R	—	R/W	R/W	R	—	R/W	R/W
POR	0	—	0	0	0	—	0	0

Bit 7 **OPA1O**: OPA1 digital output bit; positive logic (read only)
The OPA1O is "0" when the OPA1 is disabled. When O1OFM is set to 1, OPA1O is defined as OPA1 output status.
Note: For the OPA1, when executing the offset calibration function, the A1N state must be floating to ensure the value accuracy. It is not recommended to use A1N for offset calibration as there are internal components on this port.

Bit 6 Unimplemented, read as "0"

Bit 5~4 **OPA1BW1~OPA1BW0**: OPA1 bandwidth control bits (refer to OP Amplifier Electrical Characteristics)
00: 5kHz
01: 40kHz
10: 600kHz
11: 2MHz

Bit 3 **OPA0O**:OPA0 digital output bit; positive logic (read only)
The OPA0O is "0" when the OPA0 is disabled. When O0OFM is set to 1, OPA0O is defined as OPA0 output status.

Bit 2 Unimplemented, read as "0"

Bit 1~0 **OPA0BW1~OPA0BW0**: OPA0 bandwidth control bits (refer to OP Amplifier Electrical Characteristics)
00: 5kHz
01: 40kHz
10: 600kHz
11: 2MHz

• **OPA0VOS Register**

Bit	7	6	5	4	3	2	1	0
Name	O0OFM	O0RSP	O0OF5	O0OF4	O0OF3	O0OF2	O0OF1	O0OF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **O0OFM**: OPA0 normal operation or input offset voltage calibration mode selection bit
0: Normal operation
1: Offset calibration mode

Bit 6 **O0RSP**: OPA0 input offset voltage calibration reference selection bit
0: Input reference voltage comes from A0N
1: Input reference voltage comes from A0P

Bit 5~0 **O0OF5 ~ O0OF0**: OPA0 input offset voltage calibration control bits

• **OPA1VOS Register**

Bit	7	6	5	4	3	2	1	0
Name	O1OFM	O1RSP	O1OF5	O1OF4	O1OF3	O1OF2	O1OF1	O1OF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **O1OFM**: OPA1 normal operation or input offset voltage calibration mode selection bit
0: Normal operation
1: Offset calibration mode

Bit 6 **O1RSP**: OPA1 input offset voltage calibration reference selection bit
0: Input reference voltage comes from A1N
1: Input reference voltage comes from A1P

Bit 5~0 **O1OF5 ~ O1OF0**: OPA1 input offset voltage calibration control bits

Offset Calibration Procedure

Note that if OPAn inputs are pin-shared with I/O, they should be configured as OPAn input first. The operational amplifier offset calibration procedures are summarized as the following.

Step1: Set OnOFM=1 and OnRSP=1, OPAn is now under offset calibration mode, S0 and S2 on. To make sure V_{OS} as minimize as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal mode operation.

Step2: Set OnOF[5:0]=000000B then read OPnO bit.

Step3: Let OnOF[5:0]=OnOF[5:0]+1 then read OPnO bit, if OPnO bit state is changed, record the data as V_{OS1}.

Step4: Set OnOF[5:0]=111111B then read OPnO bit.

Step5: Let OnOF[5:0]=OnOF[5:0]-1 then read OPnO bit, if OPnO bit state is changed, record the data as V_{OS2}.

Step6: Restore $V_{OS}=(V_{OS1}+V_{OS2})/2$ to OnOF[5:0] bits, the calibration is finished.

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

Residue $V_{OS}=V_{OUT}-V_{IN}$

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a TM Comparator P, Comparator A match, requires microcontroller attention, its corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to its needs. The device contains several external interrupt and internal interrupt functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by internal functions including the TMs, A/D converter, Auto Conversion function and Time Bases.

