Two Way Radio 8-Bit MCU

HT98R068-1

Revision: 1.10 Date: February 13, 2012

www.holtek.com

Table of Contents

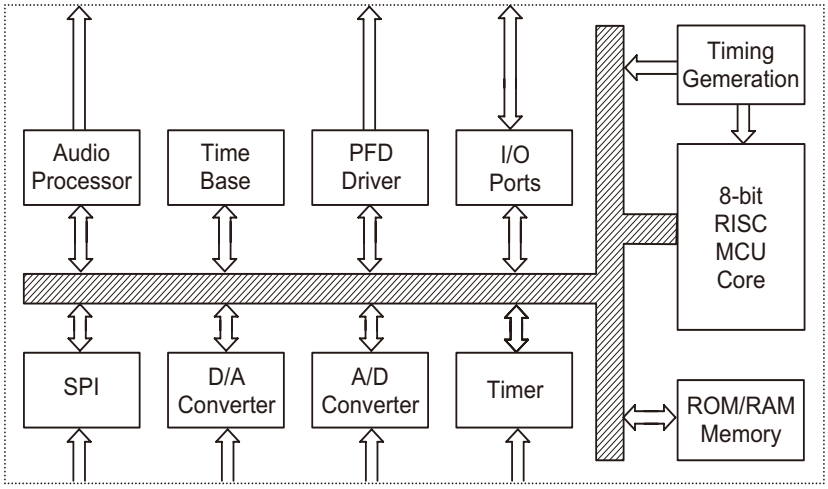
Features

- Operating voltage: 3V~5V
- Up to 0.25μs instruction cycle with 16MHz system clock
- OTP Program Memory:16K x 16
- Data Memory: 1K x8
- Up to 34 bidirectional I/O lines
- Ocillator type: external 32768Hz crystal -- LXT
- Internal PLL to generate the system clock
  - Input reference clock : 32768Hz
  - Output: 2.048MHz x4/x5/x6/x8 (optional)
- Four operational modes: Normal, Slow, Idle, Sleep
- Watchdog Timer function with WDT oscillator
- Real Time Clock function
- 8 channel 12-bit A/D converter
- 4 channel 8-bit D/A converter
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- Up to 10-level subroutine nesting
- Bit manipulation instruction
- Low voltage reset function
- Low voltage detect function
- SPI interface for external MCU control
- Two 8-bit and signal 16-bit programmable Timer/Event Counter with overflow interrupt and prescaler
- Programmable Frequency Divider -- PFD
- PGA : 5bit Operational Amplifier Gain setup
- Sub-tone processor
  - CTCSS/DCS encode/decode
- In-band-Tone processor
  - DTMF encoder/decode
  - Selective call encoder/decoder
  - User defined tone encoder/decoder
- Audio processor
  - Pre-emphasis/De-emphasis
  - Scrambler
  - Compandor
  - VOX
- Package types: 48/64-pin LQFP

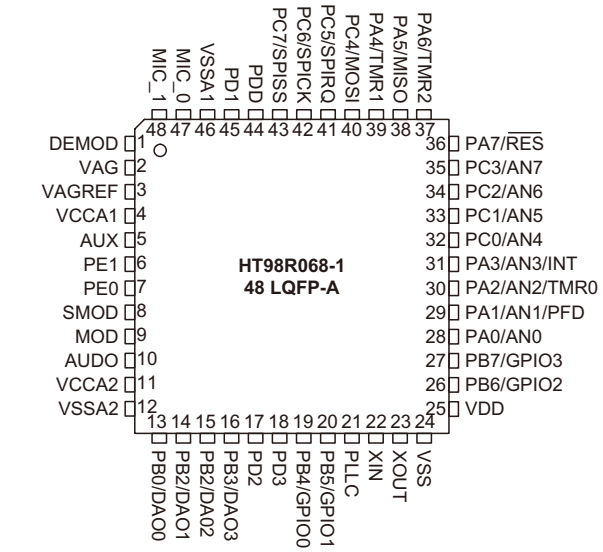
General Description

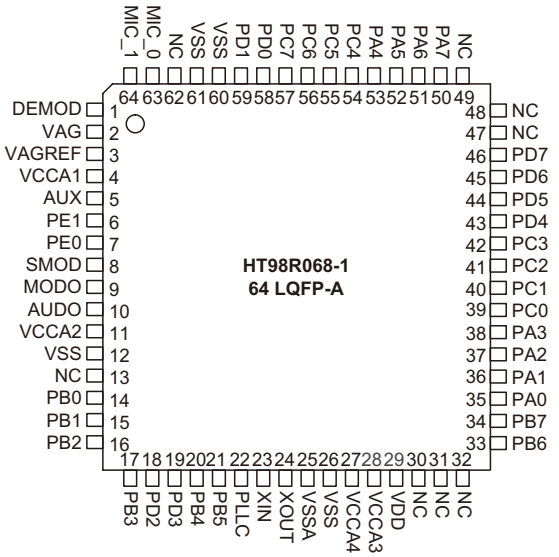
The device is an ASSP MCU for analog two way radio applications such as FRS. This device contains an 8-bit MCU, Audio Processor, ADC, DAC as well as an Operational Amplifier for peripheral interfaces. Additionally there are external interrupt functions and 16K of OTP Program Memory and 1K of SRAM Data Memory for general MCU control application. The Internal Audio Processor supports various programmable functions. These functions include CTCSS/DCSS encoding/decoding, DTMF encoding/decoding, scramble/descramble, VOX, compandor and so on. By connecting to a suitable RF module, the device provides a cost effective and flexible audio RF solution. Applications should include products in the Leisure Radio application area such as General Mobile Radios, Personal Mobile Radios, Multiple User Radios etc.

Block Diagram



Pin Assignment





Pin Description

Pad Name	Function	OPT	I/T	O/T	Description
PA0/AN0	PA0	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN0	ANCSR	AN	—	A/D channel 0
PA1/PFD/AN1	PA1	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PFD	CTRL0	—	CMOS	PFD output
	AN1	ANCSR	AN	—	A/D channel 1
PA2/TMR0/AN2	PA2	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	AN2	ANCSR	AN	—	A/D channel 2
	TMR0	—	ST	—	Timer/Event Counter 0 clock input
PA3/INT/AN3	PA3	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	—	ST	—	External interrupt input
	AN3	ANCSR	AN	—	A/D channel 3
PA4/TMR1	PA4	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TMR1	—	ST	—	Timer/Event Counter 1 clock input
PA5/MISO	PA5	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	MISO	SPICR	—	—	output data pin of slave SPI
PA6/TMR2	PA6	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TMR2	—	ST	—	Timer/Event Counter 2 clock input
PA7/$\overline{\text{RES}}$	PA7	PAWK	ST	NMOS	General purpose I/O. Register enabled wake-up.
	$\overline{\text{RES}}$	CO	ST	—	Reset input
PB0/DAO0	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	DAO0	DACC	—	AN	DAC0 output
PB1/DAO1	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	DAO1	DACC	—	AN	DAC1 output
PB2/DAO2	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	DAO2	DACC	—	AN	DAC2 output
PB3/DAO3	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	DAO3	DACC	—	AN	DAC3 output
PB4/GPIO0	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	GPIO0	SPICR	—	—	Audio processor 0 I/O
PB5/GPIO1	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	GPIO1	SPICR	—	—	AUDIO PROCESSOR 1 I/O
PB6/GPIO2	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	GPIO2	SPICR	—	—	AUDIO PROCESSOR 2 I/O
PB7/GPIO3	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	GPIO3	SPICR	—	—	AUDIO PROCESSOR 3 I/O
XIN	XIN	CO	LXT	—	Low frequency crystal input pin
XOUT	XOUT	CO	—	LXT	Low frequency crystal output pin

Pad Name	Function	OPT	I/T	O/T	Description
PC0/AN4 PC1/AN5 PC2/AN6 PC3/AN7	PCn	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	ANn	ADCSR	AN	—	A/D channel 4, 5, 6, 7
PC4/MOSI	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	MOSI	SPICR	—	—	input data pin of slave SPI
PC5/SPIRQ	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SPIRQ	SPICR	—	—	output pin of slave SPI
PC6/SPICK	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SPICK	SPICR	—	—	clock source input pin.
PC7/SPISS	PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SPISS	SPICR	—	—	AUDIO PROCESSOR select
PD0~PD7	PDn	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
PE0~PE1	PEn	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
MIC_I	MIC_I	—	—	—	Microphone OP input
MIC_O	MIC_O	—	—	—	Microphone OP output
AUX	AUX	—	—	—	Auxiliary audio input
AUDO	AUDO	—	—	—	AUDIO output
MOD	MOD	—	—	—	Audio baseband modulation output to RF module
SMOD	SMOD	—	—	—	Sub audio baseband modulation output
DEMOD	DEMOD	—	—	—	Input from RF demodulation output
VSSA1/VSSA2	VSSAn	—	—	—	Analog Block GND
VCCA1/VCCA2/ VCCA3	VCCAn	—	—	—	Analog Block VCC
VCCA4	VCCAn	—	—	—	Audio Processor VCC
VGA	VGA	—	—	—	Audio ADC analog ground output
VGAREF	VGAREF	—	—	—	Audio ADC analog ground reference bypass
PLLC	PLLC	—	—	—	Connect to low pass loop filter circuit
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, GND

Note: 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; NMOS: NMOS output

AN: Analog input or output

LXT: Low frequency crystal oscillator

Absolute Maximum Ratings

Supply Voltage ..... V_{SS} -0.3V to 6.0V
Input Voltage ..... V_{SS} -0.3V to V_{DD} +0.3V
Storage Temperature ..... -50°C to 125°C
Operating Temperature ..... -40°C to 85°C

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

Operating Temperature: -40°C to 85°C Ta=25°C Typical

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=32768Hz	2.2	—	5.5	V
		—	f_{SYS}=2MHz~12MHz	3.0	—	5.5	V
		—	f_{SYS}=16MHz	3.3	—	5.5	V
I_{DD1}	Operating Current (32k OSC+PLL)	3V	No load, f_{SYS}=4MHz ADC, DAC disable	—	1	2	mA
		5V		—	2.5	5.0	mA
I_{DD2}	Operating Current (32k OSC+PLL)	3V	No load, f_{SYS}=8MHz	—	2	4	mA
		5V	ADC, DAC disable	—	4	8	mA
I_{DD3}	Operating Current (32k OSC+PLL)	3V	No load, f_{SYS}=10MHz	—	2	4	mA
		5V	ADC, DAC disable	—	4	8	mA
I_{DD4}	Operating Current (32k OSC+PLL)	3V	No load, f_{SYS}=12MHz ADC, DAC disable	—	6	12	mA
I_{DD5}	Operating Current (32k OSC, PLL OFF)	3V	No load, f_{SYS}=32768Hz ADC, DAC disable	—	20	30	μA
		5V		—	40	60	μA
I_{DD6}	Operating Current (Audio Processing Turn On)	3V	No load, f_{SYS}=8MHz see note 2	—	15	—	mA
							mA
I_{STB1}	Standby Current (WDTOSC On, 32k OSC Off)	3V	No load, System halt	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	Standby Current (WDTOSC Off, 32k OSC Off)	3V	No load, System halt	—	—	1	μA
		5V		—	—	2	μA
I_{STB3}	Standby Current (WDTOSC Off, 32k OSC On)	3V	No load, System halt LXT OSC slowly start-up	—	—	5	μA
		5V		—	—	10	μA
V_{IL1}	Input Low Voltage For PA, PB, PC, PD, PE, TMR0, TMR1, INT	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage For PA, PB, PC, PD, PETMR0, TMR1, INT	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
V_{LVR1}	Low Voltage Reset Voltage	—	LVR 2.10V option	1.98	2.10	2.22	V
V_{LVR2}	Low Voltage Reset Voltage	—	LVR 3.15V option	2.98	3.15	3.32	V
V_{LVR3}	Low Voltage Reset Voltage	—	LVR 4.20V option	3.98	4.20	4.42	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH1}	I/O Source Current (PA, PB, PC, PD, PE)	3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
		5V		-5	-10	—	mA
I_{OL1}	I/O Sink Current (PA, PB, PC, PD, PE)	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V		10	20	—	mA
I_{OL2}	PA7 Sink Current	5V	V_{OL}=0.1V_{DD}	2	3	—	mA
R_{PH}	Pull-high Resistance (I/O)	3V	—	20	60	100	KΩ
		5V	—	10	30	50	KΩ
V_{AD}	AD Input Voltage	—	—	0	—	V_{DD}	V
E_{AD}	A/D Conversion Error	—	—	—	±0.5	±1.0	LSB
I_{ADC}	t_{AD}=1μs, Only 8Bit ADC Enable, Others Disable	3V	No load	—	0.5	1.0	mA
		5V	No load	—	1.5	3.0	mA
I_{DAC}	Only 8 Bit DAC Enable, Others Disable	3V	No load	—	300	—	μA
		5V	No load	—	500	—	μA
I_{AUDOH}	AUDO Source Current	3V	V_{OH}=0.9V_{DD}	—	-4	—	mA
I_{AUDOL}	AUDO Sink Current	3V	V_{OL}=0.1V_{DD}	—	8	—	mA
R_{AUX_ON}	AUX Input Resistance When Channel Turn On	—	—	—	300	—	Ω
R_{AUX_OFF}	AUX Input Resistance When Channel Turn Off	—	—	1	—	—	MΩ
R_{DEMOD_ON}	DEMOD Input Resistance When MUX Channel Turn On	—	—	—	300	—	KΩ
R_{DEMOD_OFF}	DEMOD Input Resistance When MUX Channel Turn Off	—	—	1	—	—	MΩ
R_{MOD_ON}	MOD Output Resistance When DAO1 OPA Turn On	—	—	—	300	—	Ω
R_{MOD_OFF}	DEMOD Input Resistance When DAO1 OPA Turn Off	—	—	—	500	—	KΩ
R_{SMOD_ON}	DEMOD Input Resistance When DAO2 OPA Turn On	—	—	—	300	—	Ω
R_{SMOD_OFF}	DEMOD Input Resistance When DAO2 OPA Turn Off	—	—	—	500	—	KΩ
R_{L1}	Load Resistance for MOD, SMOD	—	—	20	—	—	KΩ
VAG	—	—	—	—	1/2V_{DD}	—	V
VAGREF	—	—	—	0.90	1.15	1.35	V

Note: 1. PA0~PA6 is connected to external pull-up resister when $\overline{RES}$=low.

2. ADC/DAC in MCU, Audio scrambler, CTCSS, DCS, compandor and pre/de-emphasis disabled, but all other digital circuits (including the Main Clock PLL) enabled. A single analogue path is enabled through the device.

A.C. Characteristics

Operating Temperature: -40°C to 85°C Ta=25°C Typical

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS1}	System Clock (PLL)	—	3.0V~5.5V	2	—	12	MHz
		—	3.3V~5.5V	2	—	16	MHz
f_{SYS2}	System Clock (32768 Crystal)	—	—	—	32768	—	Hz
f_{TIMER}	Timer I/P Frequency (TMR)	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
		—	4.5V~5.5V	0	—	12000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES1}	MCU External Reset Low Pulse Width	—	—	1	—	—	μs
t_{RES2}	Audio Processor Reset (AUPRST) Low Pulse Width	—	—	1	—	—	μs
t_{SST1}	MCU System Start-up Time Period	—	Power-up, Reset or Wake-up from HALT MCU f_{SYS}= 32768Hz	—	1024	—	t_{SYS}
t_{SST2}	Audio Processor System Start-up Time Period From Reset	—	SPI command available after release reset	200	—	—	ms
t_{SST3}	Audio Processor System Start-up Time Period From Sleep	—	SPI command available from wakeup	2	—	—	μs
t_{STRT}	Audio Processor System Sleep Ready Time From Active	—	Audio processor into sleep after hold command	2	—	—	t_{SYS}
t_{PST} t_{FUP}	PLL Settling Time For PLL Freq Deviation < ±0.1%	—	2.2V~5.5V	6	—	10	ms
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1.00	2.00	ms
V_{MIC}	MIC Input Voltage Range	3.3V	Gain= -1 & PGAgain= 0dB	—	—	±800	mV
V_{DEMOD}	DEMOD Input Voltage Range	3.3V	PGAgain= 0dB	—	—	±800	mV
V_{AUX}	AUX Input Voltage Range	3.3V	PGAgain= 0dB	—	—	±800	mV
V_{AUDO}	AUDIO Output Voltage Range	3V	No load	0.01	—	0.99	V_{DD}
V_{MOD}	MOD Output Voltage Range	3V	No load	1/4	—	3/4	V_{DD}
V_{SMOD}	SMOD Output Voltage Range	3V	No load	1/4	—	3/4	V_{DD}

ADC Electrical Characteristics

Operating Temperature: -40°C to 85°C, Ta=25°C Typical

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
DNL	Differential Non-linearity	3V	$V_{REF}=AV_{DD}=V_{DD}$ $t_{AD}=0.5\mu s$	-2	—	+2	LSB
		5V					
INL	Integral Non-linearity	3V	$V_{REF}=AV_{DD}=V_{DD}$ $t_{AD}=0.5\mu s$	-4	—	+4	LSB
		5V					
I_{ADC}	Only ADC Enable, Others Disable	3V	No load (t_{AD}=0.5μs)	—	0.5	—	mA
		5V	No load (t_{AD}=0.5μs)	—	0.6	—	mA
t_{AD}	A/D Clock Period	2.7V~5.5V	—	0.5	—	10.0	μs
t_{ADC}	AD Conversion Time (Note)	2.7V~5.5V	12 bit ADC	—	16	—	t_{AD}
t_{ON2ST}	ADC On to ADC Start	2.7V~5.5V	—	2	—	—	μs

Note: ADC conversion time (t_{ADC})= n (bits ADC) + 4 (sampling time), the conversion for each bit needs one ADC clock(t_{AD})

DAC Electrical Characteristics

Operating Temperature: -40°C to 85°C, Ta=25°C Typical

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Analog Operating Voltage	—	Int reference voltage	2.7	—	5.5	V
V_{DA}	DA Output Voltage	—	00H~FFH, No load	0.01	—	0.99	V_{DD}
t_{DAC}	DA Conversion Time	—	V_{DD}=3V, CL=10P	—	—	5	μs
INL	Integral Non-linearity	3V	$V_{REF}=AV_{DD}=V_{DD}$	-2	—	2	LSB
DNL	Differential Non-linearity	3V	$V_{REF}=AV_{DD}=V_{DD}$	-1	—	1	LSB
THD_{DA}	Total Harmonic Distortion	—	—	—	-54	-48	db
RO	DA Output Resistance	—	—	—	5	—	kΩ

Power on Reset Electrical Characteristics

Symbol	Parameter	V_{DD}	Condition	Min.	Typ.	Max.	Unit
V_{POR1}	VDD Start Voltage To Ensure Power-on Reset And MCU Work	—	• $\overline{RES}$ pin connect to VDD or disable $\overline{RES}$ pin function • LVR and WDT disable • Without 0.1μF between VDD and VSS • f_{SYS}=32kHz, VDD drops from 2.2V to the VPOR1 voltage with a 0.50V/ms slope, then stops for 1 second, when rising from the VPOR1 voltage to 2.5V with a 0.02V/ms slope, MCU PDF and TO flag are "0" and operates normally.	—	—	100	mV
R_{POR_AC1}	VDD Rise Rate To Ensure Power-on Reset And MCU Work	—	• $\overline{RES}$ pin connected to VDD or disable $\overline{RES}$ pin function • LVR and WDT disabled • Without 0.1μF between VDD and VSS • f_{SYS}=32kHz, VDD drops from 2.2V to 0.5V with a 0.5V/ms slope, then stops for 1 second, when rising from 0.5V to 2.5V with a RPOR_ AC1 slope, MCU PDF and TO flag are "0" and operates normally.	0.035	—	—	V/ms

Symbol	Parameter	V_{DD}	Condition	Min.	Typ.	Max.	Unit
V_{DCPOR_MAX}	DC POR Maximum Voltage	—	• $\overline{RES}$ pin connected to VDD or disable $\overline{RES}$ pin function • LVR and WDT disabled • Without 0.1μF between VDD and VSS • VDD drops from 5V with a 0.1V/S slope, until the POR voltage is generated.	0.6	—	1.8	V
t_{POR1}	Power-on Reset Low Pulse Width	—	• $\overline{RES}$ pin connect to VDD or disable $\overline{RES}$ pin function • LVR and WDT disable • Without 0.1μF between VDD and VSS • VDD drops from 5.0V to 0.5V at high speed, stops for the t_{POR1} time, then arrives at 5.0V at high speed, and will generate a POR time.	2	—	—	μs
t_{POR2}	Power-on Reset Low Pulse Width	—	• $\overline{RES}$ pin connected to VDD or disable $\overline{RES}$ pin function • LVR and WDT disabled • With 0.1μF between VDD and VSS • VDD drops from 5.0V to 0.5V at high speed, stops for the t_{POR2} time, then arrives at 5.0V at high speed, and will generate a POR time.	10	—	—	μs

