HOLTEK

Two Way Radio Flash MCU

HT98F069

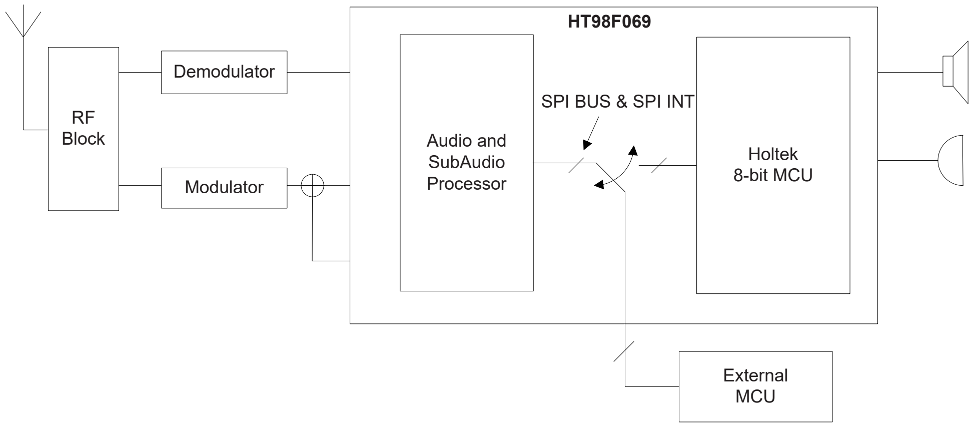
Revision: V1.20   Date: November 26, 2019

Table of Contents

Features

Audio Processor Features

- Audio Processor working frequency up to 24.576MHz
- Operating voltage: 3.3V~5.5V
- PGA: 5-bit Operational Amplifier Gain setup
- Programmable Audio Scrambler
- Sub-tone processor
  - User-defined CTCSS/DCS encoder/decoder
- In-band-tone processor
  - DTMF encoder/decoder
  - Selective call encoder/decoder
  - User-defined tone encoder/decoder
- Audio-band Processing
  - Pre-emphasis/de-emphasis
  - Scrambler
  - Compandor
  - VOX
- SPI Interface for external MCU control
- TX Output for Single-Point Modulation



Note: This integrated Audio Processor can be controlled by either the internal MCU or an external MCU.

Audio Processor

CPU Features

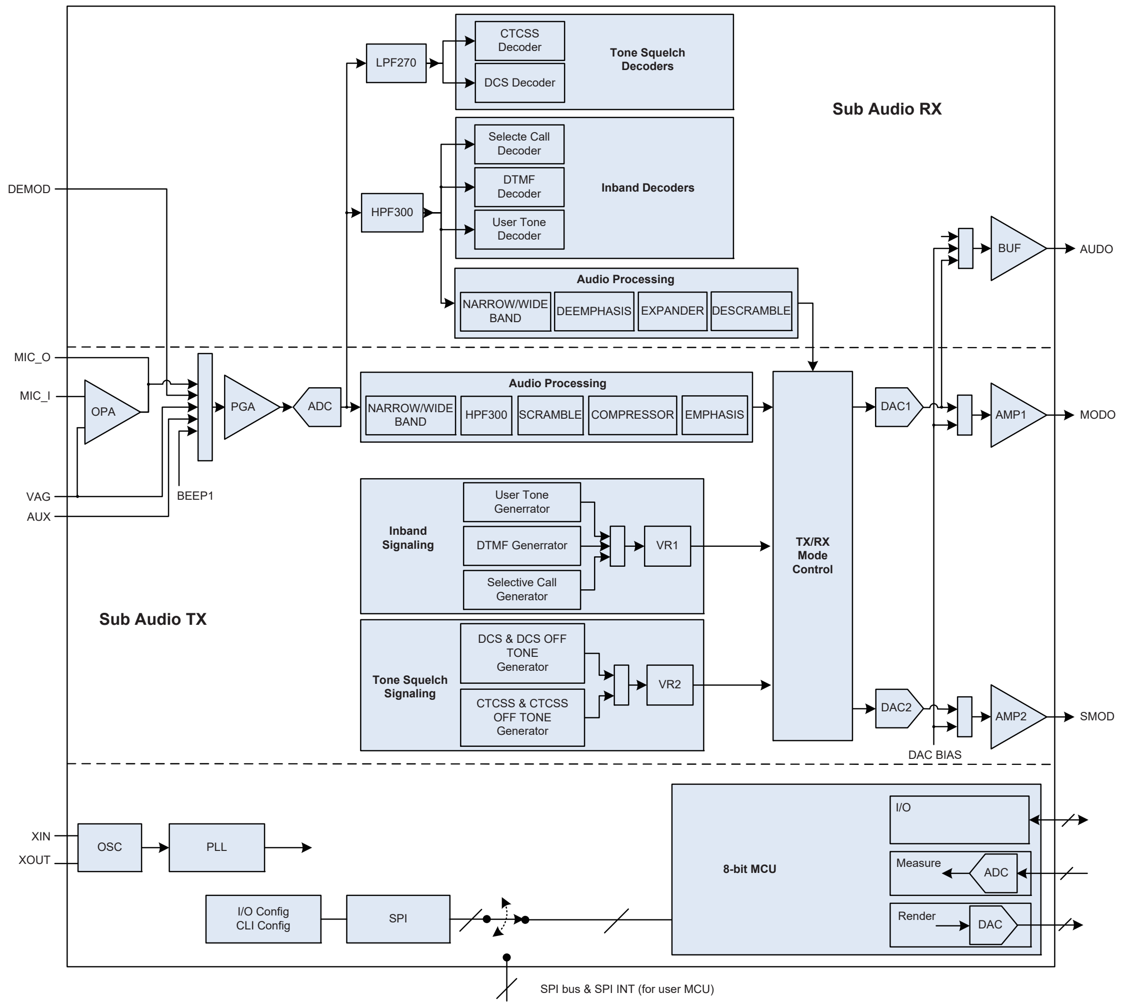
- Operating voltage:
  - f_{SYS}=32768Hz: 2.2V~5.5V
  - f_{SYS}=4.096MHz: 2.2V~5.5V
  - f_{SYS}=8.192MHz: 2.2V~5.5V
  - f_{SYS}=12.288MHz: 3.3V~5.5V
  - f_{SYS}=16.384MHz: 3.3V~5.5V
- Up to 0.24μs instruction cycle with 16.384MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Multi-mode operation: Normal, Slow, Idle and Sleep
- Oscillator types:
  - External 32768Hz low speed crystal – LXT
  - Internal 32kHz low speed RC – LIRC
- Internal PLL to generate the system clock
  - Input reference clock: 32768Hz
  - Output: 2.048MHz ×12/×16/×20/×24 (optional)
- All instructions executed in one or two instruction cycles
- Table read instruction
- 63 powerful instructions
- 10-level subroutine nesting
- Bit manipulation instruction
- Flash Program Memory: 24K×16
- RAM Data Memory: 1152×8
- Watchdog Timer function
- Up to 42 bidirectional I/O lines
- One external interrupt pin shared with I/O pin
- 8-channel 12-bit A/D Converter
- 4-channel 8-bit D/A Converter
- Two 8-bit and one 16-bit programmable Timer/Event Counters with overflow interrupt and prescaler
- Programmable Frequency Divider – PFD
- Single Time-Base function for generation of fixed time interrupt signals
- Low voltage reset/detect function
- Flash program memory can be re-programmed up to 10,000 times
- Flash program memory data retention > 10 years
- Package types: 48-pin LQFP, 64-pin LQFP

