



Enhanced A/D+LCD Type 8-Bit OTP MCU

HT46R0664

Revision: V.1.00 Date: August 12, 2011

www.holtek.com

Table of Contents

Features	5
CPU Features	5
Peripheral Features.....	5
General Description	6
Block Diagram	6
Pin Assignment	6
Pin Description	7
Absolute Maximum Ratings	9
D.C.Characteristics	10
A.C. Characteristics	11
LVD&LVR Electrical Characteristics	12
ADC Electrical Characteristics	12
Power-on Reset Characteristics	13
System Architecture	13
Clocking and Pipelining.....	13
Program Counter – PC.....	14
Stack	15
Arithmetic and Logic Unit – ALU	15
Program Memory	16
Structure.....	16
Special Vectors	16
Look-up Table.....	17
Table Program Example.....	17
Data Memory	19
Structure.....	19
General Purpose Data Memory	19
Display Memory	20
Special Purpose Data Memory	20
Special Function Register	21
Indirect Addressing Registers – IAR0, IAR1	21
Memory Pointers – MP0, MP1	21
Accumulator – ACC.....	22
Bank Pointer – BP.....	22
Program Counter Low Register – PCL.....	22
Look-up Table Registers – TBLP, TBLH.....	23
Status Register – STATUS	23
System Control Registers – CTRL0, CTRL1, CTRL2, CTRL3, CTRL4	24
Oscillator	27
Oscillator Overview	27

System Clock Configurations	27
External Crystal/Resonator Oscillator – HXT	27
External RC Oscillator – ERC	28
Internal RC Oscillator – HIRC	28
External 32768Hz Crystal Oscillator – LXT	28
LXT Oscillator Low Power Function	29
Internal Low Speed Oscillator – LIRC	29
Operating Modes	30
Mode Types and Selection	30
Operating Mode Control	30
Mode Switching	31
Standby Current Considerations	31
Wake-up	31
Watchdog Timer	33
Watchdog Timer Clock Source	33
Watchdog Timer Control Register	33
Watchdog Timer Operation	34
Reset and Initialisation	36
Reset Functions	36
Reset Initial Conditions	39
Input/Output Ports	41
Pull-high Resistors	41
Port A Wake-up	41
I/O Port Control Registers	42
Pin-shared Functions	42
Pin Remapping Configuration	43
I/O Pin Structures	43
Programming Considerations	44
Timer/Event Counter	45
Configuring the Timer/Event Counter Input Clock Source	45
Timer Registers – TMR0, TMR1	46
Timer Control Registers – TMR0C, TMR1C	46
Timer Mode	48
Event Counter Mode	49
Pulse Width Capture Mode	49
Prescaler	50
PFD Function	51
I/O Interfacing	51
Programming Considerations	51
Timer Program Example	52
Pulse Width Modulator	53
PWM Operation	53
6+2 PWM Mode	54

7+1 PWM Mode	54
PWM Output Control	55
PWM Programming Example.....	55
Analog to Digital Converter	56
A/D Overview	56
A/D Converter Data Registers – ADRL, ADRLH	56
A/D Converter Control Registers – ADCR, ACSR, ANCSR0, ANCSR1	57
A/D Operation	59
A/D Input Pins	60
Summary of A/D Conversion Steps.....	60
A/D Conversion Timing	61
Programming Considerations.....	61
A/D Transfer Function	61
A/D Programming Example.....	62
Buzzer	64
Interrupts	65
Interrupt Registers.....	65
Interrupt Operation	67
Interrupt Priority.....	69
Multi-function Interrupt	69
A/D Converter Interrupt.....	70
Timer/Event Counter Interrupt.....	70
Time Base Interrupts	70
Interrupt Wake-up Function.....	71
Programming Considerations.....	71
LCD Function	72
Display Memory	73
LCD Registers.....	73
LCD Clock Source.....	75
LCD Driver Output.....	75
LCD Voltage Source and Biasing.....	75
Low Voltage Detector – LVD	76
LVD Register	76
LVD Operation.....	77
Configuration Options.....	77
Application Circuits.....	78
Instruction Set Summary	79
Table Conventions.....	79
Instruction Definition.....	81
Package Information	91
44-pin QFP (10mm×10mm) Outline Dimensions	91

Features

CPU Features

- Operating voltage:
f_{sys}=4MHz: 2.2V–5.5V
f_{sys}=8MHz: 3.0V–5.5V
f_{sys}=12MHz: 4.5V–5.5V
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types:
External Crystal -- HXT
External RC -- ERC
External 32768Hz Crystal -- LXT
Internal RC -- HIRC
Internal 32kHz RC -- LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- Fully integrated internal 4MHz, 8MHz and 12MHz oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- Up to 6-level stack
- Bit manipulation instruction

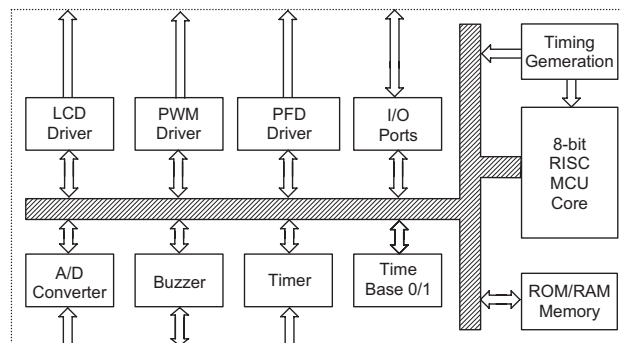
Peripheral Features

- Up to 42 bidirectional I/O lines
- Up to 12 channel 12-bit ADC
- Up to 2 channel 8-bit PWM
- Data Memory: 224×8
- Program Memory: 4k×16
- Watchdog Timer function
- 4 pin-shared external interrupts
- Up to two 8-bit programmable Timer/Event Counter with overflow interrupt and prescaler
- Low voltage reset function
- Low voltage detect function
- Time Base functions
- Buzzer function
- Package: 44QFP

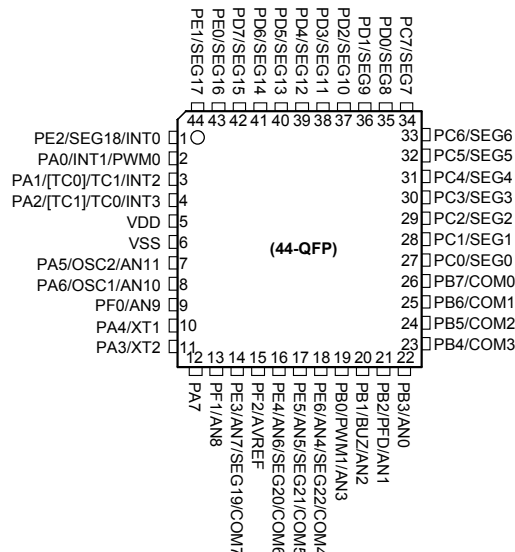
General Description

The Enhanced A/D Type with LCD is a 8-bit high performance, RISC architecture microcontroller specifically designed for applications that interface directly to analog signals and which require an LCD interface. The device includes an integrated multi-channel Analog to Digital Converter, Pulse Width Modulation outputs and an LCD driver. The benefits of integrated A/D, LCD, and PWM functions, in addition to low power consumption, high performance, I/O flexibility, timer functions, oscillator options, power down and wake-up functions, watchdog timer and low voltage reset, combine to provide device with a huge range of functional options while still main taining a high level of cost effectiveness. The fully integrated system oscillator HIRC, which requires no external components and which has three frequency selections, opens up a huge range of new application possibilities for the device, some of which may include industrial control, consumer products, household appliances subsystem controllers, etc.

Block Diagram



Pin Assignment



Note: 1. Bracketed pin names indicate non-default pinout remapping locations.

2. If the pin-shared pin functions have multiple outputs simultaneously, its pin names at the right side of the “/” sign can be used for higher priority.

Pin Description

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT1/PWM0	PA0	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT1	—	ST	—	External interrupt input
	PWM0	CTRL0	—	CMOS	PWM output
PA1/INT2/TC1/[TC0]	PA1	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT2	—	ST	—	External interrupt input
	TC1	—	ST	—	External Timer 1 clock input
	TC0	—	ST	—	External Timer 0 clock input
PA2/INT3/TC0/[TC1]	PA2	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT3	—	ST	—	External interrupt input
	TC0	—	ST	—	External Timer 0 clock input
	TC1	—	ST	—	External Timer 1 clock input
PA3/XT2	PA3	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	XT2	CO	—	LXT	Low frequency crystal pin
PA4/XT1	PA4	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	XT1	CO	LXT	—	Low frequency crystal pin
PA5/OSC2/AN11	PA5	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OSC2	CO	—	OSC	Oscillator pin
	AN11	ANCSR1	AN	—	A/D channel 11
PA6/OSC1/AN10	PA6	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OSC1	CO	OSC	—	Oscillator pin
	AN10	ANCSR1	AN	—	A/D channel 10
PA7	PA7	PAWK	ST	NMOS	General purpose I/O. Register enabled wake-up.
PB0/PWM1/AN3	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM1	CTRL0	—	CMOS	PWM1 output
	AN3	ANCSR0	AN	—	A/D channel 3
PB1/BUZ/AN2	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	BUZ	CTRL2	—	CMOS	Buzzer Output
	AN2	ANCSR0	AN	—	A/D channel 2
PB2/PFD/AN1	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PFD	CTRL0	—	CMOS	PFD output
	AN1	ANCSR0	AN	—	A/D channel 1
PB3/AN0	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN0	ANCSR0	AN	—	A/D channel 0
PB4/COM3	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	COM3	LCDO	—	COM	LCD COM port
PB5/COM2	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	COM2	LCDO	—	COM	LCD COM port
PB6/COM1	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	COM1	LCDO	—	COM	LCD COM port
PB7/COM0	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	COM0	LCDO	—	COM	LCD COM port

Pin Name	Function	OPT	I/T	O/T	Description
PC0/SEG0	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG0	LCDO	—	CMOS	LCD Segment Port
PC1/SEG1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG1	LCDO	—	CMOS	LCD Segment Port
PC2/SEG2	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG2	LCDO	—	CMOS	LCD Segment Port
PC3/SEG3	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG3	LCDO	—	CMOS	LCD Segment Port
PC4/SEG4	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG4	LCDO	—	CMOS	LCD Segment Port
PC5/SEG5	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG5	LCDO	—	CMOS	LCD Segment Port
PC6/SEG6	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG6	LCDO	—	CMOS	LCD Segment Port
PC7/SEG7	PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG7	LCDO	—	CMOS	LCD Segment Port
PD0/SEG8	PD0	PDPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG8	LCDO	—	CMOS	LCD Segment Port
PD1/SEG9	PD1	PDPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG9	LCDO	—	CMOS	LCD Segment Port
PD2/SEG10	PD2	PDPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG10	LCDO	—	CMOS	LCD Segment Port
PD3/SEG11	PD3	PDPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG11	LCDO	—	CMOS	LCD Segment Port
PD4/SEG12	PD4	PDPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG12	LCDO	—	CMOS	LCD Segment Port
PD5/SEG13	PD5	PDPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG13	LCDO	—	CMOS	LCD Segment Port
PD6/SEG14	PD6	PDPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG14	LCDO	—	CMOS	LCD Segment Port
PD7/SEG15	PD7	PDPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG15	LCDO	—	CMOS	LCD Segment Port
PE0/SEG16	PE0	PEPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG16	LCDO	—	CMOS	LCD Segment Port
PE1/SEG17	PE1	PEPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG17	LCDO	—	CMOS	LCD Segment Port
PE2/INT0/ SEG18	PE2	PEPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	INT0	—	ST	—	External interrupt input
	SEG18	LCDO	—	CMOS	LCD Segment Port
PE3/AN7/ SEG19/COM7	PE3	PEPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN7	ANCSR0	AN	—	A/D channel 7
	SEG19	LCDO	—	CMOS	LCD Segment Port
	COM7	LCDO	—	COM	LCD COM port
PE4/AN6/ SEG20/COM6	PE4	PEPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN6	ANCSR0	AN	—	A/D channel 6
	SEG20	LCDO	—	CMOS	LCD Segment Port
	COM6	LCDO	—	COM	LCD COM port

Pin Name	Function	OPT	I/T	O/T	Description
PE5/AN5/ SEG21/COM5	PE5	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN5	ANCSR0	AN	—	A/D channel 5
	SEG21	LCDO	—	CMOS	LCD Segment Port
	COM5	LCDO	—	COM	LCD COM port
PE6/AN4/ SEG22/COM4	PE6	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN4	ANCSR0	AN	—	A/D channel 4
	SEG22	LCDO	—	CMOS	LCD Segment Port
	COM4	LCDO	—	COM	LCD COM Port
PF0/AN9	PF0	PFPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN9	ANCSR1	AN	—	A/D channel 9
PF1/AN8	PF1	PFPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN8	ANCSR1	AN	—	A/D channel 8
PF2/AVREF	PF2	PFPUP	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AVREF	ACSR	AN	—	ADC Reference Input
VDD	VDD	—	PWR	—	Power Supply
VSS	VSS	—	PWR	—	Ground

Legend: I/T: Input type; O/T: Output type

OPT: Optional by configuration option (CO) or register option

PWR: Power; CO: Configuration option

ST: Schmitt Trigger input; CMOS: CMOS output; AN: Analog input

COM: LCD COM

NMOS: NMOS output

OSC: High frequency crystal oscillator

LXT: Low frequency crystal oscillator

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.....	$-50^{\circ}C$ to $125^{\circ}C$
Operating Temperature.....	$-40^{\circ}C$ to $85^{\circ}C$
I_{OH} Total	-100mA
I_{OL} Total	100mA
Total Power Dissipation	500mW

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

D.C.Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{DD}	Operating Voltage	—	f _{sys} =4MHz	2.2	—	5.5	V
			f _{sys} =8MHz	3.0	—	5.5	V
			f _{sys} =12MHz	4.5	—	5.5	V
I _{DD1}	Operating Current (HXT, HIRC, ERC)	3V	No load, f _{sys} =4MHz, ADC off	—	1	2	mA
		5V		—	2.5	5	mA
I _{DD2}	Operating Current (HXT, HIRC, ERC)	5V	ADC off	—	4	8	mA
I _{DD3}	Operating Current (HXT, HIRC, ERC)	5V	No load, f _{sys} =12MHz, ADC off	—	6	12	mA
I _{DD4}	Operating Current (HIRC+LXT, slow mode)	3V	No load, f _{sys} =32768Hz, ADC off	—	20	30	μA
		5V		—	40	60	μA
I _{STB1}	Standby Current (LIRC on, LXT off)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	μA
I _{STB2}	Standby Current (LIRC off, LXT off)	3V	No load, system HALT	—	—	1	μA
		5V		—	—	2	μA
I _{STB3}	Standby Current (LIRC off, LXT on)	3V	No load, system HALT, LXT slowly start-up	—	—	5	μA
		5V		—	—	10	μA
V _{IL1}	Input Low Voltage for PA,PB,PC,PD,PE,PF,TC0,TC1,INT	5V	—	0	—	1.5	V
		—		0	—	0.2V _{DD}	V
V _{IH1}	Input High Voltage for PA,PB,PC,PD,PE,PF,TC0,TC1,INT	5V	—	3.5	—	5	V
		—		0.8V _{DD}	—	V _{DD}	V
I _{OL1}	I/O Port Sink Current (PA,PB,PC,PD,PE,PF)	3V	V _{OL} =0.1V _{DD}	4	8	—	mA
		5V		10	20	—	mA
I _{OH1}	I/O Port, Source Current (PA,PB,PC,PD,PE,PF)	3V	V _{OH} =0.9V _{DD}	-2	-4	—	mA
		5V		-5	-10	—	mA
I _{OL2}	PA7 Sink Current	5V	V _{OL} =0.1V _{DD}	2	3	—	mA
I _{LCD_BIAS}	R-Type LCD bias Current	5V	LCDC.RSEL[1:0]=00, 1/4 bias	-20%	8.33	20%	μA
			LCDC.RSEL[1:0]=01, 1/4 bias	-20%	16.66	20%	
			LCDC.RSEL[1:0]=10, 1/4 bias	-20%	50	20%	
			LCDC.RSEL[1:0]=11, 1/4 bias	-20%	166.66	20%	
I _{LCD_OL}	LCD Common and Segment Current	3V	V _{OL} =0.1V _{DD}	210	420	—	μA
		5V		350	700	—	
I _{LCD_OH}	LCD Common and Segment Current	3V	V _{OH} =0.9V _{DD}	-80	-160	—	μA
		5V		-180	-360	—	
R _{PH}	Pull-high Resistance of I/O Ports	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}	Condition				
f _{SYS}	System clock (HXT, HIRC, ERC)	—	2.2~5.5V	0.4	—	4	MHz
			3.0~5.5V	0.4	—	8	MHz
			4.5~5.5V	0.4	—	12	MHz
f _{HIRC}	System clock (HIRC)	2.2~5.5V	—	-10%	4	+10%	MHz
		2.2~5.5V	—	-10%	8	+10%	MHz
		2.2~5.5V	—	-10%	12	+10%	MHz
		3/5V	—	-2%	4	+2%	MHz
		3/5V	—	-2%	8	+2%	MHz
		5V	—	-2%	12	+2%	MHz
		3/5V	—	-5%	4	+5%	MHz
		3/5V	—	-5%	8	+5%	MHz
		5V	Ta=0~70°C	-5%	12	+5%	MHz
		2.2~3.6V	Ta=0~70°C	-8%	4	+8%	MHz
		3.0~5.5V	Ta=0~70°C	-8%	4	+8%	MHz
		3.0~5.5V	Ta=0~70°C	-8%	8	+8%	MHz
		4.5~5.5V	Ta=0~70°C	-8%	12	+8%	MHz
		2.2~3.6V	Ta=-40°C~85°C	-12%	4	+12%	MHz
		3.0~5.5V	Ta=-40°C~85°C	-12%	4	+12%	MHz
		3.0~5.5V	Ta=-40°C~85°C	-12%	8	+12%	MHz
		4.5~5.5V	Ta=-40°C~85°C	-12%	12	+12%	MHz
f _{ERC}	System clock (ERC)	3/5V	R=120kΩ, Ta=-40°C~85°C	-10%	4	+10%	MHz
		5V	R=120kΩ	-2%	4	+2%	MHz
		5V	Ta=0~70°C, R=120kΩ	-5%	4	+5%	MHz
		5V	Ta=-40°C~85°C, R=120kΩ	-7%	4	+7%	MHz
		2.2~5.5V	Ta=-40°C~85°C, R=120kΩ	-11%	4	+11%	MHz
f _{LIRC}	Low Speed Internal RC Oscillator Clock (LIRC)	5V	—	-10%	32	+10%	kHz
		2.2V~5.5V	Ta=-40°C~85°C	-50%	32	+60%	kHz
t _{TIMER}	TCn Input Pin Minimum Pulse Width	—	—	0.3	1	1.5	μs
t _{SST}	System start-up timer period (wake-up from HALT, f _{SYS} off at HALT state)	—	f _{SYS} =HXT or LXT OSC	128	—	—	t _{sys}
			f _{SYS} =ERC or HIRC OSC	2	—	—	
	System Start-up Timer Period (Wake-up from Power down f _{SYS} on at Power down state)	—	—	2	—	—	
t _{INT}	Interrupt Minimum Pulse Width	—	—	1	3.3	5	μs
t _{RSTD}	System Reset Delay Time (Power On Reset)	—	—	25	50	100	ms
	System Reset Delay Time (Any Reset except Power On Reset)	—	—	8.3	16.7	33.3	ms