Audio Processor Characteristics

Operating Temperature: 10°C to 70°C, Ta=25°C Typical V_{DD}= 3.3V
Input stage gain = 0dB. Output stage attenuation = 0dB.
Xtal Frequency = 32.768kHz ± 0.01% (100ppm), f_{SYS1} Drift <± 0.1%
Reference signal level is 780mVrms at 1kHz with V_{DD}= 3.3V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
CTCSS Receiver							
	Sensitivity	3.3V		—	28	—	mV-peak
	Response Time	3.3V	Note1	168	210		ms
	De-response Time	3.3V	—	—	DT1+150	—	ms
DT1	Dropout Immunity	3.3V	—	0	300	8192	ms
	Frequency Range	3.3V	—	60	—	260	Hz
DCS Receiver							
	Sensitivity	3.3V		72	—	—	mVrms
	Response Time	3.3V	—	—	200	—	ms
	De-response Time	3.3V	—	—	DT2+90	—	ms
	Dropout Immunity	3.3V	—	0	—	8192	ms
CTCSS/DCS off tone							
	Sensitivity	3.3V		28	—	—	mV-peak
	Response Time	3.3V	—	—	90	—	ms
	De-response Time	3.3V	—	—	160	—	ms
Selective Call/User Tone & In-band Tone Receiver							
	Sensitivity	3.3V		—	70	—	mV-peak
	Response Time	3.3V	—	—	50	—	ms
	De-response Time	3.3V	—	—	45	—	ms
	Frequency Range	3.3V	—	300	—	3000	Hz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
DTMF Receiver							
	Sensitivity	3.3V		—	200	—	mVrms
	Response Time	3.3V	Note 2	—	45	—	ms
	De-response Time	3.3V	—	—	35	—	ms
	Frequency Tolerance	3.3V	—	—	±1.8	—	%
	Frequency Rejection	3.3V	—	—	±3.6	—	%
Audio Compandor							
	Attack Time	3.3V	—	—	4	—	ms
	Decay Time	3.3V	—	—	14	—	ms
	0dB Point	3.3V	—	—	100	—	mVrms
	Compression/Expansion Ratio	3.3V	—	—	—	—	
CTCSS Generator							
	Frequency Range	3.3V	—	67	—	275	Hz
	Tone Frequency Accuracy	3.3V	—	—	—	±0.2	%
	Total Harmonic Distortion	3.3V	—	—	1.0	3.0	%
DCS Generator							
	Bit Rate	3.3V	—	—	134.4	—	Bps
In-band Tone Generator							
	Frequency Range	3.3V	—	300	1000	3000	Hz
	Tone Frequency Accuracy	3.3V	—	—	—	±0.1	%
	Total Harmonic Distortion	3.3V	—	—	1.0	3.0	%
DTMF Generator							
	Low Group Frequency Range	3.3V	—	697	—	941	Hz
	High Group Frequency Range	3.3V	—	1209	—	1633	Hz
	Output Signal level	3.3V	—	—	600	—	mVrms
Audio channel filter							
	Received Audio	3.3V	—	300	—	3400	Hz
	12.5kHz Channel Transmitted Audio	3.3V	—	300	—	2550	Hz
	25kHz Channel Transmitted Audio	3.3V	—	300	—	3000	Hz
	Pass-band Gain	3.3V	Frequency= 1.0kHz	—	0	—	dB
	Pass-band Ripple	3.3V	Frequency= 1.0kHz	-3.0	0	+0.3	dB
	Stop-band Attenuation	3.3V	For 12.5K filter stop-band at 3.4K ,for 25K filter stop-band at 3.7K.	30	—	—	dB
	De-emphasis	3.3V	—	—	-6	—	dB/oct
	Pre-emphasis	3.3V	—	—	+6	—	dB/oct
HPF300 Filter							
	Pass-band Gain	3.3V	Frequency = 1.0kHz	—	0	—	dB
	Pass-band Ripple	3.3V	Frequency = 1.0kHz	-3.0	0	+0.3	dB
	Stop-band Attenuation	3.3V	Passband 300Hz , stopband 250Hz	30	—	—	dB
Audio Scrambler							
	Inversion Frequency	3.3V	—	—	3300	—	Hz
	Pass Band (Selecting 3300Hz Inversion Frequency)	3.3V	—	300	—	3000	Hz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Audio Expandor							
	Input Signal Range	3.3V	—	—	—	333	mVrms

Note: 1. Send a signal of 67Hz, its threshold amplitude is 28mV-peak.
2. The DTMF threshold amplitude is set as 200mVrms for both tones.
3: All adjusted or default decoder threshold should be set greater than the system noise flow, otherwise the corresponding function does not work normally. All threshold values would be scaled according to the operating voltage.
4: DT(1 and 2) setting reference to 04DE address.

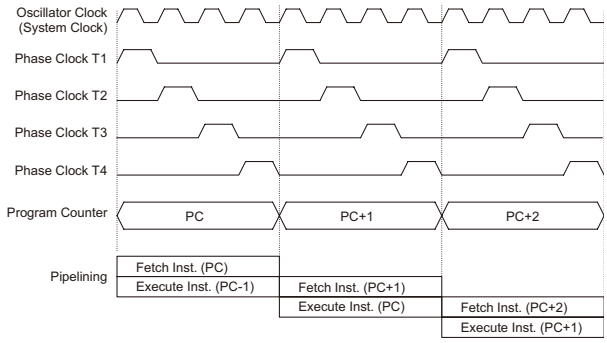
System Architecture

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

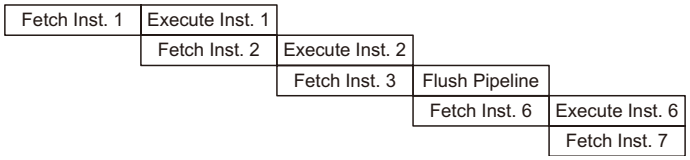
The system clock, derived from an RC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the decoding and execution 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 first obtain the actual jump or call address and then another cycle to actually execute the branch. The requirement for this extra cycle should be taken into account by programmers in timing sensitive applications.



System Clocking and Pipelining

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



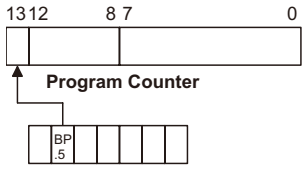
Instruction Fetching

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

Program Counter	
Program Counter High Byte	**PCL Register Low Byte**
PC13~PC8	PCL7~PCL0

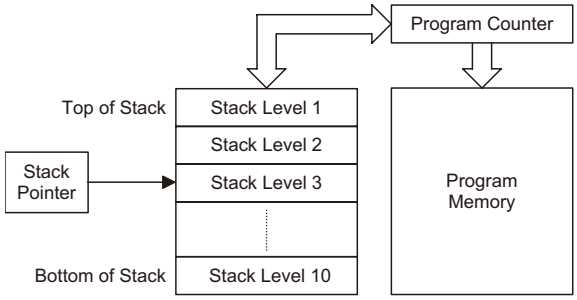


Bank Pointer (BP)

The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available for manipulation, the jumps are limited to the present page of memory, that is 256 locations. When such program jumps are executed it should also be noted that a dummy cycle will be inserted. 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. Further information on the PCL register can be found in the Special Function Register section.

Stack

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



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

Arithmetic and Logic Unit – ALU

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

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

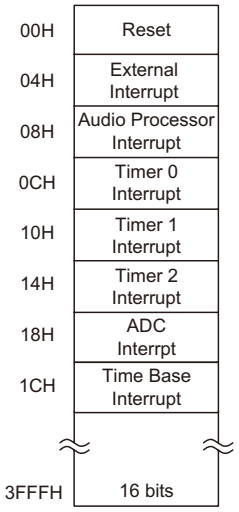
Program memory

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 devices offer 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 16Kx16. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by separate table pointer registers.

The device has its Program Memory divided into two Banks, Bank 0 and Bank 1. The required Bank is selected using Bit 5 of the BP Register.



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 initialization. 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 CTRL1 register.

- Audio Processor interrupt vector
  This vector is used by the Audio Processor interrupt. The Audio Processor interrupt is initialized by setting the Audio Processor request flag,. If the interrupt is enabled, the stack is not full and the AUPF is set, a sub-routine call will occur.

- Timer/Event 0/1/2 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 interrupt vector
  This internal vector is used by the internal Time Base. 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 and TBHP. These registers define the total address of the look-up table.

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

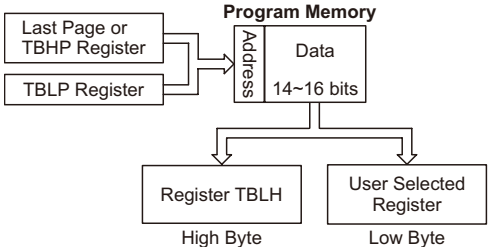


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "3F00H" which refers to the start address of the last page within the 16K words Program Memory of the device. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "3F06H" 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 table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the TBLH and subsequently cause errors if used again by the main routine. As a rule it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the

execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

Instruction(s)	Table Location													
	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	PC13	PC12	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: PC13~PC8: Current Program Counter bits.

@7~@0: Table Pointer TBLP bits.

b13~b0: Table address location bits.

Table Read Program Example

```
tempreg1 db ?              ;temporary register #1
tempreg2 db ?              ;temporary register #2
:
:
mov a,06h                  ;initialise low table pointer - note that this
address
mov tblp,a                 ;is referenced
mov a,07h                  ;initialise high table pointer
tbhp,a
:
:
tabrdl tempreg1            ;transfers value in table referenced by table
                           ;pointer data at program
                           ;memory address "3F06H" transferred to tempreg1
                           ;and TBLH
dec tblp                   ;reduce value of table pointer by one
tabrdl tempreg2            ;transfers value in table referenced by table
                           ;pointer data at program
                           ;memory address "3F05H" transferred to tempreg2
                           ;and TBLH in this
                           ;example the data "1AH" is transferred to
                           ;tempreg1 and data "0FH" to
                           ;register tempreg2
:
:
org 3F00h                  ;sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
```

Data Memory

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

Structure

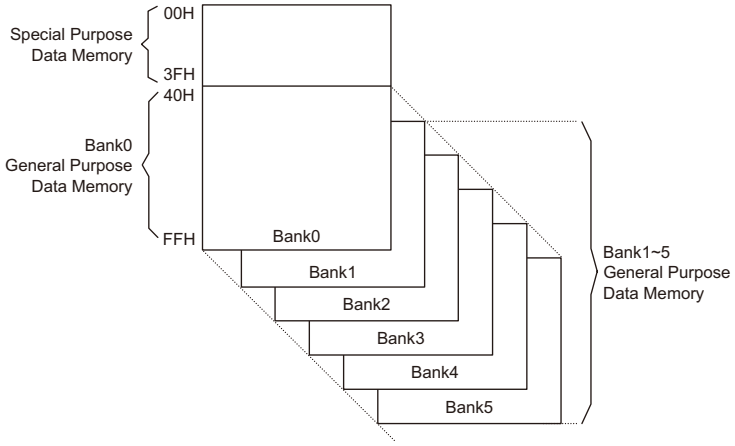
Divided into two sections, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain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

Capacity	Banks
1024x8	0: 40H~FFH 1: 40H~FFH 2: 40H~FFH 3: 40H~FFH 4: 40H~FFH 5: 40H~FFH

The two sections of Data Memory, the Special Purpose and General Purpose Data Memory are located at consecutive locations. All are implemented in RAM and are 8 bits wide but the length of each memory section is dictated by the type of microcontroller chosen. The start address of the Data Memory is the address "00H".

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 Data Memory is subdivided into six banks, which are selected using a Bank Pointer. Only data in Bank 0 can be directly addressed, data in Bank 1~5 must be indirectly addressed.



Data Memory Structure

Note: Most of the Data Memory bits can be directly manipulated using the "SET [m].i" and "CLR[m].i" with the exception of a few dedicated bits. The Data Memory can also be accessed through the memory pointer registers.

Special Purpose Data Memory

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

Special Function Registers

To ensure successful operation of the microcontroller, certain internal registers are implemented in the Data Memory area. These registers ensure correct operation of internal functions such as timers, interrupts, etc., as well as external functions such as I/O data control. The locations of these registers within the Data Memory begin at the address "00H" and are mapped from Bank 0 to Bank 5. Any unused Data Memory locations between these special function registers and the point where the General Purpose Memory begins is reserved and attempting to read data from these locations will return a value of "00H".

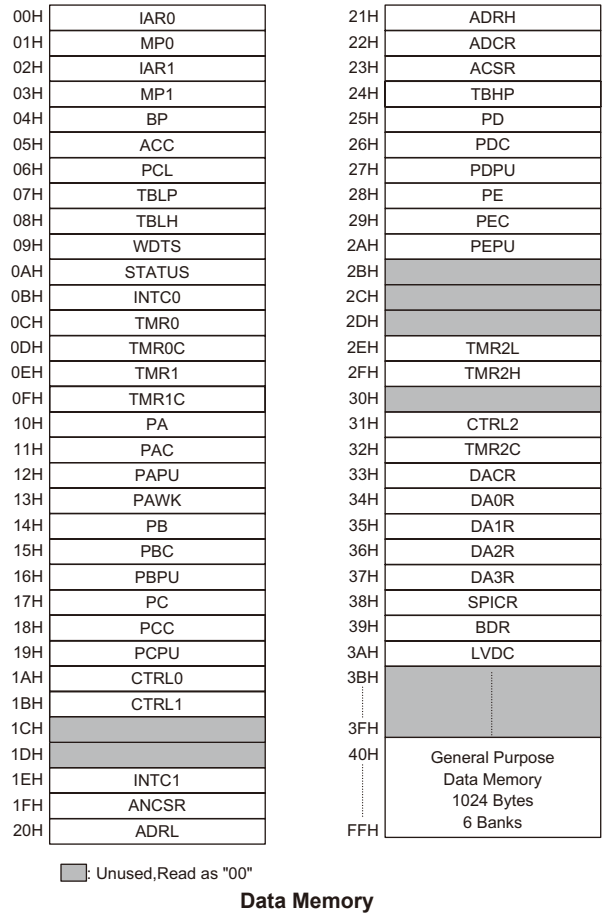
Indirect Addressing Registers – IAR0, IAR1

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

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to 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 from Bank 0 to Bank 5. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, is the address specified by the related Memory Pointer. Note that indirect addressing using MP1 and IAR1 must be used to access any data in Bank 1~Bank 5.