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 number of registers falls into two categories. The first is the INTC0~INTC1 registers which set the primary interrupts, the second is the INTEG register to set 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	—
A/D Converter	ADE	ADF	—
Auto Conversion Function	ACFE	ACFF	—
Time Bases	TBnE	TBnF	n=0~1
STM	STMPE	STMPF	—
	STMAE	STMAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	ACFF	ADF	INTF	ACFE	ADE	INTE	EMI
INTC1	STMAF	STMPF	TB1F	TB0F	STMAE	STMPE	TB1E	TB0E

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	—	ACFF	ADF	INTF	ACFE	ADE	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 **ACFF**: Auto conversion function interrupt request flag
0: No request
1: Interrupt request

Bit 5 **ADF**: A/D converter interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INTF**: INT interrupt request flag
0: No request
1: Interrupt request

Bit 3 **ACFE**: Auto conversion function interrupt control
0: Disable
1: Enable

Bit 2 **ADE**: A/D converter interrupt control
0: Disable
1: Enable

Bit 1 **INTE**: INT 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	STMAF	STMPF	TB1F	TB0F	STMAE	STMPE	TB1E	TB0E
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 **STMAF**: STM comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **STMPF**: STM comparator P match interrupt request flag
0: No request
1: Interrupt request

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

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

Bit 3 **STMAE**: STM comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **STMPE**: STM comparator P match interrupt control
0: Disable
1: Enable

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

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

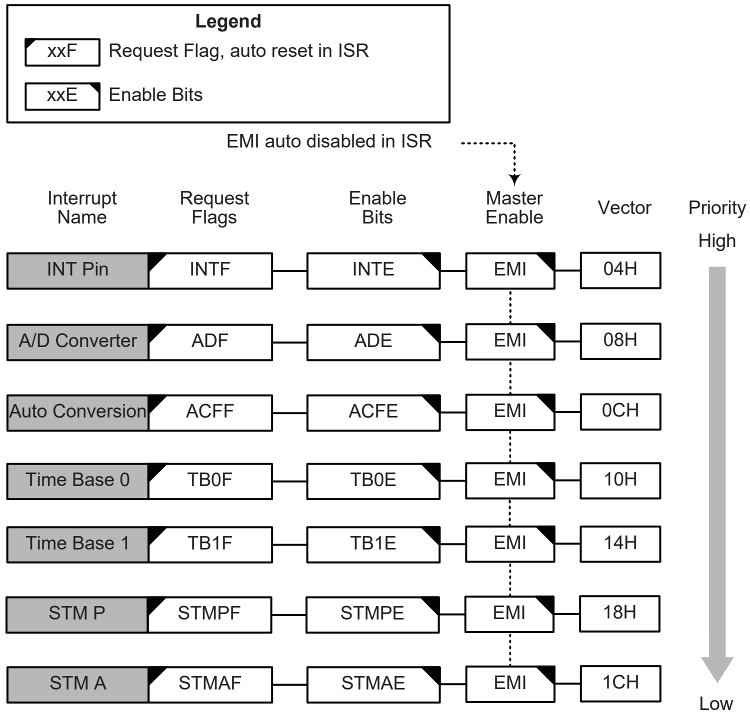
Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P, Comparator A match, 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 an “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 diagram with their order of priority. All of the interrupt sources have their own individual vector. Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.



Interrupt Structure

External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the 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 pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if its 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 bit. The pin must also be set 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 selection on the external interrupt pin 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.

A/D Converter Interrupt

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

Auto Conversion Function Interrupt

The device contains an Auto Conversion Function which has its own independent interrupt. An internal Auto Conversion Function interrupt will take place when the Auto Conversion Function interrupt request flag ACFF, is set, which occurs when the event counter value reaches the number of events pre-programmed in the NOE bit field . To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Auto Conversion Function interrupt enable bit, ACFE, must first be set. When the interrupt is enabled, the stack is not full and the condition mentioned above occurs, a subroutine call to the Auto Conversion Function interrupt vector, will take place. When the interrupt is serviced, the Auto Conversion Function interrupt flag, ACFF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Timer Module Interrupts