Note: t_{sys}=1/f_{SYS}

LVD&LVR Electrical Characteristics

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{LVR1}	Low Voltage Reset Voltage	—	LVR Enable, 2.1V option	-5%	2.1	+5%	V
V _{LVR2}			LVR Enable, 2.55V option		2.55		V
V _{LVR3}			LVR Enable, 3.15V option		3.15		V
V _{LVR4}			LVR Enable, 3.8V option		3.8		V
V _{LVD1}	Low Voltage Detector Voltage	—	LV _{DEN} =1, V _{LVD} =2.0V	-5%	2.0	+5%	V
V _{LVD2}			LV _{DEN} =1, V _{LVD} =2.2V		2.2		V
V _{LVD3}			LV _{DEN} =1, V _{LVD} =2.4V		2.4		V
V _{LVD4}			LV _{DEN} =1, V _{LVD} =2.7V		2.7		V
V _{LVD5}			LV _{DEN} =1, V _{LVD} =3.0V		3.0		V
V _{LVD6}			LV _{DEN} =1, V _{LVD} =3.3V		3.3		V
V _{LVD7}			LV _{DEN} =1, V _{LVD} =3.6V		3.6		V
V _{LVD8}			LV _{DEN} =1, V _{LVD} =4.0V		4.0		V
I _{LVR}	Additional Power Consumption if LVR is used	3V	LVR enabled	—	30	45	μA
		5V		—	60	90	μA
I _{LVD}	Additional Power Consumption if LVD is used	3V	LVD disable→LVD enable (LVR enable)	—	30	45	μA
		5V		—	60	90	μA
t _{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t _{LVD}	Low Voltage Width to Interrupt	—	—	20	45	90	μs
t _{LVDS}	LVDO stable time	—	For LVR enable, LVD off→on	15	—	—	μs
t _{SRESET}	Software Reset Width to Reset	—	—	45	90	120	μs

ADC Electrical Characteristics

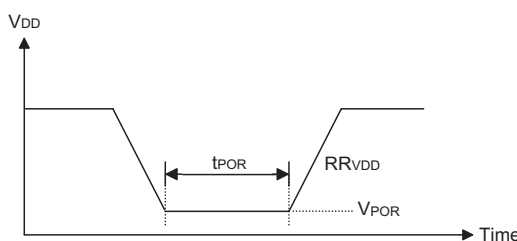
T_a=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
DNL	A/D Differential Non-linearity	3V 5V	AV _{REF} =V _{DD} , t _{AD} =0.5μs	-2	—	+2	LSB
INL	ADC Integral Non-linearity	3V 5V 5V					
I _{ADC}	Additional Power Consumption if A/D Converter is Used	3V	No load (t _{AD} =0.5μs)	—	0.5	—	mA
		5V		—	0.6	—	mA
t _{AD}	A/D Converter Clock Period	2.7V~5.5V	—	0.5	—	10	μs
t _{ADC}	A/D Conversion Time (Include Sample and Hold Time)	2.7V~5.5V	12-bit ADC	—	16	—	t _{ADC}
t _{ON2ST}	A/D Converter On-to-Start Time	—	—	2	—	—	μs

Power-on Reset 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} Raising 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



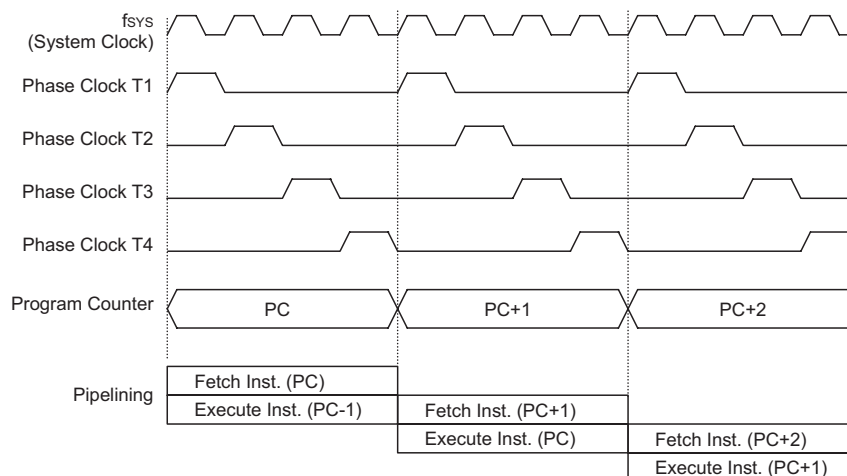
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The range of device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions. 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.

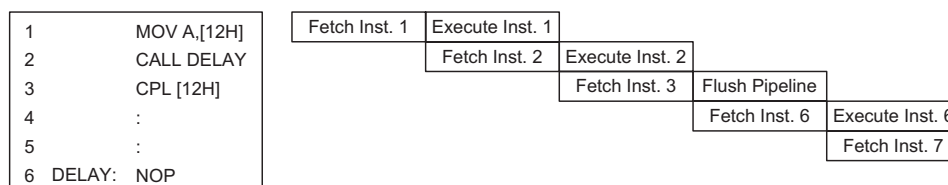
Clocking and Pipelining

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

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



System Clocking and Pipelining



Instruction Fetching

Program Counter – PC

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

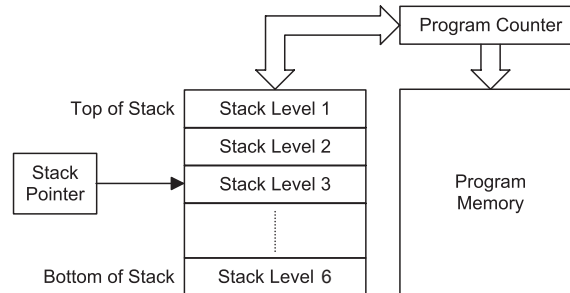
When executing instructions requiring jumping to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc, the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition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

Program Counter	
High Byte of Program	Low Byte of Program
PC11~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 in the present page of memory, which have 256 locations. When such program jumps are executed it should also be noted that a dummy cycle will be inserted. The lower byte of the Program Counter is fully accessible under program control. Manipulating the PCL might 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 organized into 6 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.



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

Arithmetic and Logic Unit – ALU

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

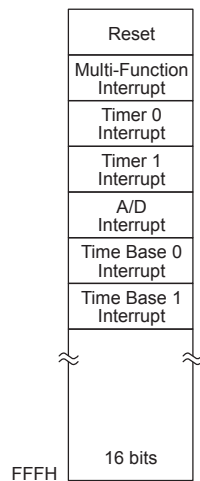
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement INCA, INC, DECA, DEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET,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

Program Memory

The Program Memory is the location where the user code or program is stored. The device is supplied with One-Time Programmable, OTP, memory where users can program their application code into the device. By using the appropriate programming tools, OTP device offers users the flexibility to freely develop their applications which may be useful during debug or for products requiring frequent upgrades or program changes.

Structure

The Program Memory has a capacity of 4k×16. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries information. Table data, which can be setup in any location within the Program Memory, is addressed by separate table pointer registers.



Program Memory Structure

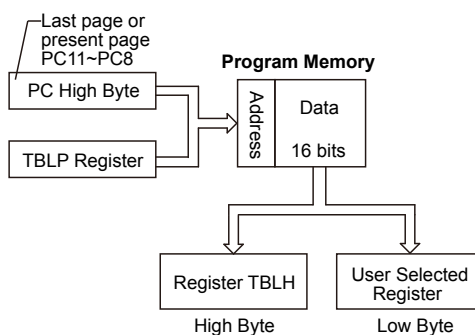
Special Vectors

Within the Program Memory, certain locations are reserved for special usage such as reset and interrupts.

- **Reset Vector**
This vector is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.
- **External interrupt vector**
This vector is used by the external interrupt. If the external interrupt pin on the device receives an edge transition, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full. The external interrupt active edge transition type, whether high to low, low to high or both is specified in the INTEG register.
- **Timer/Event 0/1 counter interrupt vector**
This internal vector is used by the Timer/Event Counters. If a Timer/Event Counter overflow occurs, the program will jump to its respective location and begin execution if the associated Timer/Event Counter interrupt is enabled and the stack is not full.
- **Time base 0/1 interrupt vector**
This internal vector is used by the internal Time Base 0/1. If a Time Base overflow occurs, the program will jump to this location and begin execution if the Time Base counter interrupt is enabled and the stack is not full.

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. These registers define the total address of the look-up table. After setting up the table pointer, the table data can be retrieved from the Program Memory using the “TABRD [m]” instructions, respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as “0”. The accompanying diagram illustrates the addressing data flow of the look-up table.



Instruction	Table Location Bits											
	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

PC11~PC8: Current Program Counter bits

@7~@0: Table Pointer TBLP bits

b11~b0: Table address location bits

Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is “F00H” which refers to the start address of the last page within the 4K Program Memory of the microcontroller.

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 “F06H” 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 “TABRDL [m]” instruction is executed. Because the TBLH register is a read-only register and cannot be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use the table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of 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 table pointer - note that this address
                    ; is referenced
mov tblp, a         ; to the last page or present page
:
:
tabrdl tempreg1     ; transfers value in table referenced by table pointer
                    ; to tempreg1 data at prog.memory address "F06H"
                    ; transferred to tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrdl tempreg2     ; transfers value in table referenced by table pointer
                    ; to tempreg2 data at prog.memory address "F05H"
                    ; transferred to tempreg2 and TBLH in this example the
                    ; data "1AH" is transferred to tempreg1 and data "0FH"
                    ; to register tempreg2 the value "00H" will be
                    ; transferred to the high byte register TBLH
:
:
org 0F00h           ; sets initial address of last page
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

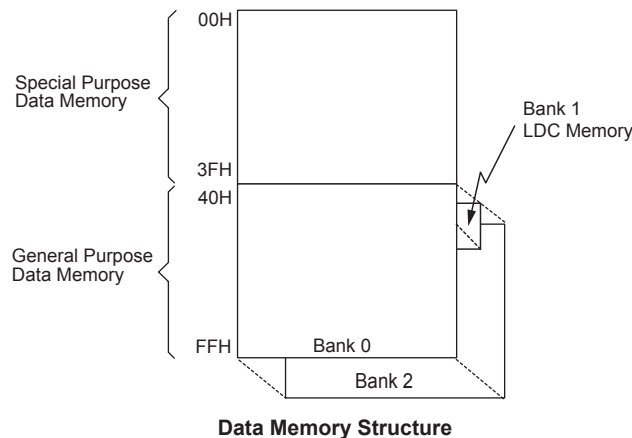
Data Memory

The Data Memory is an 8-bit wide RAM internal memory and is the location where temporary information is stored.

Divided into three sections, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control. The third area is reserved for the LCD Memory. This special area of Data Memory is mapped directly to the LCD display so data written into this memory area will directly affect the displayed data. The addresses of the LCD Memory area overlap those in the General Purpose Data Memory area. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value.

Structure

The Data Memory is subdivided into three banks, all of which are implemented in 8-bit wide RAM. The Data Memory located in Bank 0 is subdivided into two sections, the Special Purpose Data Memory and the General Purpose Data Memory. The start address of the Data Memory is the address "00H". The LCD Memory is mapped into Bank 1. Bank 2 contains only General Purpose Data Memory. As the Special Purpose Data Memory registers are mapped into all bank areas, they can subsequently be accessed from any bank location.



General Purpose Data Memory

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user program for both read and write operations. By using the "SET [m].i" and "CLR [m].i" 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. For this device, the General Purpose Data Memory, in addition to being located in Bank 0, is also stored in Bank 2.


Bank0	Bank1 (LCD RAM)	Bank2
40H~FFH	40H~56H	40H~5FH

Display Memory

The data to be displayed on the LCD display is stored in an area of fully accessible Data Memory. By writing to this area of RAM, the display output can be directly controlled by the application program. As this Memory exists in Bank 1, but have addresses which map into the General Purpose Data Memory, it is necessary to first ensure that the Bank Pointer is set to the value “01H” before accessing the Display Memory. The Display Memory can only be accessed indirectly using the Memory Pointer MP1 and the indirect addressing register IAR1. When the Bank Pointer is set to Bank 1 to access the Display Memory, if any addresses with a value less than “40H” are read, the Special Purpose Memory in Bank 0 will be accessed. Also, if the Bank Pointer is set to Bank 1, if any addresses higher than the last address in Bank 1 are read, then a value of “00H” will be returned.

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 writable 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”.

00H	IAR0	1EH	INTC1
01H	MP0	1FH	PWM0
02H	IAR1	20H	ADRL
03H	MP1	21H	ADRH
04H	BP	22H	ADCR
05H	ACC	23H	ACSR
06H	PCL	24H	MFIC
07H	TBLP	25H	PD
08H	TBLH	26H	PDC
09H	CTRL4	27H	PDPU
0AH	STATUS	28H	PE
0BH	INTC0	29H	PEC
0CH	TMR0	2AH	PEPU
0DH	TMR0C	2BH	PF
0EH	TMR1	2CH	PFC
0FH	TMR1C	2DH	PFPU
10H	PA	2EH	INTEG
11H	PAC	2FH	CTRL3
12H	PAPU	30H	LCDC
13H	PAWK	31H	CTRL2
14H	PB	32H	ANCSR0
15H	PBC	33H	ANCSR1
16H	PBPU	34H	WDTC
17H	PC	35H	LVDC
18H	PCC	36H	LVRC
19H	PCPU		
1AH	CTRL0		
1BH	CTRL1		
1CH	LCDO		
1DH	PWM1	3FH	

Special Purpose Data Memory

Special Function Register

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, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation is using 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 Bank0 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 indirectly address and track data. MP0 can only be used to indirectly address data in Bank 0 while MP1 can be used to address data in Bank 0, Bank 1 and Bank 2. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, it is the address specified by the related Memory Pointer.

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.

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.

Bank Pointer – BP

The Data Memory is divided into three Banks, known as Bank 0, Bank1 and Bank 2. A Bank Pointer, which is bit 0~1 of the Bank Pointer register is used to select the required Data Memory bank. Only data in Bank 0 can be directly addressed as data in Bank 1 and Bank2 must be indirectly addressed using Memory Pointer MP1 and Indirect Addressing Register IAR1. Using Memory Pointer MP0 and Indirect Addressing Register IAR0 will always access data from Bank 0, irrespective of the value of the Bank Pointer. Memory Pointer MP1 and Indirect Addressing Register IAR1 can indirectly address data in Bank 0, Bank 1, or Bank 2 depending upon the value of the Bank Pointer. The Data Memory is initialised to Bank 0 after a reset, except for the WDT time-out reset in the Idle/Sleep Mode, in which case, the Data Memory bank remains unaffected. It should be noted that Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from Bank 0, Bank1 or Bank 2. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer.

BP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	DMBP1	DMBP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **DMBP1, DMBP0:** Data memory bank point
 00: bank 0
 01: bank 1
 10: bank 2
 11: undefined

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

Status Register – STATUS

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

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

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

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. Note that bits 0~3 of the STATUS register are both readable and writeable bits.

STATUS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R/W	R/W	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 borrowing does not take place during a subtraction operation C is also affected by a rotate through carry instruction

System Control Registers – CTRL0, CTRL1, CTRL2, CTRL3, CTRL4

These registers are used to provide control over various internal functions. Some of these include the PFD control, PWM control, certain system clock options, the LXT oscillator low power control, buzzer function control, LCD driver clock selection, Timer clock source selection, Time Base functions division ratio, and the LXT oscillator enable control.