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



Data Memory

Indirect Addressing Program Example

```
data .section ´data´
adres1      db ?
adres2      db ?
adres3      db ?
adres4      db ?
block       db ?
code.section at 0 ´code´
org00h
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, no reference is made to specific Data Memory addresses.

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Bank Pointer – BP

In this device, the Program and Data Memory are divided into several banks. Selecting the required Program and Data Memory area is achieved using the Bank Pointer. Bit 5 of the Bank Pointer is used to select Program Memory Bank 0 or 1, while bits 0~2 are used to select Data Memory Banks 0~5.

The Data Memory is initialised to Bank 0 after a reset, except for the WDT time-out reset in the Power Down 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 within any bank. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from banks other than Bank 0 must be implemented using Indirect addressing.

As both the Program Memory and Data Memory share the same Bank Pointer Register, care must be taken during programming.

BP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PMBP0	—	—	DMBP2	DMBP1	DMBP0
R/W	—	—	R/W	—	—	R/W	R/W	R/W
POR	—	—	0	—	—	0	0	0

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 **PMBP0**: Program Memory Bank Pointer
0: Bank 0
1: Bank 1

Bit 4 ~ 3 Unimplemented, read as "0"

Bit 2 ~ 0 **DMBP2 ~ DMBP0**: Data Memory Bank Pointer
000: Bank 0
001: Bank 1
010: Bank 2
011: Bank 3
100: Bank 4
101: Bank 5
110~111: Undefined

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 any other register. Any data written into the status register will not change the TO and PDF flags. In addition operations related to the status register may give different results from those intended. The TO flag can be affected only by system power-up, a WDT time-out or executing the "HALT" or "CLR WDT" instruction. The PDF flag can be affected only by executing the "HLAT" or "CLR WDT" instruction or a system power-up.

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

In addition, on entering the interrupt sequence or executing the subroutine call, the status register will not be pushed onto stack automatically. If the contents of status are important and the subroutine can corrupt the status register, the programmer has to take precautions to save it properly.

STATUS Register

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

"x" unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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

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

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

Bit 0 **C**: Carry flag
0: no carry-out
1: an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation
C is also affected by a rotate through carry instruction.

Input/Output Ports and Control Registers

Within the area of Special Function Registers, the port PA, PB, etc data I/O registers and their associated control register PAC, PBC, etc play a prominent role. These registers are mapped to specific addresses within the Data Memory as shown in the Data Memory table. The data I/O registers, are used to transfer the appropriate output or input data on the port. The control registers specifies which pins of the port are set as inputs and which are set as outputs. To setup a pin as an input, the corresponding bit of the control register must be set high, for an output it must be set low. During program initialisation, it is important to first setup the control registers to specify which pins are outputs and which are inputs before reading data from or writing data to the I/O ports. One flexible feature of these registers is the ability to directly program single bits using the "SET [m].i" and "CLR [m].i" instructions. The ability to change I/O pins from output to input and vice versa by manipulating specific bits of the I/O control registers during normal program operation is a useful feature of this device.

System Control Register – CTRL0, CTRL1, CTRL2

These registers are used to provide control over various internal functions. Some of these include the PFD control, Audio Processor control, certain system clock options, the LXT Oscillator low power control, external Interrupt edge trigger type, Watchdog Timer enable function, Time Base function division ratio, and the LXT oscillator enable control.

CTRL0 Register

Bit	7	6	5	4	3	2	1	0
Name	PCFG	PFDCS	—	—	—	PFDC	LXTLP	CLKMOD
R/W	R/W	R/W	—	—	—	R/W	R/W	R/W
POR	0	0	—	—	—	0	0	1

Bit7 **PCFG**: I/O configuration
0: INT/TMR0/PFD pin-shared with PA3/PA2/PA1
1: INT/TMR0/PFD pin-shared with PB4/PB5/PB6

Bit6 **PFDCS**: PFD clock source
0: timer0
1: timer1

Bit5~3 Unimplemented, read as "0".

Bit2 **PFDC**: I/O or PFD
0: I/O
1: PFD

Bit1 **LXTLP**: LXT oscillator low power control function
0: LXT Oscillator quick start-up mode
1: LXT Oscillator Low Power Mode

Bit0 **CLKMOD**: system clock mode selection
0: High system clock -PLL mode.
1: Low system clock - LXT used as system clock.

CTRL1 Register

Bit	7	6	5	4	3	2	1	0
Name	INTEG1	INTEG0	TBSEL1	TBSEL0	WDTEN3	WDTEN2	WDTEN1	WDTEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	0	1	0	1	0

Bit 7, 6 **INTEG1, INTEG0**: External interrupt edge type
00: disable
01: rising edge trigger
10: falling edge trigger
11: dual edge trigger

Bit 5, 4 **TBSEL1, TBSEL0**: Time base period selection
00: $2^{10}\times (1/f_{TP})$
01: $2^{11}\times (1/f_{TP})$
10: $2^{12}\times (1/f_{TP})$
11: $2^{13}\times (1/f_{TP})$

Bit 3~0 **WDTEN3, WDTEN2, WDTEN1, WDTEN0**: WDT function enable
1010: WDT disabled
Other values: WDT enabled - Recommended value is 0101
If the "watchdog timer enable" is configuration option is selected, then the watchdog timer will always be enabled and the WDTEN3~WDTEN0 control bits will have no effect.

Note: The WDT is only disabled when both the WDT configuration option is disabled and when bits WDTEN3~WDTEN0=1010. The WDT is enabled when either the WDT configuration option is enabled or when bits WDTEN3~WDTEN0≠1010.

CTRL2 Register

Bit	7	6	5	4	3	2	1	0
Name	M1	M0	PLLD2	AUPRST	PLLEN	PLLD1	PLLD0	LXTEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	1	1	0

Bit7~6 **M1, M0**: PLL output frequency selection
00: 8.192MHz
01: 10.24MHz
10: 12.288MHz
11: 16.384MHz

Bit5 **PLLD2**: PLL output frequency divider selection (For Audio Processor)
0: PLL output frequency divided by 1
1: PLL output frequency divided by 2

Bit4 **AUPRST:** Audio Processor hardware reset bit
1→0→1: Audio Processor reset.

Bit3 **PLLEN**: PLL enable
0: disable
1: enable

Bit2~1 **PLLD1, PLLD0**: PLL output frequency divider selection (For MCU)
00: useless that will be forced to "1, 1" by hardware
01: PLL output frequency divided by 1
10: PLL output frequency divided by 2
11: PLL output frequency divided by 4

Bit0 **LXTEN**: LXT Oscillator on/off control after execution of HALT instruction
0: LXT off in Idle mode
1: LXT on in Idle mode

PLL output frequency for Audio Processor system clock (f_{ap})and divider reference clock is according to M1, M0 setting as shown below.

PLLEN	M1	M0	Clock Output
0	X	X	LOW(Logic state)
1	0	0	8.192MHz
1	0	1	10.24MHz
1	1	0	12.288MHz
1	1	1	16.384MHz

X: Don't care

Divided PLL output frequency for MCU system clock.

		PLLD1 , PLLD0	0,1	1,0	1,1
		Divide	1	2	4
PLL Freq (MHz)	8.192	f_{sys}(MHz)	8.192	4.096	2.048
	10.24		10.24	5.12	2.56
	12.288		12.288	6.144	3.072
	16.384		16.384	8.192	4.096

Note: PLLD1, PLLD0 = 0,0 is useless that will be forced to 1,1 by h/w.

Divided PLL output frequency for Audio Processor system clock.

		PLLD2	0	1
		Divide	1	2
PLL Freq (MHz)	8.192	f_{sys}(MHz)	8.192	4.096
	10.24		10.24	5.12
	12.288		12.288	6.144
	16.384		16.384	8.192

Wake-up Function Register – PAWK

When the microcontroller enters the Idle/Sleep Mode, various methods exist to wake the device up and continue with normal operation. One method is to allow a falling edge on the I/O pins to have a wake-up function. This register is used to select which Port A I/O pins are used to have this wake-up function.

Pull-high Registers – PAPU, PBPU, PCPU, PDPU, PEPU

The I/O pins, if configured as inputs, can have internal pull-high resistors connected, which eliminates the need for external pull-high resistors. This register selects which I/O pins are connected to internal pull-high resistors.

Oscillator

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

System Oscillator Overview

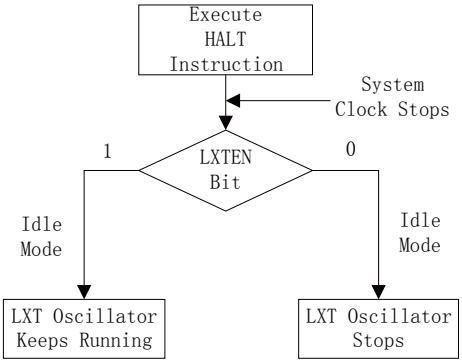
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base functions and three Timers. The system oscillator can be provided from a choice of two oscillator modes, LXT crystal oscillator and LXT crystal oscillator +PLL (by application program).

The LXT crystal oscillator is always turned on in normal mode, but PLL can be set by application program. System clock source will be provided by LXT crystal oscillator or PLL

External 32768Hz Crystal Oscillator – LXT

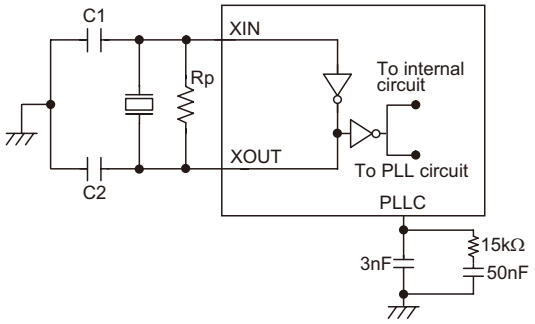
The LXT oscillator is used both as the slow system clock and also as a selectable source clock for some peripheral functions including the Watchdog Timer, Time Base, Timer/Event Counters functions etc. It must be first enabled using a configuration option.

To select the LXT oscillator to be the low speed system oscillator, the CLKMOD bit in the CTRL0 register should be set high. When a HALT instruction is executed, the system clock is stopped, but the LXTEN bit in the CTRL2 register determines if the LXT oscillator continues running when the microcontroller powers down. Setting the LXTEN bit high will enable the LXT to keep running after a HALT instruction is executed and enable the LXT oscillator to remain as a possible clock source for the Watchdog Timer, the Time-Base and the Timer/Event Counters.



LXTEN Bit Function

The LXT oscillator is implemented using a 32768Hz crystal connected to pins XIN/XOUT. However, for some crystals and to ensure oscillation and accurate frequency generation, it is normally 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 specification. The external parallel feedback resistor, Rp, may also be required.



LXT Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
32.768kHz	10pF	10pF

Note:
- C1 and C2 values are for guidance only.
- R_P=10MΩ is recommended.

32768 Hz Crystal Recommended Capacitor Values

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 CTRL0 register.

LXTLP Bit	LXT Mode
0	Quick Start
1	Low-power

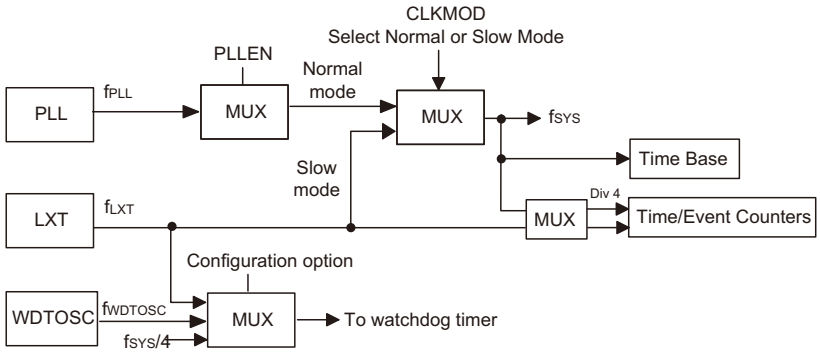
After power on, the LXTLP bit will be automatically cleared to zero ensuring that the LXT oscillator is in the Quick Start operating mode. In the Quick Start Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up it can be placed into the Low-power 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 always function normally, the only difference is that it will take more time to start up if in the Low-power mode.

Operating Modes

By using the LXT low frequency oscillator in combination with PLL, 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

For this device the LXT oscillator can run together with PLL. The CLKMOD bit in the CTRL0 register can be used to switch the system clock from the selected PLL mode or 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 Confiuration

Operating mode and MCU system clock source is as following table.

Operating Mode Control

HALT Instruction	IDLE COMMAND	PLLEN Bit	CLKMOD Bit	LXTEN Bit	LXT Oscillator	MCU System Clock	MCU Mode	Audio Processor System Clock	Audio Processor Mode
Non Executed	Non Executed	1	0	x	On	PLL	Normal	PLL	Normal
	Executed	1	0	x	On	PLL		HALT	Idle
	Non Executed	1	1	x	On	LXT	Slow	PLL	Normal
	Executed	1	1	x	On	LXT		HALT	Idle
Executed	Executed	0	1	1	On	HALT	Idle	HALT	Idle
		0	1	0	Off	HALT	Sleep		

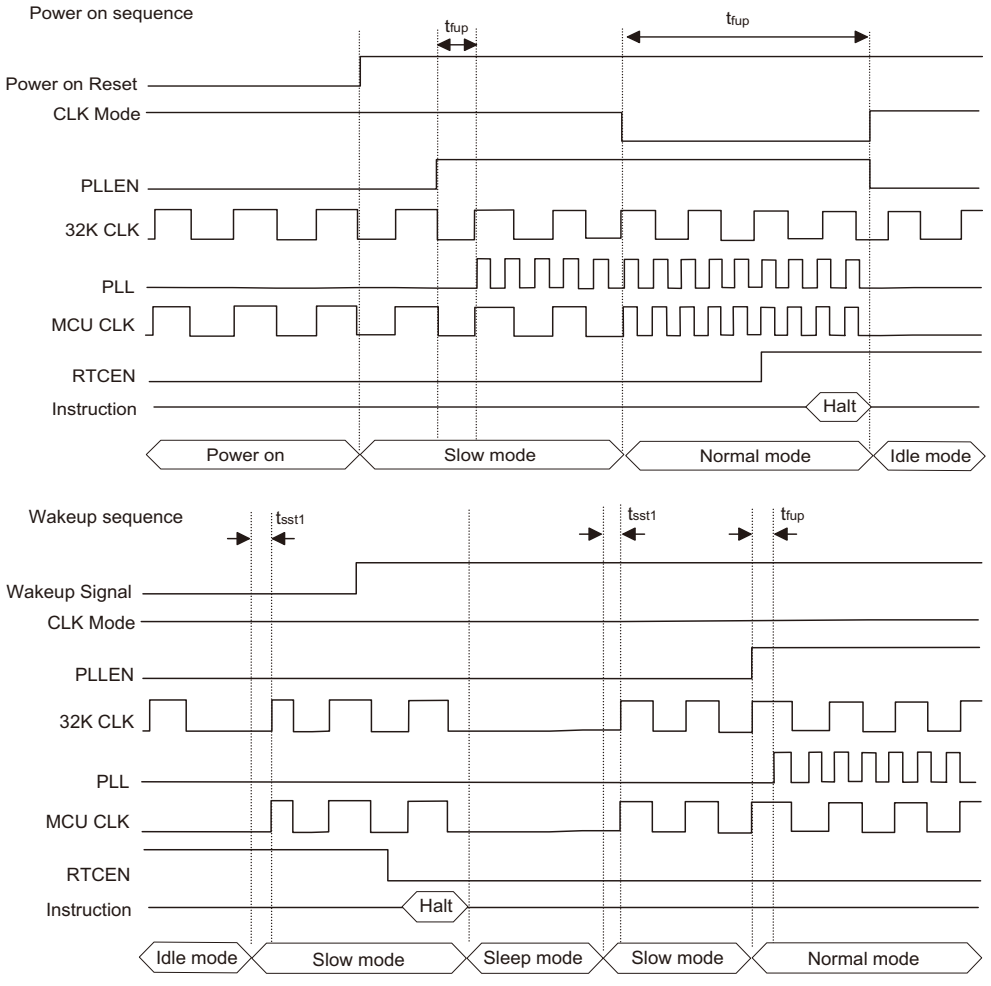
Note: 1. LXT oscillator is the 32768 Hz oscillator.

2. SST for LXT OSC is 1024 clocks.

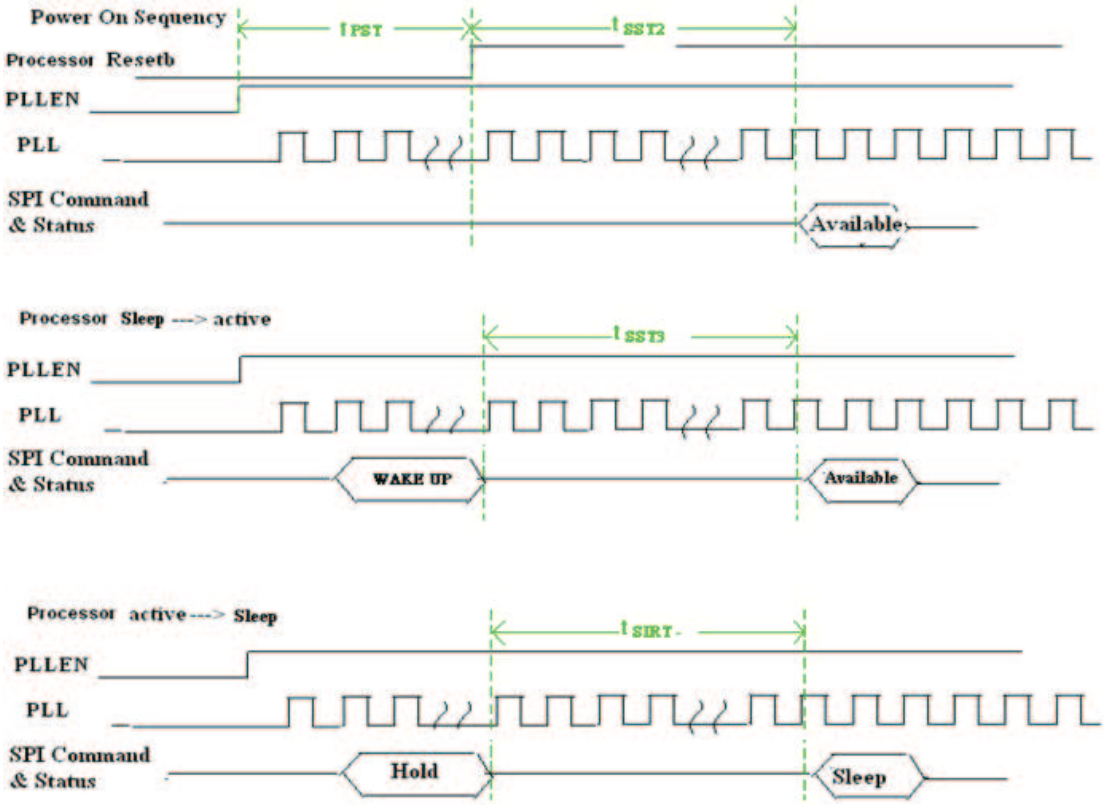
3. If PLLEN=0, CLKMOD bit will be set.

4. The Audio Processor must execute Audio Processor idle command via SPI, before MCU Executed HALT instruction or disable PLL (PLLEN=0).

The MCU operating mode switch timing are shown in the following figures.



The Audio Processor operating mode switch timing as Following figure.



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 LXT crystal oscillator or PLL. 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 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 if the WDT clock source is selected to come from the WDT or LXT oscillator. The WDT will stop if its clock source originates from the system clock
- 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 WDTOSC 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 Mode. To keep the LXT power consumption to a minimum level the LXTLP bit in the CTRL0 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:

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

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

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 woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set 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.

Wake-up Source	Oscillator Type
	Crystal
External $\overline{RES}$	$t_{RSTD} + t_{SST}$
PA Port	t_{SST}
Interrupt	
WDT Overflow	

Note: 1. t_{RSTD} (reset delay time), t_{SYS} (system clock).
2. t_{RSTD} is power-on delay, typical time= 100ms.
3. t_{SST}= 1024 $t_{SYS.}$

Wake-up Delay Time

Watchdog Timer

The Watchdog Timer, also known as the WDT, is provided to inhibit program malfunctions caused by the program jumping to unknown locations due to certain uncontrollable external events such as electrical noise.

Watchdog Timer Operation

It operates by providing a device reset when the Watchdog Timer counter overflows. Note that if the Watchdog Timer function is not enabled, then any instructions related to the Watchdog Timer will result in no operation.

Setting up the various Watchdog Timer options are controlled via the configuration options and two internal registers WDTS and CTRL1. Enabling the Watchdog Timer can be controlled by both a configuration option and the WDTEN bits in the CTRL1 internal register in the Data Memory.

Configuration Option	CTRL1 Register	WDT Function
Disable	Disable	OFF
Disable	Enable	ON
Enable	x	ON

Watchdog Timer On/Off Control

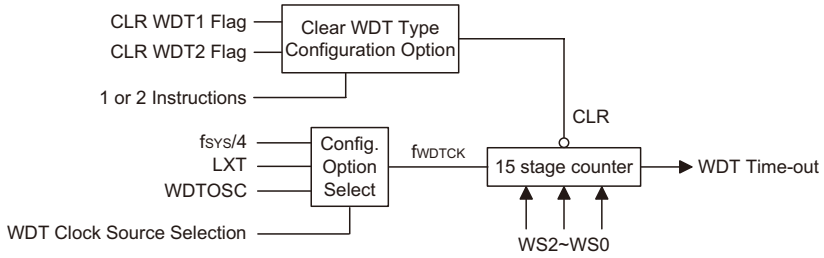
The Watchdog Timer will be disabled if bits WDTEN3~WDTEN0 in the CTRL1 register are written with the binary value 1010B and WDT configuration option is disable. This will be the condition when the device is powered up. Although any other data written to WDTEN3~WDTEN0 will ensure that the Watchdog Timer is enabled, for maximum protection it is recommended that the value 0101B is written to these bits.

The WDT clock can emanate from three different sources selected by configuration option: its own self-contained dedicated internal WDT oscillator, the instruction clock which is the system clock divided by 4 or LXT. The Watchdog Timer can be disabled by options. Note that if the WDT configuration option has been disabled, then any instruction relating to its operation will result in no operation.

The internal WDT oscillator has an approximate period of 65μs at a supply voltage of 5V. If selected, it is first divided by 256 via an 8-stage counter to give a nominal period of 17ms. Note that this period can vary with VDD, temperature and process variations. For longer WDT time-out periods the WDT prescaler can be utilized. By writing the required value to bits 0, 1 and 2 of the WDTS register, known as WS0, WS1 and WS2, longer time-out periods can be achieved. With

WS0, WS1 and WS2 all set high, the division ratio is 1:128 which gives a maximum time-out period of about 2.1s.

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 Idle/Sleep Mode, when a Watchdog Timer time-out occurs, the device will be woken up, 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 an external hardware reset, which means a low level on the external reset pin, the second is using the Clear Watchdog Timer software instructions and the third is when a HALT instruction is executed. There are two methods of using software instructions to clear the Watchdog Timer, one of which must be chosen by configuration option. The first option is to use the single "CLR WDT" instruction while the second is to use the two commands "CLR WDT1" and "CLR WDT2". For the first option, a simple execution of "CLR WDT" will clear the Watchdog Timer while for the second option, both "CLR WDT1" and "CLR WDT2" must both be executed to successfully clear the Watchdog Timer. Note that for this second option, if "CLR WDT1" is used to clear the Watchdog Timer, successive executions of this instruction will have no effect, only the execution of a "CLR WDT2" instruction will clear the Watchdog Timer. Similarly after the "CLR WDT2" instruction has been executed, only a successive "CLR WDT1" instruction can clear the Watchdog Timer.



Watchdog Timer

WDTS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	WS2	WS1	WS0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	1	1	1

bit 7~3 unimplemented, read as "0"

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

000: 2^8 t_{WDTCK}
001: 2^9 t_{WDTCK}
010: 2^{10} t_{WDTCK}
011: 2^{11} t_{WDTCK}
100: 2^{12} t_{WDTCK}
101: 2^{13} t_{WDTCK}
110: 2^{14} t_{WDTCK}
111: 2^{15} t_{WDTCK}

CTRL1 Register

Bit	7	6	5	4	3	2	1	0
Name	INTEG1	INTEG0	TBSEL1	TBSEL0	WDTEN3	WDTEN2	WDTEN1	WDTEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	0	1	0	1	0

Bit 7~4 described elsewhere

Bit 3~0 **WDTEN3, WDTEN2, WDTEN1, WDTEN0**: WDT function enable
1010: WDT disabled
Other values: WDT enabled - Recommended value is 0101
If the "watchdog timer enable" is configuration option is selected, then the watchdog timer will always be enabled and the WDTEN3~WDTEN0 control bits will have no effect.

Note: The WDT is only disabled when both the WDT configuration option is disabled and when bits WDTEN3~WDTEN0=1010. The WDT is enabled when either the WDT configuration option is enabled or when bits WDTEN3~WDTEN0≠1010.

Reset and Initialization

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

One of these registers is the Program Counter, which will be reset to zero forcing the microcontroller to begin program execution from the lowest Program Memory address. In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high.

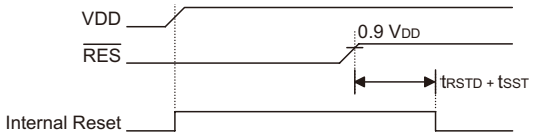
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset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

Reset Functions

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

Power-on Reset

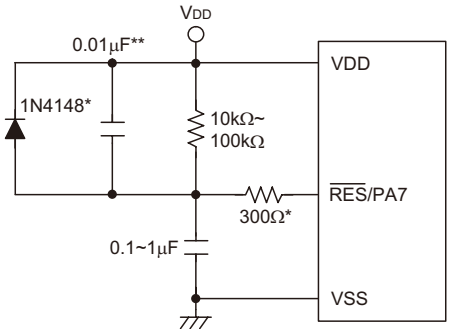
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions.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



Note: t_{RSTD} is power-on delay, typical time=100ms

Power-On Reset Timing Chart

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



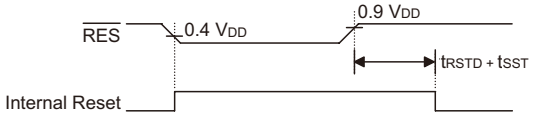
Note: "*" It is recommended that this component is added ESD protection.

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

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

$\overline{RES}$ Pin Reset

This type of reset occurs when the microcontroller is already running and the $\overline{RES}$ pin is forcefully pulled low by external hardware such as an external switch. In this case as in the case of other reset, the Program Counter will reset to zero and program execution initiated from this point.

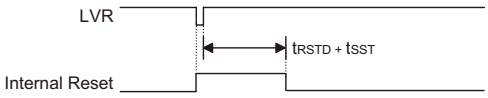


Note: t_{RSTD} is power-on delay, typical time=100ms

$\overline{RES}$ 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 selected via a configuration option. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the 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 via configuration options.

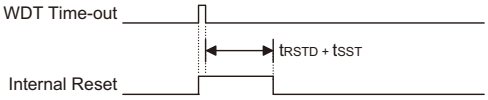


Note: t_{RSTD} is power-on delay, typical time=100ms

Low Voltage Reset Timing Chart

Watchdog Time-out Reset During Normal Operation

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

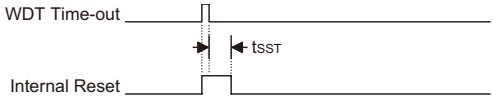


Note: t_{RSTD} is power-on delay, typical time=100ms

WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset During Idle/Sleep Mode

The Watchdog time-out Reset during Idle/Sleep 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.

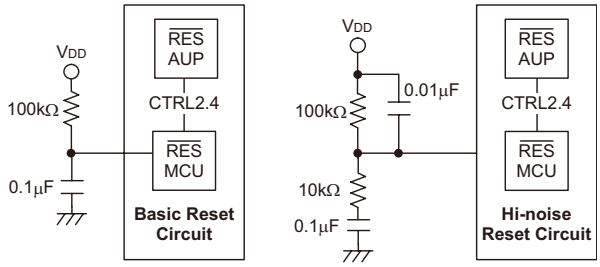


Note: The t_{SST} can be chosen to be 1024 clock cycles via configuration option.
The SST is 1024 for LXT.

WDT Time-out Reset during Idle/Sleep Timing Chart

Audio Processor Reset

The Audio Processor is reset by the software control CTRL2.4 register



Note: "*" Make the length of the wiring, which is connected to the $\overline{RES}$ pin as short as possible, to avoid noise interference.

Reset Circuit

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 Idle/Sleep function or Watchdog Timer. The reset flags are shown in the table:

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	$\overline{RES}$ or LVR reset during Normal or Slow Mode operation
1	u	WDT time-out reset during Normal or Slow Mode operation
1	1	WDT time-out reset during Idle or Sleep Mode operation

Note: "u" stands for unchanged

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

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

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers.

Register	Power On Reset	WDT Time-out (Normal Operation)	$\overline{RES}$ or LVR (Normal Operation)	$\overline{RES}$ or LVR Reset (Idle/Sleep)	WDT Time-out (Idle/Sleep)
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	-- 0- -000	-- 0- -000	-- 0- -000	-- 0- -000	-- u- -uuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	-- uu uuuu	--uu uuuu	-- uu uuuu	-- uu uuuu
WDTS	-- -- -111	-- -- -111	-- -- -111	-- -- -111	-- -- -uuu
STATUS	-- 00 xxxx	-- 1u uuuu	-- uu uuuu	-- 01 uuuu	-- 11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	$\overline{RES}$ or LVR (Normal Operation)	$\overline{RES}$ or LVR Reset (Idle/Sleep)	WDT Time-out (Idle/Sleep)
TMR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1-- -	0000 1-- -	0000 1-- -	0000 1-- -	uuuu u-- -
TMR2L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAWK	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PE	-- -- -- 11	-- -- -- 11	-- -- -- 11	-- -- -- 11	-- -- -- uu
PEC	-- -- -- 11	-- -- -- 11	-- -- -- 11	-- -- -- 11	-- -- -- uu
PEPU	-- -- -- 00	-- -- -- 00	-- -- -- 00	-- -- -- 00	-- -- -- uu
CTRL0	0000 0001	0000 0001	0000 0001	0000 0001	uuuu uuuu
CTRL1	1000 1010	1000 1010	1000 1010	1000 1010	uuuu uuuu
CTRL2	0010 0110	0010 0110	0010 0110	0010 0110	uuuu uuuu
ADRL	xxxx -- --	xxxx -- --	xxxx -- --	xxxx -- --	uuuu -- --
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	01-- 0000	01-- 0000	01-- 0000	01-- 0000	uu-- uuuu
ACSR	11-- -000	11-- -000	11-- -000	11-- -000	uu-- -uuu
ANCSR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
DACR	-- -- 0000	-- -- 0000	-- -- 0000	-- -- 0000	-- -- uuuu
DA0R	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
DA1R	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
DA2R	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
DA3R	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BDR	-- -- -- 00	-- -- -- 00	-- -- -- 00	-- -- -- 00	-- -- -- uu
SPICR	1-01 00xx	1-01 00xx	1-01 00xx	1-01 00xx	u-uu uuuu
LVDC	-- 00 -- --	-- 00 -- --	-- 00 -- --	-- 00 -- --	-- uu -- --

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

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. 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, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selectable via a register known as PAPU, PBPU, PCPU, PDPU and PEPU located in the Data Memory. The pull-high resistors are implemented using weak PMOS transistors. Note that pin PA7 does not have a pull-high resistor selection.

Port A Wake-up

If the HALT instruction is executed, the device will enter the Idle/Sleep Mode, where the system clock will stop resulting in power being conserved, 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 PA0~PA7 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.

PAWK, PAC, PAPU, PBC, PBPU, PCC, PCPU, PDC, PDPU, PEC, PEPU Register

Register Name	POR	Bit							
		7	6	5	4	3	2	1	0
PAWK	00H	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
PAC	FFH	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	00H	—	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PBC	FFH	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	00H	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PCC	FFH	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	00H	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PDC	FFH	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	00H	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PEC	03H	—	—	—	—	—	—	PEC1	PEC0
PEPU	00H	—	—	—	—	—	—	PEPU1	PEPU0

"—" Unimplemented, read as "0"

PAWKn: PA wake-up function enable
0: disable
1: enable
PACn/PBCn/PCCn/PDCn/PECn: I/O type selection
0: output
1: input
PAPUn/PBPUn/PCPUn/PDPUn/PEPUn: Pull-high function enable
0: disable
1: enable

I/O Port Control Registers

Each Port has its own control register, known as PAC, PBC, PCC, PDC and PEC 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 pin, INT, is pin-shared with an I/O pin. To use the pin as an external interrupt input the correct bits in the INTC0 register must be programmed. The pin must also be setup as an input by setting the PAC3 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, TMR0, TMR1 and TMR2 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.

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.

A/D Inputs

The device has 8 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 ANCSR0 registers, 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.

Audio Processor Pins

Pins PB4~PB7 on Port B can be used as Audio processor I/O pins. This function is controlled using the SPICR register.

Pin Remapping Configuration

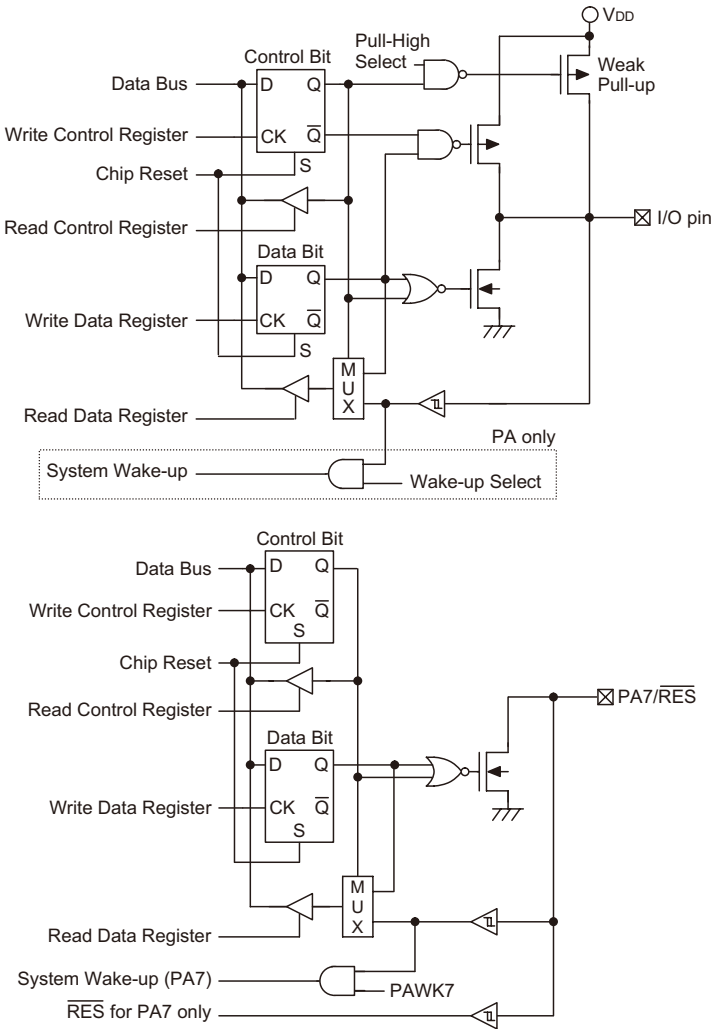
The pin remapping function enables the function pins INT, TMR0 and PFD to be located on different port pins. It is important not to confuse the Pin Remapping function with the Pin-shared function, these two functions have no interdependence.

The PCFG bit in the CTRL0 register allows the three function pins INT, TMR0 and PFD 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 these three 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 indicates its default location which is the condition after Power-on.

PCFG Bit Status		
PCFG Bit	**0**	**1**
Pin Mapping	INT/PA3 TMR0/PA2 PFD/PA1	INT/PB4 TMR0/PB5 PFD/PB6

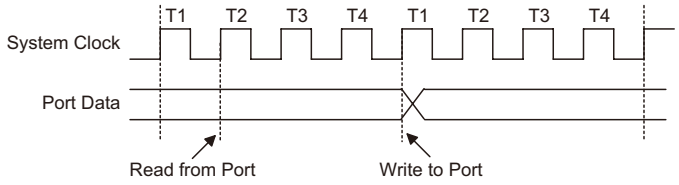
I/O Pin Structures

The diagrams illustrate the I/O pin internal structures. As the exact logical construction of the I/O pin may differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins.



Programming Considerations

Within the user program, one of the first things to consider is port initialization. After a reset, the I/O data register and I/O port control register will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high options have been selected. 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 register is first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct value into the 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.



Read Modify Write Timing

Pins PA0 to PA7 each have a wake-up function, selected via the PAWK register. When the device is in the Idle/Sleep Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the these pins. Single or multiple pins on Port A can be setup to have this function.

Timer/Event Counters

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 8-bit and one 16-bit timer. 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 gives 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 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.

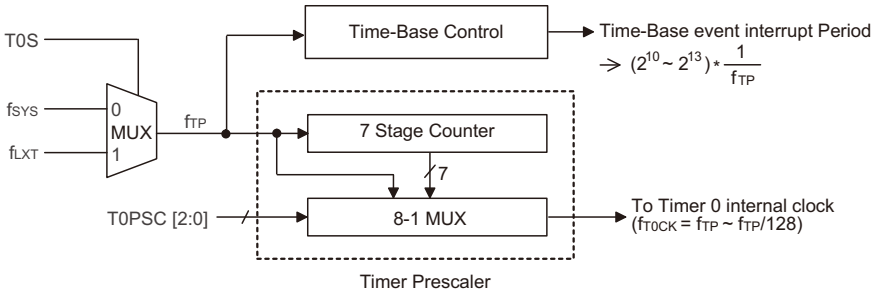
Configuring the Timer/Event Counter Input Clock Source

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. For some Timer/Event Counters, this internal clock source is first divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register bits. 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 TMRn. 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.

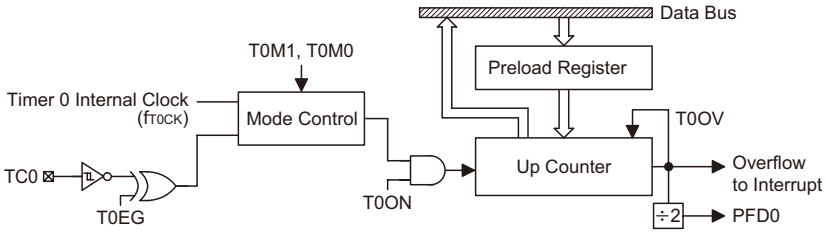
Timer Registers – TMR0, TMR1, TMR2L, TMR2H

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, TMR1, TMR2L and TMR2H. 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 for the 8-bit Timer/Event Counter or FFFFH for the 16-bit Timer/Event Counters, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting. Note that to achieve a maximum full range count of FFH or FFFFH, the preload register must first be cleared to all zeros. 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

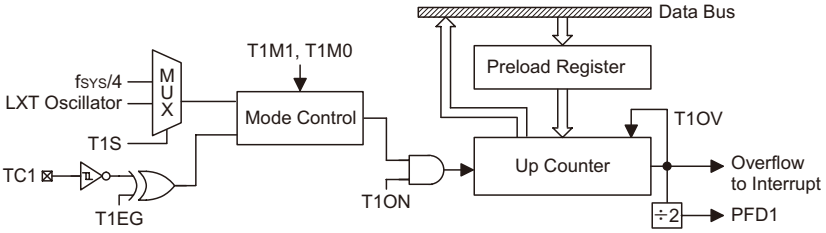
preload data register during this period will remain in the preload register and will only be written into the actual counter the next time an overflow occurs.



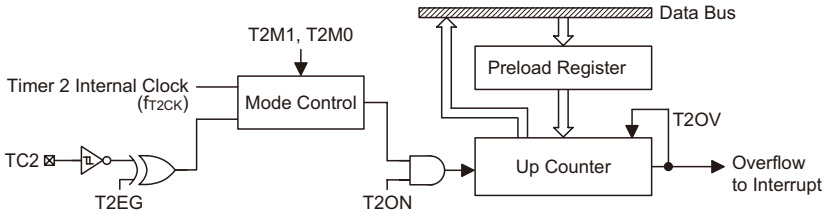
Clock Structure for Timer/Time Base



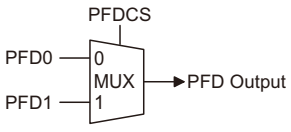
8-bit Timer/Event Counter 0 Structure



8-bit Timer/Event Counter 1 Structure



8-bit Timer/Event Counter 2 Structure



Timer Control Registers – TMR0C, TMR1C, TMR2C

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 register 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 TnS bit selects the internal clock source.

TMR0C Register

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	T0S	T0ON	T0EG	T0PSC2	T0PSC1	T0PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7, 6 **T0M1, T0M0**: Timer0 operation mode selection
00: no mode available
01: event counter mode
10: timer mode
11: pulse width capture mode

Bit 5 **T0S**: timer clock source
0: f_{SYS}
1: LXT oscillator

Bit 4 **T0ON**: Timer/event counter counting enable
0: disable
1: enable

Bit 3 **T0EG**: Event counter active edge selection
0: count on rising edge
1: count on falling edge
Pulse Width Capture active edge selection
0: start counting on falling edge, stop on rising edge
1: start counting on rising edge, stop on falling edge

Bit 2~0 **T0PSC2, T0PSC1, T0PSC0**: 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**: Timer0 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
0: count on rising edge
1: count on falling edge
Pulse Width Capture active edge selection
0: start counting on falling edge, stop on rising edge
1: start counting on rising edge, stop on falling edge

Bit 2~0 unimplemented, read as "0"

TMR2C Register

Bit	7	6	5	4	3	2	1	0
Name	T2M1	T2M0	T2S	T2ON	T2EG	T2PSC2	T2PSC1	T2PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7, 6 **T2M1, T2M0**: Timer0 operation mode selection
00: no mode available
01: event counter mode
10: timer mode
11: pulse width capture mode

Bit 5 **T2S**: Timer clock source
0: $f_{SYS}/4$
1: LXT oscillator

Bit 4 **T2ON**: Timer/event counter counting enable
0: disable
1: enable

Bit 3 **T2EG**: Event counter active edge selection
0: count on rising edge
1: count on falling edge
Pulse Width Capture active edge selection
0: start counting on falling edge, stop on rising edge
1: start counting on rising edge, stop on falling edge

Bit 2~0 **T2PSC2, T2PSC1, T2PSC0**: 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: $f_{TP}/128$

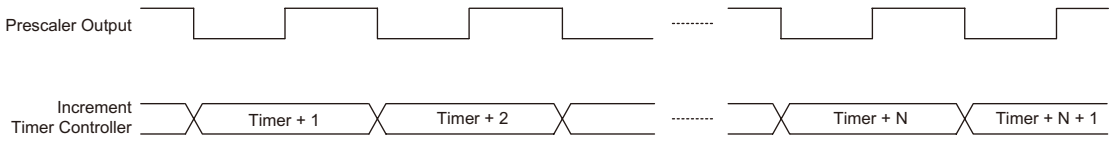
Timer Mode

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 either f_{SYS}, $f_{SYS}/4$ or the LXT oscillator. However, this timer clock source is further divided by a prescaler, the value of which is determined by the bits TnPSC2~TnPSC0 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 interrupt is one of the wake-up sources, however, the internal interrupts can be disabled by ensuring that the ETnI bits of the INTCn registe 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 TMRn 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.

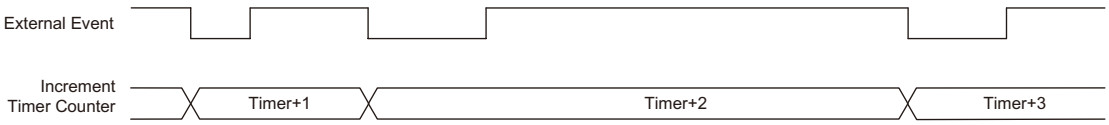
Control Register Operating Mode Select Bits for the Event Counter Mode

Bit7	Bit6
0	1

In this mode, the external timer TMRn pin, 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, 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 is in the Idle/Sleep Mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input TMRn pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timer 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} , $f_{SYS}/4$ or the LXT, is used as the internal clock for the 8-bit Timer/Event Counter and the 16-bit Timer/Event Counter. However, the clock source, f_{SYS} and $f_{SYS}/4$, for the 8-bit timer and the 16-bit timer are further divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits TnPSC2~TnPSC0, which are bits 2~0 in the Timer Control Register. 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, 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 TMRn 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, is reset to zero.

As the TMRn 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 happen. 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 configures the pin as an input.

Prescaler

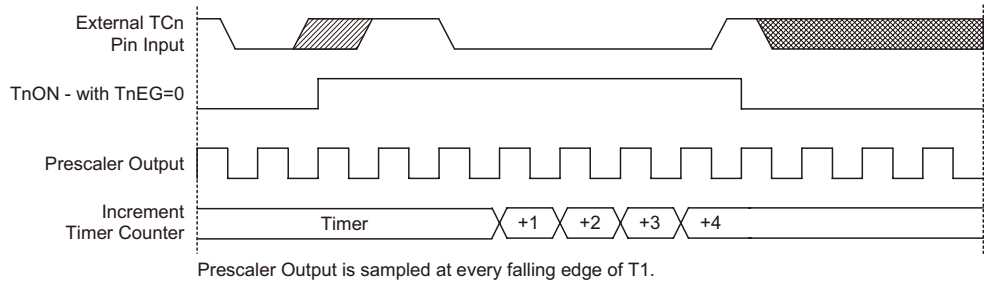
Bits T0PSC0~T0PSC2 of the TMR0C register and Bits T2PSC0~T2PSC2 of the TMR2C register can be used to define a division ratio for the internal clock source of the Timer/Event Counter enabling longer time out periods to be setup.

PFD Function

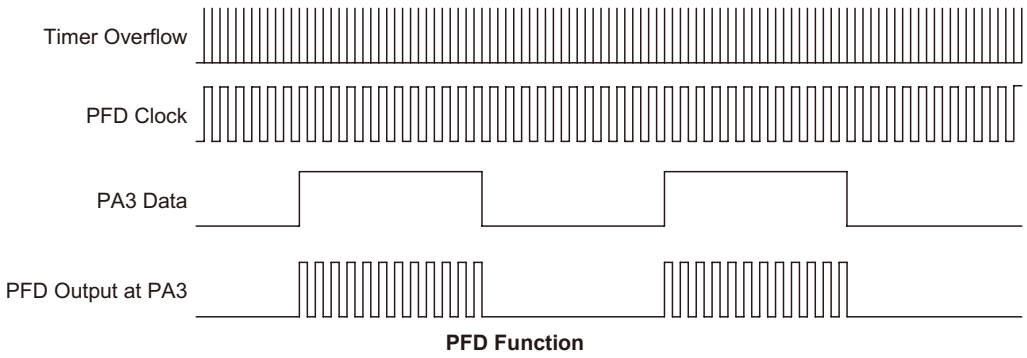
The Programmable Frequency Divider provides a means of producing a variable frequency output suitable for applications, such as piezo-buzzer driving or other interfaces requiring a precise frequency generator.

The Timer/Event Counter overflow signal is the clock source for the PFD function, which is controlled by PFDCS bit in CTRL0. For applicable 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. The counter will then 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 A control register PAC, to setup the PFD pins as outputs. PA1 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.



Pulse Width Capture Mode Timing Chart (TnEG=0)



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 select bits in the Timer/Event Counter control register, select 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 initialised 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  0ch            ;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
:           :
```

