Charger Flash MCU

HT45F5Q

Revision: V1.10 Date: December 13, 2016

www.holtek.com

Table of Contents

Features **6**
CPU Features 6
Peripheral Features 6
General Description **7**
Block Diagram **7**
Pin Assignment **8**
Pin Description **8**
Absolute Maximum Ratings **10**
D.C. Characteristics **10**
A.C. Characteristics **11**
ADC Electrical Characteristics **12**
LVD&LVR Electrical Characteristics **12**
DAC Electrical Characteristics **13**
OP Amplifier Electrical Characteristics **13**
OVP Electrical Characteristics **14**
OCP Electrical Characteristics **14**
Power Good Characteristics **15**
Power on Reset Electrical Characteristics **15**
System Architecture **16**
Clocking and Pipelining 16
Program Counter 17
Stack 17
Arithmetic and Logic Unit – ALU 18
Flash Program Memory **19**
Structure 19
Special Vectors 19
Look-up Table 19
Table Program Example 20
In Circuit Programming 21
On-Chip Debug Support – OCDS 22
RAM Data Memory **23**
Structure 23
General Purpose Data Memory 23
Special Purpose Data Memory 23

Special Function Register Description 25
Indirect Addressing Registers – IAR0, IAR1 25
Memory Pointers – MP0, MP1 25
Bank Pointer – BP 26
Accumulator – ACC 26
Program Counter Low Register – PCL 26
Look-up Table Registers – TBLP, TBLH 26
Status Register – STATUS 27

EEPROM Data Memory 29
EEPROM Data Memory Structure 29
EEPROM Registers 29
Reading Data from the EEPROM 31
Writing Data to the EEPROM 31
Write Protection 31
EEPROM Interrupt 31
Programming Considerations 32

Oscillator 33
Oscillator Overview 33
System Clock Configurations 33
Internal RC Oscillator – HIRC 34
Internal 32kHz Oscillator – LIRC 34
Supplementary Oscillator 34

Operating Modes and System Clocks 35
System Clocks 35
System Operation Modes 36
Control Register 37
Operating Mode Switching 39
Standby Current Considerations 43
Wake-up 43

Watchdog Timer 44
Watchdog Timer Clock Source 44
Watchdog Timer Control Register 44
Watchdog Timer Operation 45

Reset and Initialisation 46
Reset Functions 46
Reset Initial Conditions 49

Input/Output Ports 52
Pull-high Resistors 52
Port A Wake-up 53
I/O Port Control Registers 53
Pin-shared Functions 54
I/O Pin Structures 55
Programming Considerations 56

Timer Module – TM 57
Introduction 57
TM Operation 57
TM Clock Source 57
TM Interrupts 57
TM External Pins 58
TM Input/Output Pin Control Register 58
Programming Considerations 59
Standard Type TM – STM 60
Standard TM Operation 60
Standard Type TM Register Description 60
Standard Type TM Operating Modes 64
Analog to Digital Converter 73
A/D Overview 73
A/D Converter Register Description 74
A/D Converter Data Registers – SADOL, SADOH 74
A/D Converter Control Registers – SADC0, SADC1, PASR 74
A/D Operation 76
A/D Converter Input Signal 77
Conversion Rate and Timing Diagram 77
Summary of A/D Conversion Steps 78
Programming Considerations 79
A/D Transfer Function 79
A/D Programming Examples 80
Battery Charge Module 82
Battery Charging Constant Current and Constant Voltage Modes 82
OCP and OVP Functions 83
Battery Charge Module Registers 83
Digital to Analog Converter 84
Operational Amplifier 0 85
Interrupts 87
Interrupt Registers 87
Interrupt Operation 90
External Interrupt 91
Multi-function Interrupt 92
A/D Converter Interrupt 92
Time Base Interrupts 92
EEPROM Interrupt 93
TM Interrupts 94
OCVP Interrupt 94
Interrupt Wake-up Function 94
Programming Considerations 95

Low Voltage Detector – LVD 96
LVD Register 96
LVD Operation 97
Application Circuit 98
Instruction Set 99
Introduction 99
Instruction Timing 99
Moving and Transferring Data 99
Arithmetic Operations 99
Logical and Rotate Operation 100
Branches and Control Transfer 100
Bit Operations 100
Table Read Operations 100
Other Operations 100
Instruction Set Summary 101
Table Conventions 101
Instruction Definition 103
Package Information 112
16-pin NSOP (150mil) Outline Dimensions 113

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
- Two Oscillators:
  - Internal RC -- HIRC
  - Internal 32kHz -- LIRC
- Fully intergrated internal 8MHz oscillator requires no external components
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- Up to 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K×14
- RAM Data Memory: 64×8
- True EEPROM Memory: 32×8
- Watchdog Timer function
- Up to 8 bidirectional I/O lines
- Single pin-shared external interrupts
- One Timer Modules for time measure, compare match output, capture input, PWM output, single pulse output functions (10-bit STM×1)
- Dual Time-Base functions for generation of fixed time interrupt signals
- Multi-channel 12-bit resolution A/D converter
- Battery charge circuit
  - Build-in OCP and OVP circuits (H/W protection)
  - OPA×2 for voltage and current sense
  - 8-bit DAC
- Low voltage reset function
- Low voltage detect function
- Package type: 16-pin NSOP

General Description

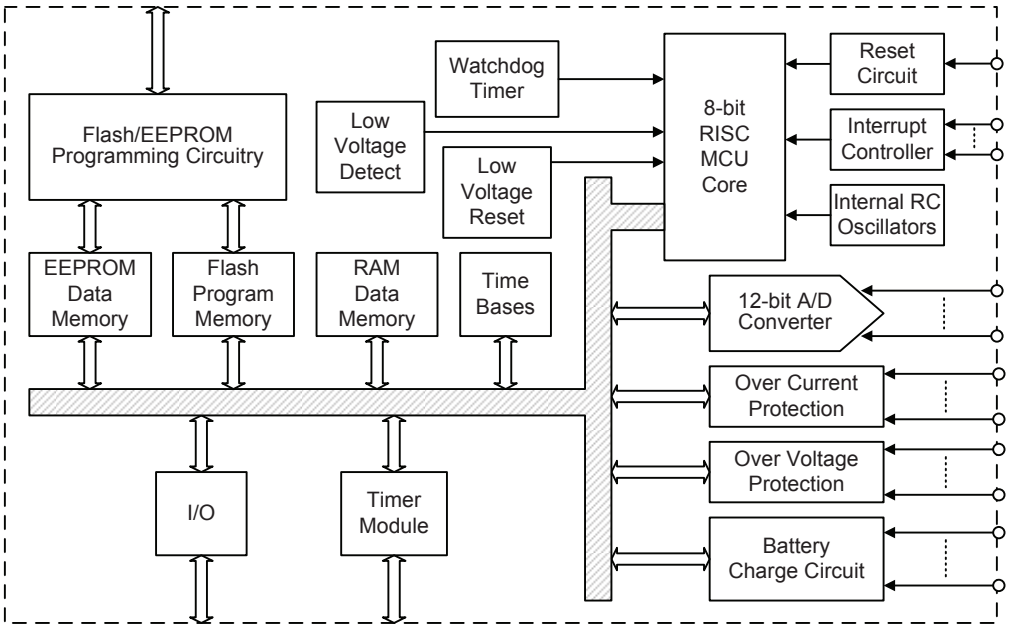
The HT45F5Q is an ASSP MCU specifically designed for Battery Charger applications. Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

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

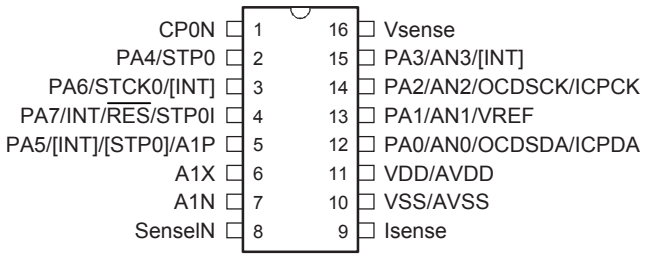
A full choice of HIRC and LIRC oscillator functions are provided including a fully integrated system oscillator which requires no external components for its implementation.

The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the device will find excellent use in applications such as electronic metering, environmental monitoring, handheld instruments, household appliances, electronically controlled tools, motor driving in addition to many others.

Block Diagram



Pin Assignment



HT45F5Q/HT45V5Q
16 NSOP-A

Note: 1. Bracketed pin names indicate non-default pinout remapping locations.
2. AVDD&VDD means the VDD and AVDD are the double bonding. VSS&AVSS means the VSS and AVSS are the double bonding.
3. The OCDSDA and OCDSCK pins are the OCDS dedicated pins

Pin Description

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

Pin Name	Function	OP	I/T	O/T	Description
PA0/AN0/ OCDSDA/ ICPDA	PA0	PAWU PAPU PASR	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN0	SADC0 PASR	AN	—	ADC input channel
	OCDSDA	—	ST	CMOS	OCDS address/data line, for EV chip only.
	ICPDA	—	ST	CMOS	ICP address/data line
PA1/AN1/ VREF	PA1	PAWU PAPU PASR	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN1	SADC0 PASR	AN	—	ADC input channel
	VREF	PASR	AN	—	ADC reference voltage input
PA2/AN2/ OCDSCK/ ICPCK	PA2	PAWU PAPU PASR	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN2	SADC0 PASR	AN	—	ADC input channel
	OCDSCK	—	ST	—	OCDS clock line, for EV chip only.
	ICPCK	—	ST	—	ICP clock line
PA3/AN3/[INT]	PA3	PAWU PAPU PASR	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN3	SADC0 PASR	AN	—	ADC input channel
	INT	PASR IFS0	ST	—	External interrupt

Pin Name	Function	OP	I/T	O/T	Description
PA4/STP0	PA4	PAWU PAPU PASR	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	STP0	PASR	ST	CMOS	STM Output
PA5/[INT]/ [STP0]/A1P	PA5	PAWU PAPU PASR	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	PASR IFS0	ST	—	External interrupt
	STP0	PASR	ST	CMOS	STM Output
	A1P	—	ST	—	OPA1 positive external input pin
PA6/ STCK0/[INT]	PA6	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	STCK0	—	ST	—	STM clock input
	INT	IFS0	ST	—	External interrupt
PA7/INT/ STP0I/$\overline{\text{RES}}$	PA7	PAWU PAPU RSTC	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	IFS0 RSTC	ST	—	External interrupt
	STP0I	RSTC	ST	CMOS	STM Input
	$\overline{\text{RES}}$	RSTC	ST	—	External reset input
A1X	A1X	—	—	AN	OPA1 output
A1N	A1N	—	AN	—	OPA1 negative input
SenseIN	SenseIN	—	AN	—	OPA1 signal input
Isense	Isense	—	AN	—	Current sense input
Vsense	Vsense	—	AN	—	Voltage sense input
CP0N	CP0N	—	AN	—	OCP negative input
VDD	VDD	—	PWR	—	Digital positive power supply
AVDD	AVDD	—	PWR	—	Analog positive power supply
VSS	VSS	—	PWR	—	Digital negative power supply
AVSS	AVSS	—	PWR	—	Analog negative power supply

Lenged: I/T: Input type
O/T: Output type
OP: Optional by register option
PWR: Power
ST: Schmitt Trigger input
CMOS: CMOS output
AN: Analog pin
*: VDD is the device power supply while AVDD is the ADC power supply. The AVDD pin is bonded together internally with VDD.
**: VSS is the device ground pin while AVSS is the ADC ground pin. The AVSS pin is bonded together internally with VSS.

Absolute Maximum Ratings

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

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (HIRC)	—	f_{SYS}=8MHz	2.2	—	5.5	V
I_{DD}	Operating Current (HIRC)	3V	No load, all peripherals off, WDT enable, LVR enable, OPA enable, OCP/OVP enable f_{SYS} = f_{HIRC} = 8MHz	—	0.8	2.0	mA
		5V		—	1.5	3.0	mA
	Operating Current (LIRC)	3V	No load, all peripherals off, OPA enable ,OCP/OVP enable f_{SYS} = f_{LIRC} = 32kHz	—	0.35	1.2	mA
		5V		—	0.52	1.3	mA
I_{IDLE}	IDLE0 Mode Standby Current (LIRC)	5V	No load, ADC off, WDT enable, LVR disable, OPA enable, OCP/OVP enable	—	0.47	1.0	mA
	IDLE1 Mode Standby Current (HIRC)	5V	No load, ADC off, WDT enable, f_{SYS}=8MHz on, OPA enable, OCP/OVP enable	—	0.89	3.0	mA
I_{SLEEP}	SLEEP0 Mode Standby Current (LIRC off)	5V	No load, ADC off, WDT disable, LVR disable, OPA enable, OCP/OVP enable	—	0.46	1.0	mA
	SLEEP1 Mode Standby Current (LIRC on)	5V	No load, ADC off, WDT enable, LVR disable, OPA enable, OCP/OVP enable	—	0.47	1.0	mA
V_{IL}	Input Low Voltage for I/O Ports except $\overline{RES}$ pin	5V	—	0	—	1.5	V
		—		0	—	0.2V_{DD}	V
	Input Low Voltage ($\overline{RES}$)	—	—	0	—	0.4V_{DD}	V
V_{IH}	Input High Voltage for I/O Ports except $\overline{RES}$ pin	5V	—	3.5	—	5	V
		—		0.8V_{DD}	—	V_{DD}	V
	Input High Voltage ($\overline{RES}$)	—	—	0.9V_{DD}	—	V_{DD}	V
I_{OL}	I/O Sink Current	3V	V_{OL}=0.1V_{DD}	16	30	—	mA
		5V		40	68	—	mA
I_{OH}	I/O Source Current	3V	V_{OH}=0.9V_{DD}	-4	-8	—	mA
		5V		-8	-16	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (HIRC)	2.2~5.5V	$f_{SYS} = f_{HIRC}$ = 8MHz	—	8	—	MHz
	System Clock (LIRC)		$f_{SYS} = f_{LIRC}$ = 32kHz	—	32	—	kHz
f_{HIRC}	System Clock (HIRC)	3V/5V	Ta = 25°C	-2%	8	+2%	MHz
		3V/5V	Ta = 0°C to 70°C	-5%	8	+5%	MHz
		2.2V~5.5V	Ta = 0°C to 70°C	-8%	8	+8%	MHz
		2.2V~5.5V	Ta = -40°C to 85°C	-12%	8	+12%	MHz
f_{LIRC}	System Clock (LIRC)	2.2V~5.5V	Ta = -40°C to 85°C	4	32	80	kHz
t_{TC}	STM Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{RES}	External Reset Minimum Low Pulse Width	—	—	10	—	—	μs
t_{INT}	External Interrupt Minimum Pulse Width	—	—	0.3	—	—	μs
t_{EERD}	EEPROM Read Time	—	—	—	2	5	t_{SYS}
t_{EEWR}	EEPROM Write Time	—	—	—	2	5	ms
t_{SST}	System Start-up Timer Period (Wake-up from Halt, f_{SYS} off at Halt, Reset Pin Reset)	—	$f_{SYS} = f_H \sim f_H / 64$, $f_H = f_{HIRC}$	10	16	22	t_{HIRC}
		—	$f_{SYS} = f_{SUB} = f_{LIRC}$	2	—	—	t_{LIRC}
	System start-up Timer Period (Wake-up from Halt, f_{SYS} on at Halt State)	—	$f_{SYS} = f_{LIRC}$	1	2	5	t_{SYS}
t_{RSTD}	System Reset Delay Time (POR Reset, LVR hardware Reset, LVR software Reset, WDT Software Reset, Reset Control Register Software Reset)	—	—	10	50	100	ms
	System Reset Delay Time (Reset Pin Reset, WDT Time-out Hardware Cold Reset)	—	—	10	16.7	50	ms