CTRL0 Register

Bit	7	6	5	4	3	2	1	0
Name	PCFG	PFDCS	PWMSEL	PWMC1	PWMC0	PFDC	—	CLKMOD
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 **PCFG:** PA2~PA1 pin-shared function Pin Remapping Control
 0: TC1/TC0 pin-shared with PA1/PA2
 1: TC1/TC0 pin-shared with PA2/PA1
- Bit 6 **PFDCS:** PFD clock source
 0: timer0
 1: timer1
- Bit 5 **PWMSEL:** PWM type selection
 0: 6+2
 1: 7+1
 This bit can be clear to “0”, but can not set to “1”.
- Bit 4 **PWMC1:** I/O or PWM1
 0: I/O
 1: PWM1
- Bit 3 **PWMC0:** I/O or PWM0
 0: I/O
 1: PWM0
- Bit 2 **PFDC:** I/O or PFD
 0: I/O
 1: PFD
- Bit 1 Unimplemented, read as “0”
- Bit 0 **CLKMOD:** System clock mode selection
 0: high speed – HIRC used as system clock
 1: low speed – LXT used as system clock, HIRC oscillator stopped

CTRL1 Register

Bit	7	6	5	4	3	2	1	0
Name	T0S1	T0S0	TB01	TB00	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

- Bit 7~6 **T0S1, T0S0**: Prescaler/TMR0 clock source
 00: $f_{TP}=f_{SYS}$
 Prescaler clock source is f_{SYS}
 TMR0 clock source is come from the output clock of Prescaler
 01: $f_{TP}=LXT$
 Prescaler clock source is LXT
 TMR0 clock source is come from the output clock of Prescaler
 10: $f_{TP}=PFD0$
 Prescaler clock source is PFD0
 TMR0 clock source is come from f_{SYS}
 11: undefined
 Note: If PWM0C or PWM1C is enabled, the clock source of Prescaler is only selected from f_{SYS} or PFD0 by assigning T0S1.
- Bit 5~4 **TB01, TB00**: Time base 0 period selection
 00: $f_s/2^{12}$
 01: $f_s/2^{13}$
 10: $f_s/2^{14}$
 11: $f_s/2^{15}$
- Bit 3~0 Unimplemented, read as “0”

CTRL2 Register

Bit	7	6	5	4	3	2	1	0
Name	LCDSEL2	LCDSEL1	LCDSEL0	BZSEL2	BZSEL1	BZSEL0	BUZC	LXTEN
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	1

- Bit 7~5 **LCDSEL2~LCDSEL0**: LCD driver clock selection
 000: $f_s/2^2$
 001: $f_s/2^3$
 010: $f_s/2^4$
 011: $f_s/2^5$
 100: $f_s/2^6$
 101: $f_s/2^7$
 110: $f_s/2^8$
 111: reserved
- Bit 4~2 **BZSEL2~BZSEL0**: BZ frequency selection
 000: $f_s/2^2$
 001: $f_s/2^3$
 010: $f_s/2^4$
 011: $f_s/2^5$
 100: $f_s/2^6$
 101: $f_s/2^7$
 110: $f_s/2^8$
 111: $f_s/2^9$
- Bit 1 **BUZC**: I/O, BUZ selection
 0: I/O
 1: BUZ

Bit 0 **LXTEN**: LXT Oscillator on/off control after execution of HALT instruction
0: LXT off in SLEEP Mode
1: LXT on in IDLE Mode

CTRL3 Register

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

"x" unknown

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LVRF**: reset caused by LVR function activation
0: not active
1: active
This bit can be clear to "0", but can not set to "1".

Bit 1 **LRF**: reset caused by LVRC setting
0: not active
1: active
This bit can be clear to "0", but can not set to "1".

Bit 0 **WRF**: reset caused by WE[4:0] setting
0: not active
1: active
This bit can be clear to "0", but can not set to "1".

CTRL4 Register

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

Bit 7~5,3 Undefined, read as "0"

Bit 4 **LXTLP**: LXT oscillator low power control function
0: LXT Oscillator quick start-up mode
1: LXT Oscillator Low Power Mode

Bit 2~0 **TB12~TB10**: Time Base 1 clock selection
000: $f_s/2^8$
001: $f_s/2^9$
010: $f_s/2^{10}$
011: $f_s/2^{11}$
100: $f_s/2^{12}$
101: $f_s/2^{13}$
110: $f_s/2^{14}$
111: $f_s/2^{15}$

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 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 functions. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options are selected through the configuration options. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

Type	Name	Freq.	Pins
External Crystal	HXT	400kHz~12MHz	OSC1/OSC2
External RC	ERC	400kHz~12MHz	OSC1
Internal High Speed RC	HIRC	4, 8 or 12MHz	—
External Low Speed RC	LXT	32768Hz	XT1/XT2
Internal Low Speed RC	LIRC	32kHz	—

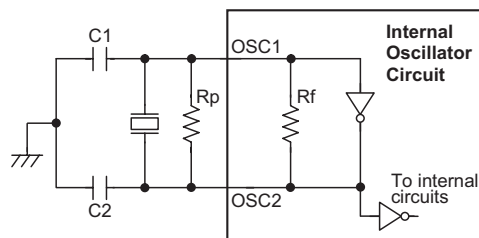
Oscillator Types

System Clock Configurations

There are five system oscillators, three high speed oscillators and two low speed oscillators. The high speed oscillators are the external crystal/ceramic oscillator – HXT, the external – ERC, and the internal RC oscillator – HIRC. The one low speed oscillator is the external 32768Hz oscillator – LXT and the internal 32kHz ($V_{DD}=5V$) oscillator – LIRC.

External Crystal/Resonator Oscillator – HXT

The simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation. However, for some crystals and most resonator types, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.

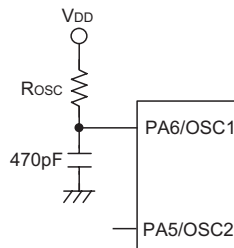


Note: 1. R_p is normally not required. C1 and C2 are required.
 2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

Crystal/Resonator Oscillator – HXT

External RC Oscillator – ERC

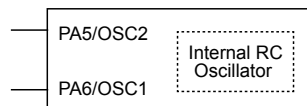
Using the ERC oscillator only requires that a resistor, with a value between 24kΩ and 1.5MΩ, is connected between OSC1 and VDD, and a capacitor is connected between OSC and ground, providing a low cost oscillator configuration. It is only the external resistor that determines the oscillation frequency; the external capacitor has no influence over the frequency and is connected for stability purposes only. 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. Here only the OSC1 pin is used and leaving OSC2 as a general I/O Port.



External RC Oscillator – ERC

Internal RC Oscillator – HIRC

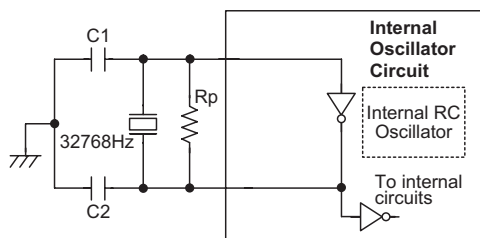
The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of either 4MHz, 8MHz or 12MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that if this internal system clock option is selected, as it requires no external pins for its operation.



Internal RC Oscillator – HIRC

External 32768Hz Crystal Oscillator – LXT

When the microcontroller enters the Idle/Sleep Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the Power-down Mode. To do this, another clock, independent of the system clock, must be provided. To do this a configuration option exists to allow a high speed oscillator to be used in conjunction with a low speed oscillator, known as the LXT oscillator. The LXT oscillator is implemented using a 32768Hz crystal connected to pins XT1/XT2. However, for some crystals, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. The external parallel feedback resistor, Rp, is required. The LXT oscillator must be used together with the HXT, ERC or HIRC register.



Note: 1. Rp, C1 and C2 are required.
 2. Although not shown pins have a parasitic capacitance of around 7pF.

External LXT Oscillator – LXT

LXT Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
32768Hz	8pF	10pF

Note: 1. C1 and C2 values are for guidance only.
 2. $R_p = 5M\Omega \sim 10M\Omega$ is recommended.

A configuration option determines if the XT1/XT2 pins are used for the LXT oscillator or as I/O pins.

- If the I/O option is selected then the XT1/XT2 pins can be used as normal I/O pins.
- If the “LXT oscillator” is selected then the 32kHz crystal should be connected to the XT1/XT2 pins.

LXT Oscillator Low Power Function

The LXT oscillator can function in one of two modes, the Quick Start Mode and the Low Power Mode. The mode selection is executed using the LXTLP bit in the CTRL4 register.

LXTLP Bit	LXT Mode
0	Quick Start
1	Low-power

After power on the LXTLP bit, it will be automatically cleared to zero ensuring that the LXT oscillator is in the Quick Start operating mode. In the Quick Start Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up it can be placed into the Low-power mode by setting the LXTLP bit high. The oscillator will continue to run but with reduced current consumption, as the higher current consumption is only required during the LXT oscillator start-up. In power sensitive applications, such as battery applications, where power consumption must be kept to a minimum, it is therefore recommended that the application program sets the LXTLP bit high about 2 seconds after power-on.

It should be noted that, no matter what condition the LXTLP bit is set to, the LXT oscillator will be always function normally, the only difference is that it will take more time to start up if it is in the Low-power mode.

Internal Low Speed Oscillator – LIRC

The LIRC is a fully self-contained free running on-chip RC oscillator with a typical frequency of 32kHz at 5V requiring no external components. When the device enters the Idle/Sleep Mode, the system clock will stop running but the WDT oscillator continues to free-run and to keep the watchdog active. If “LVR is enabled” or “WDT is enabled” or “ f_s clock source is LIRC”, LIRC Oscillator is turn on. On the contrary, LIRC Oscillator is turn off if “LVR is disabled” and “WDT is disabled” and “ f_s clock source is LXT or $f_{sys}/4$ ”.

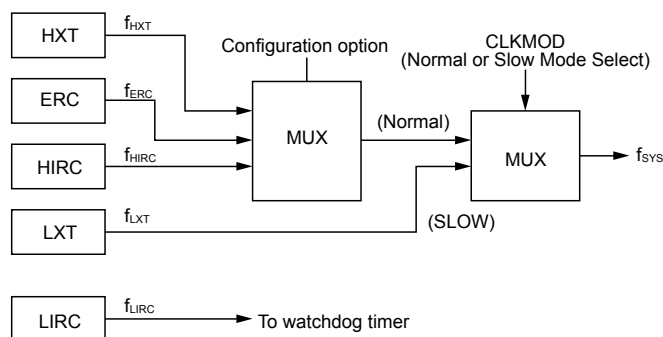
Operating Modes

By using the LXT low frequency oscillator in combination with a high frequency oscillator, the system can be selected to operate in a number of different modes. These modes are Normal, Slow, Idle and Sleep.

Mode Types and Selection

The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. With the capability of dynamically switching between fast and slow oscillators, the device has the flexibility to optimise the performance/power ratio, a feature especially important in power sensitive portable applications.

For the device the LXT oscillator can run together with any of the high speed oscillators, namely the HXT, ERC or the HIRC. The CLKMOD bit in the CTRL0 register can be used to switch the system clock from the selected high speed oscillator to the low speed LXT oscillator. When the HALT instruction is executed the LXT oscillator can be chosen to run or not using the LXTEN bit in the CTRL2 register.



System Clock Configurations

When the system enters the Sleep or Idle Mode, the high frequency system clock will always stop running. The accompanying table shows the relationship between the CLKMOD bit, the HALT instruction and the high/low frequency oscillators. The CLMOD bit can change Normal or Slow Mode.

Operating Mode Control

Operating Mode	OSC1/OSC2 Configuration			XT1/XT2 Configuration	
	HXT	ERC	HIRC	LXT	
				LXTEN=0	LXTEN=1
Normal	On	On	On	On	On
Slow	Off	Off	Off	On	On
Idle	Off	Off	Off	Off	On
Sleep	Off	Off	Off	Off	Off

Mode Switching

The device is switched between one mode and another using a combination of the CLKMOD bit in the CTRL0 register and the HALT instruction. The CLKMOD bit chooses whether the system runs in either the Normal or Slow Mode by selecting the system clock to be sourced from either a high or low frequency oscillator. The HALT instruction forces the system into either the Idle or Sleep Mode, depending upon whether the LXT oscillator is running or not. The HALT instruction operates independently of the CLKMOD bit condition. When a HALT instruction is executed and the LXT oscillator is not running, the system enters the Sleep mode, in which case, the following conditions exist:

- The system oscillator will stop running and the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting.
- The I/O ports will maintain their present condition.
- 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 Idle/Sleep Mode is to keep the current consumption of the MCU to as low a value as possible, perhaps only in the order of several micro-amps, 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. 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. If the configuration options have enabled the Watchdog Timer internal oscillator LIRC then this will continue to run when in the Idle/Sleep Mode and will thus consume some power. For power sensitive applications it may be therefore preferable to use the system clock source for the Watchdog Timer. The LXT, if configured for use, will also consume a limited amount of power, as it continues to run when the device enters the Idle/Sleep Mode. To keep the LXT power consumption to a minimum level the LXTLP bit in the CTRL4 register, which controls the low power function, should be set high.

Wake-up

After the system enters the Idle/Sleep Mode, it can be woken up from one of various sources listed as follows:

- Power-on Reset
- An external falling edge on PA0 to PA7
- A system interrupt
- A WDT overflow

If the system is woken up by Power-on 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.

Pins PA0 to PA7 can be setup via the PAWK register to permit a negative transition on the pin to wake-up the system. When a PA0 to PA7 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 wake-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 to “1” before entering the Idle/Sleep Mode, then any future interrupt requests will not generate a wake-up function of the related interrupt will be ignored.

No matter what the source of the wake-up event is, once a wake-up event occurs, there will be a time delay before normal program execution resumes. Consult the table for the related time.

Wake-up Source	Oscillator Type			
	ERC, IRC	Crystal	CLKMOD=1 LXTEN=0	CLKMOD=1 LXTEN=1
Power On Reset/LVR	$t_{RSDT}+t_{SST1}$	$t_{RSDT}+t_{SST2}$	—	—
PA port	t_{SST1}	t_{SST2}	t_{SST2}	t_{SST1}
Interrupt				
WDT Overflow				

Wake-up Delay Time

Note: 1. t_{RSDT} (reset delay time), t_{SYS} (system clock)

2. t_{RSDT} is power-on delay, typical time=50ms

3. $t_{SST1}=2t_{SYS}$

4. $t_{SST2}=128t_{SYS}$

Watchdog Timer

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

Watchdog Timer Clock Source

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

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. This register together with the corresponding configuration option 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

If the WDT configuration option is “always enable”:

10101 or 01010: Enabled

Other: Reset MCU

If the WDT configuration option is “controlled by the WDT control register”:

10101: Disabled

01010: Enabled

Other: Reset MCU

When these bits are changed by the environmental noise to reset the microcontroller, the reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the CTRL3 register will be set to 1.

Bit 2~0

WS2~WS0: WDT Time-out period selection

000: $2^8/f_s$

001: $2^{10}/f_s$

010: $2^{12}/f_s$

011: $2^{14}/f_s$

100: $2^{15}/f_s$

101: $2^{16}/f_s$

110: $2^{17}/f_s$

111: $2^{18}/f_s$

CTRL3 Register

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

Bit 7~3 “—”: Unimplemented, read as 0

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

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

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

This bit is set 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, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. Some of the Watchdog Timer options, such as always on select and clear instruction type are selected using configuration options. With regard to the Watchdog Timer enable/disable function, there are also five bits, WE4~WE0, in the WDTC register to offer additional enable/disable and reset control of the Watchdog Timer. If the WDT configuration option is determined that the WDT function is always enabled, the WE4~WE0 bits still have effects on the WDT function. When the WE4~WE0 bits value is equal to 01010B or 10101B, the WDT function is enabled. However, if the WE4~WE0 bits are changed to any other values except 01010B and 10101B, which is caused by the environmental noise, it will reset the microcontroller after 2~3 LIRC clock cycles. If the WDT configuration option is determined that the WDT function is controlled by the WDT control register, the WE4~WE0 values can determine which mode the WDT operates in. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B. The WDT function will be enabled if the WE4~WE0 bits value is equal to 01010B. If the WE4~WE0 bits are set to any other values by the environmental noise, except 01010B and 10101B, it will reset the device after 2~3 LIRC clock cycles. After power on these bits will have the value of 01010B.

Watchdog Timer Enable/Disable Control

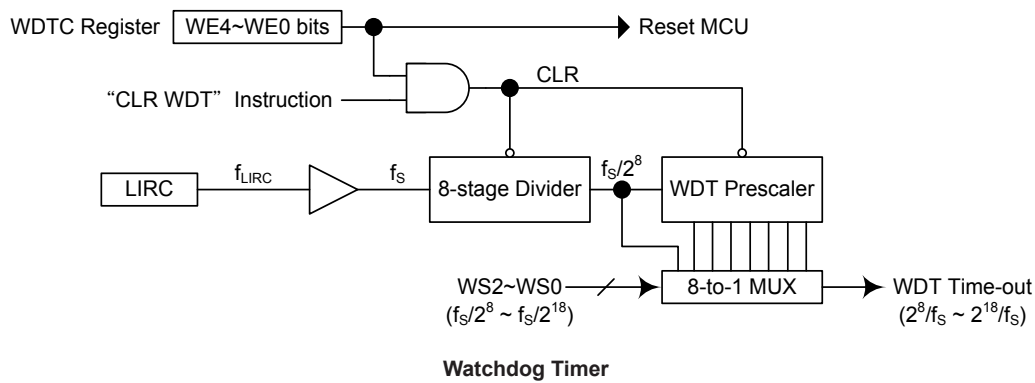
WDT Configuration Option	WE4 ~ WE0 Bits	WDT Function
Always Enable	01010B or 10101B	Enable
	Any other value	Reset MCU
Controlled by WDT Control Register	10101B	Disable
	01010B	Enable
	Any other value	Reset MCU

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer

time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bit field, the second is using the Watchdog Timer software clear instructions and the third is via a HALT instruction.

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

The maximum time out period is when the 2^{18} division ratio is selected. As an example, with a 32 kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 second for the 2^{18} division ratio, and a minimum timeout of 7.8ms for the 2^8 division ratio.



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.