General Description

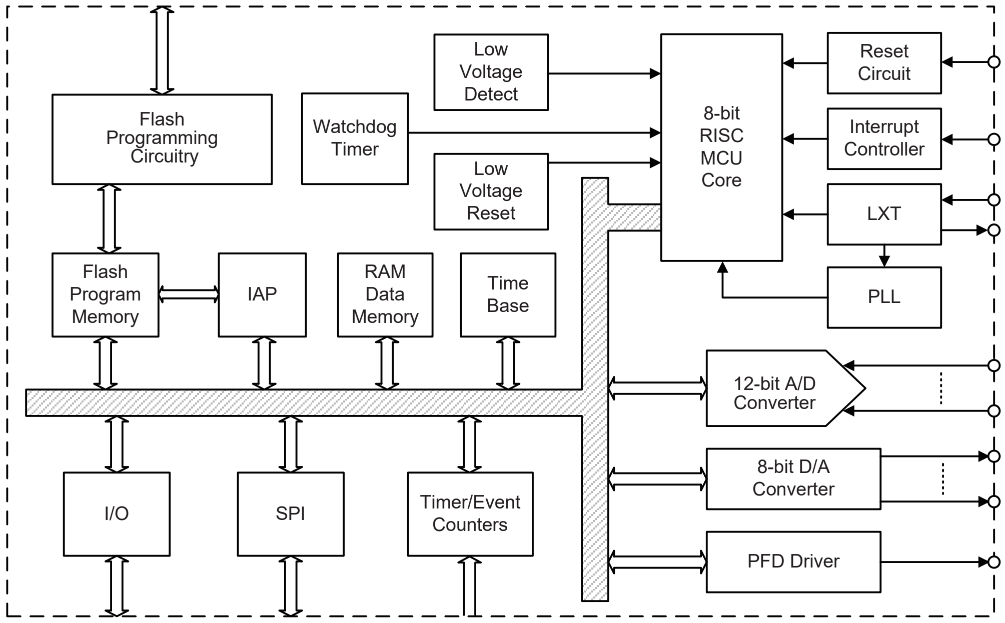
The HT98F069 device is a Flash ASSP MCU for analog two way radio applications such as FRS. It contains an 8-bit MCU, an Audio Processor, a 16-bit delta-sigma A/D converter, two 16-bit D/A converters and an operational amplifier for peripheral interfaces. It includes an 8-channel 12-bit A/D converter, a 4-channel 8-bit D/A converter, an LXT oscillator function, an external interrupt, 24K words of Flash Program Memory and 1152 bytes of SRAM Data Memory for general MCU control application. The integrated Audio Processor can be accessed by the MCU using its internal SPI interface. The Internal Audio Processor supports various programmable functions. These functions include a CTCSS/DCS encoder/decoder, DTMF encoder/decoder, scramble/descramble, VOX and compandor, etc. By connecting to a suitable RF module, this device provides a cost effective and extremely flexible FRS solution. Applications will include products in the leisure radio application area such as general mobile radios, personal mobile radios and multiple user radios.