Note: 1. $t_{SYS} = 1/f_{SYS}$

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

ADC Electrical Characteristics

Ta=25°C

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

LVD&LVR Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	V_{LVR}	—	5.5	V
V_{LVR}	Low Voltage Reset Voltage	—	LVR Enable	-5%	2.1	+5%	V
V_{LVD}	Low Voltage Detection Voltage	—	ENLVD=1, V_{LVD}=2.0V	-5%	2.0	+5%	V
		—	ENLVD=1, V_{LVD}=2.2V	-5%	2.2	+5%	V
		—	ENLVD=1, V_{LVD}=2.4V	-5%	2.4	+5%	V
		—	ENLVD=1, V_{LVD}=2.7V	-5%	2.7	+5%	V
		—	ENLVD=1, V_{LVD}=3.0V	-5%	3.0	+5%	V
		—	ENLVD=1, V_{LVD}=3.3V	-5%	3.3	+5%	V
		—	ENLVD=1, V_{LVD}=3.6V	-5%	3.6	+5%	V
		—	ENLVD=1, V_{LVD}=4.0V	-5%	4.0	+5%	V
V_{BG}	Bandgap Reference Voltage	—	—	-5%	1.09	+5%	V
I_{OP}	Operating Current	5V	LVD enable, LVR enable, VBGEN =0	—	20	25	μA
		5V	LVD enable, LVR enable, VBGEN =1	—	200	300	μA
t_{BGS}	V_{BG} Turn On Stable Time	—	—	—	—	150	μs
t_{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN = 0, LVD off → on	—	—	15	μs
		—	For LVR disable, VBGEN = 0, LVD off à→ on	—	—	150	μs
t_{LVR}	Low Voltage Width to Reset	—	—	28	240	640	μs

DAC Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.3	—	5.5	V
V_{DACO}	Output voltage Range	—	—	V_{SS}	—	V_{DD}	V
I_{DAC}	A/D Converter Reference Voltage	5V	—	—	110	500	μA

Note: DAC Voltage Formula: $\left(\frac{DAC8[7:0]}{2^8}\right)\times V_{DD}$

OP Amplifier Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
D.C. Characteristic							
V_{DD}	Operating Voltage	—	—	2.3	—	5.5	V
V_{OS}	Input Offset Voltage	5V	Without calibration (AnOF[4:0] = 10000B)	-15	—	+15	mV
			With calibration	-4	—	+4	mV
I_{SOURCE}	Output Current	5V	INP=1V,INN=0V,V_{OUT}=4.5V	3	5	—	mV
I_{SINK}		5V	INP=0V,INN=1V,V_{OUT}=0.5V	5	7	—	mV
V_{CM}	Input Common Mode Range	5V	—	V_{SS}	—	V_{DD}-1.4	V
A.C. Characteristics							
A_{OL}	Open Loop Gain	5V	—	60	80	—	dB
SR	Slew Rate	5V	No Load	—	0.3	—	V/μs
GBW	OP Gain Bandwidth	5V	V_{CM}=V_{DD}-1.4 R_L=1MΩ,C_L=100pF	1	2	—	MHz

OVP Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DDC}	OVP Operating Voltage	—	—	4.6	—	5.5	V
I_{DDC}	OVP Operating Current	5V	—	—	—	23.5	μA
V_{CMPOS}	Comparator Input Offset Voltage	5V	—	-15	—	15	mV
V_{HYS}	Hysteresis Width	—	—	20	40	60	mV
V_{CM}	Input Common Mode Range	—	—	V_{SS}	—	V_{DD}-1.4	V
OVPD	Vsense Over Voltage Detection	5V	—	-3%	3V	+3%	V

Note: The OVP module is integraged with a comparator and the internal 3V voltage is provided to the comparator negative. The OVPD verification purpose is that, when Vsense is more than 3V, the comparator outputs high level, otherwise low level.

OCP Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DDC}	OCP Operating Voltage	—	—	4.6	—	5.5	V
I_{DDC}	Comparator Operating Current	5V	—	—	—	200	μA
V_{CMPOS}	Comparator Input Offset Voltage	5V	—	-15	—	15	mV
V_{HYS}	Hysteresis Width	—	—	20	40	60	mV
V_{CM}	Input Common Mode Range	—	—	V_{SS}	—	V_{DD}-1.4	V

Power Good Characteristics

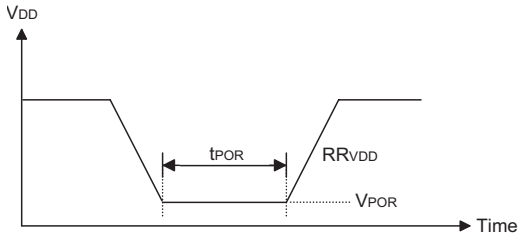
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DET}	Detection Voltage	—	—	-3%	4.6V	+3%	V
I_{DDC}	Power Good Operating Current	5V	—	—	—	85	μA
T_{PDS}	Power Good Output Stable Time	—	—	125	250	500	μS

Power on Reset Electrical Characteristics

Ta=25°C

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

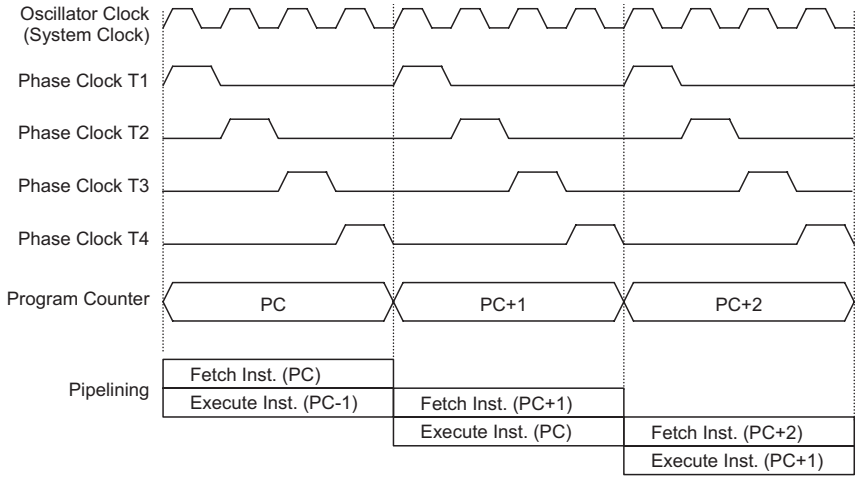


System Architecture

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

Clocking and Pipelining

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

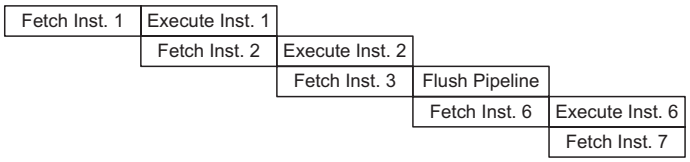


System Clock and Pipelining

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

```
1          MOV A,[12H]
2          CALL DELAY
3          CPL [12H]
4          :
5          :
6          NOP

DELAY:
```



Instruction Fetching

Program Counter

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

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

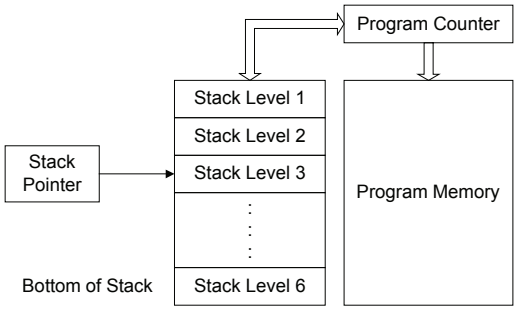
Program Counter	
Program Counter High byte	**PCL Register**
PC10~PC8	PCL7~PCL0

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

Stack

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

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



Arithmetic and Logic Unit – ALU

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

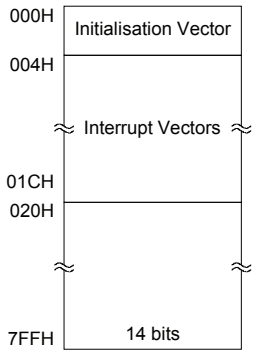
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement: INCA, INC, DECA, DEC
- Branch decision: JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET,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

Flash Program Memory

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

Structure

The Program Memory has a capacity of 2K×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 setup in any location within the Program Memory, is addressed by a separate table pointer register.



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

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

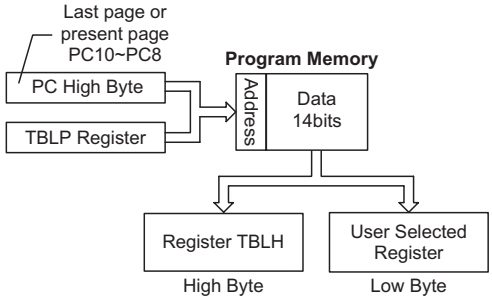


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "700H" which refers to the start address of the last page within the 2K words Program Memory of the device. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRDC [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 "TABRDC [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

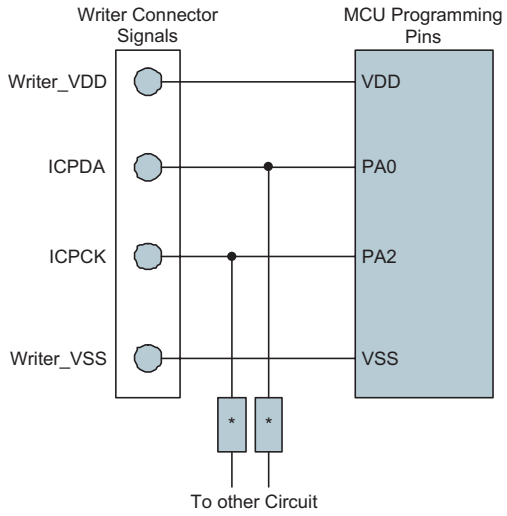
```
tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
:
mov a,06h          ; initialise low table pointer - note that this address
                   ; is referenced to the last page or  present page
mov tblp,a
:
:
tabrdc tempreg1    ; transfers value in table referenced by table pointer data at program
                   ; memory address "706H" transferred to tempreg1 and TBLH
dec tblp           ; reduce value of table pointer by one
tabrdc tempreg2    ; transfers value in table referenced by table pointer data at program
                   ; memory address "705H" transferred to tempreg2 and TBLH in this
                   ; example the data "1AH" is transferred to tempreg1 and data "0FH" to
                   ; register tempreg2
:
:
org 700h           ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

In Circuit Programming

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

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

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



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

On-Chip Debug Support – OCDS

There is an EV chip named HT45V5Q which is used to emulate the HT45F5Q device. This EV chip device also provides an "On-Chip Debug" function to debug the device during the development process. The EV chip and the actual MCU device are almost functionally compatible except for the "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the actual MCU device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For a more detailed OCDS description, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

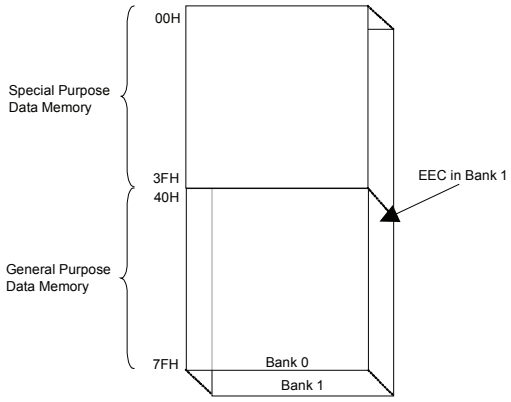
RAM Data Memory

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

Structure

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

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



Data Memory Structure

General Purpose Data Memory

There is 64 bytes of general purpose data memory which are arranged in Bank 0 and Bank1. All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user programing for both reading and writing operations. By using the bit operation instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

Special Purpose Data Memory

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

	Bank0	Bank1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1	
04H	BP	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H		
0AH	STATUS	
0BH	SMOD	
0CH	LVDC	
0DH	INTEG	
0EH	INTC0	
0FH	INTC1	
10H		
11H	MFI0	
12H		
13H		
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	IFS0	
19H	WDTC	
1AH		
1BH	TBC	
1CH	SMOD1	
1DH		
1EH	EEA	
1FH	EED	
20H	SADOL	
21H	SADOH	
22H	SADC0	
23H	SADC1	
24H		
25H	RSTC	
26H	PASR	
27H		
28H	STM0C0	
29H	STM0C1	
2AH	STM0DL	
2BH	STM0DH	
2CH	STM0AL	
2DH	STM0AH	
2EH	CHRGEN	
2FH	DAC8	
30H	DACC	
31H	SENSW	
32H	A0VOS	
33H	PGDR	
34H~3FH		
40H		EEC

: Unused;Read as 00H

Special Purpose Data Memory Structure

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

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

Memory Pointers – MP0, MP1

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

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

Indirect Addressing Program Example

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

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

Bank Pointer – BP

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

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

BP Register

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

Bit 7 ~ 1 Unimplemented, read as "0"

Bit 0 **DMBP0**: Select Data Memory Banks
0: Bank 0
1: Bank 1

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBLH

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

Status Register – STATUS

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

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

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

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

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

STATUS Register

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

"x" unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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

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

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

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

C is also affected by a rotate through carry instruction.

EEPROM Data Memory

This device contains an area of internal EEPROM Data Memory. EEPROM, which stands for Electrically Erasable Programmable Read Only Memory, is by its nature a non-volatile form of memory, with data retention even when its power supply is removed. By incorporating this kind of data memory, a whole new host of application possibilities are made available to the designer. The availability of EEPROM storage allows information such as product identification numbers, calibration values, specific user data, system setup data or other product information to be stored directly within the product microcontroller. The process of reading and writing data to the EEPROM memory has been reduced to a very trivial affair.

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 32×8 bits for this device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped and is therefore not directly accessible in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using one address register and one data register in Bank 0 and a single control register in Bank 1.

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address registers, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Bank1, cannot be directly addressed directly and can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

EEPROM Control Registers List

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

EEA Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	D4	D3	D2	D1	D0
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 **D4~D0**: Data EEPROM address
Data EEPROM address bit 4 ~ bit 0

EED Register

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

Bit 7~0 **D7~D0**: Data EEPROM data
Data EEPROM data bit 7 ~ bit 0

EEC Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3 **WREN**: Data EEPROM Write Enable
0: Disable
1: Enable
This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

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

Bit 1 **RDEN**: Data EEPROM Read Enable
0: Disable
1: Enable
This is the Data EEPROM Read Enable Bit which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.

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

Note: The WREN, WR, RDEN and RD can not be set to "1" at the same time in one instruction. The WR and RD can not be set to "1" at the same time.

Reading Data from the EEPROM

To read data from the EEPROM, the read enable bit, RDEN, in the EEC register must first be set high to enable the read function. The EEPROM address of the data to be read must then be placed in the EEA register. If the RD bit in the EEC register is now set high, a read cycle will be initiated. Setting the RD bit high will not initiate a read operation if the RDEN bit has not been set. When the read cycle terminates, the RD bit will be automatically cleared to zero, after which the data can be read from the EED register. The data will remain in the EED register until another read or write operation is executed. The application program can poll the RD bit to determine when the data is valid for reading.

Writing Data to the EEPROM

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

Write Protection

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

EEPROM Interrupt

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

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be Periodic by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Bank Pointer could be normally cleared to zero as this would inhibit access to Bank 1where the EEPROM control register exist. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process. When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write cycle is executed and then re-enabled after the write cycle starts. Note that the device should not enter the IDLE or SLEEP mode until the EEPROM read or write operation is totally complete. Otherwise, the EEPROM read or write operation will fail.

Programming Examples

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