Time Base

The device includes a Time Base function which is used to generate a regular time interval signal.

The Time Base time interval magnitude is determined using an internal 13 stage counter sets the division ratio of the clock source. This division ratio is controlled by both the TBSEL0 and TBSEL1 bits in the CTRL1 register. The clock source is selected using the T0S bit in the TMR0C register.

When the Time Base time out, a Time Base interrupt signal will be generated. It should be noted that as the Time Base clock source is the same as the Timer/Event Counter clock source, care should be taken when programming.

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 an 8-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.

Input Channels	Conversion Bits	Input Pins
8	12	PA0~PA3 PC0~PC3

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

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.

In the following table, D0~D11 is the A/D conversion data result bits.

Register	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRL	D3	D2	D1	D0	—	—	—	—
ADRH	D11	D10	D9	D8	D7	D6	D5	D4

A/D Data Registers

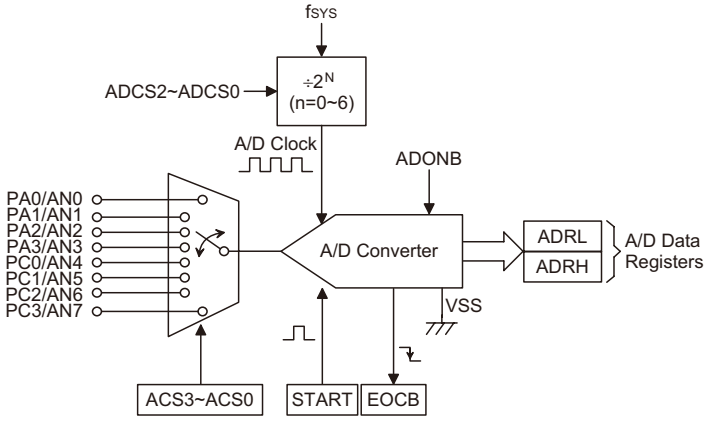
A/D Converter Control Registers – ADCR, ACSR, ANCSR

To control the function and operation of the A/D converter, four control registers known as ADCR, ACSR and ANCSR 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 8 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 control register, ANCSR, determines which pins on PA0~PA3, PC0~PC3 are used as analog inputs for the A/D converter and which pins are to be used as normal I/O pins. If the 8-bit address on PCR7~PCR0 has a value of “FFH”, then all 8 pins, namely AN0~AN7 will all be set as analog inputs. To reduce the power consumption in normal run, the ADC module can be turned off by setting ADONB=1 in ACSR. Once the ADC module is turned off, the ADC module and analog

channel have no power consumption no matter what voltage level is on analog input. If the I/O lines is selected as an I/O function and the analog input pin voltage is not equal to V_{DD} or V_{SS}, there user may have to take care I_{DD}/I_{STB} current consumed by the pin-shared logic input function no matter the ADC module is on or off. If the ADC module is turned off, then all the ADC pin-shared I/O pins will be setup as normal I/Os.



A/D Converter Structure

ADRH, ADRL Register

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

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, ACS2, ACS1, ACS0**: Select A/D channel
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000~1111: undefined, can't be used.

ACSR Register

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

Bit 7 **TEST**: for test mode use only

Bit 6 **ADONB**: ADC module power on/off control bit
0: ADC module power on
1: ADC module power off
This bit controls the power to the A/D internal function. This bit should be cleared to zero to enable the A/D converter. If the bit is set high then the A/D converter will be switched off reducing the device power consumption. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications. Note: It is recommended to set ADONB=1 before entering the IDLE/SLEEP Mode for saving power.

Bit 5~3 unimplemented, read as "0"

Bit 2~0 **ADCS2~ADCS0**: Select A/D converter clock source
000: system clock/2
001: system clock/8
010: system clock/32
011: undefined, can't be used.
100: system clock
101: system clock/4
110: system clock/16
111: undefined, can't be used.
These three bits are used to select the clock source for the A/D converter.