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

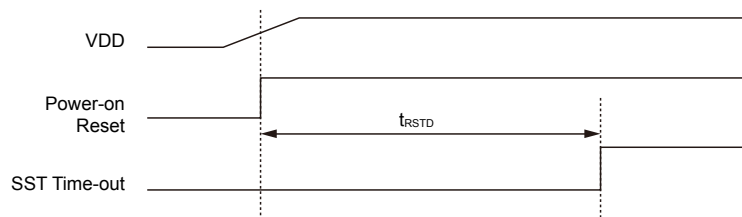
Reset Functions

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

The microcontroller has an internal RC reset function, due to unstable power on conditions. This time delay created by the RC network ensures the state of the POR remains low for an extended period while the power supply stabilizes. During this time, normal operation of the microcontroller is inhibited. After the state of the POR reaches a certain voltage value, the reset delay time t_{POR} is invoked to provide an extra delay time after which the microcontroller can begin normal operation.

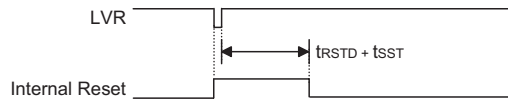


Power-On Reset Timing Chart

- **Low Voltage Reset – LVR**

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage V_{LVR} . If the supply voltage of the device drops to within a range of $0.9V \sim V_{LVR}$ such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the CTRL3 register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between $0.9V \sim V_{LVR}$ must exist for a time greater than that specified by t_{LVR} in the A.C. characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise, the LVR will reset the device after 2~3 LIRC clock cycles. When this happens, the LRF bit in the CTRL3 register will be set to 1. After power on the register will

have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters power down mode.



Low Voltage Reset Timing Chart

LVRC Register

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

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

01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V

Any other value: Generates MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage values above, an MCU reset will be generated. The reset operation will be activated after 2~3 LIRC clock cycles. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than the four defined LVR values above, will also result in the generation of an MCU reset. The reset operation will be activated after 2~3 LIRC clock cycles. However in this situation the register contents will be reset to the POR value.

CTRL3 Register

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

Bit 7~3 “—”: Unimplemented, read as 0

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

0: Not occur
1: Occurred

This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

Bit 1 **LRF**: LVR Control register software reset flag

0: Not occur
1: Occurred

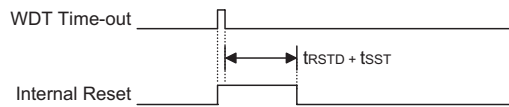
This bit is set to 1 if the LVRC register contains any non defined LVR voltage register values. This in effect acts like a software reset function. This bit can only be cleared to 0 by the application program.

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

Describe elsewhere.

- Watchdog Time-out Reset during Normal Operation

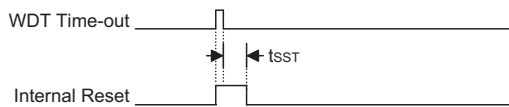
The Watchdog time-out Reset during normal operation is the same as a hardware power-on reset except that the Watchdog time-out flag TO will be set to high.



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

Note: The t_{SST} is 2 clock cycles if the system clock source is provided by ERC or HIRC. The t_{SST} is 128 clock for HXT or LXT. The t_{SST} is 128 clock for LIRC.

Reset Initial Conditions

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

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

“u” unchanged

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

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
Prescaler, Divider	Cleared
WDT, Time Base	Clear after reset, WDT begins counting
Timer/Event Counter	Timer Counter will be turned off
Input/Output Ports	I/O ports will be setup as inputs, and AN0~AN11 as A/D input pins
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 the microcontroller internal registers.

Register	Power-on Reset	LVR Reset	WDT Time-out (Normal Operation)	WDT Time-Out (Idle/Sleep)
PCL	0000 0000	0000 0000	0000 0000	0000 0000
MP0	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
MP1	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
BP	- - - - - 0 0	- - - - - 0 0	- - - - - 0 0	- - - - - u u
ACC	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
TBLP	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
TBLH	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
WDTC	0 1 0 1 0 0 1 1	0 1 0 1 0 0 1 1	0 1 0 1 0 0 1 1	u u u u u u u u
STATUS	- - 0 0 x x x x	- - u u u u u u	- - 1 u u u u u	- - 1 1 u u u u
INTC0	- 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0	- u u u u u u u
INTC1	- 0 0 0 - 0 0 0	- 0 0 0 - 0 0 0	- 0 0 0 0 0 0 0	- u u u u - u u u
TMR0	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR0C	0 0 - 0 1 0 0 0	0 0 - 0 1 0 0 0	0 0 - 0 1 0 0 0	u u u u u u u u
TMR1	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR1C	0 0 0 0 1 - - -	0 0 0 0 1 - - -	0 0 0 0 1 - - -	u u u u u - - -
CTRL4	- - - 0 - 1 1 1	- - - 0 - 1 1 1	- - - 0 - 1 1 1	- - - u - u u u
PA	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PAC	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PAWK	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PAPU	- 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0	- u u u u u u u

Register	Power-on Reset	LVR Reset	WDT Time-out (Normal Operation)	WDT Time-Out (Idle/Sleep)
(Normal Operation)	WDT Time-Out	1111 1111	1111 1111	uuuu uuuu
(Idle/Sleep)	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPUP	0000 0000	0000 0000	0000 0000	uuuu uuuu
PE	-111 1111	-111 1111	-111 1111	-uuu uuuu
PEC	-111 1111	-111 1111	-111 1111	-uuu uuuu
PEPU	-000 0000	-000 0000	-000 0000	-uuu uuuu
PF	---- -111	---- -111	---- -111	---- -uuu
PFC	---- -111	---- -111	---- -111	---- -uuu
PFPU	---- -000	---- -000	---- -000	---- -uuu
CTRL0	0000 00-0	0000 00-0	0000 00-0	uuuu uu-u
CTRL1	0000 ----	0000 ----	0000 ----	uuuu ----
CTRL2	0000 0001	0000 0001	0000 0001	uuuu uuuu
CTRL3	---- -x00	---- -uuu	---- -uuu	---- -uuu
PWM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ANCSR0	0000 0000	0000 0000	0000 0000	uuuu uuuu
ANCSR1	---- 0000	---- 0000	---- 0000	uuuu uuuu
ADCR	01-- 0000	01-- 0000	01-- 0000	uu-- uuuu
ACSR	11-0 -000	11-0 -000	11-0 -000	uu-u -uuu
ADRL	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MFIC	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTEG	1010 1010	1010 1010	1010 1010	uuuu uuuu
LCDO	---- 0000	---- 0000	---- 0000	---- uuuu
LCDC	0--- 0000	0--- 0000	0--- 0000	u--- uuuu
LVDC	--00 -000	--00 -000	--00 -000	uuuu uuuu
LVRC	0101 0101	0101 0101	0101 0101	uuuu uuuu

Note: “-” not implement

“u” means “unchanged”

“x” means “unknown”

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. Most pins can have either an input or output designation under user program control. Additionally, as there are pull-high resistors and wake-up software configurations, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities. 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.

Pull-high Resistors

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

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. After a HALT instruction forces the microcontroller into entering the Idle/Sleep Mode, the processor will remain idle or in a low-power state until the logic condition of the selected wake-up pin on Port A changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Note that pins PA0 to PA7 can be selected individually to have this wake-up feature using an internal register known as PAWK, located in the Data Memory.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAWK	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
PAPU	—	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PEPU	—	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PEC	—	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PFPU	—	—	—	—	—	PFPU2	PFPU1	PFPU0
PFC	—	—	—	—	—	PFC2	PFC1	PFC0

“—” Unimplemented, read as “0”

PAWK_n: PA wake-up function control

0: disable
1: enable

PAC_n/PBC_n/PCC_n/PDC_n/PEC_n/PFC_n: I/O type selection

0: output
1: input

PAPU_n/PBPU_n/PCPU_n/PDPU_n/PEPU_n/PFPU_n: Pull-high function control

0: disable
1: enable

I/O Port Control Registers

Each Port has its own control register, known as PAC, PBC, PCC, PDC, PEC, PFC which controls the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. 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.

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 some pins, the chosen function of the multi-function I/O pins is set by configuration options while for others the function is set by application program control.

- External Interrupt Input

The external interrupt pins, INTn, are pin-shared with I/O pins. To use the pin as an external interrupt input the correct bits in the MFIC register must be programmed. The pin must also be setup as an input by setting the Input/Output Port Control Register bit in the Port Control Register. A pull-high resistor can also be selected via the appropriate port pull-high resistor register. Note that even if the pin is setup as an external interrupt input the I/O function still remains.

- External Timer/Event Counter Input

The Timer/Event Counter pins, TC0 and TC1 are pin-shared with I/O pins. For these shared pins to be used as Timer/Event Counter inputs, the Timer/Event Counter must be configured to be in the Event Counter or Pulse Width Capture Mode. This is achieved by setting the appropriate bits in the Timer/Event Counter Control Register. The pins must also be setup as inputs by setting the appropriate bit in the Port Control Register. Pull-high resistor options can also be selected using the port pull-high resistor registers. Note that even if the pin is setup as an external timer input the I/O function still remains.

- BZ output

The BZ function output is pin-shared with an I/O pin. The output function of this pin is chosen using the CTRL2 register. If the port control register has setup the pin as an input, then the pin will function as a normal logic input with the usual pull-high selection, even if the BZ function has been selected.

- PFD Output

The PFD function output is pin-shared with an I/O pin. The output function of this pin is chosen using the CTRL0 register. Note that the corresponding bit of the port control register, must setup the pin as an output to enable the PFD output. If the port control register has setup the pin as an input, then the pin will function as a normal logic input with the usual pull-high selection, even if the PFD function has been selected.

- PWM Outputs

The PWM function whose outputs are pin-shared with I/O pins. The PWM output functions are chosen using the CTRL0 register. Note that the corresponding bit of the port control registers, for the output pin, must setup the pin as an output to enable the PWM output. If the pins are setup as inputs, then the pin will function as a normal logic input with the usual pull-high selections, even if the PWM registers have enabled the PWM function.

- A/D Inputs

The device has 12 inputs to the A/D converter. All of these analog inputs are pin-shared with I/O pins. If these pins are to be used as A/D inputs and not as I/O pins then the corresponding PCRN bits in the A/D converter control registers, ANCSR0/ANCSR1, must be properly setup. There are no configuration options associated with the A/D converter. If chosen as I/O pins, then full pull-high resistor configuration options remain, however if used as A/D inputs then any pull-high resistor configuration options associated with these pins will be automatically disconnected.

Pin Remapping Configuration

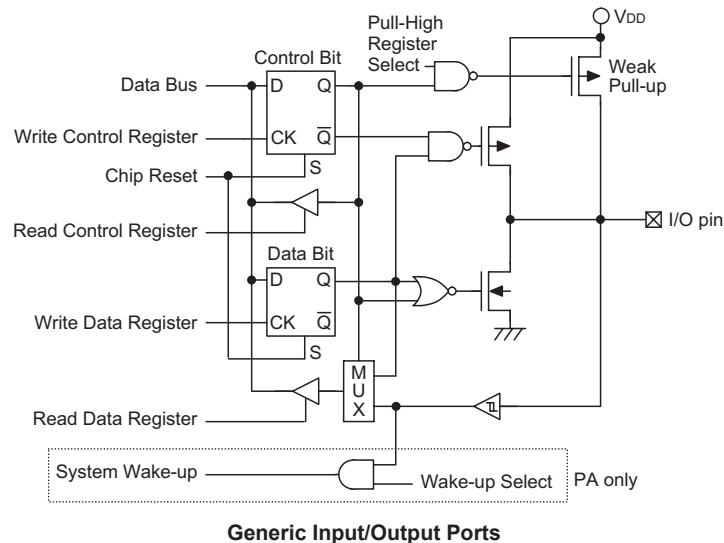
The pin remapping function enables the function pins TC0 and TC1 to be located on different port pins. It is important not to confuse the Pin Remapping function with the Pin-shared function, the two functions have no interdependence. The PCFG bit in the CTRL0 register allows the two function pins TC0 and TC1 to be remapped to different port pins. After power up, this bit will be reset to zero, which will define the default port pins to which the two functions will be mapped. Changing this bit will move the functions to other port pins.

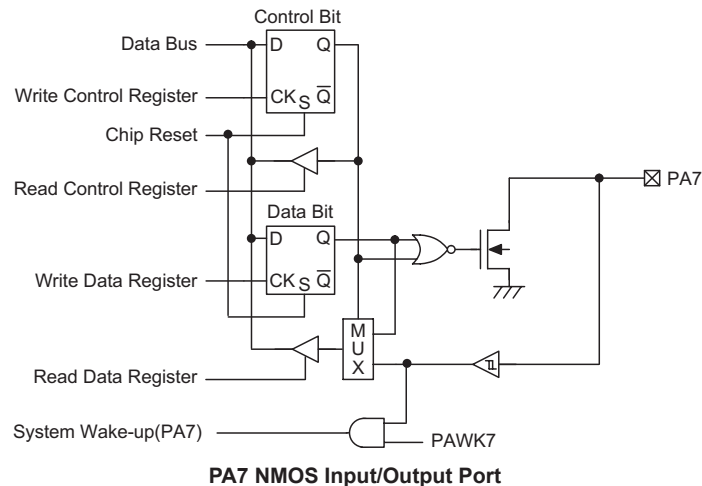
Examination of the pin names on the package diagrams will reveal that some pin function names are repeated, this indicates a function pin that can be remapped to other port pins. If the pin name is bracketed then this indicates its alternative location. Pin names without brackets indicate its default location which is the condition after Power-on.

PCFG Bit Status		
PCFG Bit	0	1
Pin Mapping	TC1/PA1 TC0/PA2	TC0/PA1 TC1/PA2

I/O Pin Structures

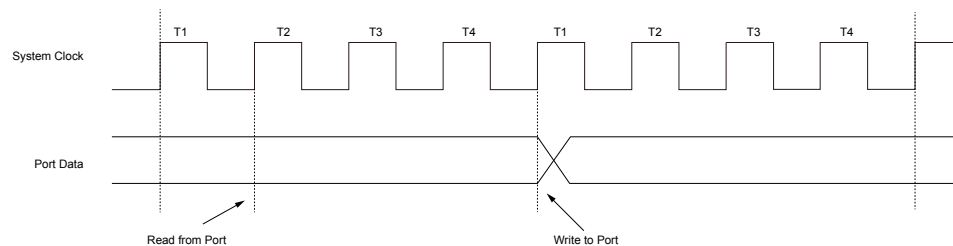
The accompanying diagrams illustrate the internal structures of some generic I/O pin types. As the exact logical construction of the I/O pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins. The wide range of pin-shared structures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





Programming Considerations

Within the user program, one of the things first to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set to high. This means that all I/O pins will be defaulted 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.

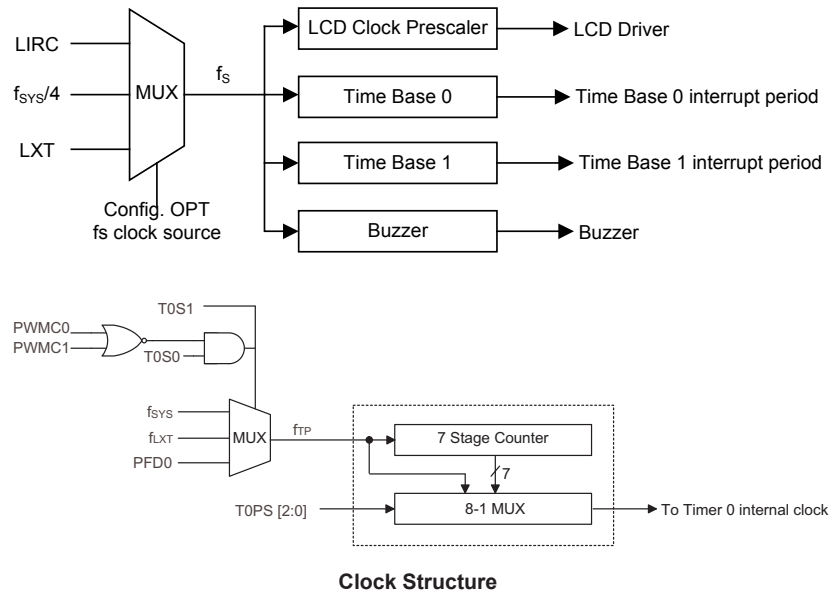
Timer/Event Counter

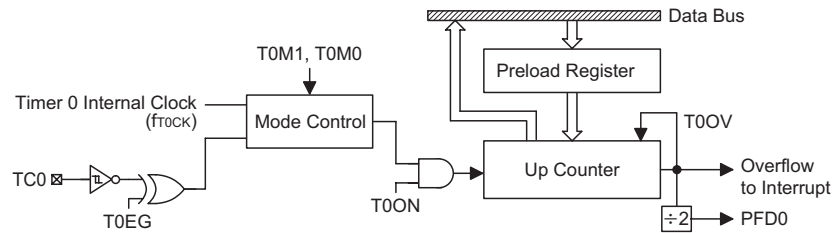
The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contains two count-up timer of 8-bit capacity. As the timers have three different operating modes, they can be configured to operate as a general timer, an external event counter or as a pulse width capture device. The provision of an internal prescaler to the clock circuitry on giving added range to the timers.

There are two types of registers related to the Timer/Event Counters. The first is the register that contains the actual value of the timer and into which an initial value can be preloaded. Reading from this register it retrieves the contents of the Timer/Event Counter. The second type of associated register is the Timer Control Register which defines the timer options and determines how the timer is to be used. The device can have the timer clock configured to come from the internal clock source. In addition, the timer clock source can also be configured to come from an external timer pin.