Block Diagram

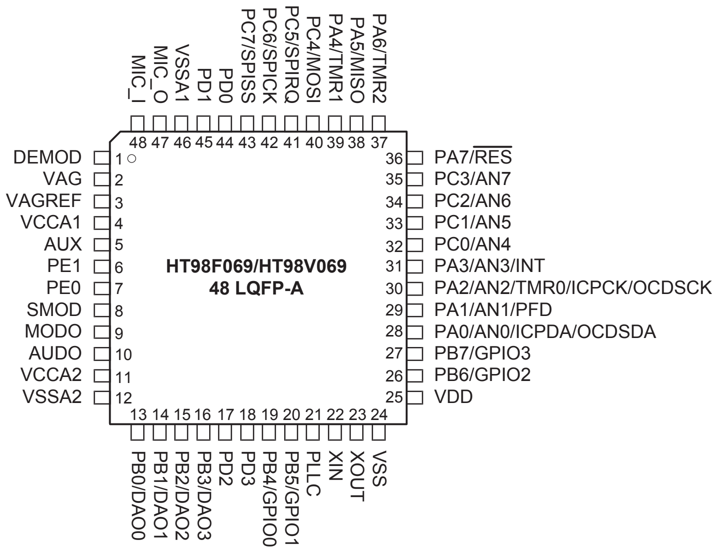
Overall Block Diagram

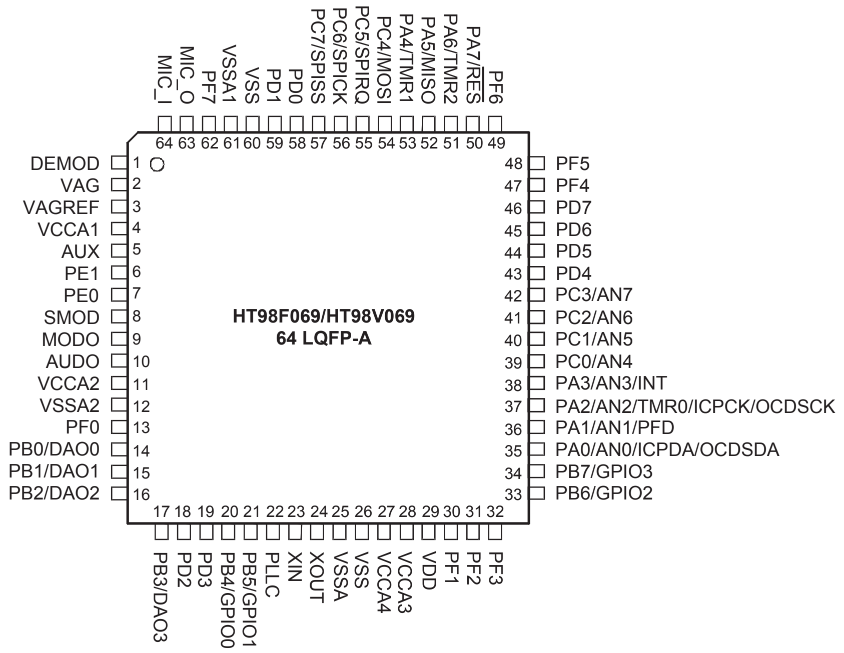


Internal 8-Bit MCU Block Diagram



Pin Assignment





Note: 1. If the pin-shared pin functions have multiple outputs simultaneously, its pin names at the right side of the "/" sign can be used for higher priority.
2. The INT, TMR0 and PFG pin functions can be remapped to other pins controlled by the same control bit. Refer to the CTRL0 register in the System Control Registers section.

Pin Description

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

Pad Name	Function	OPT	I/T	O/T	Description
PA0/AN0/ICPDA/OCDSDA	PA0	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN0	ANCSR	AN	—	A/D converter analog input 0
	ICPDA	—	—	—	ICP Serial Data/Address pin
	OCDSDA	—	—	—	OCDS Data/Address pin, for EV chip only.
PA1/AN1/PFD	PA1	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN1	ANCSR	AN	—	A/D converter analog input 1
	PFD	CTRL0	—	CMOS	PFD output
PA2/AN2/TMR0/ICPCK/OCDSCK	PA2	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	AN2	ANCSR	AN	—	A/D converter analog input 2
	TMR0	CTRL0 TMR0C	ST	—	Timer/Event Counter 0 clock input
	ICPCK	—	—	—	ICP Clock pin
	OCDSCK	—	—	—	OCDS Clock pin, for EV chip only.
PA3/AN3/INT	PA3	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN3	ANCSR	AN	—	A/D converter analog input 3
	INT	CTRL0 CTRL1 INTC0	ST	—	External interrupt input
PA4/TMR1	PA4	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TMR1	TMR1C	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	—	—	Slave SPI output data pin
PA6/TMR2	PA6	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TMR2	TMR2C	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	—	External reset pin
PB0/DAO0	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	DAO0	DACR	—	AN	DAC0 output
PB1/DAO1	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	DAO1	DACR	—	AN	DAC1 output
PB2/DAO2	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	DAO2	DACR	—	AN	DAC2 output
PB3/DAO3	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	DAO3	DACR	—	AN	DAC3 output
PB4/GPIO0	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	GPIO0	SPICR	—	—	Audio processor I/O 0
PB5/GPIO1	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	GPIO1	SPICR	—	—	Audio Processor I/O 1
PB6/GPIO2	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	GPIO2	SPICR	—	—	Audio Processor I/O 2

Pad Name	Function	OPT	I/T	O/T	Description
PB7/GPIO3	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	GPIO3	SPICR	—	—	Audio Processor I/O 3
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 converter analog input 4, 5, 6, 7
PC4/MOSI	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	MOSI	SPICR	—	—	Slave SPI input data pin
PC5/SPIRQ	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SPIRQ	SPICR	—	—	Slave SPI output pin
PC6/SPICK	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SPICK	SPICR	—	—	Slave SPI clock source input pin
PC7/SPISS	PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SPISS	SPICR	—	—	Audio Processor select pin
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.
PF0~PF7	PFn	PFPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
XIN	XIN	—	LXT	—	Low frequency crystal input pin
XOUT	XOUT	—	—	LXT	Low frequency crystal output pin
MIC_I	MIC_I	—	—	—	Microphone OP input
MIC_O	MIC_O	—	—	—	Microphone OP output
AUX	AUX	—	—	—	Auxiliary audio input
AUDO	AUDO	—	—	—	Audio output
MODO	MODO	—	—	—	Audio baseband modulation output to RF module
SMOD	SMOD	—	—	—	Sub-tone baseband modulation output
DEMOD	DEMOD	—	—	—	PGA Input from RF demodulation output
VCCA1	VCCA1	—	—	—	Analog Block VCC
VSSA1	VSSA1	—	—	—	Analog Block GND
VCCA2	VCCA2	—	—	—	Analog Block VCC
VSSA2	VSSA2	—	—	—	Analog Block GND
VAG	VAG	—	—	—	Audio ADC analog ground output
VAGREF	VAGREF	—	—	—	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