ANCSR 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 **PCR7**: Define PC3 is A/D input or not
0: Not A/D input
1: A/D input, AN7

Bit 6 **PCR6**: Define PC2 is A/D input or not
0: Not A/D input
1: A/D input, AN6

Bit 5 **PCR5**: Define PC1 is A/D input or not
0: Not A/D input
1: A/D input, AN5

Bit 4 **PCR4**: Define PC0 is A/D input or not
0: Not A/D input
1: A/D input, AN4

Bit 3 **PCR3**: Define PA3 is A/D input or not
0: Not A/D input
1: A/D input, AN3

Bit 2 **PCR2**: Define PA2 is A/D input or not
0: Not A/D input
1: A/D input, AN2

Bit 1 **PCR1**: Define PA1 is A/D input or not
0: Not A/D input
1: A/D input, AN1

Bit 0 **PCR0**: Define PA0 is A/D input or not
0: Not A/D input
1: A/D input, AN0

The START bit in the register is used to start and reset the A/D converter. When the 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 a "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 0.5μs, 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 = 100 (f_{SYS}/4)	ADCS2, ADCS1, ADCS0 = 101 (f_{SYS}/16)	ADCS2, ADCS1, ADCS0 = 011, 111
1MHz	2μs	8μs	32μs	1μs	4μs	16μs	Undefined
2MHz	1μs	4μs	16μs	500ns	2μs	8μs	Undefined
4MHz	500ns	2μs	8μs	250ns*	1μs	4μs	Undefined
8MHz	250ns*	1μs	4μs	125ns*	500ns	2μs	Undefined
12MHz	167ns*	667ns	2.67μs	83ns*	333ns*	1μs	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 and Port C. Bits PCR7~PCR0 in the ANCSR 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 PAC and PCC port control registers to enable the A/D input as when the PCR7~PCR0 bits enable an A/D input, the status of the port control register will be overridden.

Summary of A/D Conversion Steps

The following summarizes 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, ADCS1 and ADCS0 in the 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 PCR7~PCR0 bits in the ANCSR 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 "0" to "1" and then to "0" again. Note that this bit should have been originally set to "0".

- 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.

The setting up and operation of the A/D converter function is fully under the control of the application program as there are no configuration options associated with the A/D converter. 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

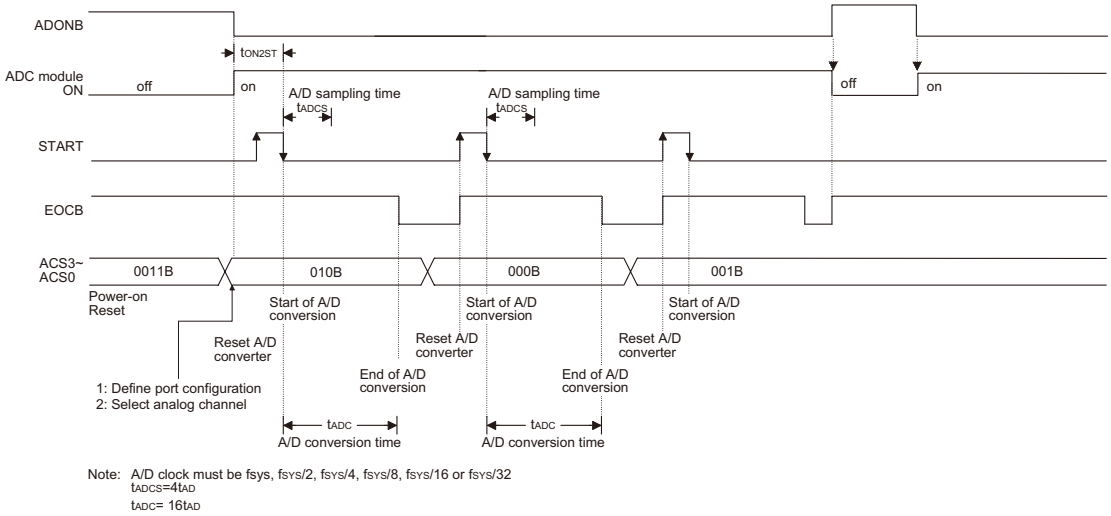
When programming, special attention must be given to the PCR[7:0] bits in the register. If these bits are all cleared to zero no external pins will be selected for use as A/D input pins allowing the pins to be used as normal I/O pins. When this happens the internal A/D circuitry will be power down. Setting 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 $V_{DD}/4096$. The diagram show 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 0.5 LSB offset is added to the A/D Converter input. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the V_{DD} level.

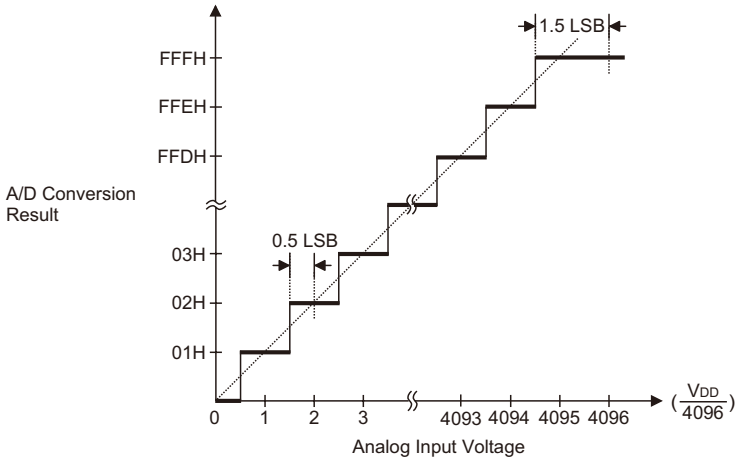
A/D Programming Example

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.



Note: A/D clock must be fsys, fSYS/2, fSYS/4, fSYS/8, fSYS/16 or fSYS/32
$t_{ADCS}=4t_{AD}$
$t_{ADC}=16t_{AD}$

A/D Conversion Timing



Ideal A/D Transfer Function

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

```
    clr     EADI            ;disable ADC interrupt
    mov     a,00000001B
    mov     ACSR,a          ;select fSYS/8 as A/D clock and ADONB=0
    mov     a,0Fh           ;setup ANCSR to configure pins AN0~AN3
    mov     ANCSR0,a
    mov     a, 00h
    mov     ANCSR1,a        ;
    mov     a,00000000B     ;
    mov     ADCR,a          ;select AN0 to be connected to the A/D converter
    :
Start_conversion:
    clr     START
    set     START           ;reset A/D
    clr     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
```