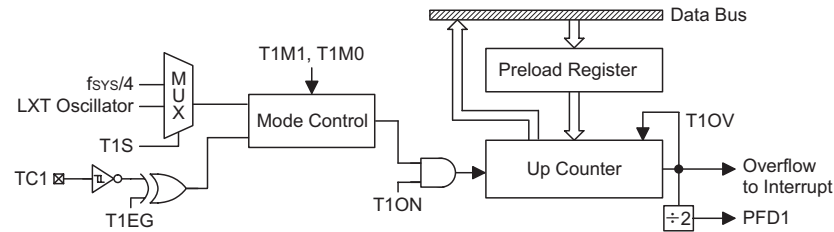
Configuring the Timer/Event Counter Input Clock Source

The Timer/Event Counter clock source can originate from various sources, an internal clock or an external pin. The internal clock source is used when the timer is in the timer mode or in the pulse width capture mode, this internal clock source is firstly divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register bits T0PS2~T0PS0. For Timer/Event Counter 0, the internal clock source can be f_{SYS} , PFD0 or the LXT Oscillator, the choice of which is determined by the T0S[1:0] bit in the CTRL1 register. For Timer/Event Counter 1, the clock source can be $f_{SYS}/4$ or the LXT Oscillator, the choice of which is determined by the T1S bit in the TMR1C register. An external clock source is used when the timer is in the event counting mode, the clock source being provided on an external timer pin TCn. Depending upon the condition of the TnEG bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.

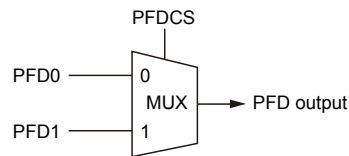




8-bit Timer/Event Counter 0 Structure



8-bit Timer/Event Counter 1 Structure



Timer Registers – TMR0, TMR1

The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. These registers are known as TMR0 and TMR1. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH at which point the timer overflows and an internal interrupt signal is generated. Then the timer value will be reset with the initial preload register value and continue counting. Note that to achieve a maximum full range count of FFH, all the preload registers must first be cleared to zero. It should be noted that after power-on, the preload registers will be in an unknown condition. Note that if the Timer/Event Counter is in an OFF condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter the next time an overflow occurs.

Timer Control Registers – TMR0C, TMR1C

The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in three different modes, the options of which are determined by the contents of their respective control register. The Timer Control Register is known as TMRnC. It is the Timer Control Register together with its corresponding timer registers that control the full operation of the Timer/Event Counter. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation. To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width capture mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair TnM1/TnM0, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as TnON, provides the basic on/off

control of the respective timer. Setting the bit high allows the counter to run. Clearing the bit stops the counter. Bits 0~2 of the Timer Control Register determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or pulse width capture mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as TnEG. The T0S1, T0S0, T1S bits select the internal clock source if used.

CTRL1 Register

Bit	7	6	5	4	3	2	1	0
Name	T0S1	T0S0	TB00	TB01	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

- Bit 7~6 **T0S1, T0S0:** Prescaler/TMR0 clock source
 00: $f_{TP}=f_{SYS}$
 Prescaler clock source is f_{SYS}
 TMR0 clock sourced from the output clock of the Prescaler
 01: $f_{TP}=LXT$
 Prescaler clock source is LXT.
 TMR0 clock sourced from the output clock of the Prescaler
 10: $f_{TP}=PFD0$
 Prescaler clock source is PFD0.
 TMR0 clock sourced from f_{SYS} .
 11: undefined
 Note: If PWM0C or PWM1C is enabled, the clock source of the Prescaler is selected to be either f_{SYS} or PFD0 using the T0S1 bit.
- Bit 5~4 **TB01, TB00:** Time base 0 period selection
 00: $f_S/2^{12}$
 01: $f_S/2^{13}$
 10: $f_S/2^{14}$
 11: $f_S/2^{15}$
- Bit 3~0 Unimplemented, read as “0”

TMR0C Register

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	—	T0ON	T0EG	T0PS2	T0PS1	T0PS0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	1	0	0	0

- Bit 7,6 **T0M1, T0M0:** Timer 0 operation mode selection
 00: no mode available
 01: event counter mode
 10: timer mode
 11: pulse width capture mode
- Bit 5 Unimplemented, read as “0”
- Bit 4 **T0ON:** Timer/event counter counting enable
 0: disable
 1: enable
- Bit 3 **T0EG:** Event counter active edge selection
 1: count on falling edge
 0: count on rising edge
 Pulse Width Capture active edge selection
 1: start counting on rising edge, stop on falling edge
 0: start counting on falling edge, stop on rising edge

Bit 2~0 **T0PS2, T0PS1, T0PS0:** Timer prescaler rate selection Timer internal clock
 000: f_{TP}
 001: $f_{TP}/2$
 010: $f_{TP}/4$
 011: $f_{TP}/8$
 100: $f_{TP}/16$
 101: $f_{TP}/32$
 110: $f_{TP}/64$
 111: $f_{TP}/128$

TMR1C Register

Bit	7	6	5	4	3	2	1	0
Name	T1M1	T1M0	T1S	T1ON	T1EG	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	1	—	—	—

Bit 7,6 **T1M1, T1M0:** Timer 1 operation mode selection
 00: no mode available
 01: event counter mode
 10: timer mode
 11: pulse width capture mode

Bit 5 **T1S:** Timer clock source
 0: $f_{SYS}/4$
 1: LXT Oscillator

Bit 4 **T1ON:** Timer/event counter counting enable
 0: disable
 1: enable

Bit 3 **T1EG:** Event counter active edge selection
 1: count on falling edge
 0: count on rising edge
 Pulse Width Capture active edge selection
 1: start counting on rising edge, stop on falling edge
 0: start counting on falling edge, stop on rising edge

Bit 2~0 Unimplemented, read as “0”

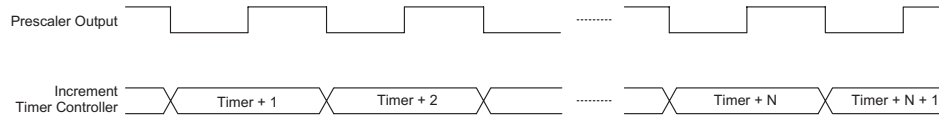
Timer Mode

In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Timer Mode.

Bit7	Bit6
1	0

In this mode the internal clock is used as the timer clock. The timer input clock source is f_{SYS} , $f_{SYS}/4$, PFD0 or the LXT oscillator. However, this timer clock source is further divided by a prescaler, the value of which is determined by the bits T0PS2~T0PS0 in the Timer Control Register. The timer-on bit, TnON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one. When the timer is full and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupts are two of the wake-up sources. However, the internal interrupts can be disabled by ensuring that the TnE bits of the INTC0 register are reset to zero.



Timer Mode Timing Chart

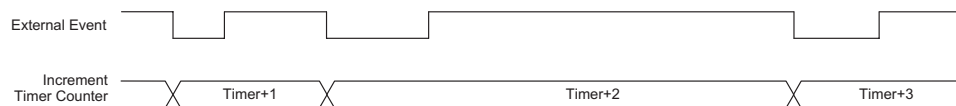
Event Counter Mode

In this mode, a number of externally changing logic events, occurring on the external timer TCn pin, can be recorded by the Timer/Event Counter. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Timer Mode.

Bit7	Bit6
0	1

In this mode, the external timer TCn, is used as the Timer/Event Counter clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit TnON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. If the Active Edge Select bit, TnEG, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the TnEG is high, the counter will increment each time the external timer pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register. It is reset to zero. As the external timer pin is shared with an I/O pin, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Event Counting Mode. The second is to ensure that the port control register configures the pin as an input. It should be noted that in the event counting mode, even if the microcontroller is in the Idle/Sleep Mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input TCn pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (TnEG=1)

Pulse Width Capture Mode

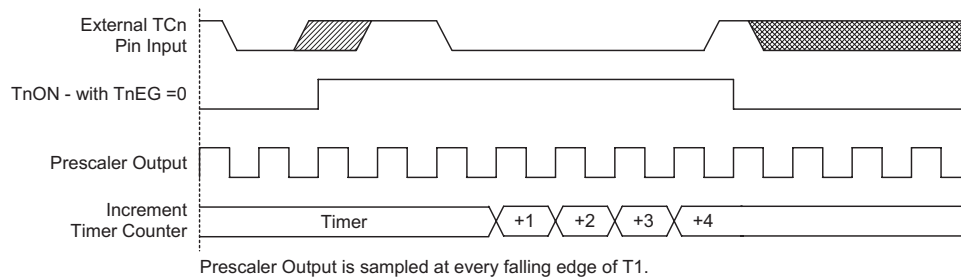
In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Pulse Width Capture Mode.

Bit7	Bit6
1	1

In this mode the internal clock, f_{SYS} , $f_{SYS}/4$, PFD0 or the LXT is used as the internal clock for the 8-bit Timer/Event Counter. However, the clock source, f_{SYS} , for the 8-bit timer is further divided

by a prescaler, the value of which is determined by the Prescaler Rate Select bits T0PS2~T0PS0, which TnON, which is bit 2~0 of the Timer Control Register, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the external timer pin. If the Active Edge Select bit TnEG, which is bit 3 of the Timer Control Register, is low, once a high to low transition has been received on the external timer pin, the Timer/Event Counter will start counting until the external timer pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. If the Active Edge Select bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. It is important to note that in the pulse width capture mode, the enable bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under program control. The residual value in the Timer/Event Counter, which can now be read by the program, therefore represents the length of the pulse received on the TCn pin. As the enable bit has now been reset, any further transitions on the external timer pin will be ignored. The timer cannot begin further pulse width capture until the enable bit is set high again by the program. In this way, single shot pulse measurements can be easily made. It should be noted that in this mode the Timer/Event Counter is controlled by logical transitions on the external timer pin and not by the logic level. When the Timer/Event Counter is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register, it is reset to zero. As the TCn pin is shared with an I/O pin, to ensure that the pin is configured to operate as a pulse width capture pin, two things have to be implemented. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the pulse width capture mode, the second is to ensure that the port control register configure the pin as an input.



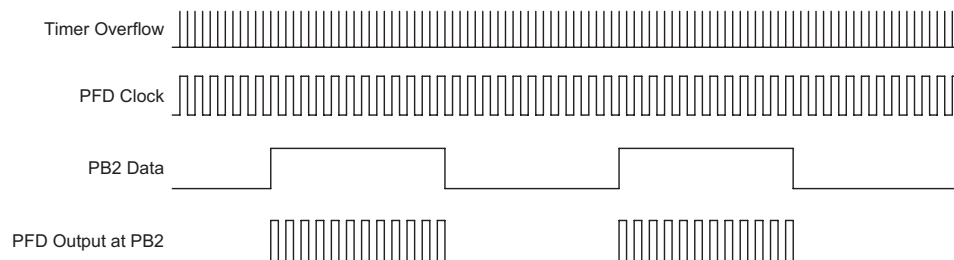
Pulse Width Capture Mode Timing Chart (TnEG=0)

Prescaler

Bits T0PS0~T0PS2 of the TMR0C register can be used to define a division ratio for the internal clock source of the Timer/Event Counter enabling longer time-out periods to be setup.

PFD Function

The Programmable Frequency Divider provides a means of producing a variable frequency output suitable for applications, such as piezo-buzzer driving or other interfaces requiring a precise frequency generator. The Timer/Event Counter overflow signal is the clock source for the PFD function, which is controlled by PFDCS bit in CTRL0. For this device the clock source can come from either Timer/Event Counter 0 or Timer/Event Counter 1. The output frequency is controlled by loading the required values into the timer prescaler and timer registers to give the required division ratio. The counter will begin to count-up from this preload register value until full, at which point an overflow signal is generated, causing both the PFD outputs to change state. Then the counter will be automatically reloaded with the preload register value and continue counting-up. If the CTRL0 register has selected the PFD function, then for PFD output to operate, it is essential for the Port B control register PBC to setup the PFD pins as outputs. PB2 must be set high to activate the PFD. The output data bits can be used as the on/off control bit for the PFD outputs. Note that the PFD outputs will all be low if the output data bit is cleared to zero. Using this method of frequency generation, and if a crystal oscillator is used for the system clock, very precise values of frequency can be generated.



PFD Function

I/O Interfacing

The Timer/Event Counter, when configured to run in the event counter or pulse width capture mode, requires the use of an external timer pin for its operation. As this pin is a shared pin it must be configured correctly to ensure that it is setup for use as a Timer/Event Counter input pin. This is achieved by ensuring that the mode selects bits in the Timer/Event Counter control register, either the event counter or pulse width capture mode. Additionally the corresponding Port Control Register bit must be set high to ensure that the pin is setup as an input. Any pull-high resistor connected to this pin will remain valid even if the pin is used as a Timer/Event Counter input.

Programming Considerations

When configured to run in the timer mode, the internal system clock is used as the timer clock source and is therefore synchronised with the overall operation of the microcontroller. In this mode when the appropriate timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector. For the pulse width capture mode, the internal system clock is also used as the timer clock source but the timer will only run when the correct logic condition appears on the external timer input pin. As this is an external event and not synchronised with the internal timer clock, the microcontroller will only see this external event when the next timer clock pulse arrives. As a result, there may be small differences in measured values requiring programmers to take this into account during programming. The same applies if the timer is configured to be in the event counting mode, which again is an external event and not synchronised with the internal system or timer clock. When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may

result in a counting error, this should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialised before using them for the first time. The associated timer enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The edge select, timer mode and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application. It is also important to ensure that an initial value is first loaded into the timer registers before the timer is switched on; this is because after power-on the initial values of the timer registers are unknown. After the timer has been initialized the timer can be turned on and off by controlling the enable bit in the timer control register. When the Timer/Event Counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the Timer/Event Counter interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in a Power-down condition. This situation may occur if the Timer/Event Counter is in the Event Counting Mode and if the external signal continues to change state. In such a case, the Timer/Event Counter will continue to count these external events and if an overflow occurs the device will be woken up from its Power-down condition. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the “HALT” instruction to enter the Idle/Sleep Mode.

Timer Program Example

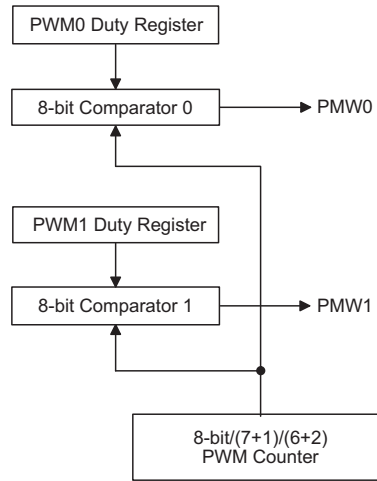
The program shows how the Timer/Event Counter registers are setup along with how the interrupts are enabled and managed. Note how the Timer/Event Counter is turned on, by setting bit 4 of the Timer Control Register. The Timer/Event Counter can be turned off in a similar way by clearing the same bit. This example program sets the Timer/Event Counters to be in the timer mode, which uses the internal system clock as their clock source.

PFD Programming Example

```
org 04h          ; external interrupt vector
org 08h          ; Timer Counter 0 interrupt vector
jmp tmr0int      ; jump here when Timer 0 overflows
:
:
org 20h          ; main program
:
:
:                ;internal Timer 0 interrupt routine
tmr0int:
:
:                ;Timer 0 main program placed here
:
:
begin:
:                ;setup Timer 0 registers
mov a,09bh      ; setup Timer 0 preload value
mov tmr0,a
mov a,081h      ; setup Timer 0 control register
mov tmr0c,a     ; timer mode and prescaler set to /2
:                ;setup interrupt register
mov a,00dh      ; enable master interrupt and both timer interrupts
mov intc0,a
:
:
set tmr0c.4     ; start Timer 0
:
:
```

Pulse Width Modulator

The device includes a multiple output 8-bit PWM function. Useful for such applications such as motor speed control, the PWM function provides outputs with a fixed frequency but with a duty cycle that can be varied by setting particular values into the corresponding PWM register.



PWM Block Diagram

PWM Operation

A single register, known as PWMn and located in the Data Memory is assigned to each Pulse Width Modulator channel. It is here that the 8-bit value, which represents the overall duty cycle of one modulation cycle of the output waveform, should be placed. To increase the PWM modulation frequency, each modulation cycle is subdivided into two or four individual modulation subsections, known as the 7+1 mode or 6+2 mode respectively. The required mode and the on/off control for each PWM channel is selected using the CTRL0 register. Note that when using the PWM, it is only necessary to write the required value into the PWMn register and select the required mode setup and on/off control using the CTRL0 register, the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware. The PWM clock source is the system clock f_{SYS} .

This method of dividing the original modulation cycle into a further 2 or 4 sub-cycles enable the generation of higher PWM frequencies which allow a wider range of applications to be served. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood. As the PWM clock is the system clock, f_{SYS} , and as the PWM value is 8-bits wide, the overall PWM cycle frequency is $f_{SYS}/256$. However, when in the 7+1 mode of operation the PWM modulation frequency will be $f_{SYS}/128$, while the PWM modulation frequency for the 6+2 mode of operation will be $f_{SYS}/64$.