Legend: I/T: Input type; O/T: Output type;
OPT: Optional by configuration option (CO) or register option;
PWR: Power; CO: Configuration option
ST: Schmitt Trigger input; AN: Analog signal;
CMOS: CMOS output; NMOS: NMOS output
LXT: Low frequency crystal oscillator

Absolute Maximum Ratings

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

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	MCU Operating Voltage	—	f_{SYS}=32768Hz	2.2	—	5.5	V
		—	f_{SYS}=4.096MHz	2.2	—	5.5	
		—	f_{SYS}=8.192MHz	2.2	—	5.5	
		—	f_{SYS}=12.288MHz	3.3	—	5.5	
		—	f_{SYS}=16.384MHz	3.3	—	5.5	
I_{DD}	Operating Current (LXT+PLL)	3V	No load, f_{SYS}=4.096MHz ADC disable, DAC disable, Audio Processor off	—	2	4	mA
		5V		—	4	8	
	Operating Current (LXT+PLL)	3V	No load, f_{SYS}=8.192MHz ADC disable, DAC disable, Audio Processor off	—	2	4	mA
		5V		—	4	8	
	Operating Current (LXT+PLL)	3.3V	No load, f_{SYS}=12.288MHz ADC disable, DAC disable, Audio Processor off	—	2	4	mA
		5V		—	4	8	
	Operating Current (LXT+PLL)	3.3V	No load, f_{SYS}=16.384MHz ADC disable, DAC disable, Audio Processor off	—	4	12	mA
		5V		—	6	16	
	Operating Current (LXT on, PLL off)	3V	No load, f_{SYS}=32768Hz ADC disable, DAC disable, Audio Processor off	—	20	30	μA
		5V		—	40	60	
	Operating Current (Audio Processor on)	3.3V	No load, f_{AP}=24.567MHz (Note2)	—	25	—	mA
I_{STB}	Standby Current (LIRC off, LXT off, PLL off: PLLEN=0&CLKMOD=1)	3V	No load, MCU powered down	—	—	1	μA
		5V		—	—	2	
	Standby Current (LIRC on, LXT off, PLL off: PLLEN=0&CLKMOD=1)	3V	No load, MCU powered down	—	—	5	μA
		5V		—	—	10	
	Standby Current (LIRC off, LXT on, PLL off: PLLEN=0&CLKMOD=1)	3V	No load, MCU powered down, LXT slowly start-up	—	—	5	μA
		5V		—	—	10	
V_{IL}	Input Low Voltage for I/O Ports	—	—	0	—	$0.3V_{DD}$	V
	Input Low Voltage for $\overline{RES}$ pin	—	—	0	—	$0.4V_{DD}$	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IH}	Input High Voltage for I/O Ports	—	—	$0.7V_{DD}$	—	V_{DD}	V
	Input High Voltage for $\overline{RES}$ pin	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OH}	I/O Source Current	3V	V_{OH}=$0.9V_{DD}$	-2	-4	—	mA
		5V		-5	-10	—	
I_{OL}	I/O Sink Current	3V	V_{OL}=$0.1V_{DD}$	4	8	—	mA
		5V		10	20	—	
R_{PH}	Pull-high Resistance for I/O Ports	3V	—	20	60	100	kΩ
		5V		10	30	50	
V_{DEMOD}	DEMOD Input Voltage Range	3.3V	PGA Gain=0dB	—	—	±1100	mV
V_{MIC}	MIC Input Voltage Range	3.3V	PGA Gain=0dB	—	—	±1100	mV
V_{AUX}	AUX Input Voltage Range	3.3V	PGA Gain=0dB	—	—	±1100	mV
V_{AUDO}	AUDO Output Voltage Range	3V	No load	0.01	—	0.99	V_{DD}
V_{MODO}	MODO 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}
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_{MODO_ON}	MODO Output Resistance When DAO1 OPA Turn On	—	—	—	300	—	Ω
R_{MODO_OFF}	MODO Output Resistance When DAO1 OPA Turn Off	—	—	—	500	—	kΩ
R_{SMOD_ON}	SMOD Output Resistance When DAO2 OPA Turn On	—	—	—	300	—	Ω
R_{SMOD_OFF}	SMOD Output Resistance When DAO2 OPA Turn Off	—	—	—	500	—	kΩ
R_L	Load Resistance for MODO and SMOD	—	—	20	—	—	kΩ
V_{AG}	Audio ADC analog ground output	—	—	-5%	$V_{DD}/2$	+5%	V

Note: 1. Each of the PA0~PA6 pins is connected to an external pull-up resistor when the $\overline{RES}$ pin is low.
2. The ADC and DAC in the MCU, Audio Scrambler, CTCSS, DCS, compandor and pre-emphasis/de-emphasis are disabled, while all the other digital circuits including the main clock PLL are enabled. A single analog path is enabled through the device.