```
MOV A, EEPROM_ADRES          ; user defined address
MOV EEA, A
MOV A, 040H                  ; setup memory pointer MP1
MOV MP1, A                   ; MP1 points to EEC register
MOV A, 01H                   ; setup Bank Pointer
MOV BP, A
SET IAR1.1                   ; set RDEN bit, enable read operations
SET IAR1.0                   ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                    ; check for read cycle end
JMP BACK
CLR IAR1                     ; disable EEPROM write
CLR BP
MOV A, EED                   ; move read data to register
MOV READ_DATA, A
```

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

```
MOV A, EEPROM_ADRES          ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA           ; user defined data
MOV EED, A
MOV A, 040H                  ; setup memory pointer MP1
MOV MP1, A                   ; MP1 points to EEC register
MOV A, 01H                   ; setup Bank Pointer
MOV BP, A
CLR   EMI
SET IAR1.3                   ; set WREN bit, enable write operations
SET IAR1.2                   ; start Write Cycle - set WR bit- executed immediately after
                             ; set WREN bit
SET EMI
BACK:
SZ IAR1.2                    ; check for write cycle end
JMP BACK
CLR IAR1                     ; disable EEPROM write
CLR BP
```

Oscillator

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

Oscillator Overview

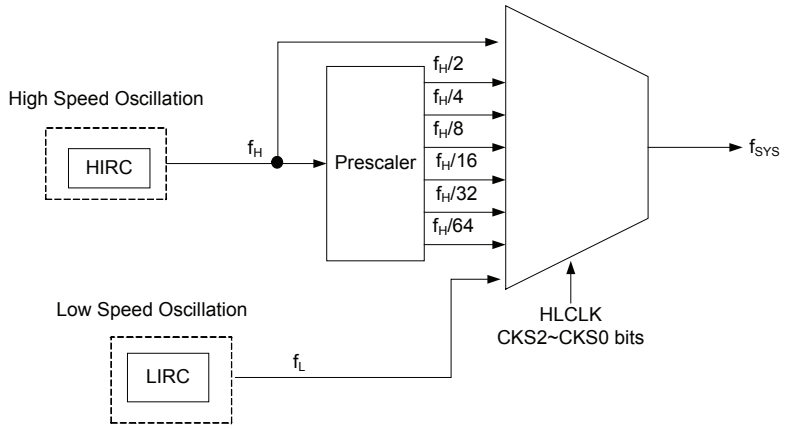
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators.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

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

Oscillator Types

System Clock Configurations

There are two methods of generating the system clock, a high speed oscillator and a low speed oscillator. The high speed oscillator is the internal 8MHz RC oscillator. The low speed oscillator is the internal 32kHz RC oscillator. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and CKS2 ~ CKS0 bits in the SMOD register and as the system clock can be dynamically selected.



System Clock Configurations

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 5V and at temperature of 25˚C degrees, the fixed oscillation frequency of the HIRC will have a tolerance within 2%.

Internal 32kHz Oscillator – LIRC

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

Supplementary Oscillator

The low speed oscillator, in addition to providing a system clock source is also used to provide a clock source to two other device functions. These are the Watchdog Timer and the Time Base Interrupts.

Operating Modes and System Clocks

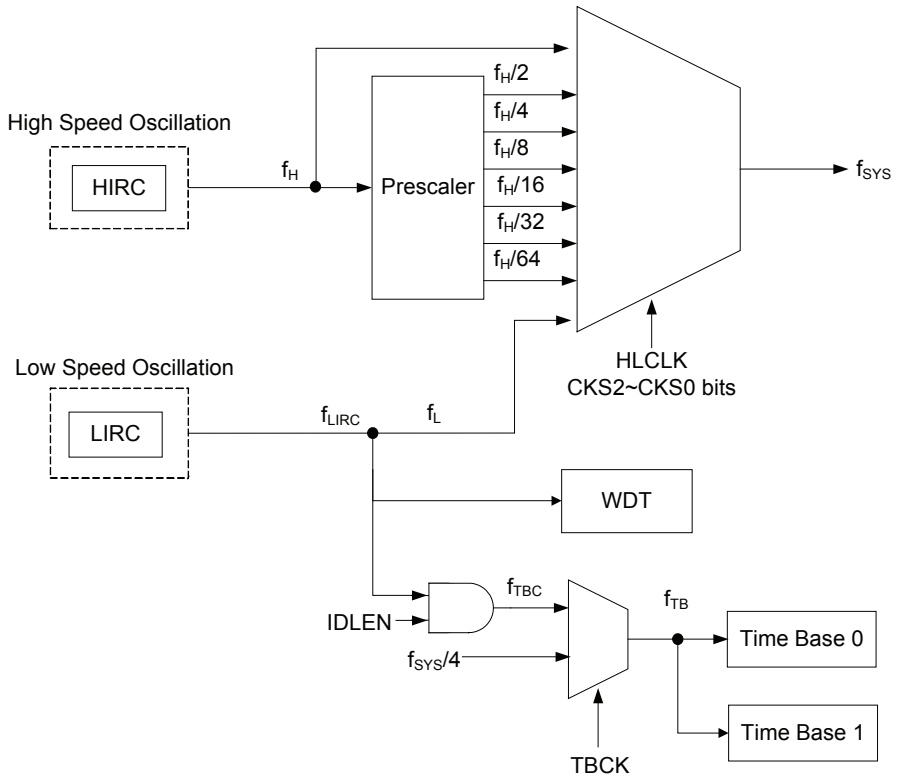
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their 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

This device has two different clock sources for both the CPU and peripheral function operation. By providing the user with 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 a low frequency, f_L, and is selected using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. The high speed system clock can be sourced from HIRC oscillator. The low speed system clock source can be sourced from the internal clock f_L. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.

There is one additional internal clock for the peripheral circuits, the Time Base clock, f_{TBC}. f_{TBC} is sourced from the LIRC oscillator. The f_{TBC} clock is used as a source for the Time Base interrupt functions and for the TM.



System Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_L from f_H, the high speed oscillation will stop to conserve the power. Thus there is no f_H~$f_H/64$ for peripheral circuit to use.

System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the NORMAL Mode and SLOW Mode. The remaining four modes, the SLEEP0, SLEEP1, IDLE0 and IDLE1 modes are used when the microcontroller CPU is switched off to conserve power.

Operating Mode	Description			
	CPU	f_{SYS}	f_{LIRC}	f_{TBC}
NORMAL mode	On	f_H~f_H/64	On	On
SLOW mode	On	f_L	On	On
IDLE0 mode	Off	Off	On	On
IDLE1 mode	Off	On	On	On
SLEEP0 mode	Off	Off	Off	Off
SLEEP1 mode	Off	Off	On	Off

NORMAL Mode

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

SLOW Mode

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

SLEEP0 Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP0 mode the CPU will be stopped, and the f_{LIRC} clock will be stopped too, and the Watchdog Timer function is disabled.

SLEEP1 Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP1 mode the CPU will be stopped. However the f_{LIRC} clocks will continue to operate if the Watchdog Timer function is enabled.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the SMOD1 register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU but some peripheral functions will remain operational such as the Watchdog Timer and TM. In the IDLE0 Mode, the system oscillator will be stopped.

IDLE1 Mode

The IDLE1 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the SMOD1 register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU but may continue to provide a clock source to keep some peripheral functions operational such as the Watchdog Timer and TM. In the IDLE1 Mode, the system oscillator will continue to run, and this system oscillator may be high speed or low speed system oscillator. In the IDLE1 Mode, the Watchdog Timer clock, f_{LIRC}, will be on.

Control Register

A single register, SMOD, is used for overall control of the internal clocks within the device.

SMOD Register

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

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

Bit 4 Unimplemented, read as "0"

Bit 3 **LTO**: Low speed system oscillator ready flag
0: Not ready
1: Ready
This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP0 mode, but after a wake-up has occurred the flag will change to a high level after 1~2 cycles if the LIRC oscillator is used.

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

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

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

SMOD1 Register

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

"x" unknown

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

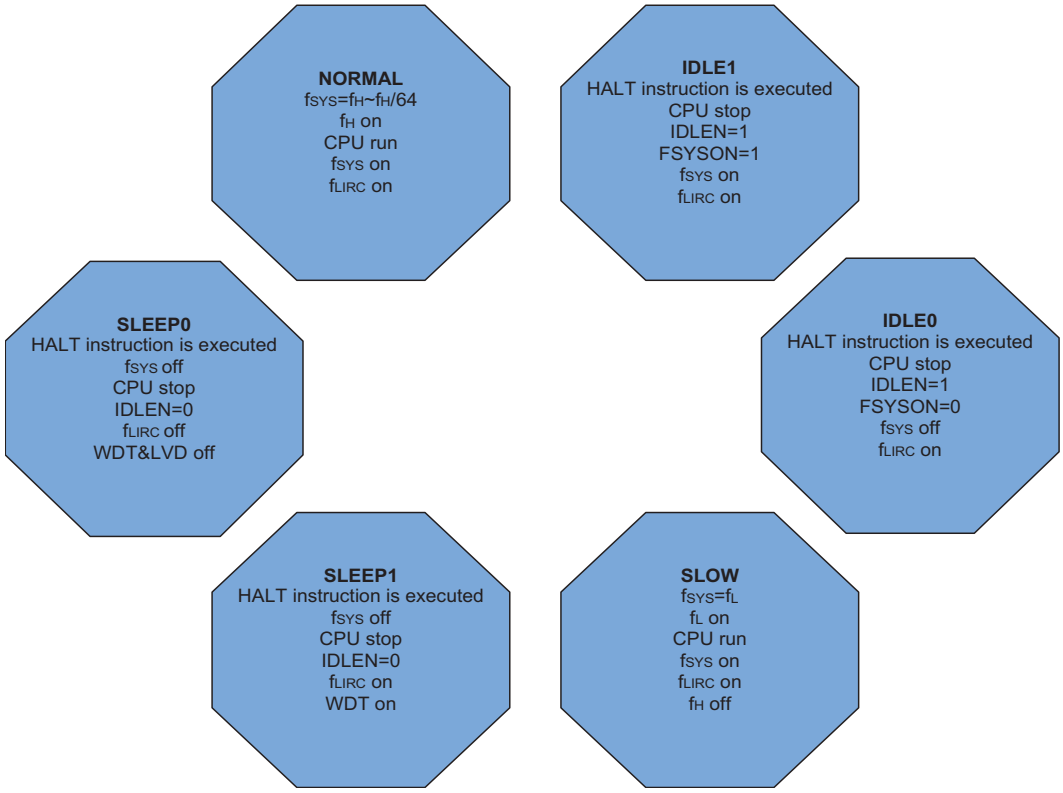
Bit 6~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset caused by RSTC setting
0: Not active
1: Active
This bit can be clear to "0", but cannot set to "1".If this bit is set, only cleared by Software or POR reset.

Bit 2 **LVRF**: LVR function reset flag
0: Not active
1: Active
This bit can be clear to "0", but can not be set to "1".

Bit 1 Unimplemented, read as "0"

Bit 0 **WRF**: WDT Control register software reset flag
0: Not occur
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.



Operating Mode Switching

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

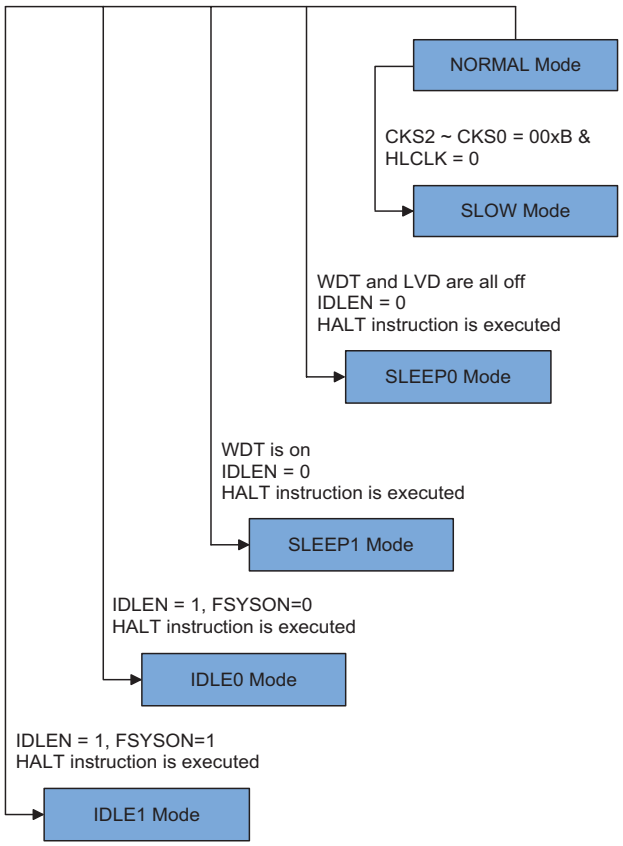
In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SMOD register and FSYSON in the SMOD1 register.

When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~$f_H/64$ or f_L. If the clock is from the f_L, the high speed clock source will stop running to conserve power. When this happens it must be noted that the $f_H/16$ and $f_H/64$ internal clock sources will also stop running, which may affect the operation of other internal functions such as the TM.

NORMAL Mode to SLOW Mode Switching

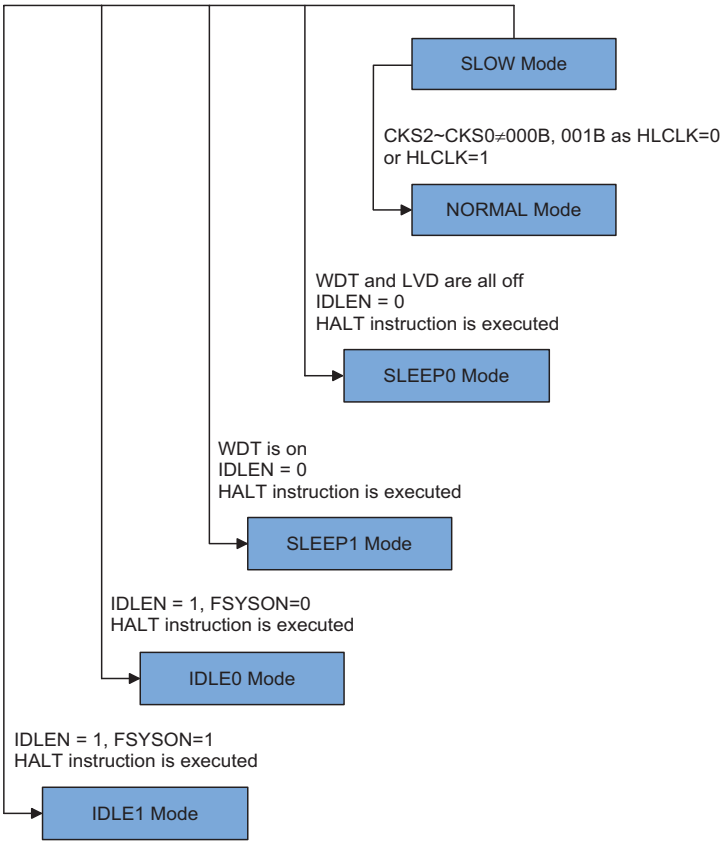
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by setting the HLCLK bit to “0” and setting the CKS2~CKS0 bits to “000” or “001” in the SMOD register.This will then use the low speed system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

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



SLOW Mode to NORMAL Mode Switching

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



Entering the SLEEP0 Mode

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

- The system clock and f_{LIRC} clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and stopped as the WDT is disabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the SLEEP1 Mode

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

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

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in SMOD1 register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

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

Entering the IDLE1 Mode

There is only one way for the device to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in SMOD1 register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The system clock and f_{LIRC} clock will be on and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting as the WDT clock source is derived from the f_{LIRC} clock.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Standby Current Considerations

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

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

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

Wake-up

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

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

If the system is woken up by an external reset, the device will experience a full system reset, however, If the device is woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flags. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the "HALT" instruction. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal f_{LIRC} clock which is supplied by the LIRC oscillator. 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. The LIRC internal oscillator has an approximate period of 32kHz at a supply voltage of 5V. However, it should be noted that this specified internal clock period can vary with V_{DD}, temperature and process variations. The WDT can be enabled/disabled using the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. The WRF software reset flag will be indicated in the SMOD1 register. These registers control the overall operation of the Watchdog Timer.

WDTC Register

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

Bit 7~ 3 **WE4 ~ WE0**: WDT function software control
10101: WDT disable
01010: WDT enable
Other values: Reset MCU
When these bits are changed to any other values by the environmental noise to reset the microcontroller, the reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit will be set to 1to indicate the reset source.

Bit 2~ 0 **WS2 ~ WS0**: WDT Time-out period selection
000: $2^8/f_{LIRC}$
001: $2^9/f_{LIRC}$
010: $2^{10}/f_{LIRC}$
011: $2^{11}/f_{LIRC}$(default)
100: $2^{12}/f_{LIRC}$
101: $2^{13}/f_{LIRC}$
110: $2^{14}/f_{LIRC}$
111: $2^{15}/f_{LIRC}$