PWM Modulation	PWM Cycle Frequency	PWM Cycle Duty
$f_{SYS}/64$ for (6+2) bits mode $f_{SYS}/128$ for (7+1) bits mode	$f_{SYS}/256$	$[PWM]/256$

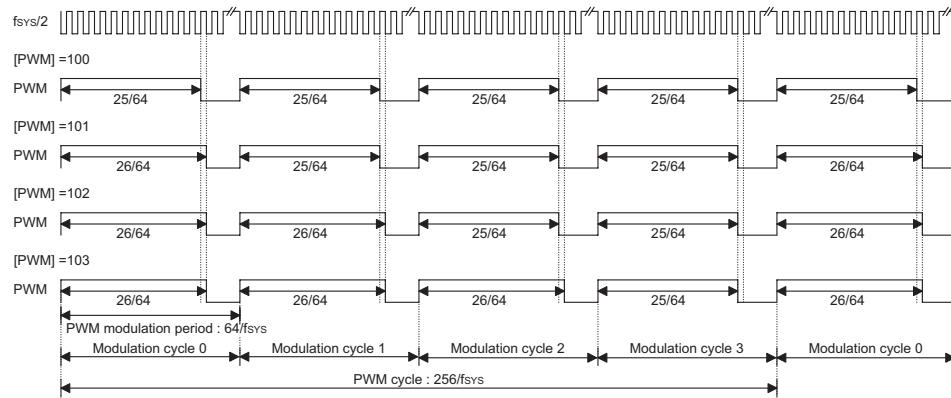
6+2 PWM Mode

Each full PWM cycle, as it is controlled by an 8-bit PWM register, has 256 clock periods. However, in the 6+2 PWM mode, each PWM cycle is subdivided into four individual sub-cycles known as modulation cycle 0~modulation cycle 3, denoted as i in the table. Each one of these four sub-cycles contains 64 clock cycles. In this mode, a modulation frequency increase of four is achieved. The 8-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit 2~bit 7 is denoted here as the DC value. The second group which consists of bit 0~bit 1 is known as the AC value. In the 6+2 PWM mode, the duty cycle value of each of the four modulation sub-cycles is shown in the following table.

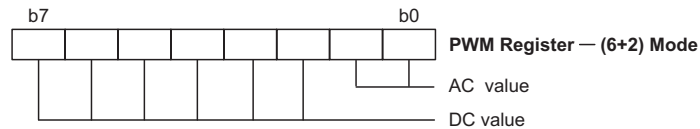
Parameter AC (0~3)	DC	DC(Duty Cycle)
Modulation cycle i ($i=0\sim3$)	$i < AC$	$(DC+1)/64$
	$i > AC$	$DC/64$

6+2 Mode Modulation Cycle Values

The following diagram illustrates the waveforms associated with the 6+2 mode of PWM operation. It is important to note how the single PWM cycle is subdivided into 4 individual modulation cycles, numbered from 0~3 and how the AC value is related to the PWM value.



6+2 PWM Mode



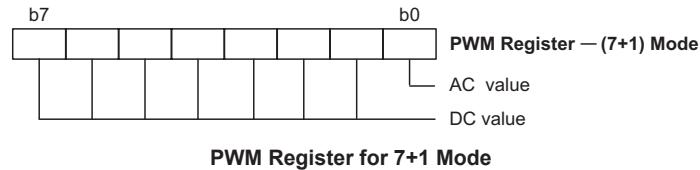
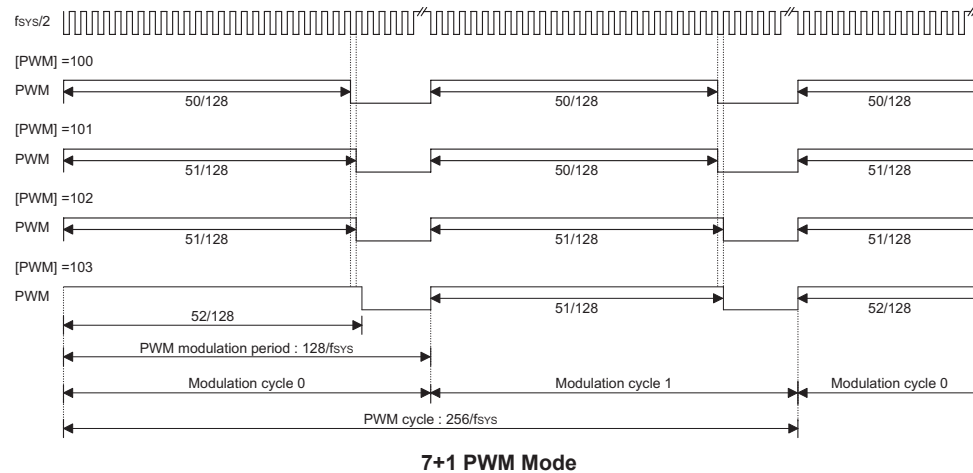
PWM Register for 6+2 Mode

7+1 PWM Mode

Each full PWM cycle, as it is controlled by an 8-bit PWM register, has 256 clock periods. However, in the 7+1 PWM mode, each PWM cycle is subdivided into two individual sub-cycles known as modulation cycle 0~modulation cycle 1, denoted as i in the table. Each one of these two sub-cycles contains 128 clock cycles. In this mode, a modulation frequency increase of two is achieved. The 8-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit 1~bit 7 is denoted here as the DC value. The second group which consists of bit 0 is known as the AC value. In the 7+1 PWM mode, the duty cycle value of each of the two modulation sub-cycles is shown in the following table.

Parameter	AC(0~1)	DC (Duty Cycle)
Modulation cycle i (i=0~1)	i<AC	(DC+1)/128
	i>=AC	DC/128

The following diagram illustrates the waveforms associated with the 7+1 mode PWM operation. It is important to note how the single PWM cycle is subdivided into 2 individual modulation cycles, numbered 0 and 1 and how the AC value is related to the PWM value.



PWM Output Control

The PWM outputs are pin-shared with the I/O pins PA0 and PB0. To operate as a PWM output and not as an I/O pin, the correct bits must be set in the CTRL0 register. A zero value must also be written to the corresponding bit in the I/O port control register PAC.0 and PBC.0 to ensure that the corresponding PWM output pin is setup as an output. After these two initial steps have been carried out, and of course after the required PWM value has been written into the PWMn register, writing a high value to the corresponding bit in the output data register PA.0 and PB.0 will enable the PWM data to appear on the pin. Writing a zero value will disable the PWM output function and force the output low. In this way, the Port data output registers can be used as an on/off control for the PWM function. Note that if the CTRL0 register has selected the PWM function, but a high value has been written to its corresponding bit in the PAC or PBC control register to configure the pin as an input, then the pin can still function as a normal input line, with pull-high resistor options.

PWM Programming Example

```

mov a, 64h                ; setup PWM value of decimal 100
mov pwm0, a
set ctrl0.5               ; select the 7+1 PWM mode
set ctrl0.3               ; select pin PA0 to have a PWM function
clr pac.0                 ; setup pin PA0 as an output
set pa.0                  ; enable the PWM output
:
:
clr pa.0                  ; disable the PWM output_ pin PA0 forced low

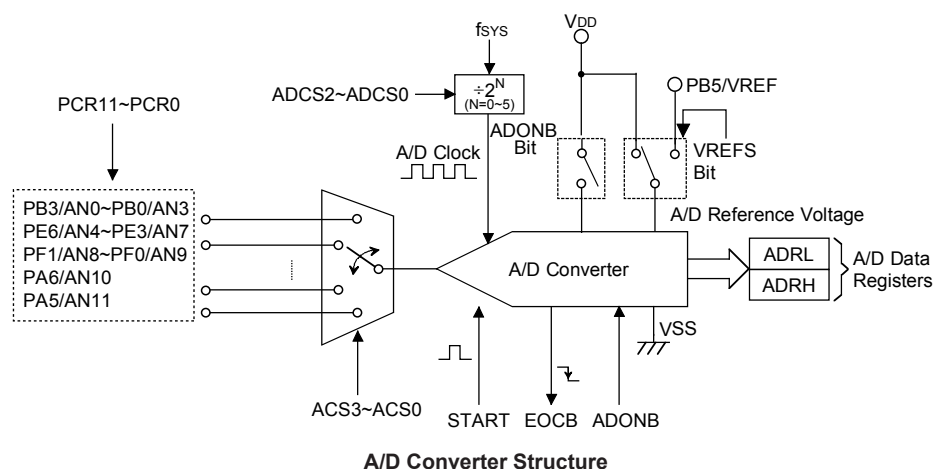
```

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 12-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 either a 12-bit digital value.



A/D Converter Structure

A/D Converter Data Registers – ADRL, ADRH

The device, which has an internal 12-bit A/D converter, requires two data registers, a high byte register, known as ADRH, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. Only the high byte register, ADRH, utilises its full 8-bit contents. The low byte register utilises only 4 bit of its 8-bit contents as it contains only the lowest bits of the 12-bit converted value.

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADRL	D3	D2	D1	D0	—	—	—	—
ADRH	D11	D10	D9	D8	D7	D6	D5	D4

ADRH, ADRL Register

Bit	ADRH								ADRL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	—	—	—	—
R/W	R	R	R	R	R	R	R	R	R	R	R	R	—	—	—	—
POR	X	X	X	X	X	X	X	X	X	X	X	X	—	—	—	—

"X" unknown

"—": Unimplemented, read as "0"

D11~D0: ADC conversion data

A/D Converter Control Registers – ADCR, ACSR, ANCSR0, ANCSR1

To control the function and operation of the A/D converter, four control registers known as ANCSR0, ANCSR1, ADCR and ACSR are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, which pins are used as analog inputs and which are used as normal I/Os, the A/D clock source as well as controlling the start function and monitoring the A/D converter end of conversion status.

The ACS3~ACS0 bits in the ADCR register define the channel number. As the device contains only one actual analog to digital converter circuit, each of the individual 12 analog inputs must be routed to the converter. It is the function of the ACS3~ACS0 bits in the ADCR register to determine which analog channel is actually connected to the internal A/D converter. The ANCSR0/ANCSR1 control register also contains the PCR11~PCR0 bits which determine whether the A/D function pins are connected to external pin. If the 12-bit address on PCR11~PCR0 has a value of “FFFH”, then all pins will all be set as analog inputs. Note that if the PCR11~PCR0 bits are all set to zero, then all the pins will be setup as normal I/Os.

ANCSR0 Register

Bit	7	6	5	4	3	2	1	0
Name	PCR7	PCR6	PCR5	PCR4	PCR3	PCR2	PCR1	PCR0
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 **PCR[7:0]**: define I/O lines or analog inputs configuration to port
0: digital I/O. Pin is assigned as an I/O line or pin-shared function
1: analog input. Pin is switched to analog input

ANCSR1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PCR11	PCR10	PCR9	PCR8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unused, read as “0”

Bit 3~0 **PCR[11:8]**: define I/O lines or analog inputs configuration to port
0: digital I/O. Pin is assigned as a I/O line or pin-shared function
1: analog input. Pin is switched to analog input

ADCR Register

Bit	7	6	5	4	3	2	1	0
Name	START	EOCB	—	—	ACS3	ACS2	ACS1	ACS0
R/W	R/W	R	—	—	R/W	R/W	R/W	R/W
POR	0	1	—	—	0	0	0	0

Bit 7 **START**: Start the A/D conversion
0→1→0: start
0→1: reset the A/D converter and set EOCB to “1”

This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process. When the bit is set high the A/D converter will be reset.

Bit 6 **EOCB**: End of A/D conversion flag
0: A/D conversion ended
1: A/D conversion in progress

This read only flag is used to indicate when an A/D conversion process has completed. When the conversion process is running the bit will be high.

- Bit 5~4 unimplemented, read as “0”
- Bit 3~0 **ACS3~ACS0:** Select A/D channel
- 0000: AN0
 - 0001: AN1
 - 0010: AN2
 - 0011: AN3
 - 0100: AN4
 - 0101: AN5
 - 0110: AN6
 - 0111: AN7
 - 1000: AN8
 - 1001: AN9
 - 1010: AN10
 - 1011: AN11
 - 1100: AN8
 - 1101: AN9
 - 1110: AN10
 - 1111: AN11

These are the A/D channel select control bits. As there is only one internal hardware A/D converter each of the eight A/D inputs must be routed to the internal converter using these bits.

ACSR Control Register

Bit	7	6	5	4	3	2	1	0
Name	TEST	ADONB	—	VREFS	—	ADCS2	ADCS1	ADCS0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	1	1	—	0	—	0	0	0

- Bit 7 **TEST:** For test mode use only
- Bit 6 **ADONB:** ADC MODULE on/off control bit
- 0: ADC module is on
 - 1: ADC module is off
- Bit 5 unimplemented, read as “0”
- Bit 4 **VREFS:** Selecte ADC reference voltage
- 0: internal ADC power
 - 1: AVREF pin
- Bit 3 unimplemented, read as “0”
- Bit 2~0 **ADCS2~ADCS0:** Select ADC clock source
- 000: $f_{SYS}/2$
 - 001: $f_{SYS}/8$
 - 010: $f_{SYS}/32$
 - 011: undefined
 - 100: f_{SYS}
 - 101: $f_{SYS}/4$
 - 110: $f_{SYS}/16$
 - 111: undefined

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

A/D Operation

The START bit in the register is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the EOCB bit in the ADCR register will be set to “1” 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 EOCB bit in the ADCR register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to “0” by the microcontroller after a conversion cycle has ended. 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 EOCB bit in the ADCR register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle. The clock source for the A/D converter, which originates from the system clock f_{SYS} , is first divided by a division ratio, the value of which is determined by the ADCS2, ADCS1 and ADCS0 bits in the ACSR register. Controlling the power on/off function of the A/D converter circuitry is implemented using the value of the ADONB bit.

Although the A/D clock source is determined by the system clock f_{SYS} , and by bits ADCS2, ADCS1 and ADCS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the minimum value of permissible A/D clock period, t_{AD} , is 0.5 μ s, care must be taken for system clock speeds in excess of 4MHz. For system clock speeds in excess of 4MHz, the ADCS2, ADCS1 and ADCS0 bits should not be set to “000”. Doing so will give A/D clock periods that are less than the minimum A/D clock period which may result in inaccurate A/D conversion values.

Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

f_{SYS}	A/D Clock Period(t_{AD})						
	ADCS2, ADCS1, ADCS0 =000 ($f_{SYS}/2$)	ADCS2, ADCS1, ADCS0 =001 ($f_{SYS}/8$)	ADCS2, ADCS1, ADCS0 =010 ($f_{SYS}/32$)	ADCS2, ADCS1, ADCS0 =100 (f_{SYS})	ADCS2, ADCS1, ADCS0 =101 ($f_{SYS}/4$)	ADCS2, ADCS1, ADCS0 =110 ($f_{SYS}/16$)	ADCS2, ADCS1, ADCS0 =011,111
1MHz	2 μ s	8 μ s	32 μ s	1 μ s	4 μ s	16 μ s	Undefined
2MHz	1 μ s	4 μ s	16 μ s	500ns	2 μ s	8 μ s	Undefined
4MHz	500ns	2 μ s	8 μ s	250ns*	1 μ s	4 μ s	Undefined
8MHz	250ns*	1 μ s	4 μ s	125ns*	500ns	2 μ s	Undefined
12MHz	167ns*	667ns	2.67 μ s	83ns*	333ns*	1 μ s	Undefined

A/D Clock Period Examples

A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins on Port A, B, E, F. Bits PCR7~PCR0 in the ANCSR0 register and PCR11~PCR8 in the ANCSR1 register, determine whether the input pins are setup as normal input/output pins or whether they are setup as analog inputs. In this way, pins can be changed under program control to change their function from normal I/O operation to analog inputs and vice versa. Pull-high resistors, which are setup through register programming, apply to the input pins only when they are used as normal I/O pins, if setup as A/D inputs the pull-high resistors will be automatically disconnected. Note that it is not necessary to first setup the A/D pin as an input in the I/O port control registers to enable the A/D input as when the PCRn bits enable an A/D input, the status of the port control register will be overridden.

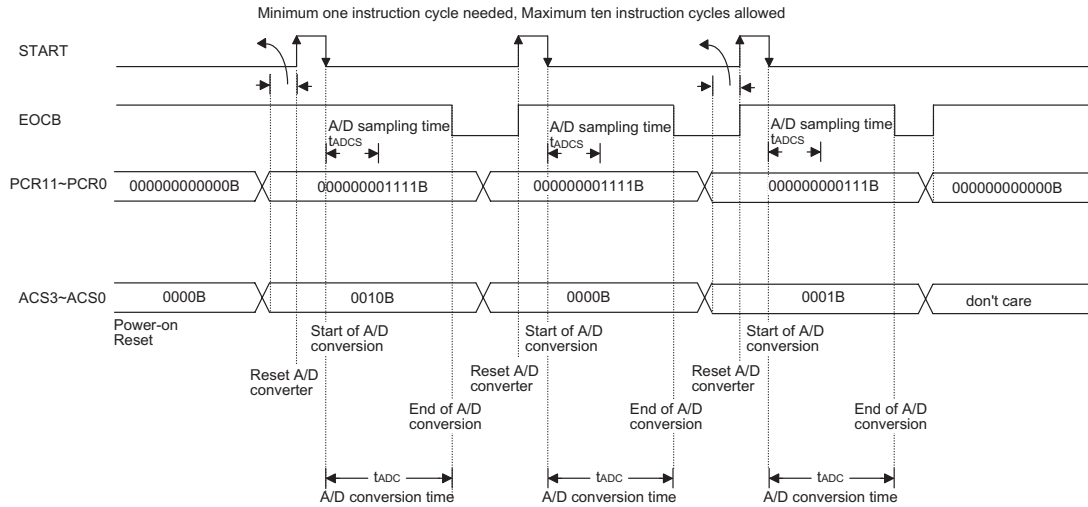
Summary of A/D Conversion Steps

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

- Step 1
Select the required A/D conversion clock by correctly programming bits ADCS2~ADCS0 in the ACSR register.
- Step 2
Select which pins are to be used as A/D inputs and configure them as A/D input pins by correctly programming the PCR11~PCR0 bits in the ANCSR0 and ANCSR1 register.
- Step 3
Enable the A/D by clearing the ADONB in the ACSR register to zero.
- Step 4
Select which channel is to be connected to the internal A/D converter by correctly programming the ACS3~ACS0 bits which are also contained in the register.
- Step 5
If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, the INTC0 interrupt control register must be set to “1”, the A/D converter interrupt bit, ADE, must also be set to “1”.
- Step 6
The analog to digital conversion process can now be initialised by setting the START bit in the ADCR register from low to high and then low again. Note that this bit should have been originally cleared to zero.
- Step 7
To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

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

The accompanying diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing.