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{PLL}	PLL Clock	3.3V~5.5V	M[1:0]=00	-1%	24.576	+1%	MHz
			M[1:0]=01		32.768		
			M[1:0]=10		40.960		
			M[1:0]=11		49.152		
f_{SYS}	System Clock (LXT)	—	—	—	32768	—	Hz
f_{TIMER}	Timer Input Frequency (TMRn)	2.2V~5.5V	—	0	—	4	MHz
		3.3V~5.5V	—	0	—	8	
		4.5V~5.5V	—	0	—	12	
f_{LIRC}	Low Speed RC Oscillator Clock	5V	Ta=25°C	-10%	32	+10%	kHz
		5V±0.5V	Ta= -40°C~85°C	-40%	32	+40%	
		2.2V~5.5V	Ta= -40°C~85°C	-50%	32	+60%	
t_{RES}	MCU External Reset Low Pulse Width	—	—	10	—	—	μs
	Audio Processor Reset (AUPRST) Low Pulse Width	—	—	1	—	—	μs
t_{SST1}	MCU System Start-up Time Period (Power On Reset or Wake-up from Power Down Mode)	—	MCU f_{SYS}=32768Hz	-10%	1024	—	t_{SYS}
t_{SST2}	Audio Processor System Start-up Time Period (Wake-up from Reset)	—	—	100	—	—	ms
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{FUP}	PLL Settling Time (32768Hz to 24.576/32.768/40.96/49.152MHz, PLL Frequency Deviation < ±0.1%)	3.3V	LXT on, PLL off→on	—	—	10	ms
		5V					
t_{RSTD}	System Reset Delay Time (Power On Reset, LVR Hardware Reset, LVRC/WDTC Software Reset)	—	—	25	50	100	ms
	System Reset Delay Time ($\overline{RES}$ pin reset, WDT hardware reset, IAP reset)	—	—	8.3	16.7	33.3	ms

Note: 1. $t_{SYS}=1/f_{SYS}$
2. M[1:0] are PLLCR register bit 1 and bit 0.

LVR/LVD Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	-5%	2.1	+5%	V
		—	LVR enable, voltage select 2.55V		2.55		
		—	LVR enable, voltage select 3.15V		3.15		
		—	LVR enable, voltage select 3.8V		3.8		
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 2.0V	-5%	2.0	+5%	V
		—	LVD enable, voltage select 2.2V		2.2		
		—	LVD enable, voltage select 2.4V		2.4		
		—	LVD enable, voltage select 2.7V		2.7		
		—	LVD enable, voltage select 3.0V		3.0		
		—	LVD enable, voltage select 3.3V		3.3		
		—	LVD enable, voltage select 3.6V		3.6		
		—	LVD enable, voltage select 4.0V		4.0		
t_{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVDS}	LVDO Stable Time	—	LVR enable, LVD off → on	—	—	15	μs
		—	LVR disable, LVD off → on	—	—	150	μs

A/D Converter Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
AV_{DD}	Operating Voltage	—	$V_{REF}=AV_{DD}$	2.7	5.0	5.5	V
V_{AD}	A/D Converter Input Voltage	—	$V_{REF}=AV_{DD}$	0	—	AV_{DD}	V
DNL	A/D Differential Non-linearity	3V	$V_{REF}=AV_{DD}=V_{DD}$ $t_{AD}=0.5\mu s$	-3	—	+3	LSB
		5V					
		3V	$V_{REF}=AV_{DD}=V_{DD}$ $t_{AD}=10\mu s$				
		5V					
INL	A/D Integral Non-linearity	3V	$V_{REF}=AV_{DD}=V_{DD}$ $t_{AD}=0.5\mu s$	-4	—	+4	LSB
		5V					
		3V	$V_{REF}=AV_{DD}=V_{DD}$ $t_{AD}=10\mu s$				
		5V					
I_{ADC}	Operating Current (only A/D enable, others disable)	3V	No load ($t_{AD}=0.5\mu s$)	—	1.0	2.0	mA
		5V		—	1.5	3.0	
t_{AD}	A/D Converter Clock Period	2.7V~5.5V	—	0.5	—	10	μs
t_{ADC}	A/D Conversion Time (Include Sample and Hold Time)	2.7V~5.5V	—	—	16	—	t_{AD}
t_{ADS}	A/D Sampling Time	2.7V~5.5V	—	—	4	—	t_{AD}
t_{ON2ST}	A/D Converter On-to-Start Time	2.7V~5.5V	—	2	—	—	μs

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

D/A Converter Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
AV_{DD}	Operating Voltage	—	—	2.7	5.0	5.5	V
DNL	D/A Differential Non-linearity	5V	$V_{REF}=AV_{DD}=V_{DD}$	-0.5	—	+0.5	LSB
INL	D/A Integral non-linearity	5V	$V_{REF}=AV_{DD}=V_{DD}$	-1	—	+1	LSB
I_{DAC}	Operating Current	3V	No load, $V_{REF}=AV_{DD}=V_{DD}$	—	300	—	μA
		5V		—	300	—	
R_O	R2R Output Resistance	5V	—	—	10	—	kΩ

Audio Processor Characteristics

Operating Temperature: -40°C~85°C, Ta=25°C, V_{DD}=3.3V
Input stage gain=0dB. Output stage attenuation=0dB
LXT frequency=32768Hz ± 0.01% (100ppm), f_{SYS} drift < ±0.1%
Reference signal level is 300mVrms at 1kHz with V_{DD}=3.3V