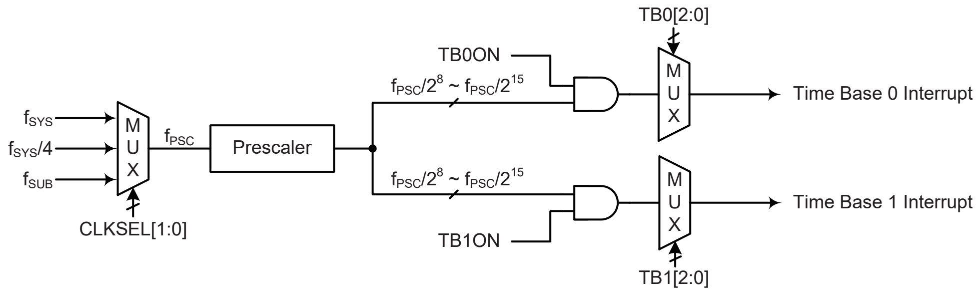
The Standard type TM has two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. For all of the TM types there are two interrupt request flags and two enable control bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or 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, 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 TM interrupt vector locations will take place. When the TM interrupt is serviced, the TM interrupt request flag will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. It is controlled by the overflow signals from the internal timers. When this happens its interrupt request flag TBnF will be set. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI and Time Base enable bit, TBnE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to its interrupt vector location will take place. When the interrupt is serviced, the interrupt request flag, TBnF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupts 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 TBnC register to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL[1:0] bits in the PSCR register.



Time Base Interrupts

• PSCR Register

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

Bit 7~3 Unimplemented, read as "0"

Bit 2 **PSCEN**: Prescaler clock enable control
0: Disable
1: Enable
This PSCEN bit is the prescaler clock enable/disable control bit. When the prescale clock is not in use, clearing this bit to zero can reduce extra power consumption.

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

• TBnC Register (n=0~1)

Bit	7	6	5	4	3	2	1	0
Name	TBnON	—	—	—	—	TBn2	TBn1	TBn0
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TBnON**: Time Base n control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TBn2~TBn0**: Time Base n time-out period selection
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

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.

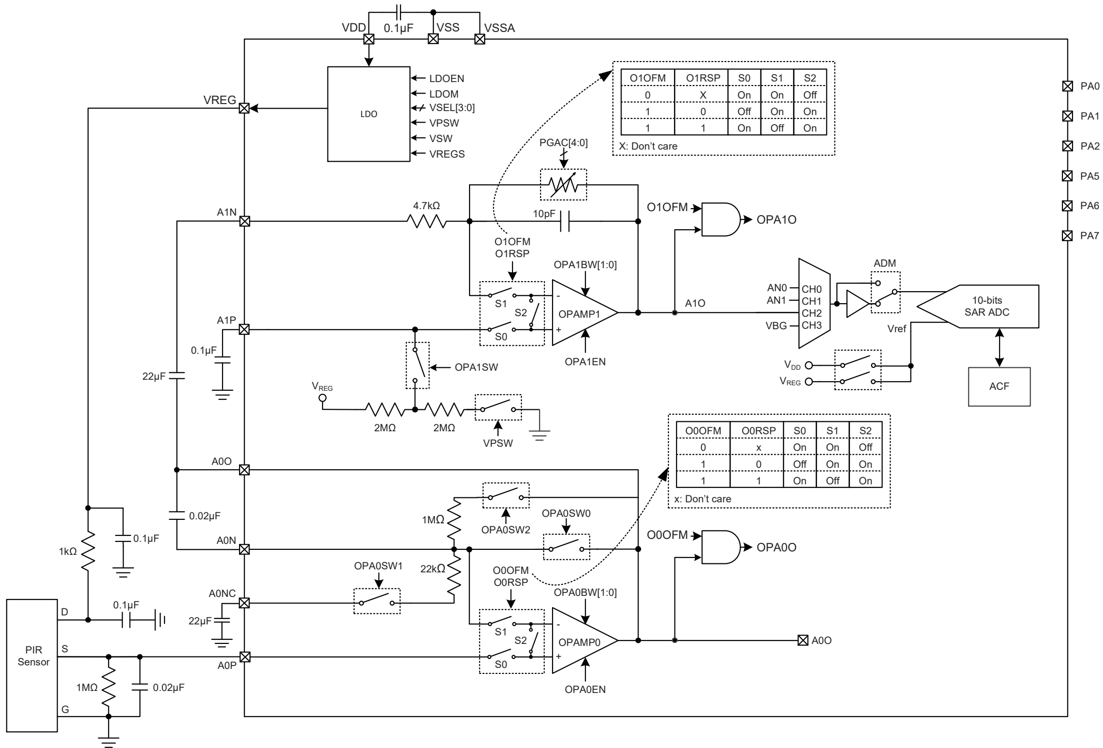
It is recommended that programs do not use the “CALL” instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged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