A/D Conversion Timing

Note: A/D clock must be $f_{SYS}/2$, $f_{SYS}/8$ or $f_{SYS}/32$

$$t_{ADCS}=4t_{AD}$$

$$t_{ADC}=16t_{AD}$$

A/D Conversion Timing

After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is $16t_{AD}$ where t_{AD} is equal to the A/D clock period.

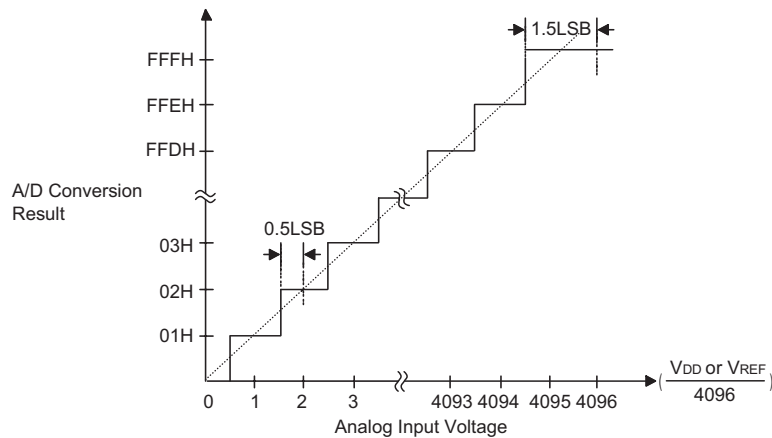
Programming Considerations

When programming, the special attention must be given to the PCR[11:0] bits in the register. If these bits are all cleared to zero, no external pins will be selected for use as A/D input pins allowing the pins to be used as normal I/O pins. When this happens, the internal A/D circuitry will be power down. Setting the ADONB bit high has the ability to power down the internal A/D circuitry, which may be an important consideration in power sensitive applications.

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} voltage, this gives a single bit analog input value of V_{DD} divided by 4096. The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter.

Note that to reduce the quantisation error, a 0.5LSB offset is added to the A/D Converter input. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5LSB below where they would change without the offset, and the last full scale digitized value will change at a point 1.5LSB below the V_{DD} level.



Ideal A/D Transfer Function

A/D Programming Example

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the EOCB bit in the ADCR 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 EOCB polling method to detect the end of conversion

```

clr ADE                      ; disable ADC interrupt
mov a,00000001B
mov ACSR,a                   ; select fsys/8 as A/D clock and ADONB=0
mov a,00000000B              ; setup ANCSR1/ANCSR0 register to configure Port
                              ; as A/D inputs

mov ANCSR1,a
mov a,00000001B
mov ANCSR0,a                 ; and select AN0 to be connected to the A/D
                              ; converter

:
:
Start_conversion:
clr START                    ; reset A/D
set START                    ; start A/D
Polling_EOC:
sz EOCB                      ; poll the ADCR register EOCB bit to detect end
                              ; of A/D conversion
jmp polling_EOC              ; continue polling
mov a,ADRL                   ; read low byte conversion result value
mov adrl_buffer,a            ; save result to user defined register
mov a,ADRH                   ; read high byte conversion result value
mov adrh_buffer,a            ; save result to user defined register
:
jmp start_conversion          ; start next A/D conversion

```

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

```
clr ADE                                ; disable ADC interrupt
mov a,00000001B
mov ACSR,a                             ; select fSYS/8 as A/D clock and ADONB=0
mov a,00000000B                         ; setup ANCSR1/ANCSR0 register to configure Port
                                         ; as A/D inputs

mov ANCSR1,a
mov a,00000001B
mov ANCSR0,a                           ; and select AN0 to be connected to the A/D
                                         ; converter

:
:
Start_conversion:
clr START
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_:
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,ADRL                              ; read low byte conversion result value
mov adrl_buffer,a                      ; save result to user defined register
mov a,ADRH                              ; read high byte conversion result value
mov adrh_buffer,a                      ; save result to user defined register
:
:
EXIT_ISR:
mov a,status_stack
mov STATUS,a                           ; restore STATUS from user defined memory
mov a, acc_stack                        ; restore ACC from user defined memory
clr ADF                                ; clear ADC interrupt flag
reti
```

Note: To power off ADC module, it is necessary to set ADONB as “1”.

Buzzer

Operating in a similar way to the Programmable Frequency Divider, the Buzzer function provides a means of producing a variable frequency output, suitable for applications such as Piezo-buzzer driving or other external circuits that require a precise frequency generator. The buzzer output pin BUZ, is pin-shared with the I/O pin, PB1. The buzzer is driven by the internal clock source, which then passes through a divider, the division ratio of which is selected by CTRL2 register to provide a range of buzzer frequencies from $f_s/2^2$ to $f_s/2^9$. The clock source that generates f_s , which in turn controls the buzzer frequency, can originate from three different sources, the LIRC oscillator, LXT oscillator or $f_{sys}/4$. The choice of which is determined by the f_s clock source configuration option.

CTRL2 register

Bit	7	6	5	4	3	2	1	0
Name	LCDSEL2	LCDSEL1	LCDSEL0	BZSEL2	BZSEL1	BZSEL0	BUZC	LXTEN
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	1

Bit 7~5 Described elsewhere

Bit 4~2 **BZSEL2~BZSEL0**: BZ frequency select

000: $f_s/2^2$

001: $f_s/2^3$

010: $f_s/2^4$

011: $f_s/2^5$

100: $f_s/2^6$

101: $f_s/2^7$

110: $f_s/2^8$

111: $f_s/2^9$

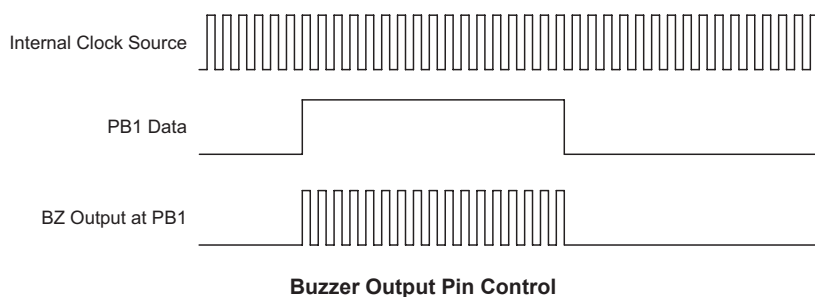
Bit 1 **BUZC**: I/O or Buzzer function select

0: I/O

1: BUZ

Bit 0 Described elsewhere

If the register has selected pin PB1 to function as a buzzer output, then for correct buzzer operation it is essential that the pin must be setup as output by setting bit PBC.1 of the PBC port control register to zero. The PB1 data bit in the PB data register must also be set high to enable the buzzer outputs, if set low, the pin PB1 will remain low. In this way the single bit PB1 of the PB register can be used as an on/off control for buzzer pin output. The data setup on pin PB1 has no effect on the buzzer output.



Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter or Time Base require 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 four external interrupts and multiple internal interrupts. The external interrupts are controlled by the action of the external interrupt pins, while the internal interrupt is controlled by the Timer/Event Counters and Time Base overflows.

Interrupt Registers

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by using three registers, INTC0, INTC1 and MFIC. By controlling the appropriate enable bits in these registers each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request flag will be set by the microcontroller. The global enable flag cleared to zero will disable all interrupts.

INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	1	0	1	0	1	0

- Bit 7~6 **INT3S1, INT3S0:** Interrupt edge control for INT3 pin
00: disable
01: rising edge
10: falling edge
11: both rising and falling edge
- Bit 5~4 **INT2S1, INT2S0:** Interrupt edge control for INT2 pin
00: disable
01: rising edge
10: falling edge
11: both rising and falling edge
- Bit 3~2 **INT1S1, INT1S0:** Interrupt edge control for INT1 pin
00: disable
01: rising edge
10: falling edge
11: both rising and falling edge
- Bit 1~0 **INT0S1, INT0S0:** Interrupt edge control for INT0 pin
00: disable
01: rising edge
10: falling edge
11: both rising and falling edge

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	T1F	T0F	MFF	T1E	T0E	MFE	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 **T1F**: Timer/Event Counter 1 interrupt request flag
0: inactive
1: active
- Bit 5 **T0F**: Timer/Event Counter 0 interrupt request flag
0: inactive
1: active
- Bit 4 **MFF**: Multi-function interrupt request flag
0: inactive
1: active
- Bit 3 **T1E**: Timer/Event Counter 1 interrupt enable
0: disable
1: enable
- Bit 2 **T0E**: Timer/Event Counter 0 interrupt enable
0: disable
1: enable
- Bit 1 **MFE**: Multi-function 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	TB0F	ADF	—	TB1E	TB0E	ADE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

- Bit 7 Unimplemented, read as "0"
- Bit 6 **TB1F**: Time Base 1 interrupt request flag
0: inactive
1: active
- Bit 5 **TB0F**: Time Base 0 interrupt request flag
0: inactive
1: active
- Bit 4 **ADF**: A/D Conversion interrupt request flag
0: inactive
1: active
- Bit 3 Unimplemented, read as "0"
- Bit 2 **TB1E**: Time Base 1 interrupt enable
0: disable
1: enable
- Bit 1 **TB0E**: Time Base 0 interrupt enable
0: disable
1: enable

Bit 0 **ADE:** A/D Conversion interrupt enable
 0: disable
 1: enable

MFIC Register

Bit	7	6	5	4	3	2	1	0
Name	INT3F	INT2F	INT1F	INT0F	INT3E	INT2E	INT1E	INT0E
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 **INT3F:** INT3 interrupt request flag
 0: inactive
 1: active

Bit 6 **INT2F:** INT2 interrupt request flag
 0: inactive
 1: active

Bit 5 **INT1F:** INT1 interrupt request flag
 0: inactive
 1: active

Bit 4 **INT0F:** INT0 interrupt request flag
 0: inactive
 1: active

Bit 3 **INT3E:** INT3 interrupt enable
 0: disable
 1: enable

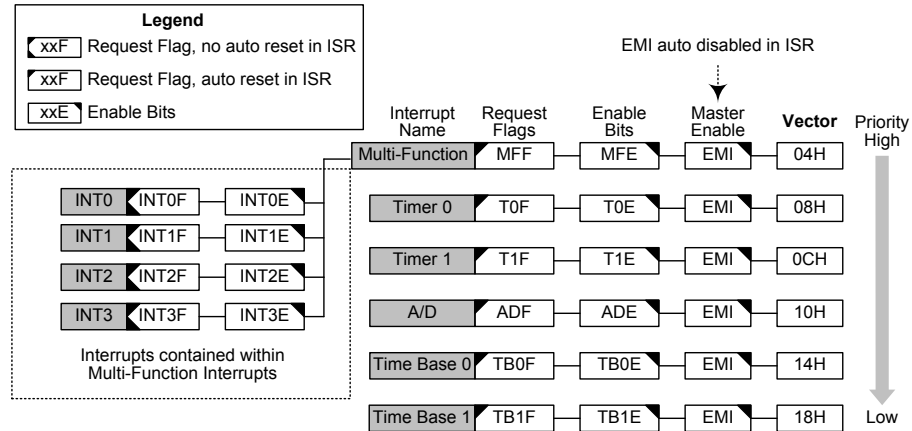
Bit 2 **INT2E:** INT2 interrupt enable
 0: disable
 1: enable

Bit 1 **INT1E:** INT1 interrupt enable
 0: disable
 1: enable

Bit 0 **INT0E:** INT0 interrupt enable
 0: disable
 1: enable

Interrupt Operation

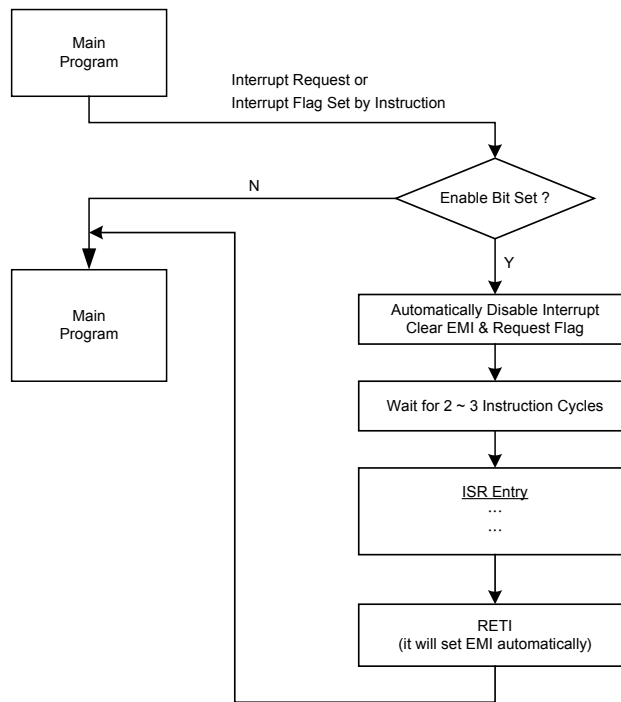
A Timer/Event Counter overflow, a Time Base event or an active edge on the external interrupt pin will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, the Program Counter, which stores the address of the 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 statement 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 instruction, 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 following diagram with their order of priority.



Interrupt Structure

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the 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.

When an interrupt request is generated it takes 2 or 3 instruction cycles before the program jumps to the interrupt vector. If the device is in the Sleep or Idle Mode and is woken up by an interrupt request then it will take 3 cycles before the program jumps to the interrupt vector.



Interrupt Flow

Interrupt Priority

Interrupts, occurring in the interval between the rising edges of two consecutive T2 pulses, will be serviced on the latter of the two T2 pulses, if the corresponding interrupts are enabled. In case of simultaneous requests, the following table shows the priority that is applied. These can be masked by resetting the EMI bit.

Interrupt Source	Priority	Vector
Multi-Function interrupt(External interrupt 0~3)	1	04H
Timer/Event Counter 0 overflow	2	08H
Timer/Event Counter 1 overflow	3	0CH
ADC interrupt	4	10H
Time Base 0 interrupt	5	14H
Time Base 1 interrupt	6	18H

In cases where both external and internal interrupts are enabled and where an external and internal interrupt occurs simultaneously, the external interrupt will always have priority and will therefore be serviced first. Suitable masking of the individual interrupts using the interrupt registers can prevent simultaneous occurrences.

Multi-function Interrupt

The device contains a Multi-function interrupt. Unlike the other independent interrupts, this interrupt has no independent source, but rather is formed from external interrupt source. A Multi-function interrupt request will take place when the Multi-function interrupt request flag, MFF is set. The Multi-function interrupt flag will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within the Multi-function interrupt occurs, a subroutine call to the Multi-function interrupt vector will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

The type of transition that will trigger the external interrupts, whether high to low, low to high or both is determined by the INTnS0 and INTnS1 bits, in the INTEG register. These bits can also disable the external interrupt function.

INTnS1	INTnS0	Edge Trigger Type
0	0	External interrupt disable
0	1	Rising edge Trigger
1	0	Falling edge Trigger
1	1	Both edge Trigger

The external interrupt pins are pin-shared with the I/O pins and can only be configured as external interrupt pins if the corresponding external interrupt enable bit in the MFIC register has been set and the edge trigger type has been selected using the INTEG register. The pins must also be setup as an input by setting the port control registers.

A/D Converter Interrupt

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

Timer/Event Counter Interrupt

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, TnE, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TnF, is set, a situation that will occur when the relevant Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter overflow occurs, a subroutine call to the relevant timer interrupt vector, will take place. When the interrupt is serviced, the timer interrupt request flag, TnF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Time Base Interrupts

Time Base Interrupts functions are 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.

CTRL1 Register

Bit	7	6	5	4	3	2	1	0
Name	T0S1	T0S0	TB01	TB00	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

Bit 7~6 Described elsewhere

Bit 5~4 **TB01, TB00:** Time base 0 period selection
 00: $f_s/2^{12}$
 01: $f_s/2^{13}$
 10: $f_s/2^{14}$
 11: $f_s/2^{15}$

Bit 3~0 Unimplemented, read as “0”

CTRL4 Register

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

Bit 7~3 Described elsewhere

Bit 2~0 **TB12~TB10**: Time Base 1 clock selection

000: $f_s/2^8$

001: $f_s/2^9$

010: $f_s/2^{10}$

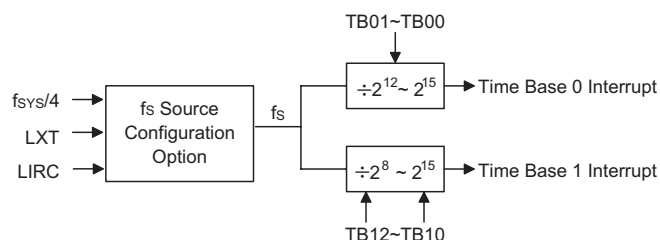
011: $f_s/2^{11}$

100: $f_s/2^{12}$

101: $f_s/2^{13}$

110: $f_s/2^{14}$

111: $f_s/2^{15}$



Time Base Interrupts

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins, a low power supply voltage or 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 a software instruction.

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flag, MFF, 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 entering the 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 in advance.