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

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

```
    clr     EADI            ;disable ADC interrupt
    mov     a,00000001B
    mov     ACSR,a          ;select fSYS/8 as A/D clock and ADONB=0
    mov     a,0Fh           ;setup ANCSR to configure pins AN0~AN3
    mov     ANCSR0,a
    mov     a, 00h
    mov     ANCSR1,a        ;
    mov     a,00000000B     ;
    mov     ADCR, a         ;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     EADI            ;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".

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 requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs.

The device provides an external interrupt and multiple internal interrupts. The external interrupt is controlled by the action of the external interrupt pin, while the internal interrupts are controlled by the Timer/Event Counters overflow and the Timer base overflow.

Interrupt Register

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by using two registers, INTC0 and INTC1. By controlling the appropriate enable bits in this 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 if cleared to zero will disable all interrupts.

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	T0F	AUPF	EIF	T0E	AUPE	EEI	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 **T0F**: Timer/Event Counter 0 interrupt request flag
0: inactive
1: active

Bit 5 **AUPF**: Audio Processor request flag
0: inactive
1: active

Bit 4 **EIF**: External interrupt request flag
0: inactive
1: active

Bit 3 **T0E**: Timer/Event Counter 0 interrupt enable
0: disable
1: enable

Bit 2 **AUPE**: Audio Processor interrupt enable
0: disable
1: enable

Bit 1 **EEI**: External interrupt enable
0: disable
1: enable

Bit 0 **EMI**: Master interrupt global enable
0: disable
1: enable

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	TBF	ADF	T2F	T1F	TBE	ADE	T2E	T1E
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 **TBF**: Time Base request flag
0: inactive
1: active

Bit 6 **ADF**: A/D request flag
0: inactive
1: active

Bit 5 **T2F**: Internal Timer/Event Counter 2 request flag
0: inactive
1: active

Bit 4 **T1F**: Internal Timer/Event Counter 1 request flag
0: inactive
1: active

Bit 3 **TBE**: Time Base interrupt enable
0: disable
1: enable

Bit 2 **ADE**: A/D converter interrupt enable
0: disable
1: enable

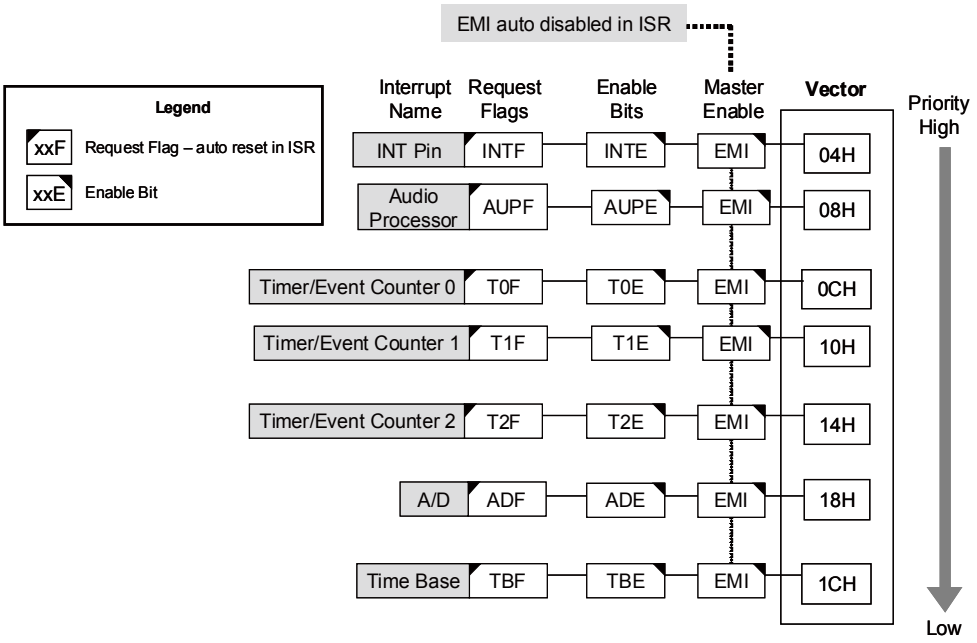
Bit 1 **T2E**: Timer/Event Counter 2 interrupt enable
0: disable
1: enable

Bit 0 **T1E**: Timer/Event Counter 1 interrupt enable
0: disable
1: enable

Interrupt Operation

A Timer/Event Counter overflow, an active edge on the external interrupt pin, a serial data byte transmitted or received completion, or a Time Base event 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.

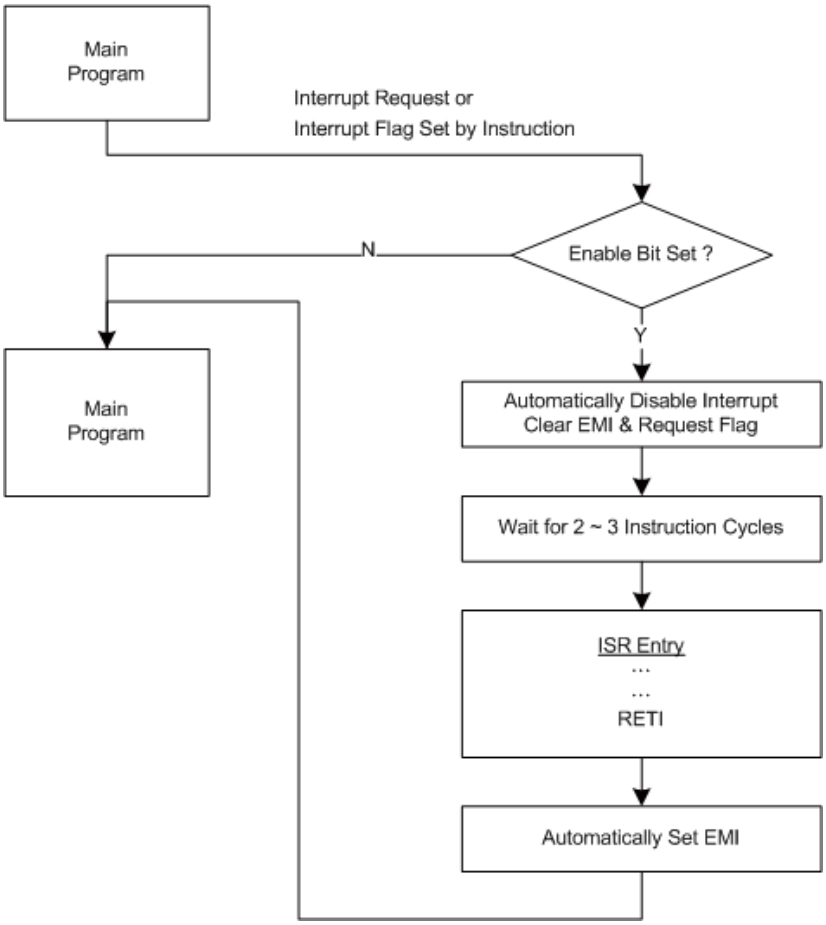
The various interrupt enable bits, together with their associated request flags, are shown in the following diagram with their order of priority.



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 cycle 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
External interrupt	1	04H
Audio Processor interrupt	2	08H
Timer/Event Counter 0 overflow	3	0CH
Timer/Event Counter 1 overflow	4	10H
Timer/Event Counter 2 overflow	5	14H
A/D interrupt	6	18H
Time base counter overflow	7	1CH

Interrupt Subroutine Vector

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.

External Interrupt

For an external interrupt to occur, the global interrupt enable bit, EMI, and external interrupt enable bit, INTE, must first be set. An actual external interrupt will take place when the external interrupt request flag, INTF, is set, a situation that will occur when an edge transition appears on the external INT line. The type of transition that will trigger an external interrupt, whether high to low, low to high or both is determined by the INTEG0 and INTEG1 bits, which are bits 6 and 7 respectively, in the CTRL1 control register. These two bits can also disable the external interrupt function.

INTEG1	INTEG0	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 pin is pin-shared with the I/O pin PA3 and can only be configured as an external interrupt pin if the corresponding external interrupt enable bit in the INTC0 register has been set and the edge trigger type has been selected using the CTRL1 register. The pin must also be setup as an input by setting the corresponding PAC.3 bit in the port control register. When the interrupt is enabled, the stack is not full and an active transition appears on the external interrupt pin, a subroutine call to the external interrupt vector at location 04H, will take place. When the interrupt is serviced, the external interrupt request flag, EIF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor connections on this pin will remain valid even if the pin is used as an external interrupt input.

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 n 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 Interrupt

The internal Time base Counter interrupt is initialized by setting the Time base overflow interrupt request flag, which is normally caused by a timer overflow. After the interrupt is enabled, and the stack is not full, and the TBF bit is set, a subroutine call occurs. The related interrupt request flag TBF is reset, and the EMI bit is cleared to disable further mask-able interrupts.

A/D Converter Interrupt

The A/D Converter interrupt is initialized by setting the A/D converter request flag, caused by an end of A/D conversion. When the interrupt is enabled, the stack is not full and the ADF is set, a sub-routine call will occur. The related interrupt request flag ADF will be reset and the EMI bit cleared to disable further interrupts.

Audio Processor Interrupt

The Audio Processor interrupt is initialized by setting the Audio Processor request flag,. When Audio Processor returns a data, the SPICR[0] transits from "1" to "0" . If the interrupt is enabled, the stack is not full and the AUPF is set, a sub-routine call will occur. The related interrupt request flag AUPF will be reset and the EMI bit cleared to disable further interrupts.

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 re- main in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program. It is recommended that programs do not use the CALL instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a CALL subroutine is executed in the interrupt subroutine.

Every interrupt has the capability of waking up the microcontroller when it is in 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 to the memory at the beginning of the interrupt service routine.

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

SPI Function

SPI interface is one method to communicate with Audio Processor. The command groups types can be referred to chapter of Audio Processor Interface.

SPI format and control register as following.

SPICR Register

Bit	7	6	5	4	3	2	1	0
Name	IEMC	—	ERAM	SPISS	SPICK	MOSI	MISO	SPIRQ
R/W	R/W	—	R/W	R/W	R/W	R/W	R	R
POR	1	—	0	1	0	0	x	x

Bit 7 I**EMC**: Mode control bit
0: external MCU mode (pad control) PAD of PA5/PC4/PC5/PC6/PC7 Pin shared MISO/ MOSI/ SPIRQ/ SPICK/SPISS
1: internal mode (MCU control)
In external MCU mode, the Audio Processor can be controlled by external MCU via SPI interface. SPISS/SPICK/MOSI can not read/write and MISO/SPIRQ read only when SPIRQ falling edge occur, it will not effecting AUPF to prevents interrupts occurring

Bit 6 unimplemented, read as "0"

Bit 5 **ERAM**: PAD of PB4/PB5/PB6/PB7 Pin shared AUDIO PROCESSORIO0/1/2/3
0: Disable
1: Enable

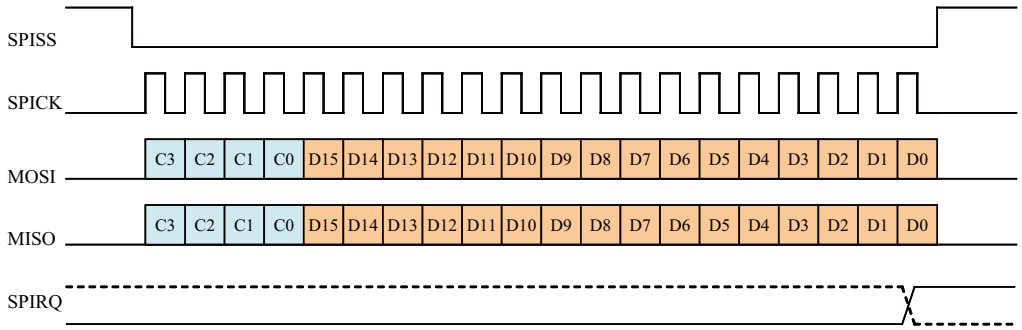
Bit 4 **SPISS**: Select AUDIO PROCESSOR
0: Select
1: No select

Bit 3 **SPICK**: SPI clock output to AUDIO PROCESSOR
0: low
1: high

Bit 2 **MOSI**: Output command/data into AUDIO PROCESSOR
0: low
1: high

Bit 1 **MISO**: Input command/data from AUDIO PROCESSOR

Bit 0 **SPIRQ**: Set to 0 by Audio processor, when Audio processor interrupt occurs. this bit will be set to 1 by Audio processor after MCU read command/data successfully.



BEEP Function

This function is used Application program building a square wave on speak or Audio Processor through BEEP0/1 pad.

BDR Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	BEEP1	BEEP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit7~6 unimplemented, read as "0"

Bit 1 **BEEP1**: output signal
0: low
1: high
When BEEP1 Function is useful, if Audio Processor souce is selected beep1, the Beep 1 pad is connect to Audio Processor

Bit 0 **BEEP0**: output signal
0: low
1: high
When BEEP0 Function is useful, if speak souce is selected beep0, the Beep 0 pad is connect to speak.
The two bits are output pin signals, set or clear BDR data can modify beep0/beep1 status. If the two functions are not useful, modify beep0/beep1 data are not affected beep1/0 pad status.

Digital to Analog Converter – DAC

The device includes 4-channel 8-bit Digital to Analog Converter function. This function allows digital data contained in the device to generate audio signals.

Operation

The data to be converted is stored in four registers DA0R, DA1R, DA2R and DA3R. The four bits in the control register DACR provides DACn on/off control. The DACn outputs are channeled to pin DAO0~DAO3 which are pin-shared with I/O pin PB0~PB3. When the DACn is enabled by setting the ENn high, then the original I/O function will be disabled, along with any pull-high resistor options. The DAC output reference voltage is the power supply voltage V_{DD}.

DACR Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	EN3	EN2	EN1	EN0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 unimplemented, read as "0"

Bit 3 **EN3**: DAC3 disable/enable control
0: disable
1: enable

Bit 2 **EN2**: DAC2 disable/enable control
0: disable
1: enable

Bit 1 **EN1**: DAC1 disable/enable control
0: disable
1: enable

Bit 0 **EN0**: DAC0 disable/enable control
0: disable
1: enable

DA3R Register

Bit	7	6	5	4	3	2	1	0
Name	DA37	DA36	DA35	DA34	DA33	DA32	DA31	DA30
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x" unknown

Bits 7~0 **DA37~ DA30**: DAC3 output data

DA2R Register

Bit	7	6	5	4	3	2	1	0
Name	DA27	DA26	DA25	DA24	DA23	DA22	DA21	DA20
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x" unknown

Bits 7~0 **DA27~ DA20**: DAC2 output data

DA1R Register

Bit	7	6	5	4	3	2	1	0
Name	DA17	DA16	DA15	DA14	DA13	DA12	DA11	DA10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x" unknown

Bits 7~0 **DA17~ DA10**: DAC1 output data

DA0R Register

Bit	7	6	5	4	3	2	1	0
Name	DA07	DA06	DA05	DA04	DA03	DA02	DA01	DA00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x" unknown

Bits 7~0 **DA07~ DA00**: DAC0 output data

Low Voltage Detector – LVD

The device has a Low Voltage Detector function, known as LVD. This enables the device to monitor the power supply voltage, VDD, 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.

LVD Register

The Low Voltage Detector is controlled using a single register, LVDC, and configuration options. The voltage threshold level to be detected is determined using a configuration option, therefore cannot be modified by the application program.

LVDC Register

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

Bit 7~6 unimplemented, read as "0"

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

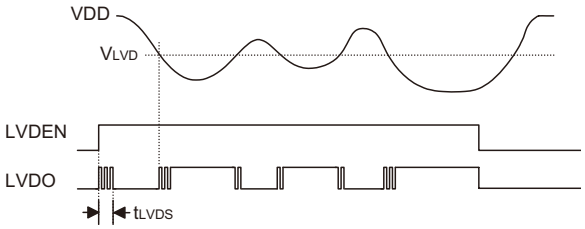
Bit 4 **LVDEN**: Low Voltage Detector Control
0: Disable
1: Enable

Bit 3~0 unimplemented, read as "0"

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, VDD, with a pre-specified voltage level voltage level setup using a configuration option. When the power supply voltage, VDD, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is powered down the low voltage

detector will remain active if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay tLVDS should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the VDD voltage may rise and fall rather slowly, at the voltage nears that of VLVD, there may be multiple bit LVDO transitions.



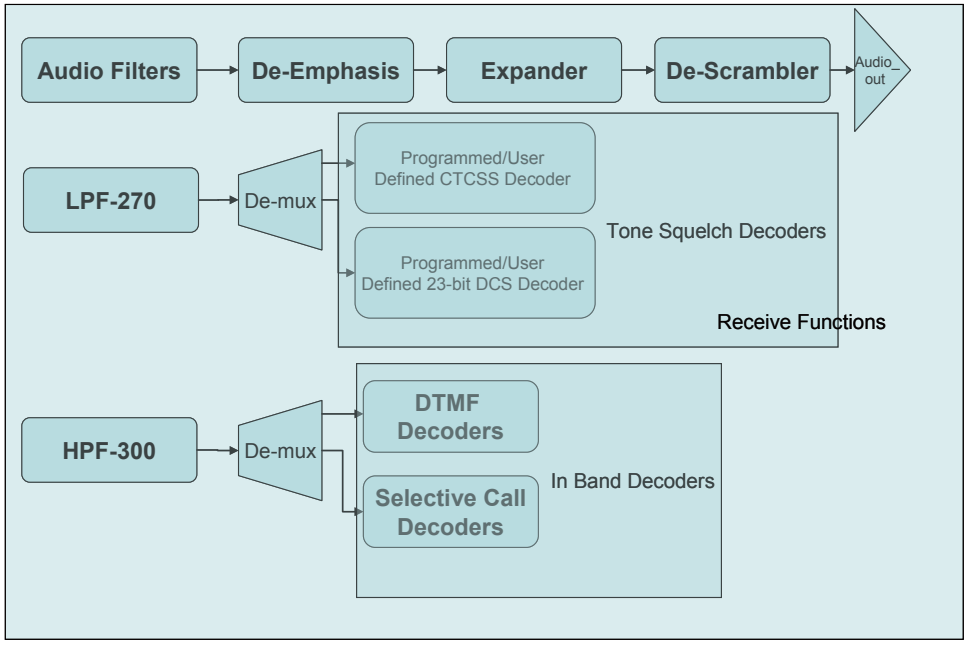
LVD Operation

Audio Processor

The device includes an audio processor function for processing of signals received on the RF carrier. Aimed specifically at low cost radio communication products the audio processor section contains all the transmitter and receiver functions required to process the necessary in-band audio signals such as compression, expansion, scrambling and de-scrambling.

Audio Receiver

The following diagram shows the main functional blocks of the receiver audio processor.

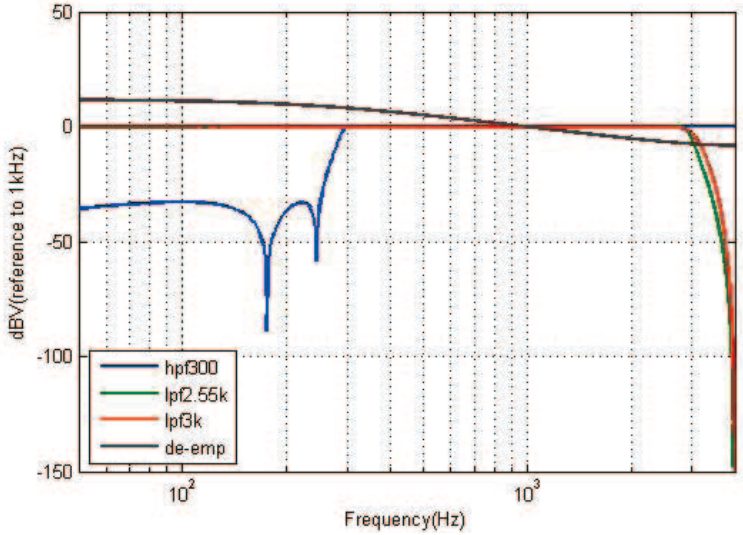


Receiver Block Diagram

Receiver Audio Filters

The incoming signal is first filtered, as shown in the block diagram, to remove sub-audio components and to filter out high frequency noise. After this low and high pass filtering is executed, the residual audio signal is then routed to the AUDIO output. Individual filters are available for the following:

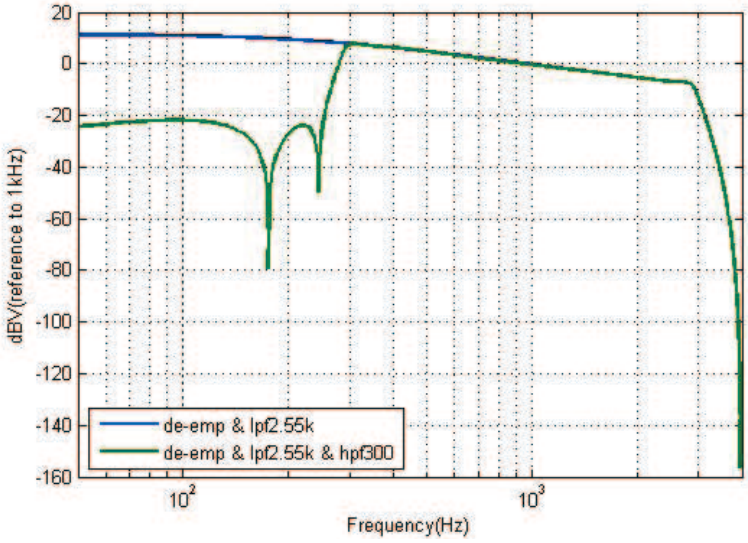
- 300Hz High Pass for rejection of sub audio signals
- 2.55kHz Low Pass for Narrow Bandwidth channel operation
- 3.0kHz Low Pass for Wide Bandwidth channel operation



Receiver Audio Filter Frequency Response

Receiver De-emphasis

The device includes a selectable pre-emphasis filter with a response as shown in the following figure.



Audio Frequency De-emphasis

Receiver De-scrambler

If desired, the transmitter section of the device can be setup to scramble the transmitted audio signals by inversing the transmitted audio frequencies. In the receiver mode, a descrambler is used to convert the received scrambled signals back to their original audio frequency content. The inversion central frequency can be programmed using the CLI Scramble Frequency register. The default inversion central frequency value is 3300Hz.

Receiver Expander

The receiver section includes an expander function to expand any signals that may have been compressed by the transmitter. This function is optional and is only used if the transmitter is transmitting compressed signals. Compressing and expanding the audio signals increases greatly the signal dynamic range.

Receiving and Decoding CTCSS or DCS Codes

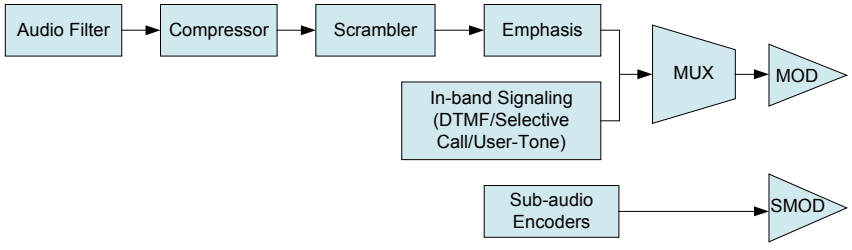
The device can accurately detect valid CTCSS tones rapidly which prevents losing the start of audio or data transmissions. It can also continuously monitor the detected tones low risk of false drop out. The DCS code is in NRZ format and is transmitted at a rate of 134.4±0.4bps. The device is able to decode any 23-bit pattern in either of the two DCS modulation modes as defined by TIA/EIA-603. The device can also rapidly detect valid DCS code quickly enough to avoid losing the beginning of audio transmissions.

Rx In-Band Tone Decoder

The in-band tone can be summarized into Selective call, DTMF and User-tone categories. For Selective call, the HT98R068-1 will decode a set of EEA defined tone sets, however this may be changed by the users to any valid tone sets within its operational range by setting related registers. This ensures that the device can remain compatible with all available tone systems in use. The HT98R068-1 does not implement automatic repeat tone insertion or deletion as this depends upon the user protocol.

Audio Transmitter

The following diagram shows the main function blocks of the transmitter audio processor.

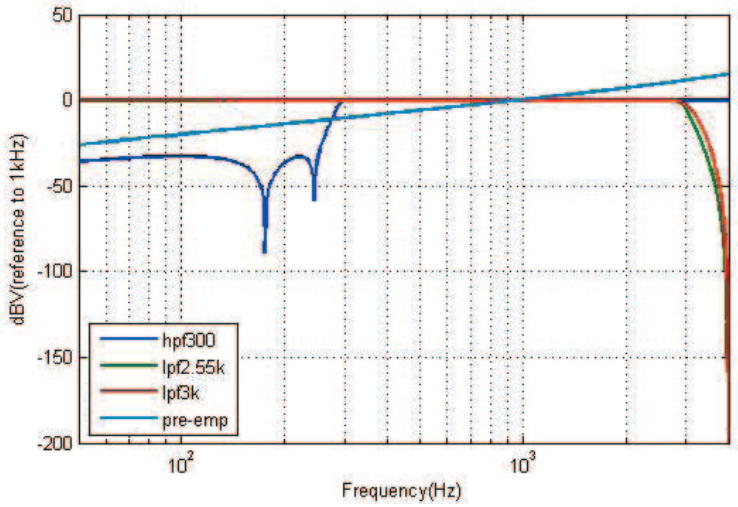


Transmitted Functions

Transmitter Block Diagram

Transmitter Audio Filters

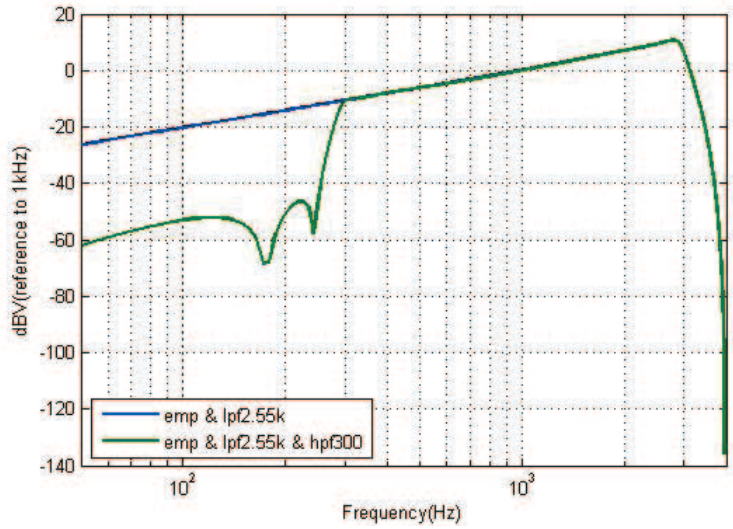
The transmitter audio filters include a 25kHz wide-band and a 12.5kHz narrow-band channel filter. The filter frequency responses are shown in the following figures.



Transmitter Audio Filter Frequency Response

Transmitter pre-emphasis

The device includes a selectable pre-emphasis filter with +6dB per octave from 300Hz to 3000Hz, with a response as shown in the following figure..



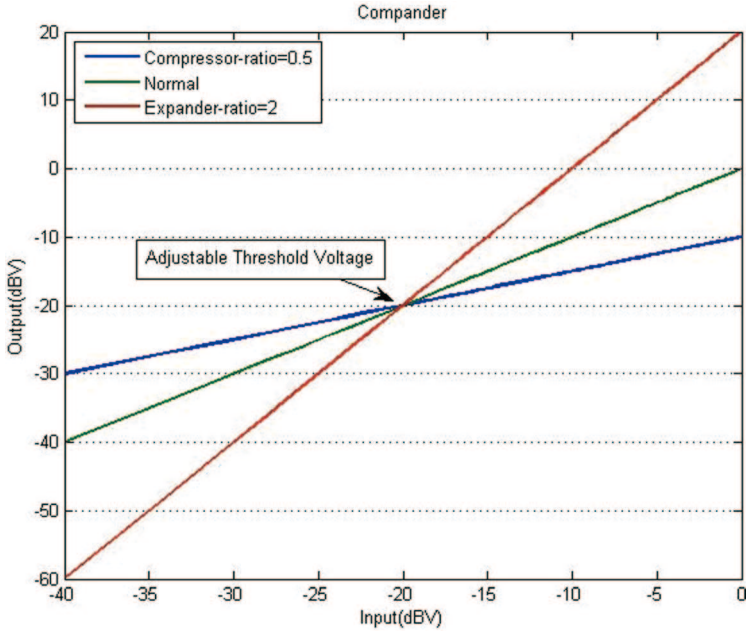
Transmitter Audio Frequency Pre-emphasis

Transmitter Scrambler

The device includes an optional frequency inversion scrambler for both the transmit and receive modes. This scrambles the transmitted audio band signals, which can then be de-scrambled in the receiver. The central inversion frequency is programmed using registers. The default value is 3300Hz. The central inversion frequency can be changed not only in the idle mode but also in the active Rx or Tx mode.

Transmitter Compressor

The device incorporates an optional syllabic compandor for both the transmit and receive mode. This compresses the audio band signals before transmission to enhance the overall dynamic range. The relationship between the input and output voltage of the compandor is shown in the following figure.



Compander voltage-voltage response

Transmitter Sub-audio Encoders

Sub-audio signaling is available in the audio band below 260Hz. When sub-audio signaling is enabled, the 300Hz HPF in the audio section should also be enabled to remove the sub-audio signaling from the actual audio signals (for both Tx and Rx). Both CTCSS tones and DCS codes are supported. Except for the 51 defined CTCSS tones, users can define their specified tones using registers. The DCS coder/decoder includes a 23-bit mode with both normal and inverse modulation formats and a 134Hz end of transmission burst.

Transmitter In-Band Signaling

The device includes a programmable in-band tone set or an arbitrary single user-tone between 300Hz and 3000Hz. By default, the device will use an EEA selective call defined tone set; however this may be changed to any valid tone within its operational range using registers. As well as the selective call defined tone sets, standard dual tone multiple frequency, DTMF, is also supported in the in-band signaling mode.

Supported Different Combination Functions

For optimal power consumption, the audio processor can be set to operate with different operation frequencies for different functional requirements. The accompanying table provides a reference for the required audio processor system clock for different applications.

Receiver

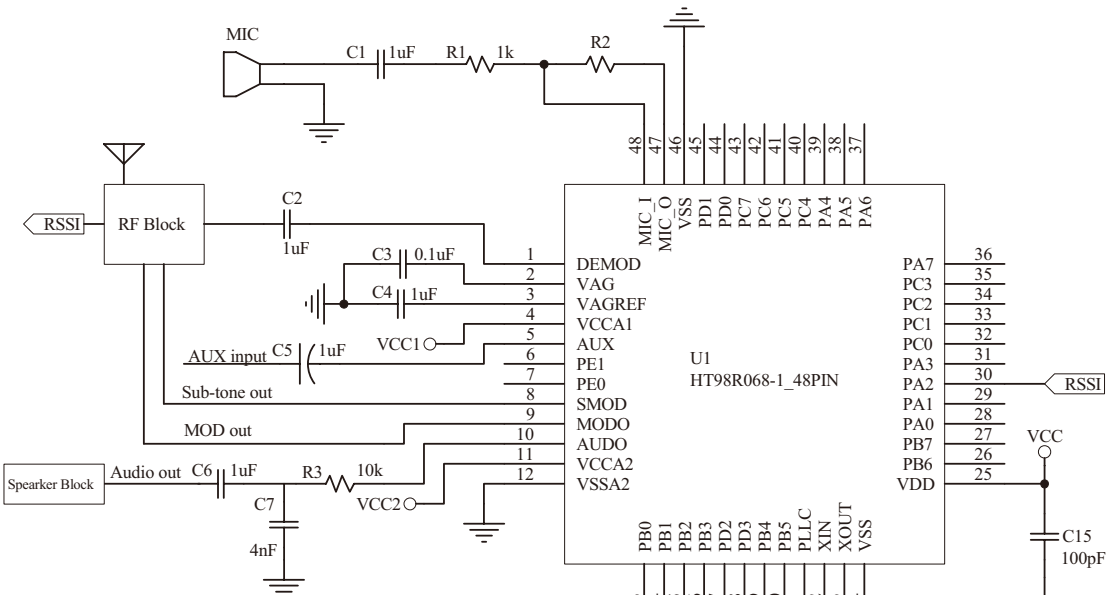
Voice Band Processing	Sub-Audio	Voice Band Signaling	Required System Clock (f_{ap})
Basic Voice Filter	CTCSS	None	12M
	inv DCS, DCS	None	16M
	CTCSS	DTMF Decoder	16M
	sub-audio not supported	A. None	8M
		B. DTMF Decoder	16M
		C. Selective Call Decoder	16M
Basic Voice Filter + De-emphasis	inv DCS, DCS / CTCSS	None	16M
	sub-audio not supported	A. None	8M
		B. DTMF Decoder	16M
Basic Voice Filter + Expander	inv DCS, DCS / CTCSS	None	16M
	sub-audio not supported	A. None	16M
		B. DTMF Decoder	16M
Basic Voice Filter + De-emphasis + Expander	sub-audio not supported	None	12M
	CTCSS	DTMF Decoder	16M
	inv DCS, DCS	None	16M
Basic Voice Filter + De-emphasis + Expander+Scramble	sub-audio not supported	None	12M
	CTCSS	None	16M
	DCS	None	16M
VOX	None	None	4M

Transmitter

Voice Band Processing	Sub-Audio	Voice Band Signaling	Required System Clock (f_{ap})