Every interrupt has the capability of waking up the microcontroller when it is in the 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 an RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

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

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch Operation			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read Operation			
TABRD [m]	Read table (specific page or current page) to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

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

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

3. For the "CLR WDT1" and "CLR WDT2" instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both "CLR WDT1" and "CLR WDT2" instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry
Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation ACC ← ACC + [m] + C
Affected flag(s) OV, Z, AC, C

ADCM A,[m] Add ACC to Data Memory with Carry
Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation [m] ← ACC + [m] + C
Affected flag(s) OV, Z, AC, C

ADD A,[m] Add Data Memory to ACC
Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.
Operation ACC ← ACC + [m]
Affected flag(s) OV, Z, AC, C

ADD A,x Add immediate data to ACC
Description The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.
Operation ACC ← ACC + x
Affected flag(s) OV, Z, AC, C

ADDM A,[m] Add ACC to Data Memory
Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation [m] ← ACC + [m]
Affected flag(s) OV, Z, AC, C

AND A,[m] Logical AND Data Memory to ACC
Description Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation ACC ← ACC ″AND″ [m]
Affected flag(s) Z

AND A,x Logical AND immediate data to ACC
Description Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.
Operation ACC ← ACC ″AND″ x
Affected flag(s) Z

ANDM A,[m] Logical AND ACC to Data Memory
Description Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.
Operation [m] ← ACC ″AND″ [m]
Affected flag(s) Z

CALL addr Subroutine call

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

Operation: Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s): None

CLR [m] Clear Data Memory

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

Operation: [m] ← 00H

Affected flag(s): None

CLR [m].i Clear bit of Data Memory

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

Operation: [m].i ← 0

Affected flag(s): None

CLR WDT Clear Watchdog Timer

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

Operation: WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

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

Operation: WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

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

Operation: WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CPL [m] Complement Data Memory

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

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

Affected flag(s): Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation: ACC ← $\overline{[m]}$

Affected flag(s): Z

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

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

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

Affected flag(s): C

DEC [m] Decrement Data Memory

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

Operation: [m] ← [m] − 1

Affected flag(s): Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation: ACC ← [m] − 1

Affected flag(s): Z

HALT Enter power down mode

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

Operation: TO ← 0
PDF ← 1

Affected flag(s): TO, PDF

INC [m] Increment Data Memory

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

Operation: [m] ← [m] + 1

Affected flag(s): Z

INCA [m] Increment Data Memory with result in ACC

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

Operation: ACC ← [m] + 1

Affected flag(s): Z

JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None

MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None

MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None

MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None

NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None

OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ x
Affected flag(s)	Z

ORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″OR″ [m]
Affected flag(s)	Z

RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None

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

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

Operation: Program Counter ← Stack
ACC ← x

Affected flag(s): None

RETI Return from interrupt

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

Operation: Program Counter ← Stack
EMI ← 1

Affected flag(s): None

RL [m] Rotate Data Memory left

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

RR [m] Rotate Data Memory right

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

Operation: $ACC \leftarrow ACC - [m] - \overline{C}$

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

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

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

Operation: $[m] \leftarrow ACC - [m] - \overline{C}$

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

SDZ [m] Skip if decrement Data Memory is 0

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation: [m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation: ACC ← [m] − 1
Skip if ACC=0

Affected flag(s): None

SET [m] Set Data Memory

Description: Each bit of the specified Data Memory is set to 1.

Operation: [m] ← FFH

Affected flag(s): None

SET [m].i Set bit of Data Memory

Description: Bit i of the specified Data Memory is set to 1.