These three bits determine the division ratio of the Watchdog Timer sourece clock, which in turn determines the timeout period.

SMOD1 Register

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

"x" unknown

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

Bit 6~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset caused by RSTC setting
0: Not active
1: Active
This bit can be clear to "0", but cannot set to "1".If this bit is set, only cleared by Software or POR reset.

Bit 2 **LVRF**: LVR function reset flag
0: Not active
1: Active
This bit can be clear to "0", but can not be set to "1".

Bit 1 Unimplemented, read as "0"

Bit 0 **WRF**: WDT Control register software reset flag
0: Not occur
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

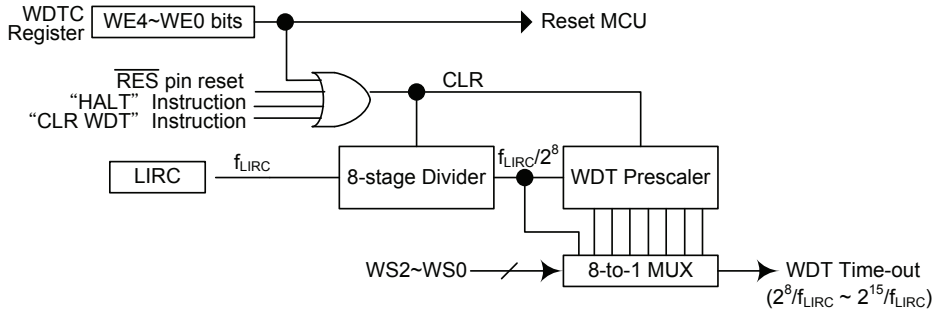
The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear WDT instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to additional enable/disable and reset control of the Watchdog Timer.

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

Watchdog Timer Enable/Disable Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Four methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT reset, which means a value other than 01010B and 10101B is written into the WE4~WE0 bit locations, the second is an external hardware reset, which means a low level on the external reset pin, the third is using the Watchdog Timer software clear instruction and the fourth is via a HALT instruction. There is only one method of using software instruction to clear the Watchdog Timer. That is to use the single "CLR WDT" instruction to clear the WDT.

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



Watchdog Timer

Reset and Initialisation

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

In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{RES}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high.

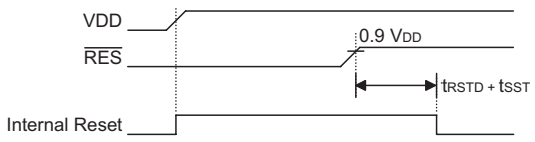
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply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

Reset Functions

There are several ways in which a microcontroller reset can occur, through events occurring both internally and externally:

Power-on Reset

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



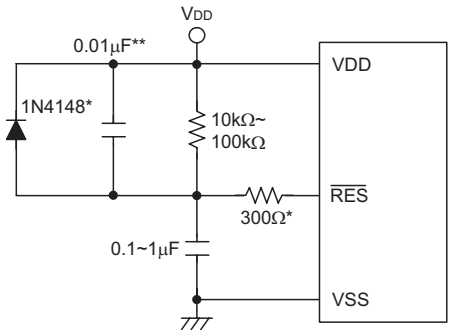
Note: t_{RSTD} is power-on delay, typical time=50ms

Power-On Reset Timing Chart

$\overline{\text{RES}}$ Pin Reset

Although the microcontroller has an internal RC reset function, if the V_{DD} power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{\text{RES}}$ pin, whose additional time delay will ensure that the $\overline{\text{RES}}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{\text{RES}}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.

For most applications a resistor connected between VDD and the $\overline{\text{RES}}$ pin and a capacitor connected between VSS and the $\overline{\text{RES}}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{\text{RES}}$ pin should be kept as short as possible to minimize any stray noise interference. For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



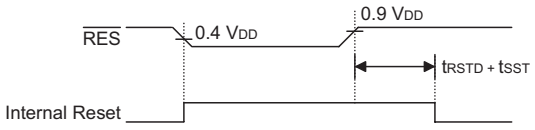
Note: "*" It is recommended that this component is added for added ESD protection

"**" It is recommended that this component is added in environments where power line noise is significant

External $\overline{\text{RES}}$ Circuit

More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

Pulling the $\overline{\text{RES}}$ Pin low using external hardware will also execute a device reset. In this case, as in the case of other resets, the Program Counter will reset to zero and program execution initiated from this point.



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

$\overline{\text{RES}}$ Reset Timing Chart

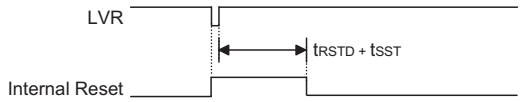
- **RSTC External Reset Register**

Bit	7	6	5	4	3	2	1	0
Name	RSTC7	RSTC6	RSTC5	RSTC4	RSTC3	RSTC2	RSTC1	RSTC0
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7 ~ 0 **RSTC7 ~ RSTC0**: PA7/$\overline{\text{RES}}$ selection
01010101: Configured as PA7 pin or other function
10101010: Configured as $\overline{\text{RES}}$ pin
Other Values: MCU reset (reset will be active after 2~3 LIRC clock for debounce time)
All reset will reset this register as POR value except WDT time out Hardware warm reset.

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provide an MCU reset should the value fall below a certain predefined level. The LVR function is always enabled during the normal and slow modes with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the SMOD1 register will also be set to1. For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~ V_{LVR} must exist for greater than the value t_{LVR} specified in the LVR characteristics. If the low voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} is 2.1V, the LVR will reset the device after 2~3 LIRC clock cycles. Note that the LVR function will be automatically disabled when the device enters the SLEEP/IDLE mode.



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

Low Voltage Reset Timing Chart

- **SMOD1 Register**

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

"x" unknown

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

Bit 6~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset caused by RSTC setting
0: Not active
1: Active
This bit can be clear to "0", but cannot set to "1".If this bit is set, only cleared by Software or POR reset.

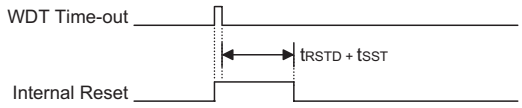
Bit 2 **LVRF**: LVR function reset flag
0: Not active
1: Active
This bit can be clear to "0", but can not be set to "1".

Bit 1 Unimplemented, read as "0"

Bit 0 **WRF**: WDT Control register software reset flag
0: Not occur
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 Time-out Reset during Normal Operation

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

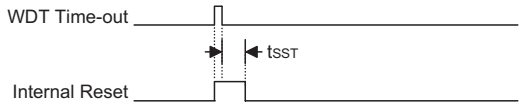


Note: t_{RSTD} is power-on delay, typical time=16.7ms

WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

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

Note: "u" stands for unchanged

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

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

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers. 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)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)*
Program Counter	000H	000H	000H	000H	000H
MP0	1xxx xxxx	1xxx xxxx	1xxx xxxx	1xxx xxxx	1uuu uuuu
MP1	1xxx xxxx	1xxx xxxx	1xxx xxxx	1xxx xxxx	1uuu uuuu
BP	---- --0	---- --0	---- --0	---- --0	---- ---u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
SMOD	000- 0011	000- 0011	000- 0011	000- 0011	uuu- uuuu
LVDC	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
INTEG	---- --00	---- --00	---- --00	---- --00	---- --uu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuu-
MFI0	--00 --00	--00 --00	--00 --00	--00 --00	--uu --uu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS0	---- --00	---- --00	---- --00	---- --00	---- -uuu
WDTC	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	0011 -111	0011 -111	0011 -111	0011 -111	uuuu –uuu
SMOD1	0--- 0x-0	0--- uu-u	0--- uu-u	0--- uu-u	u--- uu-u
INTEG	---- --00	---- --00	---- --00	---- --00	---- --uu
EEA	---0 0000	---0 0000	---0 0000	---0 0000	---u uuuu
EED	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADOL (ADRFS=0)	xxxx ----	xxxx ----	xxxx ----	xxxx ----	uuuu ----
SADOL (ADRFS=1)	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADOH (ADRFS=0)	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADOH (ADRFS=1)	---- xxxx	---- xxxx	---- xxxx	---- xxxx	---- uuuu
SADC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
PASR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0C0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0C1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0DL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
STM0DH	---- --00	---- --00	---- --00	---- --00	---- --uu
STM0AL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0AH	---- --00	---- --00	---- --00	---- --00	---- --uu
CHRGEN	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
DAC8	1000 0000	1000 0000	1000 0000	1000 0000	uuuu uuuu
DACC	1--- ----	1--- ----	1--- ----	1--- ----	u--- ----
SENSW	--01 0101	--01 0101	--01 0101	--01 0101	--uu uuuu
A0VOS	0001 0000	0001 0000	0001 0000	0001 0000	uuuu uuuu
PGDR	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
EEC	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu

Note: "*" stands for warm reset
"-" not implement
"u" stands for "unchanged"
"x" stands for "unknown"

Input/Output Ports

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

The device provide 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.

I/O Control Register List

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PASR	PAS7	PAS6	PAS5	PAS4	PAS3	PAS2	PAS1	PAS0
IFS0	—	—	—	—	—	—	INTPS1	INTPS0

Pull-high Resistors

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

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

PAPU Register

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

Bit 7 ~ 0 I/O Port A bit 7~ bit 0 Pull-High Control
0: Disable
1: Enable

Port A Wake-up

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

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

PAWU Register

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

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

I/O Port Control Registers

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

PAC Register

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

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

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 PASR and IFS0 registers which can select the desired functions of the multi-function pin-shared pins.

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

PASR Register

Bit	7	6	5	4	3	2	1	0
Name	PAS7	PAS6	PAS5	PAS4	PAS3	PAS2	PAS1	PAS0
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 **PAS7~PAS6**: Pin-Shared Control Bits
00: PA5/INT
01: STP0
10: PA5/INT
11: A1P

Bit 5 **PAS5**: Pin-Shared Control Bit
0: PA4
1: STP0

Bit 4 **PAS4**: Pin-Shared Control Bit
0: PA3/INT
1: AN3

Bit 3 **PAS3**: Pin-Shared Control Bit
0: PA2
1: AN2

Bit 2~1 **PAS2~PAS1**: Pin-Shared Control Bits
00: PA1
01: PA1
10: VREF
11: AN1

Bit 0 **PAS0**: Pin-Shared Control Bit
0: PA0
1: AN0

IFS0 Register

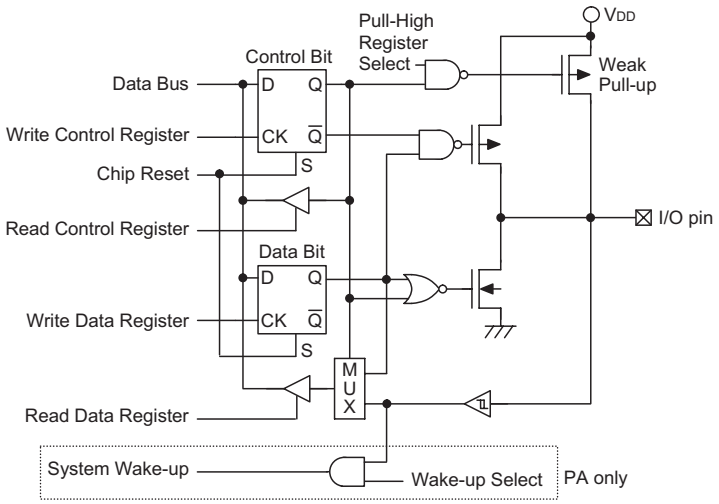
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INTPS1	INTPS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

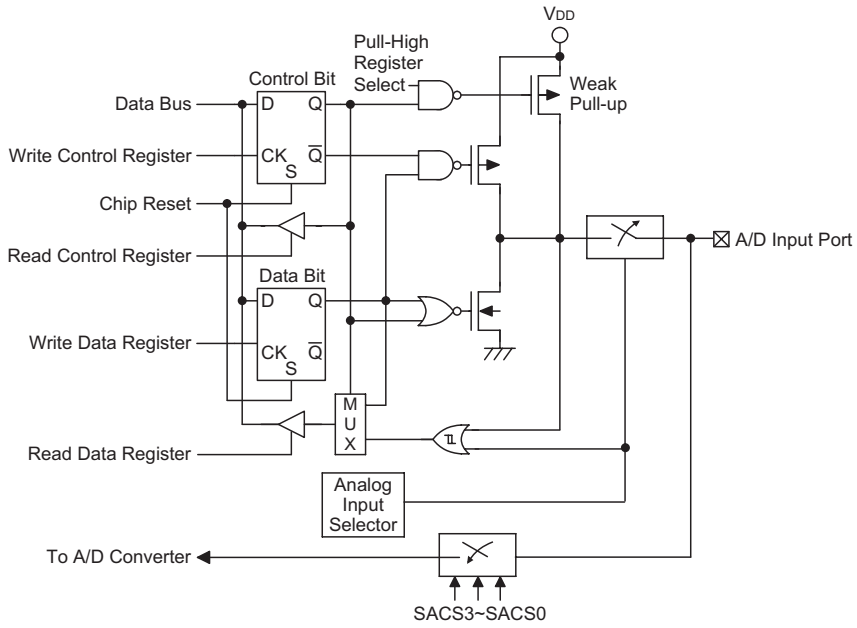
Bit 1~0 **INTPS1, INTPS0**: INT Pin Remapping Control