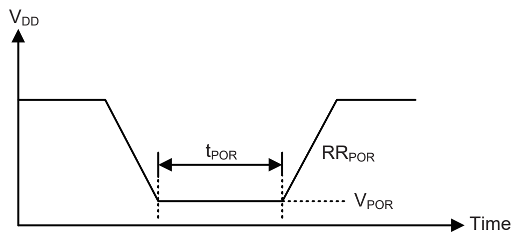
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
CTSS Receiver							
	Sensitivity (Pure Tone)	3.3V	—	—	20	—	mV_{P-P}
	Response Time	3.3V	—	—	150	—	ms
	De-response Time	3.3V	—	—	DT1+154	—	ms
DT1	Dropout Immunity	3.3V	—	0	300	8192	ms
	Frequency Range	3.3V	—	60	—	260	Hz
DCS Receiver							
	Sensitivity	3.3V	—	60	—	—	mV_{P-P}
	Response Time	3.3V	—	—	150	—	ms
	De-response Time	3.3V	—	—	DT2+120	—	ms
DT2	Dropout Immunity	3.3V	—	0	—	8192	ms
Selective Call / User Tone & Audio Tone Receiver							
	Sensitivity (Pure Tone)	3.3V	—	—	16	—	mVrms
	Response Time	3.3V	—	—	32	—	ms
	De-response Time	3.3V	—	—	30	—	ms
	Dropout Immunity	3.3V	—	—	25	—	ms
	Frequency Range	3.3V	—	400	—	3000	Hz
DTMF Receiver							
	Sensitivity	3.3V	—	—	25	—	mVrms
	Response Time	3.3V	—	—	35	—	ms
	De-response Time	3.3V	—	—	35	—	ms
	Frequency Tolerance	3.3V	—	—	±2.5	—	%
	Frequency Rejection	3.3V	—	—	±3.5	—	%
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	—	—	2:1	—	
CTCSS Generator							
	Frequency Range	3.3V	—	60	—	260	Hz
	Frequency Accuracy	3.3V	—	—	—	±0.3	%
	Amplitude Tolerance	—	—	-1.0	0	+1.0	dB
	Harmonic Distortion	—	—	—	2.0	4.0	%
Audio Channel Filter							
	Received Audio HPF	3.3V	—	300	—	3400	Hz
	12.5kHz Channel Transmitted Audio	3.3V	—	300	—	2550	Hz
	25kHz Channel Transmitted Audio	3.3V	—	300	—	3000	Hz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
	Pass-band Gain	3.3V	Frequency=1.0kHz	—	0	—	dB
	Pass-band Ripple	3.3V	Frequency=1.0kHz	-2.0	0	+0.5	dB
	Stop-band Attenuation	3.3V	For 12.5kHz channel, stop band at 3.4kHz; For 25kHz channel, stop band at 3.7kHz;	30	—	—	dB
	Residual Hum and Noise	3.3V	—	—	-50	—	dB
	De-emphasis	3.3V	—	—	+6	—	dB/oc
	Pre-emphasis	3.3V	—	—	-6	—	dB/oc
Audio Scrambler							
	Inversion Frequency	3.3V	—	—	3300	—	Hz
	Pass Band	3.3V	—	300	—	3000	Hz
Audio Expandor							
	Input Signal Range	3.3V	—	—	300	—	mVrms

Power-on Reset Characteristics

Ta=25°C

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



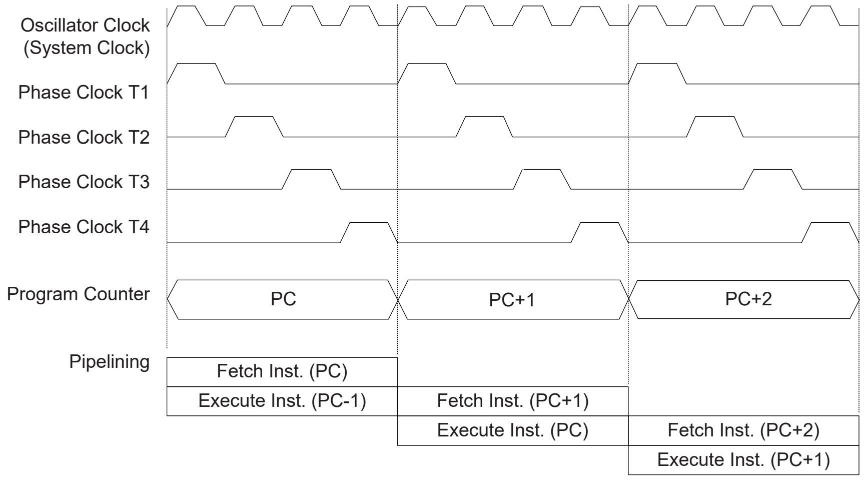
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to the internal system architecture. The range of device take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions. An 8-bit wide ALU is used in practically all operations of the instruction set. It 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 system with maximum reliability and flexibility.

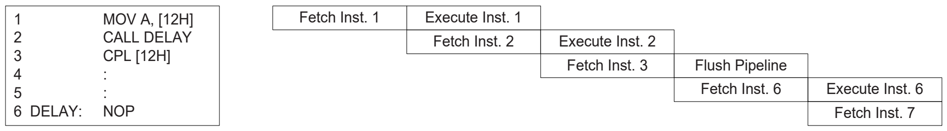
Clocking and Pipelining

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

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



System Clocking and Pipelining



Instruction Fetching

Program Counter – PC

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demand a jump to a non-consecutive Program Memory address. As the device memory capacity is greater than 8K words, the Program Memory address may be located in a certain program memory bank which is selected by the program memory bank pointer bits, PMBP1~PMBP0. It must be noted that only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by user.

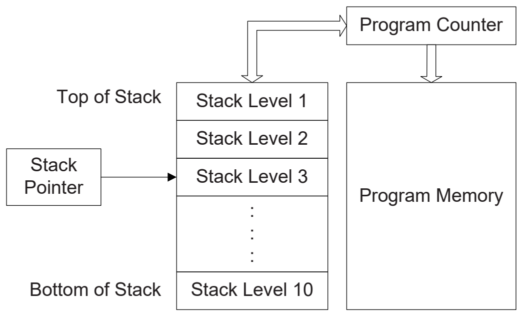
When executing instructions requiring jumping to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc, the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition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

Program Counter	
High Byte of Program	**Low Byte of Program**
PMBP1~PMBP0, PC12~PC8	PCL7~PCL0

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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The device 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.

Arithmetic and Logic Unit – ALU

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

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

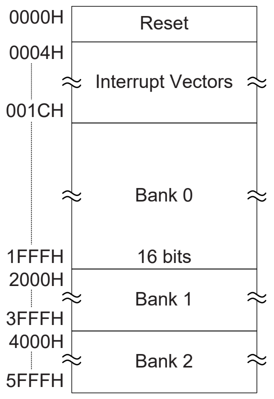
Flash Program Memory

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

Structure

The Program Memory has a capacity of 24K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries information. Table data, which can be set in any location within the Program Memory, is addressed by a separate table pointer register.