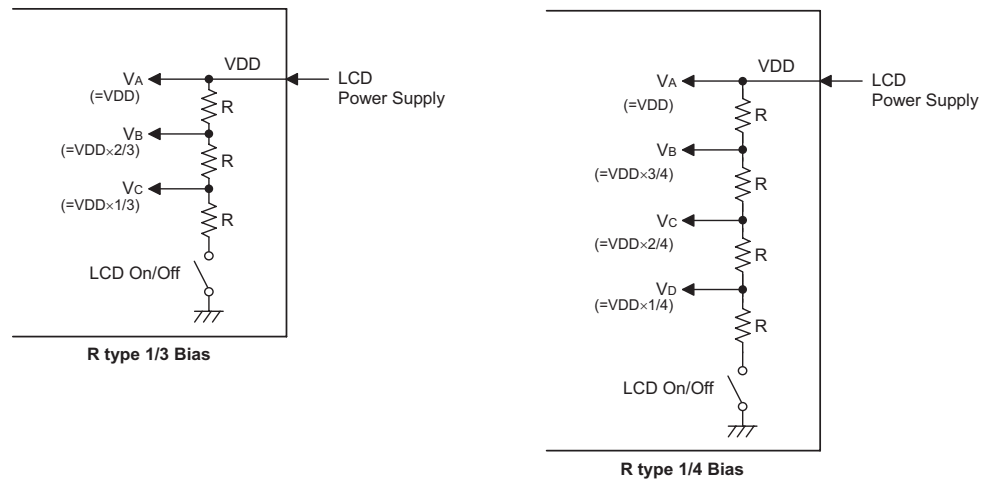
To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program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

LCD Function

For large volume applications, which incorporate an LCD in their design, the use of a custom display rather than a more expensive character based display reduces costs significantly. However, the corresponding COM and SEG signals required, which vary in both amplitude and time, to drive such a custom display require many special considerations for proper LCD operation to occur. This device contains an LCD Driver function, which with their internal LCD signal generating circuitry and various options, will automatically generate these time and amplitude varying signals to provide a means of direct driving and easy interfacing to a range of custom LCDs.

Duty	Driver No.	Bias	Bias Type	Wave Type
1/4	23×4	1/3 or 1/4	R	A or B
1/8	19×8			

LCD Selections

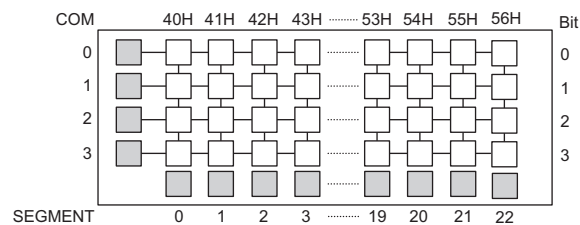


R Type Bias Voltage Levels

Display Memory

An area of Data Memory is especially reserved for use for the LCD display data. This data area is known as the Display Memory. Any data written here will be automatically read by the internal display driver circuits, which will in turn automatically generate the necessary LCD signals. Therefore any data written into this Memory will be immediately reflected into the actual display connected to the microcontroller. As the Display Memory addresses overlap those of the General Purpose Data Memory, it stored in its own independent Bank 1 area. The Data Memory Bank to be used is chosen by using the Bank Pointer, which is a special function register in the Data Memory, with the name, BP. To access the Display Memory therefore requires first that Bank 1 is selected by writing a value of 01H to the BP register. After this, the memory can then be accessed by using indirect addressing through the use of Memory Pointer MP1. With Bank 1 selected, then using MP1 to read or write to the memory area, starting with address 40H, will result in operations to the Display Memory. Directly addressing the Display Memory is not applicable and will result in a data access to the Bank 0 General Purpose Data Memory.

The accompanying Display Memory Map diagrams shows how the internal Display Memory is mapped to the Segments and Commons of the display for the device.



LCD Registers

Control Registers in the Data Memory, are used to control the various setup features of the LCD Driver. There are three control register for the LCD function, CTRL2, LCDC and LCDO. Various bits in these registers control functions such as duty type, bias type, bias resistor selection as well as overall LCD enable and disable. The LCDEN bit in the LCDC register, which provide the overall LCD enable/disable function, will only be effective when the device is in the Normal, Slow or Idle Mode. If the device is in the Sleep Mode then the display will always be disabled. Bits RSEL0 and RSEL1 in the LCDC register select the internal bias resistors to supply the LCD panel with the correct bias voltages. A choice to best match the LCD panel used in the application can be selected also to minimise bias current. The TYPE bit in the same register is used to select whether Type A or Type B LCD control signals are used.

The LCDO register is used to determine if the output function of display pins are used as segment drivers or CMOS outputs or I/O pins. The bits LCDSEL2~LCDSEL0 in the CTRL2 register are used to select LCD Clock Source.

CTRL2 register

Bit	7	6	5	4	3	2	1	0
Name	LCDSEL2	LCDSEL1	LCDSEL0	BZSEL2	BZSEL1	BZSEL0	BUZC	LXTEN
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	1

Bit 7~5 **LCDSEL2~LCDSEL0:** LCD Driver clock

000: fs/2²
001: fs/2³
010: fs/2⁴
011: fs/2⁵
100: fs/2⁶
101: fs/2⁷
110: fs/2⁸
111: reserved

Bit 4~0 Described elsewhere

LCDC register

Bit	7	6	5	4	3	2	1	0
Name	TYPE	—	—	—	BIAS	RSEL1	RSEL0	LCDEN
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **TYPE:** LCD Type A or Type B

0: Type A
1: Type B

Bit 6~4 Unused bit, read as “0”

Bit 3 **BIAS:** Define LCD Bias

0: 1/3 Bias; 1/4 Duty
1: 1/4 Bias; 1/8 Duty

Bit 2~1 **RSEL1, RSEL0:** Select resistor for R type LCD bias current

00: 600kΩ
01: 300kΩ
10: 100kΩ
11: 30kΩ

Bit 0 **LCDEN:** LCD enable/disable control

0: disable
1: enable

LCDO register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	LCDO3	LCDO2	LCDO1	LCDO0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unused bit, read as “0”

Bit 3 **LCDO3:** LCD COM or I/O function control bit(COM0~COM3)

0: I/O
1: LCD COM

Bit 2 **LCDO2:** LCD SEG or I/O function control bit(SEG16~SEG22)

0: I/O or A/D function
1: LCD SEG or COM

- Bit 1 **LCDO1:** LCD SEG or I/O function control bit(SEG8~SEG15)
 0: I/O
 1: LCD SEG
- Bit 0 **LCDO0:** LCD SEG or I/O function control bit(SEG0~SEG7)
 0: I/O
 1: LCD SEG

LCD Clock Source

The LCD clock source is the internal clock signal, f_s , divided using an internal divider circuit. The f_s internal clock is supplied by the LXT oscillator, $f_{sys}/4$ or the LIRC oscillator; the choice of which is determined by a configuration option. For proper LCD operation, this arrangement is provided to generate an ideal LCD clock source frequency of 4kHz.

The options of LCD clock frequency are listed in the following table

f_s Clock Source	LCD Clock Selection
LIRC	$LIRC/2^3$
LXT	$LXT/2^3$
$f_{sys}/4$	$f_{sys}/2^4 \sim f_{sys}/2^{10}$

LCD Driver Output

The nature of Liquid Crystal Display requires that only AC voltage can be applied to its pixel as the application of DC voltage to LCD pixel may cause permanent damage. For this reason the relative contrast of an LCD display is controlled by the actual RMS voltage applied to each pixel, which is equal to the RMS value of the voltage on the COM pin minus the voltage applied to the SEG pin. This differential RMS voltage must be greater than the LCD saturation voltage for the pixel to be on and less than the threshold voltage for the pixel to be off. The requirement to limit the DC voltage to zero and to control as many pixels as possible with a minimum number of connections, requires that both a time and amplitude signal is generated and applied to the application LCD. These time and amplitude varying signals are automatically generated by the LCD driver circuits in the microcontroller. What is known as the duty determines the number of common lines used, which are also known as backplanes or COMs. The duty, which is chosen by a control bit to have a value of 1/4, 1/8 and which equates to a COM number of 4, 8, therefore defines the number of time divisions within each LCD signal frame. Two types of signal generation are also provided, known as Type A and Type B, the required type is selected via the TYPE bit in the LCDC register. Type B offers lower frequency signals, however lower frequencies may introduce flickering and influence display clarity.

LCD Voltage Source and Biasing

The time and amplitude varying signals generated by the LCD Driver function require the generation of several voltage levels for their operation. The number of voltage levels used by the signal depends upon the value of the BIAS bit in the LCDC register. The device has “R” type biasing only. LCD voltage source is V_{DD} . For the R type 1/3 bias selection, five voltage levels V_{SS} , V_A , V_B , V_C and V_D are utilised. The voltage V_A is equal to V_{DD} , V_B is equal to $V_{DD} \times 2/3$, while V_C is equal to $V_{DD} \times 1/3$. For the R type 1/4 bias selection, five voltage levels V_{SS} , V_A , V_B , V_C and V_D are utilised. The voltage V_A is equal to V_{DD} , V_B is equal to $V_{DD} \times 3/4$, V_C is equal to $V_{DD} \times 2/4$, while V_D is equal to $V_{DD} \times 1/4$. In addition to selecting 1/3 or 1/4 bias, several values of bias resistor can be chosen using bits in the LCDC register. Different values of internal bias resistors can be selected using the RSEL0 and RESEL1 bits in the LCDC register.

Low Voltage Detector – LVD

The device has a Low Voltage Detector function, also known as LVD. This enabled the device to monitor the power supply voltage, V_{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, $V_{LVD2} \sim V_{LVD0}$, 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 LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

LVDC Register

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

Bit 7~6 Unimplemented, read as "0"

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

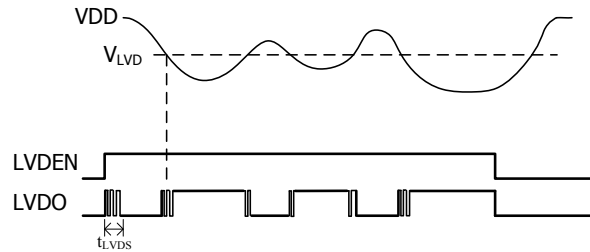
Bit 4 **LVDEN**: Low Voltage Detector Control
0: disable
1: enable

Bit 3 Unimplemented, read as "0"

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

LVD Operation

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



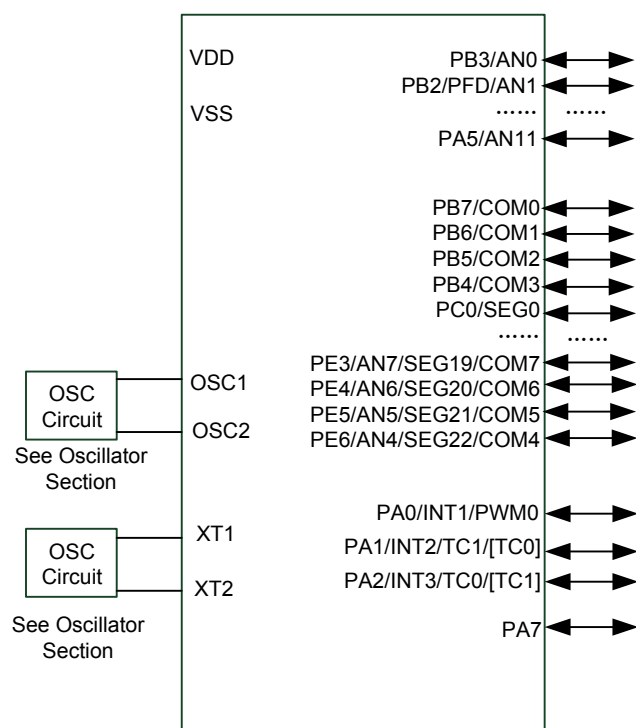
LVD Operation

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later using the application program. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
1	Watchdog Timer Function: Always Enable or by S/W control
2	fs clock source: LIRC, f_{SYS} or LXT
3	HIRC Frequency Selection: 4MHz, 8MHz or 12MHz
4	System oscillator configuration: HXT, HIRC, ERC, HIRC+LXT
5	Always Enabled or Application Program Enabled

Application Circuits



Instruction Set Summary

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

Table Conventions

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

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

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1 ^{Note}	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1 ^{Note}	None
SET [m].i	Set bit of Data Memory	1 ^{Note}	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1 ^{Note}	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1 ^{Note}	None
SZ [m].i	Skip if bit i of Data Memory is zero	1 ^{Note}	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1 ^{Note}	None
SIZ [m]	Skip if increment Data Memory is zero	1 ^{Note}	None
SDZ [m]	Skip if decrement Data Memory is zero	1 ^{Note}	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1 ^{Note}	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1 ^{Note}	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 to TBLH and Data Memory	2 ^{Note}	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2 ^{Note}	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1 ^{Note}	None
SET [m]	Set Data Memory	1 ^{Note}	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	1 ^{Note}	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	$ACC \leftarrow ACC + [m] + C$
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	$[m] \leftarrow ACC + [m] + C$
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	$ACC \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC + x$
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	$[m] \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC \text{ "AND" } [m]$
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 bitwise logical AND operation. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC \text{ "AND" } x$
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	$[m] \leftarrow ACC \text{ "AND" } [m]$
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 \leftarrow Program Counter + 1 Program Counter \leftarrow addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] \leftarrow 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 \leftarrow 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 \leftarrow 0 PDF \leftarrow 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 \leftarrow 0 PDF \leftarrow 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 \leftarrow 0 PDF \leftarrow 0
Affected flag(s)	TO, PDF

CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	$[m] \leftarrow \overline{[m]}$
Affected flag(s)	Z
CPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow \overline{[m]}$
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	$[m] \leftarrow ACC + 00H$ or $[m] \leftarrow ACC + 06H$ or $[m] \leftarrow ACC + 60H$ or $[m] \leftarrow ACC + 66H$
Affected flag(s)	C
DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	$[m] \leftarrow [m] - 1$
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	$ACC \leftarrow [m] - 1$
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	$TO \leftarrow 0$ $PDF \leftarrow 0$
Affected flag(s)	TO, PDF
INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	$[m] \leftarrow [m] + 1$
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	$ACC \leftarrow [m] + 1$
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	$Program\ Counter \leftarrow addr$
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	$ACC \leftarrow [m]$
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	$ACC \leftarrow x$
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	$[m] \leftarrow ACC$
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	$ACC \leftarrow ACC \text{ "OR" } [m]$
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	$ACC \leftarrow ACC \text{ "OR" } x$
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	$[m] \leftarrow ACC \text{ "OR" } [m]$
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	$\text{Program Counter} \leftarrow \text{Stack}$
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	$\text{Program Counter} \leftarrow \text{Stack}$ $ACC \leftarrow x$
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	$\text{Program Counter} \leftarrow \text{Stack}$ $EMI \leftarrow 1$
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	$[m].(i+1) \leftarrow [m].i; (i = 0 \sim 6)$ $[m].0 \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i = 0 \sim 6)$ $ACC.0 \leftarrow [m].7$
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	$[m].(i+1) \leftarrow [m].i; (i = 0 \sim 6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i = 0 \sim 6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
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	$[m].i \leftarrow [m].(i+1); (i = 0 \sim 6)$ $[m].7 \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1); (i = 0 \sim 6)$ $ACC.7 \leftarrow [m].0$
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	$[m].i \leftarrow [m].(i+1); (i = 0 \sim 6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1); (i = 0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
SBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C
SBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C
SDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] - 1$ 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	$ACC \leftarrow [m] - 1$
Affected flag(s)	Skip if $ACC = 0$ None

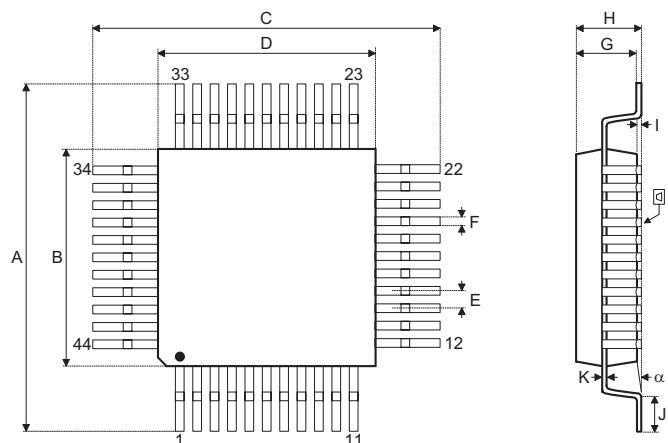
SET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	$[m] \leftarrow FFH$
Affected flag(s)	None
SET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	$[m].i \leftarrow 1$
Affected flag(s)	None
SIZ [m]	Skip if 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	$[m] \leftarrow [m] + 1$ 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	$ACC \leftarrow [m] + 1$ 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 $[m].i \neq 0$
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	$ACC \leftarrow ACC - [m]$
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	$[m] \leftarrow ACC - [m]$
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	$ACC \leftarrow ACC - x$
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	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
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	$ACC \leftarrow [m]$ 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 to TBLH and Data Memory
Description	The program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
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

44-pin QFP (10mm×10mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.512	—	0.528
B	0.390	—	0.398
C	0.512	—	0.528
D	0.390	—	0.398
E	—	0.031	—
F	—	0.012	—
G	0.075	—	0.087
H	—	—	0.106
I	0.010	—	0.020
J	0.029	—	0.037
K	0.004	—	0.008
L	—	0.004	—
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	13.00	—	13.40
B	9.90	—	10.10
C	13.00	—	13.40
D	9.90	—	10.10
E	—	0.80	—
F	—	0.30	—
G	1.90	—	2.20
H	—	—	2.70
I	0.25	—	0.50
J	0.73	—	0.93
K	0.10	—	0.20
L	—	0.10	—
α	0°	—	7°

Holtek Semiconductor Inc. (Headquarters)

No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
<http://www.holtek.com.tw>

Holtek Semiconductor Inc. (Taipei Sales Office)

4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan
Tel: 886-2-2655-7070
Fax: 886-2-2655-7373
Fax: 886-2-2655-7383 (International sales hotline)

Holtek Semiconductor Inc. (Shenzhen Sales Office)

5F, Unit A, Productivity Building, No.5 Gaoxin M 2nd Road, Nanshan District, Shenzhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

Holtek Semiconductor (USA), Inc. (North America Sales Office)

46729 Fremont Blvd., Fremont, CA 94538, USA
Tel: 1-510-252-9880
Fax: 1-510-252-9885
<http://www.holtek.com>

Copyright© 2011 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>.