Basic Voice Filter	inv DCS, DCS / CTCSS	A. None	12M
		B. Tone Generator	16M
	sub-audio not supported	A. None	8M
		B. Tone Generator	12M
Basic Voice Filter + Emphasis	inv DCS, DCS / CTCSS	A. None	12M
		B. Tone Generator	16M
	sub-audio not supported	A. None	8M
		B. Tone Generator	12M
Basic Voice Filter + Compress	inv DCS, DCS / CTCSS	A. None	16M
		B. Tone Generator	16M
	sub-audio not supported	A. None	12M
		B. Tone Generator	12M
Basic Voice Filter + Emphasis + Compress	inv DCS, DCS / CTCSS	A. None	16M
	sub-audio not supported	A. None	12M
		B. Tone Generator	16M

Audio Signal Routing

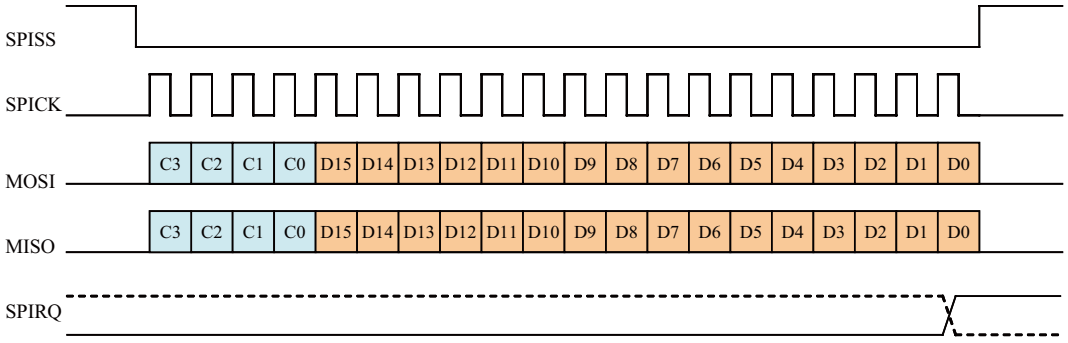
The device has a flexible audio signal routing input and output structure to route signals to and from the audio processor. The routing setup paths is controlled using the path selection register which is described in a different section. The gain of the internal Programmable Gain Amplifier shown in the block diagram is also setup using a register, the details of which are described in a different section.



Audio Signal Routing Block Diagram

MCU Interfacing

The Audio Processor communicates with the integrated MCU using a series of command registers. Communication with external MCUs is conducted using the SPICR register and the PC4~PC7 I/O lines which have shared use as an SPI interface.



SPISS from the MCU: This pin is an input pin and active low.

SPICK from the MCU: This pin is the clock source input pin.

MOSI (SMOSI) from the MCU: This pin is the slave SPI input data pin. MOSI refers to Master Output Slave Input.

MISO (SMISO) from the Audio Processor: This pin is the slave SPI output data pin. MISO refers to Master Input Slave Output.

SPIRQ from the Audio Processor: This pin is the slave SPI output pin. It is used to improve the CLI interface access data speed.

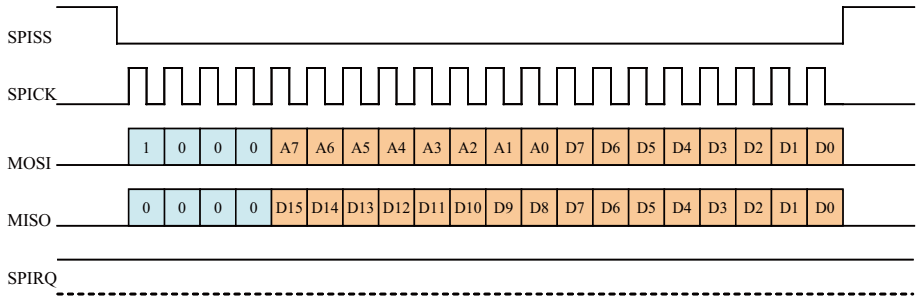
The first half byte C3~C0 form the SPI command which is followed by two bytes D15~D0 of data

Command Groups

There are two types of command for the host to communicate with the audio processor. One is an I/O command group and the other is a CLI command group. The I/O command group is related to the switching on/off function. The CLI command group is related to the audio processing functions.

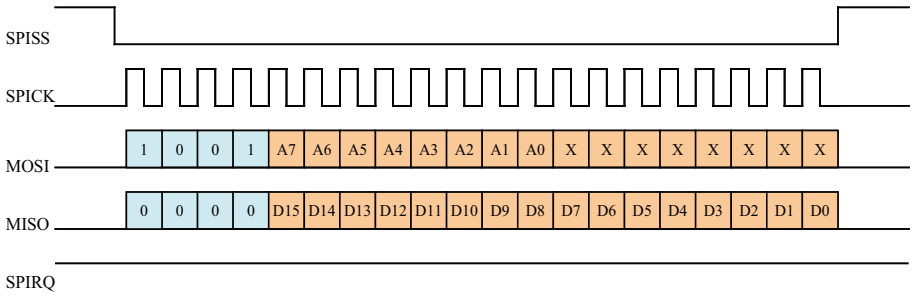
IO Command Group Write – C [3:0]= "1000"

The MCU will send 8 bits of IO command address and 8 bits of IO command data to Audio Processor. The host device sends IO register write command to Audio Processor. Waveform is showed as follows:

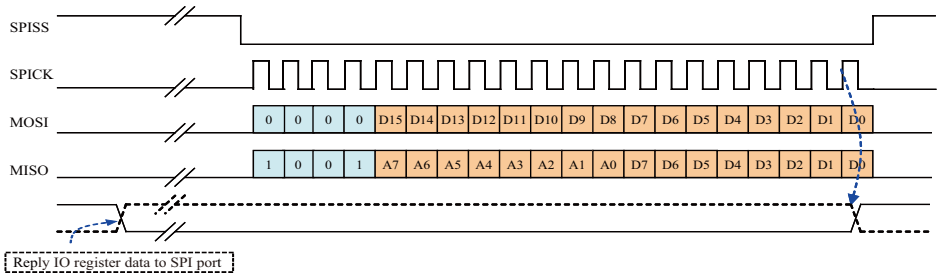


I/O Command Group Read – C [3:0]= "1001"

The host device will read 8 bits of the IO command address and 8 bits of the IO command data from the Audio Processor. The IO command group requires a dual SPI command cycle. During the first cycle, the IO read command (IO address) is sent to the Audio Processor. During the second cycle, the Audio Processor returns its IO command data back to the host. The following two waveforms show the IO command read timing.



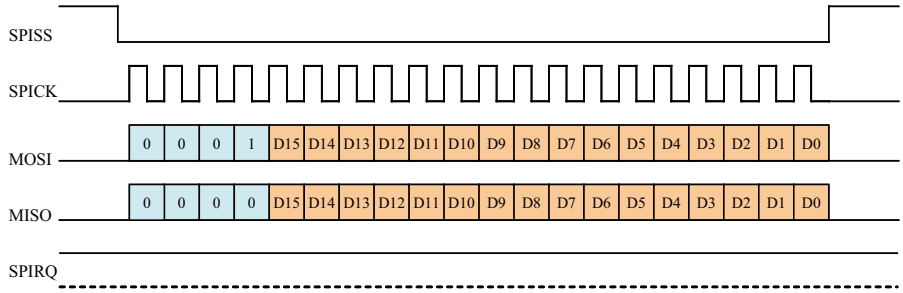
During the first cycle, the IO command (IO address) is sent to the Audio Processor



During the second cycle, the Audio Processor returns its IO command data to the host

CLI Command Group

The following waveform shows the host sending a CLI command to the Audio processor.



Write CMD Details

CLI_CMD	Major	Minor	Multi	Length
4'b**0001**	4'b**0100**	4'b**0000**	4'b**1000**	4'b**0010**
4'b0001	Address[15:0]			
4'b0001	Data[15:0]			

Reply:

CLI_CMD	Major	Minor	Multi	Length
4'b0001	4'b0100	4'b0000	4'b0000	4'b0000

Note: When the audio processor replies with '14000' this means that the write data conditions has been met.

Read CMD Details

CLI_CMD	Major	Minor	Multi	Length
4'b**0001**	4'b**0100**	4'b**0001**	4'b**1000**	4'b**0001**
4'b0001	Address[15:0]			

Reply:

CLI_CMD	Major	Minor	Multi	Length
4'b0001	4'b0100	4'b0001	4'b1000	4'b0001
4'b0001	Data[15:0]			

Command Group Summary

The I/O Commands are summarised in the following table.

Address / Bit	7	6	5	4	3	2	1	0
00~10	Reserved							
11	—	Operation Mode		Audio Band Mode			Sub Audio Mode	
12~19	Reserved							
1A	—			PGA_B				
1B	PGAI_S			AUDO_S			SDAO2	SDAO1
1C~1D	Reserved							
1E	/EN_DAC2	/EN_DAC1	EN_AMP2	EN_AMP1	EN_BUF	EN_MIC	EN_PGA	1'b1
1F~21	Reserved							
22	—	IRQ	DTMF INT	Selective call INT	CTCSS INT	DCS INT	Off_Tone INT	VOX INT

Address / Bit	7	6	5	4	3	2	1	0
23	—		DTMF Event	Selective call Event	CTCSS Event	DCS Event	Off_Tone Event	VOX Event
24~28	Reserved							
29	—						VOX Threshold Status	
2A	—				Selective Call			
2B	Sub_Inv	Sub Audio Tone						
2C	EN_Scram	EN_Comp	EN_Emp	EN_NBW	EN_WBW	EN_HPF300	EN_VOX	EN_AGC
2D	—				DTMF Tone			
2E	—				Selective Call Finder			
2F	—				DTMF Finder			
30	—							CTC_Anti-tone Event
31	—						EN_CTC_Rx_Anti-tone	EN_CTC_Tx_Anti-tone
51	2'b1		EN_AUD0	2'b1		0	EN_AUD1	0

I/O Command Group Detail

The details behind each I/O command are provided in the following tables.

Mode Control - 11h Address

Bit	7	6	5	4	3	2	1	0
Name	—	Operation Mode		In-band Tone Mode			Sub Audio Mode	

Bit7 Unimplemented, ignore this value

Bit6~5 **Operation Mode**: Audio processor operation mode selection
01: SLOW mode (VOX used only)
10: TX mode
11: RX mode

Bit4~2 **In-band Tone Mode**: In-band tone signal selection
001: User Tones (**to DA1**)
010: Selective Call signal
100: DTMF
Others : ALL Disable

Bit1~0 **Sub Audio Mode**: Sub tone selection
01: DCS
10: CTCSS
Others: ALL Disable

PGA Gain Control - 1Ah Address

Bit	7	6	5	4	3	2	1	0
Name	—			PGA_B				

Bit7~5 Unimplemented, ignore this value

Bit4~0 **PGA_B**: PGA gain x setting
The PGA Gain adjustment range is defined as (1 + x/4) , where x = 0 ~ 31,Initial reset value is '00000'

Path Selection - 1Bh Address

Bit	7	6	5	4	3	2	1	0
Name	PGAI_S			AUDO_S			SDAO2	SDAO1

Bit7~5 **PGAI_S**: PGA input source selection
000: MICO
001: DEMOD
010: AUX
011: BEEP1
1xx:VAG

Bit4~2 **AUDO_S**: AUDO output buffer source selection
001: DAC1 (output range = 1/4 ~ 3/4 V_{DD})
011: BEEP0
100 : DAC common-mode bias (V_{DD}/2)
Other : Must not be used

Bit1 **SDAO2**: DAC2 input source selection
0: DAO2 input = DAC
1: DAO2 input = internal common-mode bias

Bit0 **SDAO1**: DAC1 input source selection
0: DAO1 input = DAC
1: DAO1 input = internal common-mode bias

Power Control - 1Eh Address

Bit	7	6	5	4	3	2	1	0
Name	$\overline{\text{EN\_DAC2}}$	$\overline{\text{EN\_DAC1}}$	EN_AMP2	EN_AMP1	EN_BUF	EN_MIC	EN_PGA	1'b1

Bit7 **$\overline{\text{EN\_DAC2}}$**: DAC2 on/off control
0: Enable DAC2
1: Disable DAC2

Bit6 **$\overline{\text{EN\_DAC1}}$**: DAC1 on/off control
0: Enable DAC1
1: Disable DAC1

Bit5 **EN_AMP2**: AMP2 on/off control
0: Disable AMP2
1: Enable AMP2

Bit4 **EN_AMP1**: AMP1 on/off control
0: Disable AMP1
1: Enable AMP1

Bit3 **EN_BUF**: Buffer on/off control
0: Disable Buffer
1: Enable Buffer

Bit2 **EN_MIC**: MIC on/off control
0: Disable MIC
1: Enable MIC

Bit1 **EN_PGA**: PGA on/off control
0: Disable PGA
1: Enable PGA

Bit0 **1'b1**: Reserved bit, must be set high

Event Interrupt Mask - 22h Address

Bit	7	6	5	4	3	2	1	0
Name	—	IRQ	DTMF INT	Selective call INT	CTCSS INT	DCS INT	Off_Tone INT	VOX INT

Bit7 Unimplemented, ignore this value

Bit6 **IRQ**: Interrupt request - (DTMF, Selective call, CTC, DCS, VOX) issued on the MCU PE
0: Disable IRQ interrupt request
1: Enable IRQ interrupt request

Bit5 **DTMF INT**: DTMF event interrupt issue enable/disable selection
0: Disable CTCSS interrupt request
1: Enable CTCSS interrupt request

Bit4 **Selective call INT**: Selective Call event interrupt issue enable/disable selection
0: Disable CTCSS interrupt request
1: Enable CTCSS interrupt request

Bit3 **CTCSS INT**: CTCSS event interrupt issue enable/disable selection
0: Disable CTCSS interrupt request
1: Enable CTCSS interrupt request

Bit2 **DCS INT**: DCS event interrupt issue enable/disable selection
0: Disable DCS interrupt request
1: Enable DCS interrupt request

Bit1 **Off_Tone INT**: Off_Tone event interrupt issue enable/disable selection
0: Disable Off_Tone interrupt request
1: Enable Off_Tone interrupt request

Bit0 **VOX INT**: VOX event interrupt issue enable/disable selection
0: Disable VOX interrupt request
1: Enable VOX interrupt request

Event Status - 23h Address

Bit	7	6	5	4	3	2	1	0
Name	—	—	DTMF Event	Selective Call Event	CTCSS Event	DCS Event	Off_Tone Event	VOX Event

Bit7~6 Unimplemented, ignore this value

Bit5 **DTMF Event**: DTMF code detection
0: No detected DTMF event change
1: DTMF event status has detected changet

Bit4 **Selective Call Event**: Selective Call detection.
0: No detected Selective call event change
1: Selective call event status has detected change

Bit3 **CTCSS Event**: CTCSS code detection
0: No detected CTCSS event change
1: CTCSS event status has detected change

Bit2 **DCS Event**: DCS code detection
0: No detected DCS event change
1: DCS event status has detected change

Bit1 **Off_Tone Event**: Off_Tone code detection
0: No detected Off_Tone event change
1: Off_Tone event status has detected change

Bit0 **VOX Event**: VOX signal above high or below low threshold
0: No detected VOX event change
1: VOX event status has detected change

VOX Threshold Status - 29h Address

Bit	7	6	5	4	3	2	1	0
Name	—						VOX Threshold Status	

Bit7~2 Unimplemented, ignore this value

Bit1~0 **VOX Threshold Status**: VOX high/low threshold data compare result
01:Below Low Threshold
10:Above High Threshold
Others: Don't care

Selective Call Tone - 2Ah Address

Bit	7	6	5	4	3	2	1	0
Name	—				Selective Cal			

Bit7~4 Unimplemented, ignore this value

Bit3~0 **Selective Call**: Selective call tone number selection
Bit3-0: 0000 – 1111 assigned selective call number 0 – F, following EEA protocol.
The 16 number tones are indicated in the address range: 04e0 – 04ff

Sub-audio Tone - 2Bh Address

Bit	7	6	5	4	3	2	1	0
Name	Sub_Inv	Sub Audio Tone						

Bit7 **Sub_Inv**: DCS inverted select bit
In DCS mode:
0: Non-inverted DCS code
1: Inverted DCS code
In CTCSS mode: Don't care

Bit6~0 **Sub Audio Tone**: Sub Audio Tone Channel Number
Bit6-0: Select DCS or CTCSS tone number, see the following table

Sub Audio Mode	Number(Dec)	Code(Dec) / Tone
DCS	0	User defined DCS code
	1-83	DCS code 1~83
	85~126	No tone
	127	DCS turn off tone (**to DA2**)
CTCSS	0	User defined CTCSS tone
	1~51	CTCSS tone 1~51
	53~127	No tone

Sub-audio Tone Number Table

Audio Control - 2Ch Address

Bit	7	6	5	4	3	2	1	0
Name	EN_Scram	EN_Comp	EN_Emp	EN_NBW	EN_WBW	EN_HPF300	EN_VOX	EN_AGC

Bit7 **EN_Scram**: Audio Scrambling on/off control
0: Disable Audio Scrambling
1: Enable Audio Scrambling

Bit6 **EN_Comp**: Audio Compandor on/off control
0: Disable Audio Compandor
1: Enable Audio Compandor

Bit5 **EN_Emp**: Audio Pre/De-emphasis on/off control
0: Disable Audio Pre/De-emphasis
1: Enable Audio Pre/De-emphasis

Bit4 **EN_NBW**: Narrow band-width channel on/off control
0: Disable narrow band-width
1: Enable narrow band-width

Bit3 **EN_WBW**: Wide band-width channel on/off control
0: Disable wide band-width
1: Enable wide band-width

Bit2 **EN_HPF300**: Audio 300Hz HPF on/off control
0: Disable Audio 300Hz HPF
1: Enable Audio 300Hz HPF

Bit1 **EN_VOX**: VOX detected on/off control
0: Disable VOX detected
1: Enable VOX detected

Bit0 **EN_AGC**: AGC function on/off control
0: Disable AGC
1: Enable AGC
When the AGC was enabled, it was suggested that the mic op gain set 5. However, the correspond gain should be scaled corresponding to the operation voltage(i.e. * Volt/3.3).

DTMF Tone - 2Dh Address

Bit	7	6	5	4	3	2	1	0
Name	—				DTMF Tone			

Bit7~4 Unimplemented, ignore this value

Bit3~0 **DTMF Tone**: DTMF number selection
Bit3-0: 0000 – 1111 assigned DTMF number, see the following table.

Low Group (Hz)	High Group (Hz)	Digital	B3,B2,B1,B0
697	1209	1	0001
697	1336	2	0010
697	1477	3	0011
770	1209	4	0100
770	1336	5	0101
770	1477	6	0110
852	1209	7	0111

Low Group (Hz)	High Group (Hz)	Digital	B3,B2,B1,B0
852	1336	8	1000
852	1477	9	1001
941	1209	*	1011
941	1336	0	1010
941	1477	#	1100
697	1633	A	1101
770	1633	B	1110
842	1633	C	1111
941	1633	D	0000

DTMF Data Output Table

Selective Call Finder - 2Eh Address

Bit	7	6	5	4	3	2	1	0
Name	—				Selective Call Finder			

Bit7~5 Unimplemented, ignore this value

Bit4~0 **Selective Call Finder**: Detecting Selective Call number

These four bits indicate the detected Selective call tone number.
In the receiver mode, if the selective call mode is enabled, the processor will calculate the demodulation signal whether or not a tone exists. When a tone is detected , the calculation result will be auto stored in the selective call finder register.

DTMF Finder - 2Fh Address

Bit	7	6	5	4	3	2	1	0
Name	—				DTMF Finder			

Bit7~5 Unimplemented, ignore this value

Bit4~0 **DTMF Finder**: Detected DTMF tone number

These four bits indicate the detected DTMF tone number.
In the receiver mode, if the DTMF mode is enabled, the processor will calculate the demodulation signal whether or not a tone exists. When a tone is detected , the calculation result will be auto stored in the DTMF finder register.

Evnet2 Status - 30h Address

Bit	7	6	5	4	3	2	1	0
Name	—							CTC_Anti-tone Event

Bit7~1 Unimplemented, ignore this value

Bit0 CTC_Anti-tone Event: CTCSS anti-tone detection
0: No detected CTCSS Anti-tone event change
1: CTCSS Anti-tone event status is detected change

Event2 Control - 31h Address

Bit	7	6	5	4	3	2	1	0
Name	—						EN_CTC_Rx_ Anti-tone	EN_CTC_Tx_ Anti-tone

Bit7~2 Unimplemented, ignore this value

Bit1 **EN_CTC_Rx_Anti-tone**: CTCSS Rx anti-tone on/off control
0: Disable CTCSS detect anti-tone in Rx mode.
1: Enable CTCSS detect anti-tone in Rx mode

Bit 0 **EN_CTC_Tx_Anti-tone**: CTCSS Tx anti-tone on/off control
When CTCSS is in the Tx mode, the EN_CTC_Anti bit can to enable CTCSS anti-tone signal. If this control bit 0→1 or 1→0 both have anti-tone effect (phase reversal 180˚).

Audio Processor Control - 51h Address

Bit	7	6	5	4	3	2	1	0
Name	2'b1		EN_AUD0	2'b1		0	EN_AUD1	0

Bit7~6, 4~3 **2'b1**: Reserved bit, must be all set high.

Bit5, 1 **EN_AUD0/1**: Audio processor turn on/off selection
If the system is to be placed into the standby status to save power, this two bit should be set high. The EN_AUP0 and EN_AUP1 bit stops the control audio processor operation.
10: Enable Audio processor
01: Disable Audio processor

Bit2, 0 **0**: Zero, must be cleared to zero.

CLI Command Group Summary

The CLI Commands are summarised in the following table.

Address / Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
012a	Compressor Threshold															
012b	Expander Threshold															
012c ~ 0133	Reserved															
0134	0													AGC Step		
0135~0139	Reserved															
013a	Scrambler Inv. Freq. – parameter1															
013b	Scrambler Inv. Freq. – parameter2															
013c ~ 01c3	Reserved															
01c4	DTMF Power Threshold															
01c5~01dc	Reserved															
01dd	Off tone Accepted Voltage Register															
01de	Off tone Released Voltage Register															
01df~01e1	Reserved															
01e2	DCS Accepted Voltage Register															
01e3~0323	Reserved															
0324	Selective Call/User-Defined Tone Accepted Voltage Register															
0325	Selective Call/User-Defined Tone Released Voltage Register															
0326~04c9	Reserved															
04ca	CTCSS Power Released Threshold															
04cb	VR1								VR2							

Address / Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
04cd	VOX Threshold High															
04ce	VOX Threshold Low															
04cf	Reserved															
04d0	User CTCSS – Parameter1															
04d1	User CTCSS – Parameter2															
04d2	0								VR3							
04d3	0						VR4									
04d4	Reserved															
04d5	0						VR5									
04d6	—													CTCSS Freq.		
04d7~04d8	Reserved															
04d9	User-Tone Freq. – Parameter1															
04da	User-Tone Freq. – Parameter2															
04db	CTCSS Power Threshold															
04dc	User defined DCS codes[15:0]															
04dd	0										User defined DCS codes[22:16]					
04de	Drop Time Immunity															
04df	0	0	Soft Limiter													
04e0	Selective_Call_0 – Parameter1															
04e1	Selective_Call_0 – Parameter2															
04e2	Selective_Call_1 – Parameter1															
04e3	Selective_Call_1 – Parameter2															
04e4	Selective_Call_2 – Parameter1															
04e5	Selective_Call_2 – Parameter2															
04e6	Selective_Call_3 – Parameter1															
04e7	Selective_Call_3 – Parameter2															
04e8	Selective_Call_4 – Parameter1															
04e9	Selective_Call_4 – Parameter2															
04ea	Selective_Call_5 – Parameter1															
04eb	Selective_Call_5 – Parameter2															
04ec	Selective_Call_6 – Parameter1															
04ed	Selective_Call_6 – Parameter2															
04ee	Selective_Call_7 – Parameter1															
04ef	Selective_Call_7 – Parameter2															
04f0	Selective_Call_8 – Parameter1															
04f1	Selective_Call_8 – Parameter2															
04f2	Selective_Call_9 – Parameter1															
04f3	Selective_Call_9 – Parameter2															
04f4	Selective_Call_A – Parameter1															
04f5	Selective_Call_A – Parameter2															
04f6	Selective_Call_B – Parameter1															
04f7	Selective_Call_B – Parameter2															
04f8	Selective_Call_C – Parameter1															
04f9	Selective_Call_C – Parameter2															
04fa	Selective_Call_D – Parameter1															

Address / Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
04fb	Selective_Call_D – Parameter2															
04fc	Selective_Call_E – Parameter1															
04fd	Selective_Call_E – Parameter2															
04fe	Selective_Call_F – Parameter1															
04ff	Selective_Call_F – Parameter2															

Note: 1. When the different operation mode changes such as an Rx or Tx mode change, a soft_reset cmd "10000" and the initial parameters should be previously transmitted.

2. In the Rx mode, a mute path (except under Vox monitoring) should be selected before the RSSI signal is large enough. As the DCS/CTCSS detection time is also related with RSSI signal detection, the RF module must take into account this RSSI detection response time.

The following is a CLI command programming example.

Ex: Adjust the in-band tone and the sub-audio band tone level to a maximum.

- Write condition

  Master write data:

  14082 / Initialise the CLI write command

  104CB / Setup the 04CB register to be written to

  1FFFF / Write "FFFF" to the 04CB register.

  Audio processor reply:

  14000 / After receiving this reply, the above command is successfully accepted

- Read Condition

  Master read data:

  14181 /Initialise the CLI read command

  104CB / Setup the 04CB register to be read

  Audio processor reply:

  14181 /After receiving these two reply, above command is successfully accepted 1FFFF

CLI Command Group Detail

- Tx Compandor Threshold - **012Ah** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Tx Compandor Threshold															