The device has its Program Memory divided into three Banks, Bank 0, Bank 1 and Bank 2. The required Bank is selected using the Bit 6~5 bits of the BP register.



Program Memory Structure

Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 0000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be set 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 "TABRD[m]" or "TABRDL[m]" instructions, respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as "0".

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

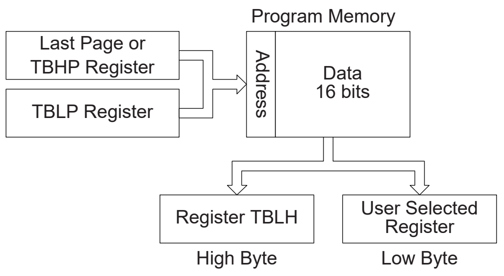


Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "1F00H" which is located in ROM Bank 2 and refers to the start address of the last page within the 24K words Program Memory of the device.

The table pointer low byte register is set here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "5F06H" 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 specific page pointed by the TBHP register if the "TABRD[m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRDL [m]" instruction is executed.

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

Table Read Program Example

```
rombank 2 code2
ds .section ‘data’
tempreg1 db?       ; temporary register #1
tempreg2 db?       ; temporary register #2
code0 .section 'code'
mov a,06h          ; initialise table pointer - note that this address is referenced
mov tblp,a         ; to the last page or the page that TBHP pointed
mov a,05fh         ; initialise high table pointer
mov tbhp,a         ; it is not necessary to set TBHP if executing tabrdl
:
:
tabrd tempreg1     ; transfers value in table referenced by table pointer
                   ; register pair to tempreg1
tabrdl tempreg1    ; data at program memory address "5F06H" transferred to
                   ; tempreg1 and TBLH
dec tblp           ; reduce value of table pointer by one
tabrd tempreg2     ; transfers value in table referenced by table pointer
                   ; register pair to tempreg2
tabrdl tempreg2    ; data at program memory address "5F05H" transferred to
                   ; tempreg2 and TBLH
                   ; in this example the data "1AH" is transferred to
                   ; tempreg1 and data "0FH" to tempreg2
                   ; the value "00H" will be transferred to the high byte register TBLH
:
:
code2 .section 'code'
org 1F00h          ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

In Circuit Programming – ICP

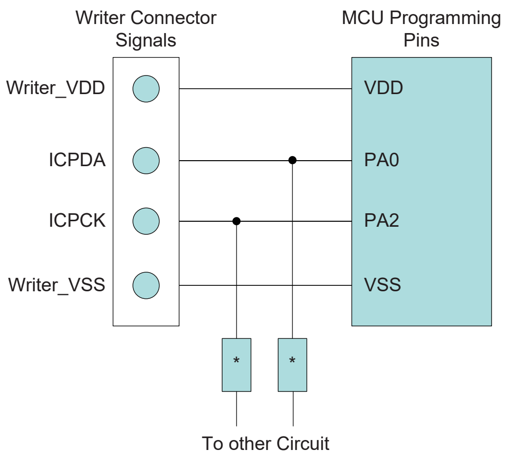
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

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

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

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



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

On-Chip Debug Support – OCDS

There is an EV chip named HT98V069 which is used to emulate the HT98F069 device. The EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

In Application Programming – IAP

The device offers IAP function to update data or application program to Flash ROM. Users can define any ROM location for IAP, but there are some features which user must notice in using IAP function.

- Erase Page: 64 words/page
- Writing Word: 64 words/time
- Reading Word: 1 word/time

In Application Programming Registers

The Address registers, FARL and FARH, and the Data registers, FD0L/FD0H, FD1L/FD1H, FD2L/FD2H, FD3L/FD3H, together with the Control registers, FC0, FC1 and FC2, located in Data Memory Bank 1, are the corresponding Flash access registers for IAP. As indirect addressing is the only way to access all these registers, all read and write operations to the registers must be performed using the Indirect Addressing Register, IAR1, and the Memory Pointer, MP1. Because the data, address and control registers are located at the address of 31H~3DH in Data Memory Bank 1, the desired value ranged from 31H~3DH must be written into the MP1 Memory Pointer and the value "01H" must also be written into the Bank Pointer, BP.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	—	CLWB
FARL	A7	A6	A5	A4	A3	A2	A1	A0
FARH	—	A14	A13	A12	A11	A10	A9	A8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Registers List

• **FC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CFWEN**: Flash Memory Write enable control
0: Flash Memory write function is disabled
1: Flash Memory write function has been successfully enabled

When this bit is cleared to 0 by application program, the Flash Memory write function is disabled. Note that writing a "1" into this bit results in no action. This bit is used to indicate that the Flash Memory write function status. When this bit is set to 1 by hardware, it means that the Flash Memory write function is enabled successfully. Otherwise, the Flash Memory write function is disabled as the bit content is zero.