00: INT on PA7 (default)
01: INT on PA3
10: INT on PA5
11: INT on PA6

I/O Pin Structures

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



Generic Input/Output Structure



A/D Input/Output Structure

Programming Considerations

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

Port A has the additional capability of providing wake-up functions. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the Port A pins. Single or multiple pins on Port A can be setup to have this function.

Timer Module – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions the device includes only one Timer Module, abbreviated to the name TM. The TM is 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. This TM has two individual interrupts. The addition of input and output pins for this TM ensures that users are provided with timing units with a wide and flexible range of features.

Introduction

The device contains only one Standard Type TM unit, with its individual reference name, TM, and its type is 10-bit STM. The main features of TM are summarised in the accompanying table.

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

TM Function Summary

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 counter whose value is then compared with the value of pre-programmed internal comparators. When the free running counter has the same value as the pre-programmed comparator, known as a compare match situation, a TM interrupt signal will be generated which can clear the counter and perhaps also change the condition of the TM output pin. The internal TM counter is driven by a user selectable clock source, which can be an internal clock or an external pin.

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the ST0CK2~ST0CK0 bits in the STM control registers. The clock source can be a ratio of either the system clock f_{SYS} or the internal high clock f_H, the f_{TBC} clock source or the external STCK0 pin. The STCK0 pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

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

TM External Pins

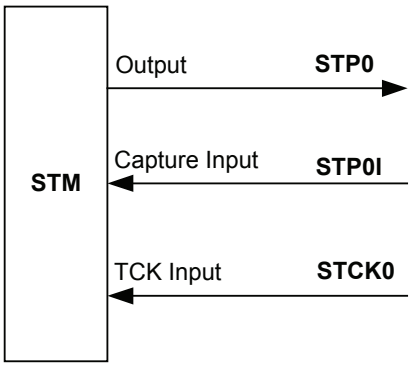
The TM has two TM input pins, with the label STCK0 and STP0I. The TM input pin STCK0, is essentially a clock source for the TM and is selected using the ST0CK2~ST0CK0 bits in the STM0C0 register. This external TM input pin allows an external clock source to drive the internal TM. This external TM input pin is shared with other functions but will be connected to the internal TM if selected using the ST0CK2~ST0CK0 bits. The TM input pin can be chosen to have either a rising or falling active edge.

The other TM input pin, STP0I, 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 ST0IO1 and ST0IO0 bits in the STM0C1 register.

The TM has one output pin with the label STP0. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external STP0 output pin is also the pin where the TM generates the PWM output waveform. As the TM output pins are pin-shared with other function, the TM output function must first be setup using registers. A single bit in one of the registers determines if its associated pin is to be used as an external TM output pin or if it is to have another function.

TM Input/Output Pin Control Register

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

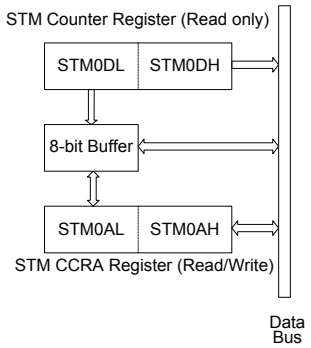


STM Function Pin Control Block Diagram

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA register, and CCRP register pair for Periodic Timer Module, 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 register and CCRP registers are implemented in the way shown in the following diagram and accessing the register is carried out CCRP low byte register using the following access procedures. Accessing the CCRA or CCRP low byte register without following these access procedures will result in unpredictable values.



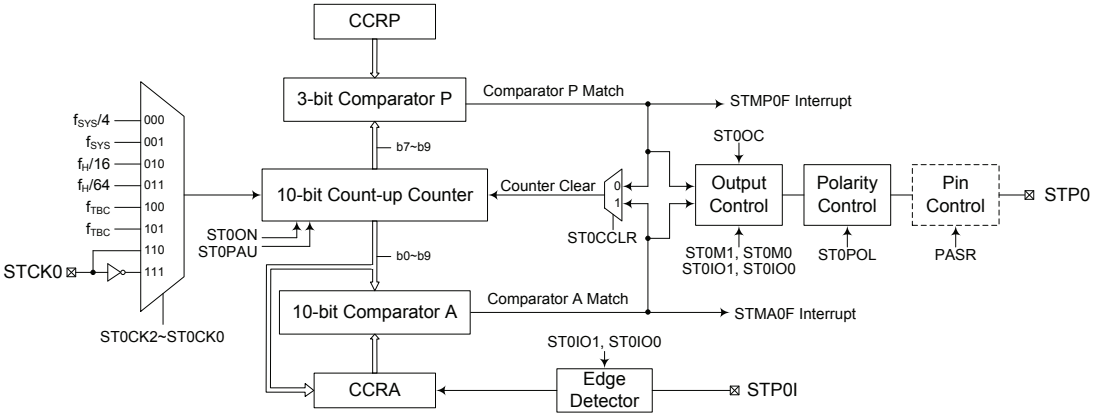
The following steps show the read and write procedures:

- Writing Data to CCRA
  - Step 1. Write data to Low Byte STM0AL
    – note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte STM0AH
    – 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 STM0DH, STM0AH
    – 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 STM0DL, STM0AL
    – 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 TM can be controlled with two external input pins and can drive one external output pin.

TM Type	TM Name	TM Input Pin	TM Output Pin
10-bit STM	STM	STCK0, STP0I	STP0



Standard Type TM Block Diagram

Standard TM Operation

At its core is a 10-bit count-up counter which is driven by a user selectable internal clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is 3-bit wide whose value is compared with the highest 3 bits in the counter while the CCRA is the 10 bits and therefore compares with all counter bits.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the ST0ON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a TM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources and can also control an output pin. All operating setup conditions are selected using relevant internal registers.

Standard Type TM Register Description

Overall operation of the Standard TM is controlled using 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 as well as three CCRP bits.

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STM0C0	ST0PAU	ST0CK2	ST0CK1	ST0CK0	ST0ON	ST0RP2	ST0RP1	ST0RP0
STM0C1	ST0M1	ST0M0	ST0IO1	ST0IO0	ST0OC	ST0POL	ST0DPX	ST0CCLR
STM0DL	D7	D6	D5	D4	D3	D2	D1	D0
STM0DH	—	—	—	—	—	—	D9	D8
STM0AL	D7	D6	D5	D4	D3	D2	D1	D0
STM0AH	—	—	—	—	—	—	D9	D8

10-bit Standard TM Register List

STM0C0 Register

Bit	7	6	5	4	3	2	1	0
Name	ST0PAU	ST0CK2	ST0CK1	ST0CK0	ST0ON	ST0RP2	ST0RP1	ST0RP0
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 **ST0PAU**: 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 **ST0CK2~ST0CK0**: Select STM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: f_{TBC}
110: STCK0 rising edge clock
111: STCK0 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_{TBC} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **ST0ON**: 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, 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 or the PWM output Mode or Single Pulse Output Mode then the STM output pin will be reset to its initial condition, as specified by the ST0OC bit, when the ST0ON bit changes from low to high.

Bit 2~0 **ST0RP2~ ST0RP0**: 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 ST0CCLR bit is set to zero. Setting the ST0CCLR 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.

STM0C1 Register

Bit	7	6	5	4	3	2	1	0
Name	ST0M1	ST0M0	ST0IO1	ST0IO0	ST0OC	ST0POL	ST0DPX	ST0CCLR
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 **ST0M1~ ST0M0**: 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 ST0M1 and ST0M0 bits. In the Timer/Counter Mode, the STM output pin state is undefined.

Bit 5~4 **ST0IO1~ ST0IO0:** 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 STP0I
01: Input capture at falling edge of STP0I
10: Input capture at falling/rising edge of STP0I
11: Input capture disabled

Timer/counter Mode:
Unused

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

In the Compare Match Output Mode, the ST0IO1~ST0IO0 bits determine how the TM output pin changes state when a compare match occurs from the Comparator A. The TM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the ST0IO1~ST0IO0 bits are both zero, then no change will take place on the output. The initial value of the TM output pin should be setup using the ST0OC bit. Note that the output level requested by the ST0IO1~ST0IO0 bits must be different from the initial value setup using the ST0OC bit otherwise no change will occur on the TM output pin when a compare match occurs. After the TM output pin changes state it can be reset to its initial level by changing the level of the ST0ON bit from low to high.

In the PWM Mode, the ST0IO1 and ST0IO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to change the values of the ST0IO1 and ST0IO0 bits only after the TM has been switched off. Unpredictable PWM outputs will occur if the ST0IO1 and ST0IO0 bits are changed when the TM is running.

Bit 3 **ST0OC**: STM Output control bit

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

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

This is the output control bit for the 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 ST0ON bit changes from low to high.

Bit 2 **ST0POL**: STM Output polarity Control
0: Non-invert
1: Invert

This bit controls the polarity of the STM 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 **ST0DPX**: 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 **ST0CCLR**: Select STM Counter clear condition
0: STM Comparator P match
1: STM Comparator A match

This bit is used to select the method which clears the counter. Remember that the Standard STM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the ST0CCLR 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 ST0CCLR bit is not used in the PWM output mode, Single Pulse or Input Capture Mode.

STM0DL 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 STM Counter Low Byte Register bit 7 ~ bit 0
STM 10-bit Counter bit 7 ~ bit 0

STM0DH 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 STM Counter High Byte Register bit 1 ~ bit 0
STM 10-bit Counter bit 9 ~ bit 8

STM0AL 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 STM CCRA Low Byte Register bit 7 ~ bit 0
STM 10-bit Counter bit 7 ~ bit 0

STM0AH 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 STM CCRA High Byte Register bit 1 ~ bit 0
STM 10-bit Counter bit 9 ~ bit 8

Standard Type TM Operating Modes

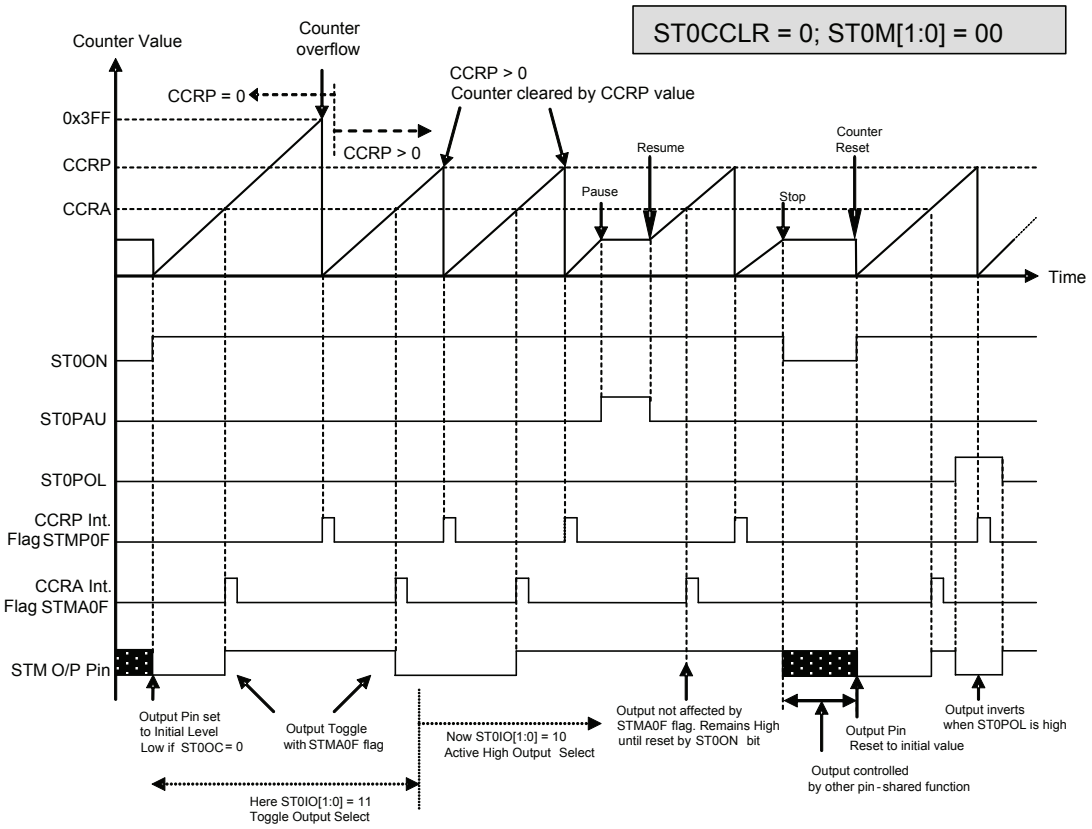
The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the ST0M1 and ST0M0 bits in the STM0C1 register.

Compare Output Mode

To select this mode, bits ST0M1 and ST0M0 in the STM0C1 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 ST0CCLR 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 STMA0F and STMP0F interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the ST0CCLR bit in the STM0C1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMA0F interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when ST0CCLR is high no STMP0F interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set 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 STMA0F 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 STMA0F interrupt request flag is generated after a compare match occurs from Comparator A. The STMP0F 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 ST0IO1 and ST0IO0 bits in the STM0C1 register. The STM output pin can be selected using the ST0IO1 and ST0IO0 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 ST0ON bit changes from low to high, is setup using the ST0OC bit. Note that if the ST0IO1 and ST0IO0 bits are zero then no pin change will take place.