Bit15~0 **Tx Compandor Threshold**: Setup
These 16-bits are used for setting the threshold voltage to determine if the signal is to be compressed or expanded. The default value is 0x515C (100mVrms@3.3V). When the input signal is larger than threshold voltage, the signal will be compressed otherwise the signal will be expanded. When the input signal is equal to threshold voltage, the signal will not be changed. After the compression and expansion process, the signal will be transmitted. Refer to the application notes for detailed register setting.

- Tx Compandor Threshold - **012Bh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Rx Compandor Threshold															

Bit15~0 **Rx Compandor Threshold**: Setting Rx Compandor threshold
These 16-bits are used for setting the threshold voltage to determine if the signal is to be compressed or expanded The default value is 0x0595 (100mVrms@3.3V). When the received signal is larger than the threshold voltage, the signal will be expanded otherwise the signal will be compressed. When the input signal is equal to threshold voltage, the signal will not be changed. After the compression and expansion process, the signal will revert back to the original input signal. Refer to the application notes for detailed register setting.

- AGC Step - **0134h** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	0													AGC Step		

Bit15~3 0: Zero, must be cleared to zero.

Bit15~0 **AGC Step** – Setting AGC attack/released time

The corresponding bits influence the AGC attack and release time. When AGC function was enabled, the bits should be set previously.

0x0000: no AGC (default)
0x0001: 3500ms
0x0002: 1600ms
0x0003: 1300ms
0x0004: 1100ms
0x0005: 900ms
0x0006: 800ms
0x0007: 600ms

- Scrambler Inversion Frequency - **013ah** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Scrambler Inversion Frequency – parameter1															

Bit15~0 **Scrambler Inversion Frequency – parameter1**: Scrambler inversion central frequency parameter1

- Scrambler Inversion Frequency - **013bh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Scrambler Inversion Frequency – parameter2															

Bit15~0 **Scrambler Inversion Frequency – parameter2**: Scrambler inversion central frequency parameter2

Scrambler inversion frequency range: 2.6kHz~3.4kHz. Default: 3.3kHz

Note: If user needs to define Scrambler inversion central frequency by self, 013a and 013b must be used at the same time.

- DTMF Power Threshold - **01C4h** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DTMF Power Threshold															

Bit15~0 **DTMF Power Threshold**: DTMF power threshold detector

This register is used to setup the minimum detected DTMF signal level. The default value is Vth=0xFF5B (200mVrms/600mVpp @3.3V) for both tones. Refer to the application notes for detailed register setting

- Off tone Accepted Voltage Register - **01DDh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Off Tone Accepted Voltage Register															

Bit15~0 **Off tone accepted voltage parameter [15:0]**: User defines the acknowledged Off tone power bit15-bit0. The default value was set 0xff2b (28mV-peak @3.3V).

- Off Tone Released Voltage Register - **01DEh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Off Tone Released Voltage Register															

Bit15~0 **Off tone released voltage parameter [15:0]**: User defines the deceased Off Tone power bit15-bit0. The default value was set 0xff05 (15mV-peak @3.3V).

- DCS Accepted Voltage Register - **01E2h** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DCS Accepted Voltage Register															

Bit15~0 **DCS Accepted Voltage parameter [15:0]**: User defines the sensitivity of DCS squelch signal. However, the corresponding value should be set greater than the overall system noise floor. The default value was set 0x00ff (72mVrms/200mVp-p @3.3V).

- Selective Call/User-Defined Tone Accepted Voltage Register - **0324h** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Selective Call/User-Defined Tone Accepted Voltage Register															

Bit15~0 **Selective Call/User-Defined Tone Accepted Voltage parameter [15:0]**: User defines the acknowledged Selective Call/User-Defined Tone power bit15-bit0. The default value was set 0xff60, which denotes 70mV-peak operating at 3.3V.

- Selective Call/User-Defined Tone Released Voltage Register - **0325h** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Selective Call/User-Defined Tone Released Voltage Register															

Bit15~0 Selective Call/User-Defined Tone Released Voltage parameter [15:0]: User defines the deceased Selective Call/User-Defined Tone power bit15-bit0. The default value was set 0xff30, which denotes 25mV-amplitude operating at 3.3V.

- CTCSS Release Power Threshold – **04CAh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CTCSS Power Released Threshold															

Bit15~0 CTCSS power released threshold voltage.

This register is to set the minimum CTCSS signal level that is to be released. The default value is 0Xff05 (15mV-peak@3.3V).

- In-Band Tone/Sub-audio Tx Level - **04CBh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	VR1								VR2							

Bit15~8 **VR1[7:0]**: In-Band Tone Tx Level

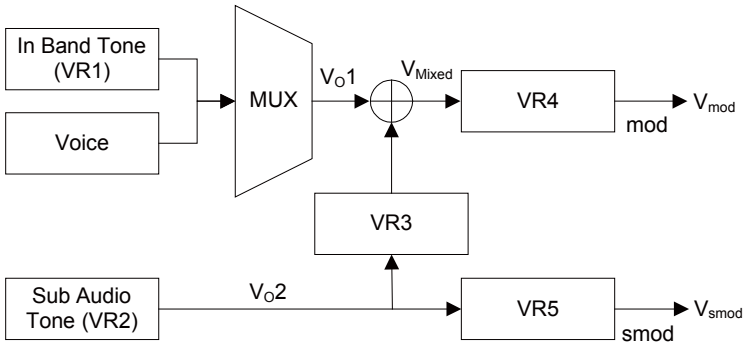
This register is to adjust the user-tone, selective call and DTMF transmission volume. Refer to the 'Modulation path block diagram' for information on optimal parameter usage. The VR1 setting is based on the formula: Vo1= Vvoice * 1.1 if bit2~b4='000' in the I/O command address 11, otherwise $Vo1=0.75V_{DD}$ * (VR1)/256; where Vvoice is the ADC input voltage.

Note: The actual output voltage will be a little degraded due to the DAC1 filter circuit.

Bit7~0 **VR2[7:0]**: Sub Audio Tone Tx Level

This register is to adjust the sub audio CTCSS and DCS transmission volume. Refer to the 'Modulation path block diagram' for information on optimal parameter usage. The VR2 setting is based on the formula:$Vo2= V_{DD}$ * (VR2)/256

Note: The actual output voltage will be a little degraded due to the DAC2 filter circuit.



Modulation Path Block Diagram

- Mixed Gain - **04D2** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	0								VR3							

Bit15~8 **0**: Zero, must be cleared to zero.

Bit7~0 **VR3**: Mixed Tone Gain Adjustment.

This register is used to adjust the mixed tone gain levels.

Refer to the 'Modulation path block diagram' for optimal parameter details..

The VR3 setting is based on the formula: $V_{mixed} = V_{o1} * (512-VR3)/512 + V_{o2} * (VR3/512)$

Note: V_{o1} can be output from the In-band tone tuned by VR1 or Voice

- MOD Gain - **04D3** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	0						VR4									

Bit15~10 **0**: Zero, must be cleared to zero.

Bit9~0 **VR4**: MOD Gain Adjustment.

This register is used to adjust MOD pin output gain levels.

Refer to the 'Modulation path block diagram' for optimal parameter details.

The VR4 setting is based on the formula: $V_{MOD} = V_{mixed} * (VR4/1024)$

Note: The actual output voltage will be a little degraded due to the DAC1 filter circuit.

- SMOD Gain - 04D5 Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	0						VR5									

Bit15~10 **0**: Zero, must be cleared to zero.

Bit9~0 **VR5**: SMOD Gain Adjustment.

This register is used to adjust the SMOD output pin gain levels.

Refer to the 'Modulation path block diagram' for optimal parameter details.

The VR5 setting is based on the formula: $V_{SMOD} = V_{o2} * VR5/1024$

Note: The actual output voltage will be a little degraded due to the DAC2 filter circuit.

- VOX Threshold High - **04CDh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	VOX Threshold High															

Bit15~0 **VOX Threshold High**: VOX high level threshold.

This register is used to set the VOX high level threshold voltage. The default value is 0x00CC (-15.17dBm@3.3V). When the input signal is larger than the threshold voltage, this means the input signal from microphone is valid. The transmission function will be on. At this time bit0 and bit 1 in the I/O command register address 29 will reply with 0x02. Otherwise these two bits will reply with 0x01.

Refer to the application notes for detailed register setting.

- VOX Threshold Low - **04CEh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	VOX Threshold Low															

Bit15~0 **VOX Threshold Low**: VOX low level threshold.

This register is used to set the VOX low level threshold voltage. The default value is 0x0088 (-17.78dBm@3.3V). To avoid mistaking an input signal for noise and switching off the transmission, the low level threshold is used to setup the turn off threshold. When the input signal is less than the threshold voltage, bit0 and bit 1 in the I/O command register address 29 will reply with 0x01. Otherwise these two bits will reply with 0x02

Refer to the application notes for detailed register setting.

- User CTCSS Tone_1/2 - **04D0h** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	User CTCSS – parameter1															

Bit15~0 User defined CTCSS tone parameter1.

Refer to the application notes for detailed register setting.

- User CTCSS Tone_2/2 - **04D1h** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	User CTCSS – parameter2															

Bit15~0 User defined CTCSS tone parameter2.

Refer to the application notes for detailed register setting.

User Defined CTCSS Tone frequency range: 67 Hz~275 Hz.

Note: If it is necessary to a CTCSS tone by itself, 04d0 and 04d1must be used at the same time.

- CTCSS Frequency Accept Variance - **04D6** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	—													CTCSS Freq.		

Bit15~3 Unimplemented, ignore this value.

Bit2~0 **CTCSS Freq. accept**: CTCSS frequency accept variance.

3’b001: 0.595% ~ 1.975%
3’b010: 0.885% ~ 2.63%
3’b011: 1.195% ~ 3.01%
3’b100: 1.485% ~ 3.385%
3’b101: 1.77% ~ 3.545%
3’b110: 2.05% ~ 3.91%

- User Audio Tone_1/2 - **04D9h** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	User-Tone Freq. – parameter1															

Bit15~0 User Tone Generator frequency parameter1

Refer to the application notes for detailed register setting.

- User Audio Tone_2/2 - **04DAh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	User-Tone Freq. – parameter2															

Bit15~0 User Tone Generator frequency parameter2.

Refer to the application notes for detailed register setting.

Default:1kHz.

Note: If it is necessary to define the user-tone frequency by itself, registers 04d9 and 04da must be used at the same time.

- CTCSS Power Threshold - **04DBh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CTCSS Power Threshold															

Bit15~0 CTCSS Power Threshold.

This register is to set the minimum CTCSS signal level that is to be detected. The default value is 0xFF2B (28mV-peak@3.3V).

When the received signal larger than threshold voltage, bit 3 in the I/O command register address 23 will be set to 1. Otherwise the bit will be cleared to 0.

Refer to the application notes for detailed register setting.

- User DCS Codes_1/2 - **04DCh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	User DCS Codes[15:0]															

Bit15~0 **User DCS parameter[15:0]**: User define DCS parameter bit15~bit0

Refer to the application notes for detailed register setting.

- User DCS Codes_2/2 - **04DDh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	0									User DCS parameter[22:16]						

Bit15~7 **0**: Zero, must be cleared to zero.

Bit6~0 **User DCS parameter[22:16]**: User defined DCS parameter bit22-bit16.

Refer to the application notes for detailed register setting.

Note: If it is necessary to define the DCS tone by itself, registers 04dc and 04dd must be used at the same time. if 4dc and 4de the bits set 1, output is 0, if the bits set 0, output is 1.

- Sub-audio Drop Time - **04DEh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Drop Time Immunity															

Bit15~0 **Drop Time Immunity**: DCS/CTCSS drop out time.

The drop out time is the maximum allowed DCS/CTCSS detection loss duration after the Tx/Rx is connected.

The drop out time = **Drop Time Immunity/4**. The setting range is from 0 to 32767.

Default = b'0000 0100 1011 0000 (300ms).

- MOD Amplitude Limiter - **04DFh** Address

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	0	0	Soft Limiter													

Bit15~14 **0**: Zero, must be fill b'0'.

Bit13~0 **Soft Limiter**: Tx amplitude limiter threshold to VCO.
In Tx mode, this limiter will constrain the MOD output voltage levels.
Refer to the application notes for detailed register setting.

- Selective_Call_0-F

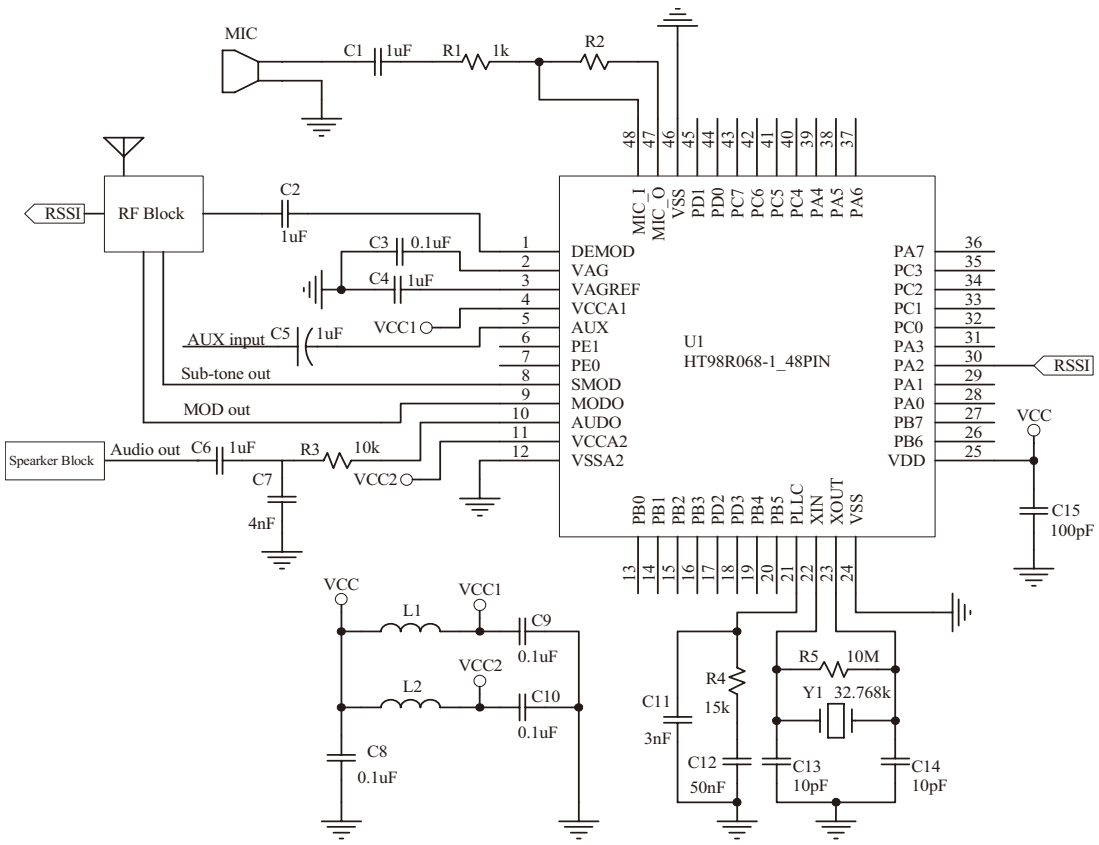
Address / Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
04E0	Selective_Call_0: parameter1															
04E1	Selective_Call_0: parameter2															
04E2	Selective_Call_1: parameter1															
04E2	Selective_Call_1: parameter2															
04E4	Selective_Call_2: parameter1															
04E5	Selective_Call_2: parameter2															
04E6	Selective_Call_3: parameter1															
04E7	Selective_Call_3: parameter2															
04E8	Selective_Call_4: parameter1															
04E9	Selective_Call_4: parameter2															
04EA	Selective_Call_5: parameter1															
04EB	Selective_Call_5: parameter2															
04EC	Selective_Call_6: parameter1															
04ED	Selective_Call_6: parameter2															
04EE	Selective_Call_7: parameter1															
04EF	Selective_Call_7: parameter2															
04F0	Selective_Call_8: parameter1															
04F1	Selective_Call_8: parameter2															
04F2	Selective_Call_9: parameter1															
04F3	Selective_Call_9: parameter2															
04F4	Selective_Call_A: parameter1															
04F5	Selective_Call_A: parameter2															
04F6	Selective_Call_B: parameter1															
04F7	Selective_Call_B: parameter2															
04F8	Selective_Call_C: parameter1															
04F9	Selective_Call_C: parameter2															
04FA	Selective_Call_D: parameter1															
04FB	Selective_Call_D: parameter2															
04FC	Selective_Call_E: parameter1															
04FD	Selective_Call_E: parameter2															
04FE	Selective_Call_F: parameter1															
04FF	Selective_Call_F: parameter2															

The address 04e0-04ff allows the user to define parameters for selective call tones from tone 0 – tone F. The adjustment frequency range is 3.5kHz~0.3kHz. The device follows the EEA protocol (refer to the accompanying table). Every selective call tone set has 2 parts: parameter1 and parameter2, that put two registers in the audio processor. For details regarding register value settings, refer to the application notes.

Tone Number	Frequency (Hz)				
(HEX)	EIA	EEA	CCIR	ZVEI 1	ZVEI 2
0	600	1981	1981	2400	2400
1	741	1124	1124	1060	1060
2	882	1197	1197	1160	1160
3	1023	1275	1275	1270	1270
4	1164	1358	1358	1400	1400
5	1305	1446	1446	1530	1530
6	1446	1540	1540	1670	1670
7	1587	1640	1640	1830	1830
8	1728	1747	1747	2000	2000
9	1869	1860	1860	2200	2200
A	2151	1055	2400	2800	885
B	2435	930	930	810	810
C	2010	2247	2247	970	740
D	2295	991	991	885	680
E	459	2110	2110	2600	970
F	No Tone	2400	1055	680	2600

Selective Call Reference Sets Table

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

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

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

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

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

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

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

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC ″AND″ x

Affected flag(s) Z

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

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

Operation: [m] ← ACC ″AND″ [m]

Affected flag(s): Z

CALL addr Subroutine call

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

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

Affected flag(s): None

CLR [m] Clear Data Memory

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

Operation: [m] ← 00H

Affected flag(s): None

CLR [m].i Clear bit of Data Memory

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

Operation: [m].i ← 0

Affected flag(s): None

CLR WDT Clear Watchdog Timer

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

Operation: WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

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

Operation: WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT	Enter power down mode
Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.
Operation	TO ← 0 PDF ← 1
Affected flag(s)	TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z

INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z

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

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

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

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

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC "OR" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − C

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

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

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

Operation [m] ← ACC − [m] − C

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m] = 0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

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

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

Operation Skip if [m].i = 0

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

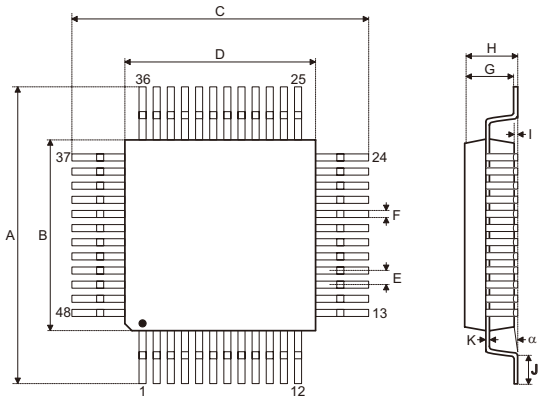
Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website (http://www.holtek.com.tw/english/literature/package.pdf) for the latest version of the package information.

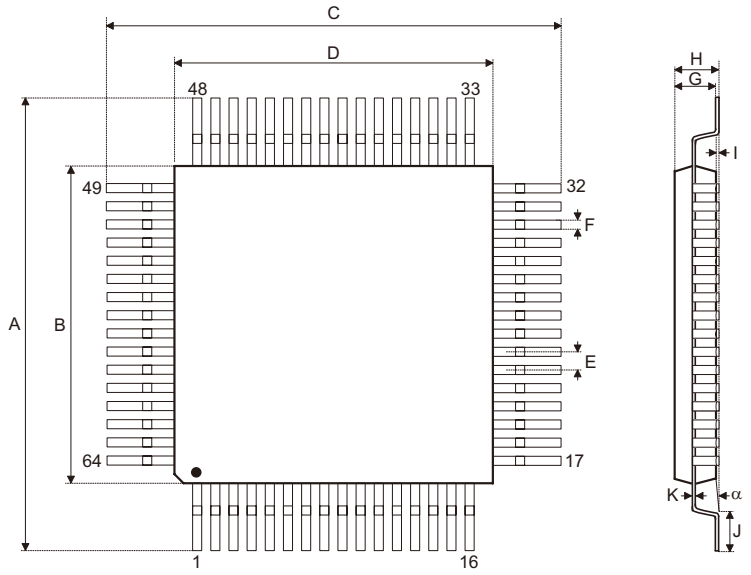
48-pin LQFP (7mmx7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.350	—	0.358
B	0.272	—	0.280
C	0.350	—	0.358
D	0.272	—	0.280
E	—	0.020	—
F	—	0.008	—
G	0.053	—	0.057
H	—	—	0.063
I	—	0.004	—
J	0.018	—	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	8.90	—	9.10
B	6.90	—	7.10
C	8.90	—	9.10
D	6.90	—	7.10
E	—	0.50	—
F	—	0.20	—
G	1.35	—	1.45
H	—	—	1.60
I	—	0.10	—
J	0.45	—	0.75
K	0.10	—	0.20
α	0°	—	7°

64-pin LQFP (7mmx7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.350	—	0.358
B	0.272	—	0.280
C	0.350	—	0.358
D	0.272	—	0.280
E	—	0.016	—
F	0.005	—	0.009
G	0.053	—	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	—	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	8.90	—	9.10
B	6.90	—	7.10
C	8.90	—	9.10
D	6.90	—	7.10
E	—	0.40	—
F	0.13	—	0.23
G	1.35	—	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	—	0.75
K	0.09	—	0.20
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 (China) Inc. (Dongguan Sales Office)
Building No.10, Xinzhu Court, (No.1 Headquarters), 4 Cuizhu Road, Songshan Lake, Dongguan, China 523808
Tel: 86-769-2626-1300
Fax: 86-769-2626-1311, 86-769-2626-1322

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