Operation: [m].i ← 1

Affected flag(s): None

SIZ [m] Skip if increment Data Memory is 0

Description: The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation: [m] ← [m] + 1
Skip if [m]=0

Affected flag(s): None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description: The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation: ACC ← [m] + 1
Skip if ACC=0

Affected flag(s): None

SNZ [m].i Skip if bit i of Data Memory is not 0

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

Operation: Skip if [m].i ≠ 0

Affected flag(s): None

SUB A,[m] Subtract Data Memory from ACC

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

Operation: ACC ← ACC − [m]

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

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

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

Operation: [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation: ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation: [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s): None

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

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

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

Affected flag(s): None

SZ [m] Skip if Data Memory is 0

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

Operation: Skip if [m]=0

Affected flag(s): None

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

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

Operation: ACC ← [m]
Skip if [m]=0

Affected flag(s): None

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

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

Operation: Skip if [m].i=0

Affected flag(s): None

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

Description: The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s): None

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

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

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

Affected flag(s): None

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

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

Operation: ACC ← ACC ″XOR″ [m]

Affected flag(s): Z

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

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

Operation: [m] ← ACC ″XOR″ [m]

Affected flag(s): Z

XOR A,x Logical XOR immediate data to ACC

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

Operation: ACC ← ACC ″XOR″ x

Affected flag(s): Z

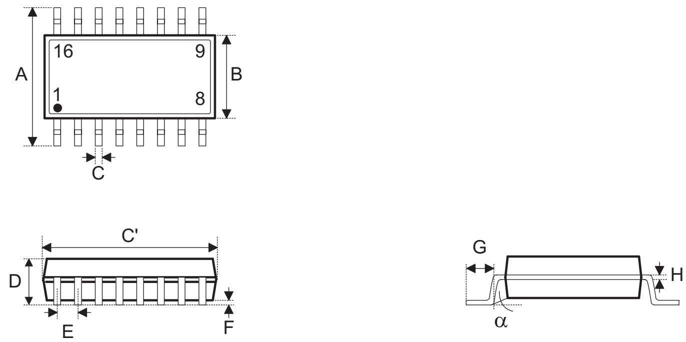
Package Information

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

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

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

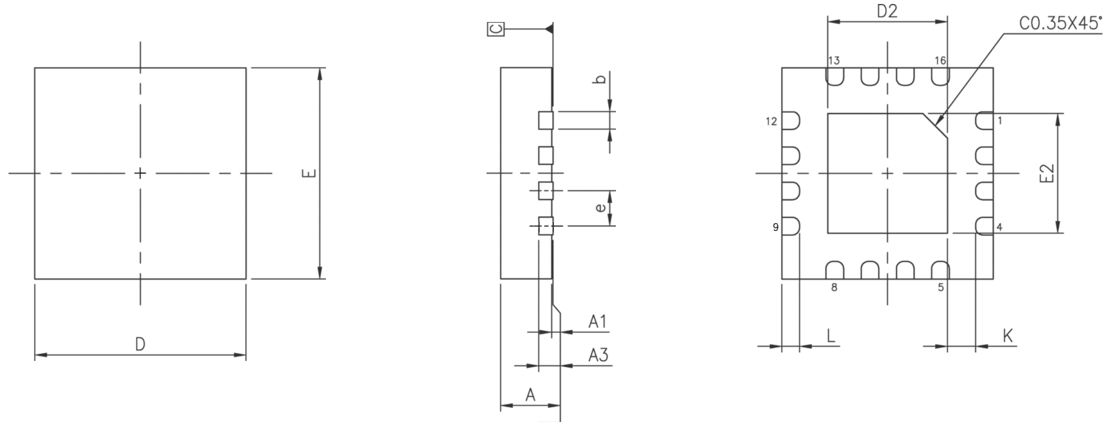
16-pin NSOP (150mil) Outline Dimensions



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

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.00 BSC	—
B	—	3.90 BSC	—
C	0.31	—	0.51
C’	—	9.90 BSC	—
D	—	—	1.75
E	—	1.27 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

SAW Type 16-pin QFN (3mm×3mm, FP0.25mm) 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	—	0.118 BSC	—
E	—	0.118 BSC	—
e	—	0.020 BSC	—
D2	0.063	0.067	0.069
E2	0.063	0.067	0.069
L	0.008	0.010	0.012
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	—	0.20 BSC	—
b	0.18	0.25	0.30
D	—	3.00 BSC	—
E	—	3.00 BSC	—
e	—	0.50 BSC	—
D2	1.60	1.70	1.75
E2	1.60	1.70	1.75
L	0.20	0.25	0.30
K	0.20	—	—