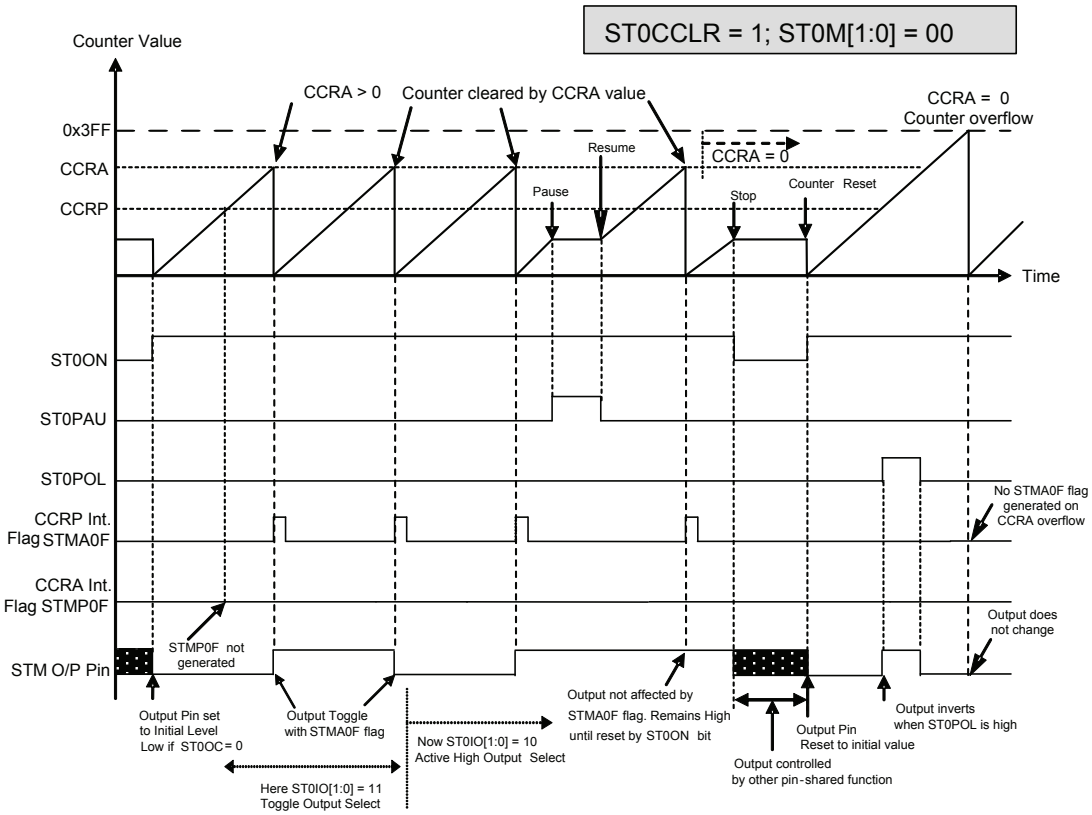


Compare Match Output Mode – ST0CCLR=0

Note: 1. With ST0CCLR = 0 a Comparator P match will clear the counter

2. The TM output pin controlled only by the STMA0F flag

3. The output pin reset to initial state by a ST0ON bit rising edge



Compare Match Output Mode – ST0CCLR=1

Note: 1. With ST0CCLR = 1 a Comparator A match will clear the counter

2. The TM output pin controlled only by the STMA0F flag

3. The output pin reset to initial state by a ST0ON rising edge

4. The STMP0F flag is not generated when ST0CCLR = 1

Timer/Counter Mode

To select this mode, bits ST0M1 and ST0M0 in the STM0C1 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 by setting pin-share function register.

PWM Output Mode

To select this mode, bits ST0M1 and ST0M0 in the STM0C1 register should be set to 10 respectively and also the ST0IO1 and ST0IO0 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 ST0CCLR 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 ST0DPX bit in the STM0C1 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 ST0OC bit in the STM0C1 register is used to select the required polarity of the PWM waveform while the two ST0IO1 and ST0IO0 bits are used to enable the PWM output or to force the STM output pin to a fixed high or low level. The ST0POL bit is used to reverse the polarity of the PWM output waveform.

- **10-bit STM, PWM Output Mode, Edge-aligned Mode, ST0DPX=0**

CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	128	256	384	512	640	768	896	1024
Duty	CCRA							

If f_{SYS} = 16MHz, TM clock source is $f_{SYS}/4$, CCRP = 100b and CCRA =128,

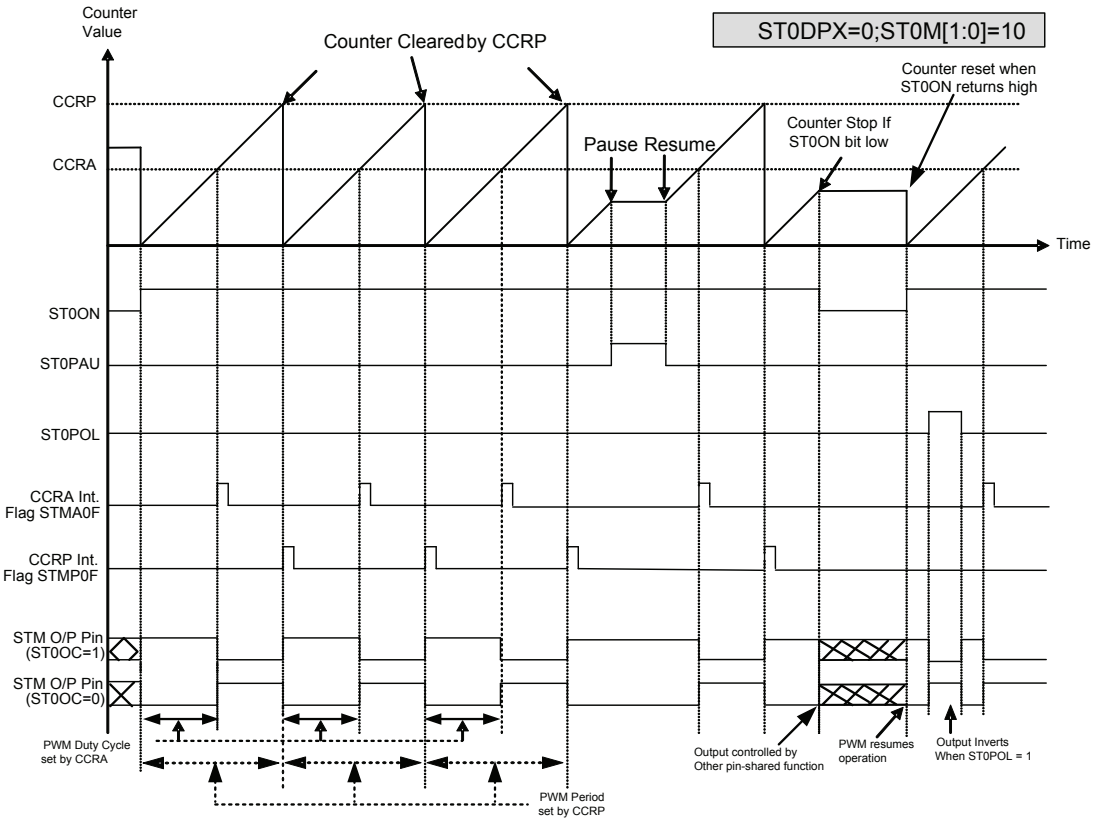
The STM PWM output frequency = ($f_{SYS}/4$) / 512 = $f_{SYS}/2048$ = 7.8125kHz, duty = 128/512 = 25%.

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

- **10-bit STM, PWM Output Mode, Edge-aligned Mode, ST0DPX=1**

CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	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.



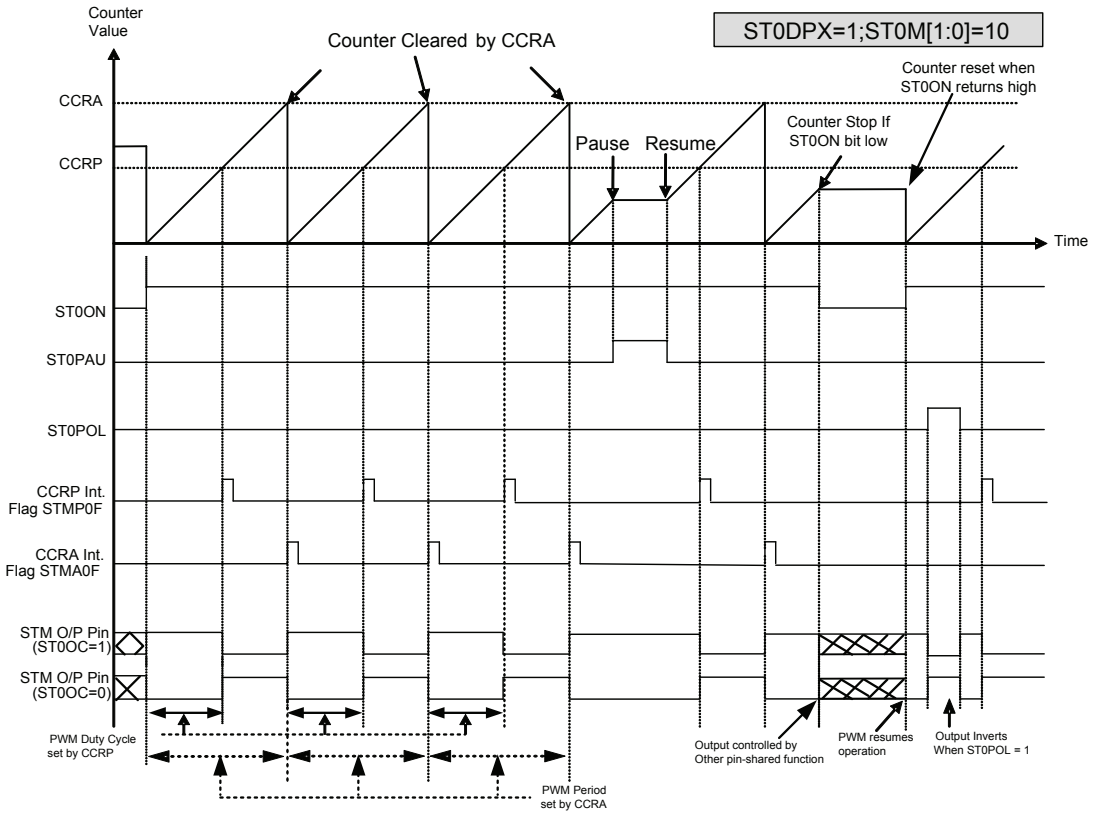
PWM Output Mode – ST0DPX = 0

Note: 1. Here ST0DPX = 0 - Counter cleared by CCRP

2. A counter clear sets PWM Period

3. The internal PWM function continues running even when ST0IO[1:0] = 00 or 01

4. The ST0CCLR bit has no influence on PWM operation



PWM Output Mode - ST0DPX = 1

Note: 1. Here ST0DPX = 1 - Counter cleared by CCRA

2. A counter clear sets PWM Period

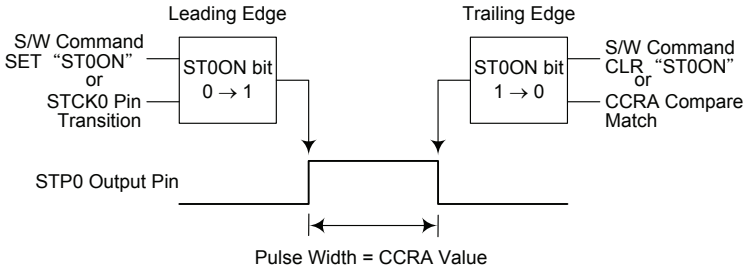
3. The internal PWM function continues even when ST0IO[1:0] = 00 or 01

4. The ST0CCLR bit has no influence on PWM operation

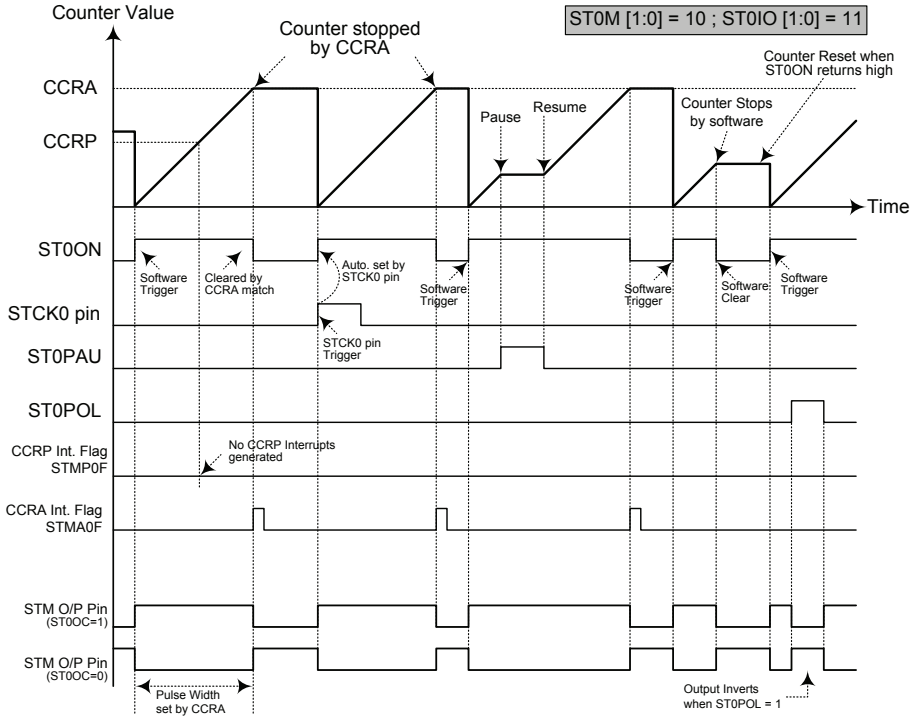
Single Pulse Mode

To select this mode, bits ST0M1 and ST0M0 in the STM0C1 register should be set to 10 respectively and also the ST0IO1 and ST0IO0 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 ST0ON bit, which can be implemented using the application program. However in the Single Pulse Mode, the ST0ON bit can also be made to automatically change from low to high using the external STCK0 pin, which will in turn initiate the Single Pulse output. When the ST0ON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The ST0ON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the ST0ON bit is cleared to zero, which can be implemented using the application program or when a compare match occurs from Comparator A.



Single Pulse Generation



Single Pulse Mode

Note: 1. Counter stopped by CCRA match
2. CCRP is not used
3. The pulse is triggered by setting the ST0ON bit high
4. In the Single Pulse Mode, ST0IO [1:0] must be set to "11" and can not be changed.

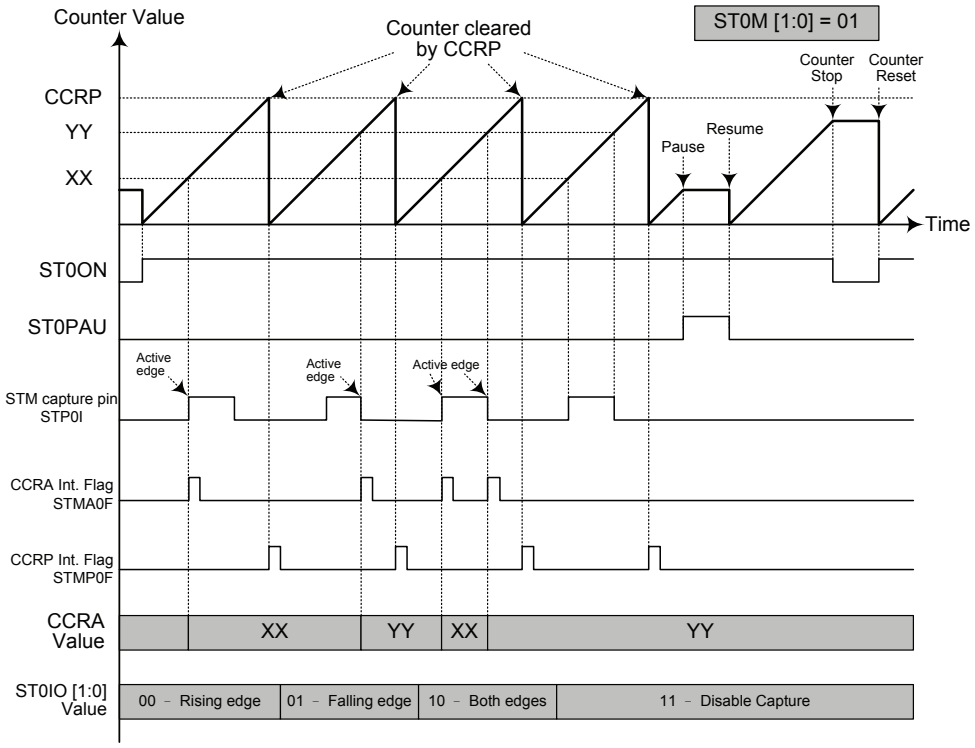
However a compare match from Comparator A will also automatically clear the ST0ON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate a STM interrupt. The counter can only be reset back to zero when the ST0ON bit changes from low to high when the counter restarts. In the Single Pulse Mode CCRP is not used. The ST0CCLR and ST0DPX bits are not used in this Mode.

Capture Input Mode

To select this mode bits ST0M1 and ST0M0 in the STM0C1 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 STP0I, whose active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the ST0IO1 and ST0IO0 bits in the STM0C1 register. The counter is started when the ST0ON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the STP0I the present value in the counter will be latched into the CCRA registers and a STM interrupt generated. Irrespective of what events occur on the STP0I the counter will continue to free run until the ST0ON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a STM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The ST0IO1 and ST0IO0 bits can select the active trigger edge on the STP0I to be a rising edge, falling edge or both edge types. If the ST0IO1 and ST0IO0 bits are both set high, then no capture operation will take place irrespective of what happens on the STP0I, however it must be noted that the counter will continue to run.

The ST0CCLR and ST0DPX bits are not used in this Mode.



Capture Input Mode

Note: 1. ST0M[1:0] = 01 and active edge set by the ST0IO[1:0] bits

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

3. The ST0CCLR and ST0DPX bits are not used

4. No output function – ST0OC and ST0POL bits are not used

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

Analog to Digital Converter

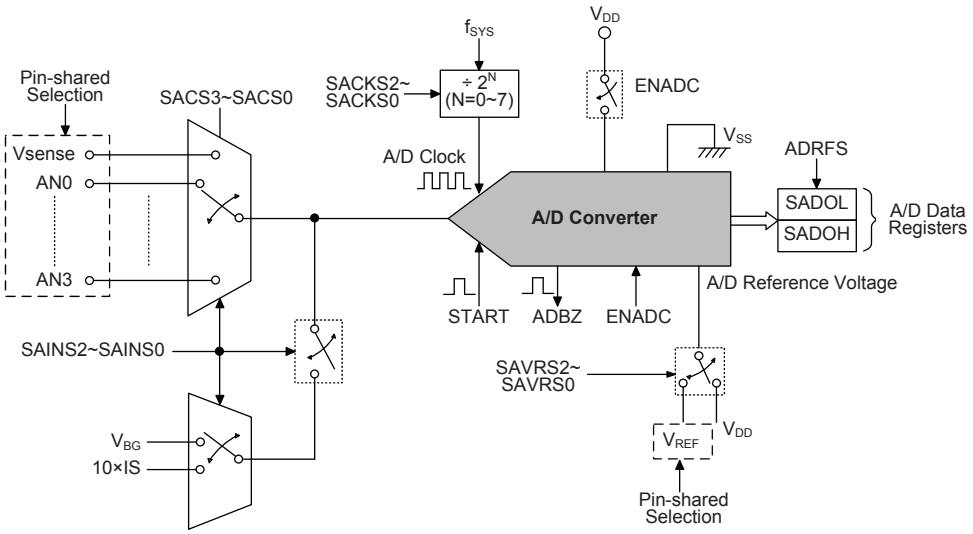
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

A/D Overview

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

Input Channels	A/D Channel Select Bits	Input Pins
5+2	SAINS2~SAINS0, SACS3~SACS0	AN0~AN3 and Vsense VBG, 10×IS

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



A/D Converter Structure

A/D Converter Register Description

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

Name	Bit							
	7	6	5	4	3	2	1	0
SADOL(ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL(ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH(ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH(ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ENADC	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

A/D Converter Data Registers – SADOL, SADOH

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

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

A/D Data Registers

A/D Converter Control Registers – SADC0, SADC1, PASR

To control the function and operation of the A/D converter, several control registers known as SADC0 and SADC1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input. If the SAINS2~SAINS0 bits are set to "000", the external analog channel input is selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected to be converted. If the SAINS2~SAINS0 bits are set to "001", the V_{BG} voltage is selected to be converted. If the SAINS2~SAINS0 bits are set to "010", the OPA output voltage is selected to be converted. The internal analog signals can be derived from the A/D converter supply power, V_{DD}, or internal reference voltage, V_{REF}. If the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off to avoid the signal contention.

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

SADC0 Register

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

Bit 7 **START**: Start the A/D conversion
0→1→0: Start A/D conversion
0→1: Reset the A/D converter and set ADBZ to 0
1→0: Start A/D conversion and set ADBZ to 1

Bit 6 **ADBZ**: ADC busy flag
0: A/D conversion ended or no conversion
1: A/D is busy

Bit 5 **ENADC**: ADC enable/disable control register
0: ADC disable
1: ADC enable

Bit 4 **ADRFS**: A/D output data format selection bit
0: ADC output data format à SADOH=D[11:4]; SADOL=D[3:0]
1: ADC output data format à SADOH=D[11:8]; SADOL=D[7:0]

Bit 3~0 **SACS3~SACS0**: ADC input channels selection
0000: ADC input channel comes from AN0
0001: ADC input channel comes from AN1
0010: ADC input channel comes from AN2
0011: ADC input channel comes from AN3
0100: ADC input channel comes from Vsense
Other values: ADC input is floating

SADC1 Register

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

Bit 7~5 **SAINS2~SAINS0**: Internal ADC input channel selection bit
000: ADC input only comes from external pin
001: ADC input also comes from V_{BG}
010: ADC input also comes from 10×IS
011: GND
100: GND
101: The same as 000
110: The same as 000
111: The same as 000

Bit 4~3 **SAVRS1~SAVRS0**: ADC reference voltage selection
00: Reference voltage only comes from V_{REF}
01: Reference voltage only comes from V_{DD}
10: The same as 00
11: The same as 00
Note: When Select V_{REF} as ADC reference voltage, the pin share control bits (PAS2, PAS1) is (1, 0) to select VREF as input.

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

A/D Operation

The START bit is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the ADBZ bit in the SADC0 register will be cleared to zero and the analog to digital converter will be reset. It is the START bit that is used to control the overall start operation of the internal analog to digital converter.

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

Although the A/D clock source is determined by the system clock f_{SYS}, and by bits SACKS2~ SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended value of permissible A/D clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the SACKS2~SACKS0 bits should not be set to 000B or 11xB. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values.

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

The reference voltage supply to the A/D Converter can be supplied from either the internal ADC power or from an external reference sources supplied on pin VREF voltage. The desired selection is made using the SAVRS1~ SAVRS0 bits. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bits should be properly configured to disable other pin functions.

A/D Converter Input Signal

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

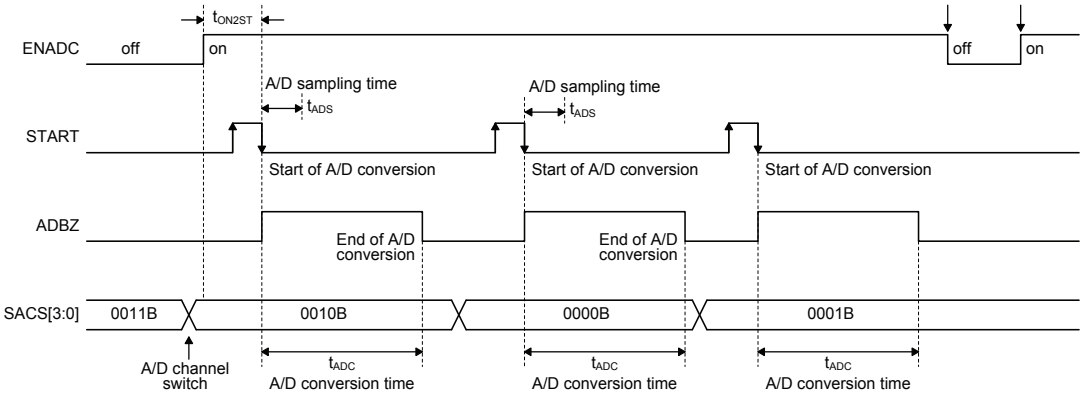
The A/D converter has its own reference voltage pin, VREF, however the reference voltage can also be supplied from the power supply pin, a choice which is made through the SAVRS[1:0] in the SADC1 register. The analog input values must not be allowed to exceed the value of V_{REF}.

Conversion Rate and Timing Diagram

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

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

However, there is a usage limitation on the next A/D conversion after the current conversion is complete. When the current A/D conversion is complete, the converted digital data will be stored in the A/D data register pair and then latched after half an A/D clock cycle. If the START bit is set to 1 in half an A/D clock cycle after the end of A/D conversion, the converted digital data stored in the A/D data register pair will be changed. Therefore, it is recommended to initiate the next A/D conversion after a certain period greater than half an A/D clock cycle at the end of current A/D conversion.



A/D Conversion Timing

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion frequency by SACKS2~ SACKS0
- Step 2
  Enable the ADC by set ENADC=1
- Step 3
  Select which pins will be configure as ADC analog inputs
- Step 4
  If input comes from I/O, set SAINS[2:0]=000 and then set SACS bit fields to corresponding PA input
  If input comes from internal input, set SAINS[2:0] to corresponding internal input source
- Step 5
  Select reference voltage comes from external V_{REF} or V_{DD} by SAVRS[1:0]
  Note: If select V_{REF} as reference voltage, (PAS2, PAS1) = (1, 0)
- Step 6
  Select ADC output data format by ADRFS
- Step 7
  If ADC 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 bits, ADE, must both set high in advance.
- Step 8
  The A/D converter procedure can now be initialized by set START from low to high and then low again
- Step 9
  If ADC is under conversion, ADBZ=1. After A/D conversion process is completed, the ADBZ flag will go low, and then output data can be read from SADOH and SADOL registers. If the ADC interrupt is enabled and the stack is not full, data can be acquired by interrupt service program. Another way to get the A/D output data is polling the ADBZ flag.

Programming Considerations

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

A/D Transfer Function

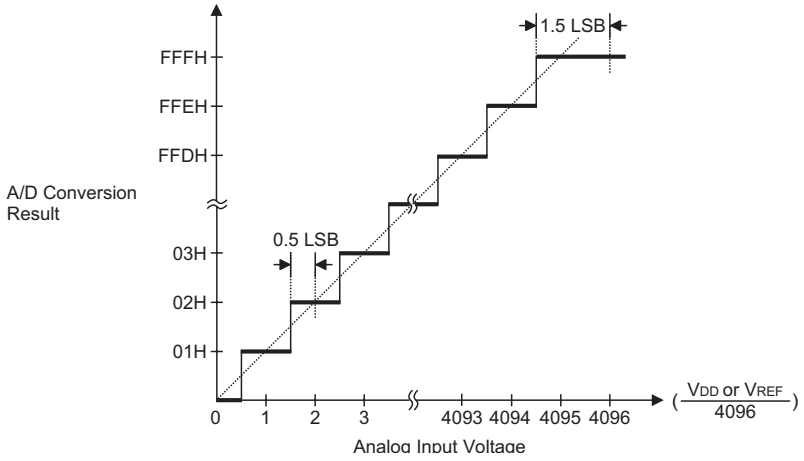
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the V_{DD} or V_{REF} voltage, this gives a single bit analog input value of V_{DD} or V_{REF} divided by 4096.

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

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

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

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



Ideal A/D Transfer Function

A/D Programming Examples

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

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

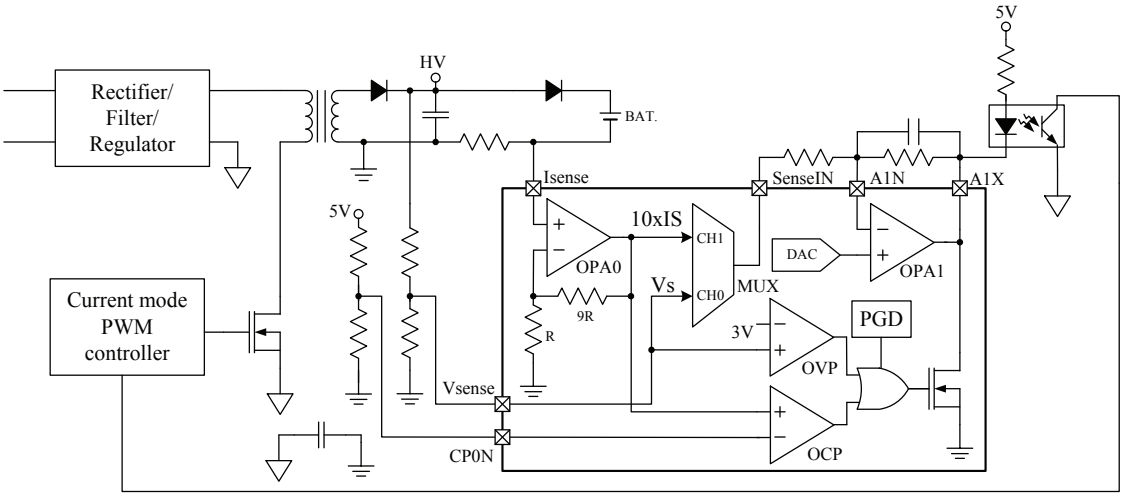
```
clr  ADE              ; disable ADC interrupt
mov  a,03H
mov  SADC1,a          ; select fSYS/8 as A/D clock and switch off the bandgap reference
                      ; voltage
mov  a,01h            ; setup PASR to configure pin AN0
mov  PASR,a
mov  a,00h
mov  SADC0,a          ; enable and connect AN0 channel to A/D converter
set  ENADC
:
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
mov  SADC1,a          ; select fSYS/8 as A/D clock and switch off the bandgap reference
                      ; voltage
mov  a,01h            ; setup PASR to configure pin AN0
mov  PASR,a
mov  a,00h
mov  SADC0,a          ; enable and connect AN0 channel to A/D converter
set  ENADC
Start_conversion:
clr  START            ; high pulse on START bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
clr  ADF              ; clear ADC interrupt request flag
set  ADE              ; enable ADC interrupt
set  EMI              ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov  acc_stack,a      ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a   ; save STATUS to user defined memory
:
:
mov  a,SADOL          ; read low byte conversion result value
mov  SADOL_buffer,a   ; save result to user defined register
mov  a,SADOH          ; read high byte conversion result value
mov  SADOH_buffer,a   ; save result to user defined register
:
:
EXIT_INT_ISR:
mov  a,status_stack
mov  STATUS,a         ; restore STATUS from user defined memory
mov  a,acc_stack      ; restore ACC from user defined memory
reti
```

Battery Charge Module

The device contains a battery charge module which consists of the battery charging constant current(CC) or constant voltage(CV) modes and the OVP/OCP functions. The constant current signal is from Isense pin while the constant voltage signal is from Vsense pin. The OVP or OCP circuitry uses respectively the external pins, named Vsense, Isense, and CP0N, to detect 10×IS voltage or Vs voltage and this output voltage is used to modify the duty of the Current mode PWM controller to control the charging current and charging voltage.



Battery Charge Module Structure

Note: 1. The input voltage range of Isense should be less than 0.36V at 5V.

2. When Vsense voltage is more than 3V and V_{DD} is more than or equal to 4.6V, OVP output is high, the OCVP interrupt occurs and A1X pin output low level.

3. When 10×IS voltage is more than CP0N and V_{DD} is more than or equal to 4.6V, OCP output is high, the OCVP interrupt occurs and A1X pin output low level.

Battery Charging Constant Current and Constant Voltage Modes

The battery charging current is measured using a resistor to produce a voltage which is input to the OPA0 via Isense pin. Then the Isense voltage is amplified 10 times by an OPA0 to produce 10×IS voltage. This voltage is input to a MUX channel 1 and ADC internal channel.

When MUX select CH1 from the 10×IS voltage and OPA1 positive voltage from DAC, if the 10×IS voltage is less than DAC voltage, A1X output signal is transmitted to current mode PWM controller via a photo-coupler to indirectly increase PWM duty cycle of the power MOS driving port of the current mode PWM control circuits.

The battery charging voltage is measured using external two resistors to produce a voltage which is input to the MUX via Vsense pin. This voltage is the same as Vs. When MUX channel select CH0, the Vs voltage will be sent out via SenseIN pin and external resister to the OPA1 negative input. If OPA1 positive from 8-bit DAC input, decide DAC value as decide battery charging voltage, because A1X of OPA1 output is transmitted Vs and DAC difference via a photo-coupler to current mode PWM controller, if Vs voltage is less than DAC voltage, A1X is transmitted to current mode PWM controller via a photo-coupler to indirectly increases PWM duty cycle of the power MOS driving port of the current mode PWM control circuits.

OCP and OVP Functions

The OCP function is used to monitor the battery charging current, which is converted to a voltage using a resistor, and the voltage signal is input to the OPA0 via the Isense pin. Then the Isense voltage is amplified 10 times by an OPA0 to produce 10×IS voltage. If 10×IS voltage is more than CP0N and PGD(Power Good Detection) detects that the device power supply is ready, V_{DD} is more than or equal to 4.6V, the OCVP interrupt will occur when corresponding interrupt is enabled and will force A1X output Low.

The OVP function is used to monitor the battery charging voltage, which is converted to a voltage using external two resistors, and the voltage is input to the Vsense pin. If the Vsense pin input voltage is more than 3V and PGD(Power Good Detection) detects that the device power supply is ready, V_{DD} is more than and equal to 4.6V, the OCVP interrupt will occur when corresponding interrupt is enabled and will force A1X output Low.

Battery Charge Module Registers

As the battery charge module is complex, the overall function is controlled by several registers and the corresponding register definitions are described in the accompanying sections.

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CHRGEN	CHGEN7	CHGEN6	CHGEN5	CHGEN4	CHGEN3	CHGEN2	CHGEN1	CHGEN0
DAC8	D7	D6	D5	D4	D3	D2	D1	D0
DACC	ENDAC	—	—	—	—	—	—	—
SENSW	—	—	MUXS5	MUXS4	MUXS3	MUXS2	MUXS1	MUXS0
A0VOS	A0FM	A0RSP	A0X	A0OF4	A0OF3	A0OF2	A0OF1	A0OF0
PGDR	—	—	—	—	—	—	—	PGDF

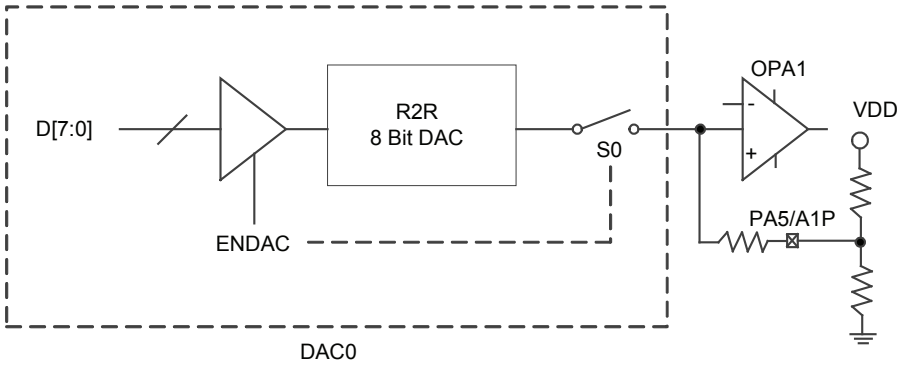
CHRGEN Register

Bit	7	6	5	4	3	2	1	0
Name	CHGEN7	CHGEN6	CHGEN5	CHGEN4	CHGEN3	CHGEN2	CHGEN1	CHGEN0
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 **CHGEN7~CHGEN0**: MUX, DAC and OPA0 related control registers modification
10101010: The related registers could be modified
Other values: Ignore registers modify
When these bits are changed to any other values except 10101010, the DAC8, DACC, SENSW and A0VOS registers cannot be modified.

Digital to Analog Converter

The battery charge module contains an 8-bit DAC. The DAC is used to set a reference charging current or reference charging voltage using DAC8 register.



Note: 1. The ENDAC has interlocking relationships with S0

When ENDAC=0, DAC disable and S0 off.

When ENDAC=1, DAC enable and S0 on.

2. OPA1 positive input voltage can be selected from external pin, which is named A1P.

DAC8 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	1	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**: DAC output control code

The DAC output voltage is calculated using the following equation:

8-bit DAC output voltage = $\left(\frac{DAC8[7:0]}{2^8}\right)\times V_{DD}$

DACC Register

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

Bit 7 **ENDAC**: DAC and S0 control

0: DAC disable&S0 off

1: DAC enable&S0 on

Bit 6~0 Unimplemented, read as "0"

SENSW Register

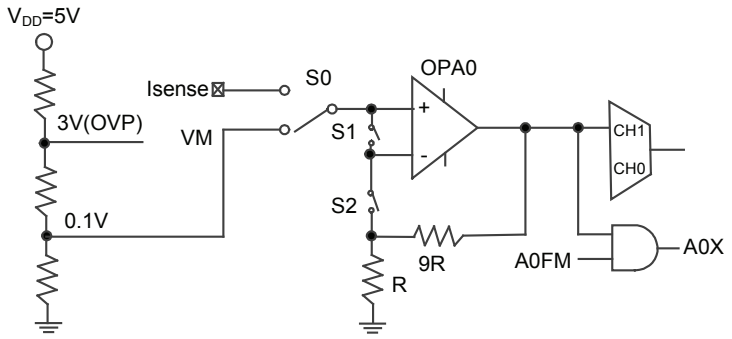
Bit	7	6	5	4	3	2	1	0
Name	—	—	MUXS5	MUXS4	MUXS3	MUXS2	MUXS1	MUXS0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	1	0	1	0	1

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **MUXS5~MUXS0**: MUX channel selection
010101: CH0 (Switch to Vsense pin input)
101010: CH1 (Switch to OPA0 input)
Other values: Keep the current switch state unchanged.
For example, when the MUX[5:0]=101010, switch to CH1, but when MUX[5:0] are changed to 111111, switch state, CH1, is unchanged, till when the MUX[5:0] is set to 010101, switch state will be changed to CH0.

Operational Amplifier 0

The battery charge module contains an operational amplifier 0, which only plays a role in the Battery charging constant current mode.



A0VOS Register

Bit	7	6	5	4	3	2	1	0
Name	A0FM	A0RSP	A0X	A0OF4	A0OF3	A0OF2	A0OF1	A0OF0
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **A0FM**: Operational amplifier mode or offset calibration mode
0: Operational amplifier mode (S1 off and S2 on)
1: Offset calibration mode (S1 on and S2 off)

Bit 6 **A0RSP**: Operational amplifier input voltage selection bit
0: Input voltage comes from Isense pin
1: Input voltage comes from internal VM reference voltage

Bit 5 **A0X**: Operational amplifier output; positive logic. This bit is read only.

Bit 4~0 **A0OF4~A0OF0**: Operational amplifier offset calibration data bits

OPA0 Functions

The OPA0 can operate together with the MUX, DAC and OPA1 as shown in the main functional blocks of the Battery charging circuit.

The OPA0 provides its input voltage offset to be adjustable by using common mode input to calibrate the offset.

The calibration steps are as following:

- 1. Set A0FM=1 to setup the offset calibration mode, here S1 on and S2 off.
- 2. Set A0RSP to select which input pin is to be used as the reference voltage – Isense pin or VM.
- 3. Adjust A0OF4~A0OF0 until the output status changes
- 4. Set A0FM = 0 to restore the normal mode.

Note: 1. When calibration, the device can detect the OPA output status by A0X bit.

2. VM voltage is 0.1V at V_{DD}=5V.

3. After OPA0 offset calibration, set the A0RSP bit by the actual applications.

- **PGDR Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **PGDF**: Power Good Detection ready flag
0: Detect V_{DD}<4.6V
1: Detect V_{DD}≥4.6V
Note: The device can detect the V_{DD} current status by PGDF bit.

Interrupts

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

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers depends upon the device chosen but fall into three categories. The first is the INTC0~INTC1 registers which setup the primary interrupt, the second is the MFI0 register which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INT Pin	INTE	INTF	—
Time Base	TBnE	TBnF	n=0 or 1
Multi-function	MF0E	MF0F	—
OCVP function	OCVPF	OCVPE	—
EEPROM	DEE	DEF	—
A/D Converter	ADE	ADF	—
TM	STMA0E	STMA0F	—
	STMP0E	STMP0F	

Interrupt Register Bit Naming Conventions

Interrupt Register Contents

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	MF0F	TB0F	INTF	MF0E	TB0E	INTE	EMI
INTC1	TB1F	ADF	DEF	OCVPF	TB1E	ADE	DEE	OCVPE
MFI0	—	—	STMA0F	STMP0F	—	—	STMA0E	STMP0E

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**: Define INT interrupt active edge
00: Disable Interrupt
01: Rising Edge Interrupt
10: Falling Edge Interrupt
11: Dual Edge Interrupt

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	MF0F	TB0F	INTF	MF0E	TB0E	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 **MF0F**: Multi-function 0 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **TB0F**: Time Base 0 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **INTF**: INT Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **MF0E**: Multi-function 0 Interrupt Control
0: Disable
1: Enable

Bit 2 **TB0E**: Time Base 0 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	TB1F	ADF	DEF	OCVPF	TB1E	ADE	DEE	OCVPE
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 **TB1F**: Time Base 1 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **ADF**: A/D Converter Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **DEF**: Data EEPROM Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **OCVPF**: OCVP Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **TB1E**: Time Base 1 Interrupt Control
0: Disable
1: Enable

Bit 2 **ADE**: A/D Converter Interrupt Control
0: Disable
1: Enable

Bit 1 **DEE**: Data EEPROM Interrupt Control
0: Disable
1: Enable

Bit 0 **OCVPE**: OCVP Interrupt Control
0: Disable
1: Enable

MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	STMA0F	STMP0F	—	—	STMA0E	STMP0E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5 **STMA0F**: STM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **STMP0F**: STM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **STMA0E**: STM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **STMP0E**: STM Comparator P match interrupt control
0: Disable
1: Enable

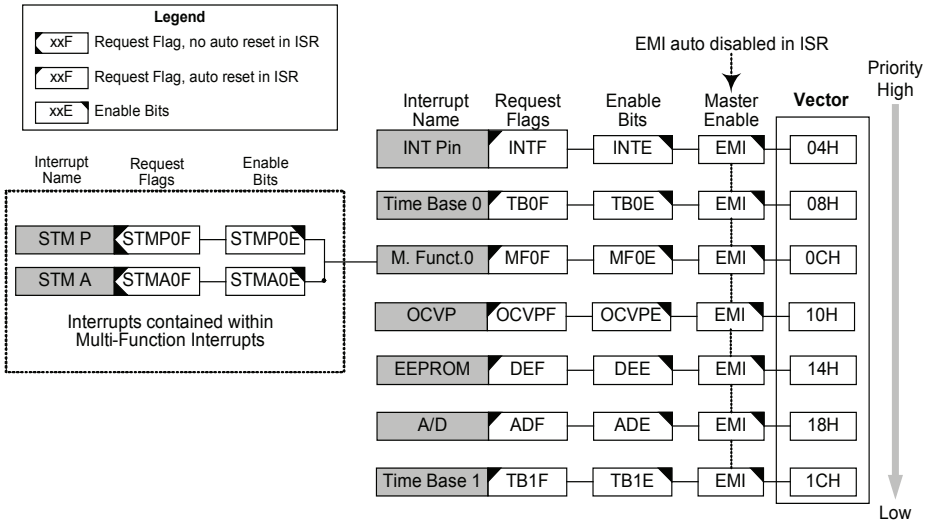
Interrupt Operation

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

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

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

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



Interrupt Structure

External Interrupt

The external 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 the 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 by setting the pin-shared registers. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that the 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.

Multi-function Interrupt

Within this device there is one Multi-function interrupt. Unlike the other independent interrupts, this interrupt has no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupts.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flag, MF0F are set. The Multi-function interrupt flag will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within the Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related Multi-Function request flag, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function Interrupt flag will be automatically reset when the interrupt is serviced, the request flag from the original source of the Multi-function interrupts, namely the TM Interrupts, will not be automatically reset and must be manually reset by the application program.

A/D Converter Interrupt

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

Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Their clock sources originate from the internal clock source f_{TB}. This f_{TB} input clock passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source that generates f_{TB}, which in turn controls the Time Base interrupt period, can originate from several different sources, as shown in the System Operating Mode section.

TBC Register

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

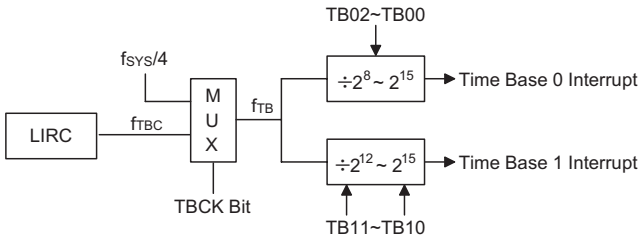
Bit 7 **TBON**: TB0 and TB1 Control bit
0: Disable
1: Enable

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

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

Bit 3 Unimplemented, read as "0"

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



Time Base Interrupt

EEPROM Interrupt

An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the respective EEPROM Interrupt vector, will take place. When the EEPROM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, and the EEPROM interrupt request flag, DEF, will also be automatically cleared.

TM Interrupts

The TM has two interrupts. All of the TM interrupts are contained within the Multi-function Interrupt. For the TM there are two interrupt request flag STMP0F and STMA0F and two enable bits STMP0E and STMA0E. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or comparator A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the respective TM Interrupt enable bit, and associated Multi-function interrupt enable bit, MF0E, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant TM Interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MF0F flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

OCVP Interrupt

The OCVP interrupt is controlled by the two internal comparators. An OCVP interrupt request will take place when the OCVP interrupt request flag, OCVPF, is set, a situation that will occur when the comparators output changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and OCVP interrupt enable bit, OCVPE, must first be set. When the interrupt is enabled, the stack is not full and the comparator input generates a comparator output transition, a subroutine call to the OCVP interrupt vector, will take place. When the OCVP Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, and the OCVP interrupt request flag, OCVPF, will also be automatically cleared.

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, a low power supply voltage or comparator input change may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

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

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MF0F, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

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

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

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

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

Low Voltage Detector – LVD

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

LVD Register

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

LVDC Register

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

Bit 7~6 Unimplemented, read as "0"

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

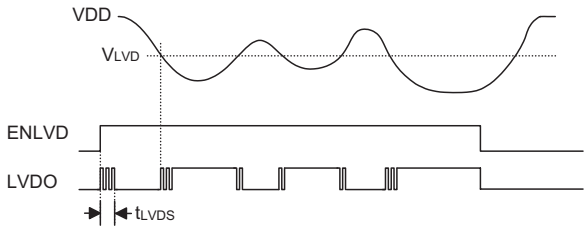
Bit 4 **ENLVD**: Low Voltage Detector Control
0: Disable
1: Enable

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

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

LVD Operation

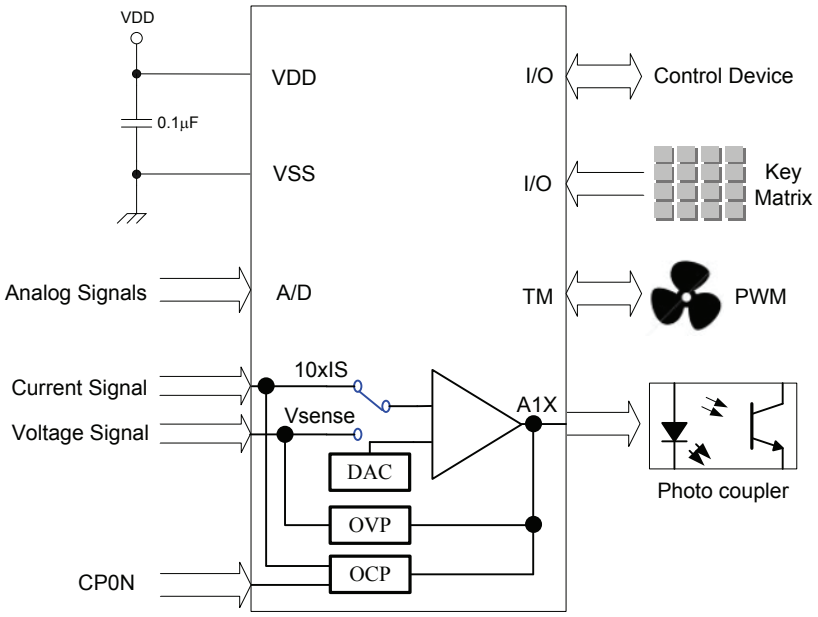
The Low Voltage Detector function operates by comparing the power supply voltage, VDD, with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.0V and 4.0V. When the power supply voltage, VDD, falls below this pre-determined value, the LVDO bit will be set high indicating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



LVD Operation

The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. When the device is in SLEEP mode the Low Voltage Detector will disable, even if the ENLVD bit is high.

Application Circuit



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

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

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

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the "CLR WDT1" and "CLR WDT2" instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both "CLR WDT1" and "CLR WDT2" instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − C

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

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

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

Operation [m] ← ACC − [m] − C

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

SDZ [m] Skip if decrement Data Memory is 0

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

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

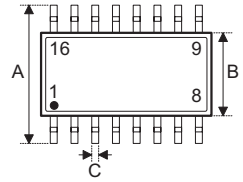
Package Information

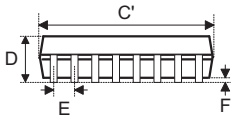
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

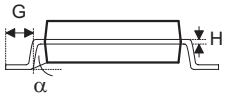
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Further Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- Packing Meterials Information
- Carton information

16-pin NSOP (150mil) Outline Dimensions







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

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

Copyright© 2016 by HOLTEK SEMICONDUCTOR INC.

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