Bit 6~4 **FMOD2~FMOD0**: Mode selection
000: Write Program Memory
001: Page erase Program Memory
010: Reserved
011: Read Program Memory
100: Reserved
101: Reserved
110: FWEN mode – Flash Memory write function enable mode
111: Reserved

Bit 3 **FWPEN**: Flash Memory Write procedure enable control
0: Disable
1: Enable
When this bit is set to 1 and the FMOD field is set to "110", the IAP controller will execute the "Flash memory write function enable" procedure. Once the Flash memory write function is successfully enabled, it is not necessary to set the FWPEN bit any more.

Bit 2 **FWT**: Flash ROM write control bit
0: Do not initiate Flash Memory write or Flash Memory write process is completed
1: Initiate Flash Memory write process
This bit can be set by software only, when the write process is completed, hardware will clear the FWT bit.

Bit 1 **FRDEN**: Flash Memory read enabled bit
0: Flash Memory read disable
1: Flash Memory read enable

Bit 0 **FRD**: Flash Memory read control bit
0: Do not initiate Flash Memory read or Flash Memory read process is completed
1: Initiate Flash Memory read process
This bit can be set by software only, when the read process is completed, hardware will clear the FRD bit.

- **FC1 Register**

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

Bit 7~0 **D7~D0**: whole chip reset
When user writes 55H to this register, it will generate a reset signal to reset whole chip.

- **FC2 Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **CLWB**: Flash Memory Write buffer clear control
0: Do not initiate Write Buffer Clear or Write Buffer Clear process is completed
1: Initiate Write Buffer Clear process
This bit can be set by software only, when the clear write buffer process is completed, hardware will clear the CLWB bit.

- **FARL Register**

Bit	7	6	5	4	3	2	1	0
Name	A7	A6	A5	A4	A3	A2	A1	A0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 Flash Memory Address [7:0]

- **FARH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	A14	A13	A12	A11	A10	A9	A8
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 6~0	Flash Memory Address [14:8]

- **FD0L Register**

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

Bit 7~0 The first Flash Memory data [7:0]

- **FD0H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 The first Flash Memory data [15:8]

- **FD1L Register**

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

Bit 7~0 The second Flash Memory data [7:0]

- **FD1H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 The second Flash Memory data [15:8]

- **FD2L Register**

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

Bit 7~0 The third Flash Memory data [7:0]

- **FD2H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 The third Flash Memory data [15:8]

- **FD3L Register**

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

Bit 7~0 The fourth Flash Memory data [7:0]

- **FD3H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 The fourth Flash Memory data [15:8]

Flash Memory Write Function Enable Procedure

In order to allow users to change the Flash Memory data through the IAP control registers, users must first enable the Flash Memory write operation by the following procedure:

Step 1. Write "110" into the FMOD2~FMOD0 bits to select the FWEN mode.

Step 2. Set the FWPEN bit to "1". The step 1 and step 2 can be executed simultaneously.

Step 3. The pattern data with a sequence of 00H, 04H, 0DH, 09H, C3H and 40H must be written into the FD1L, FD1H, FD2L, FD2H, FD3L and FD3H registers respectively.

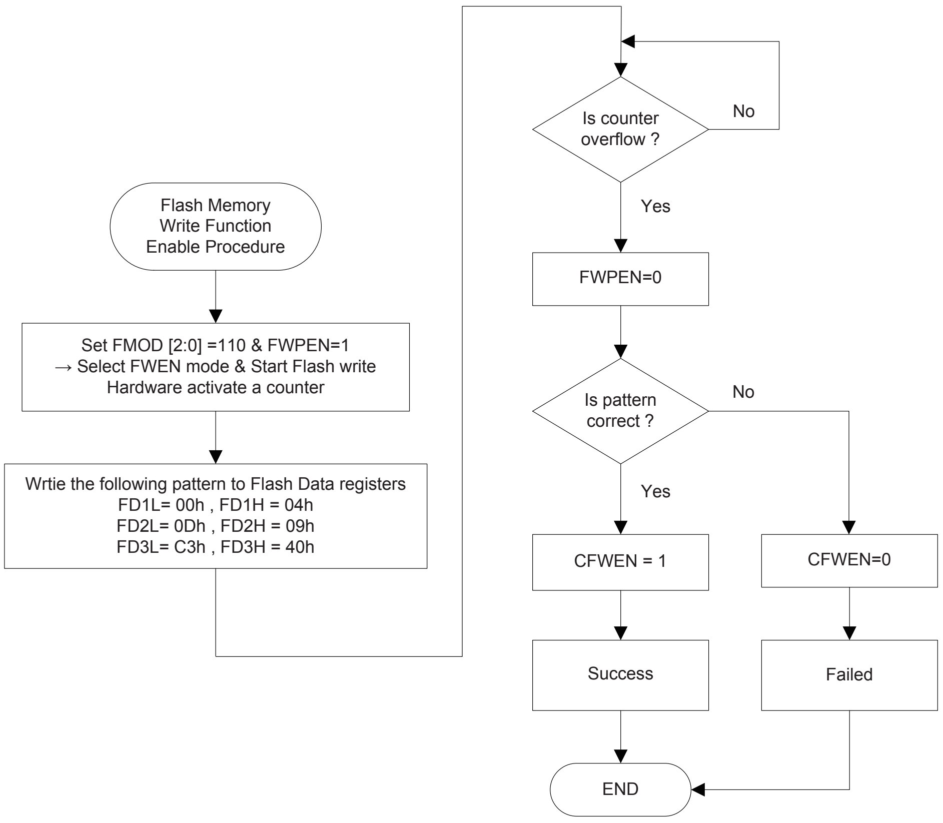
Step 4. A counter with a time-out period of 300μs will be activated to allow users writing the correct pattern data into the FD1L/FD1H~FD3L/FD3H register pairs. The counter clock is derived from the LIRC oscillator.

Step 5. If the counter overflows or the pattern data is incorrect, the Flash Memory write operation will not be enabled and users must again repeat the above procedure. Then the FWPEN bit will automatically be cleared to 0 by hardware.

Step 6. If the pattern data is correct before the counter overflows, the Flash Memory write operation will be enabled and the FWPEN bit will automatically be cleared to 0 by hardware. The CFWEN bit will also be set to 1 by hardware to indicate that the Flash Memory write operation is successfully enabled.

Step 7. Once the Flash Memory write operation is enabled, the user can change the Flash ROM data through the Flash control register.

Step 8. To disable the Flash Memory write operation, the user can clear the CFWEN bit to 0.


Flash Memory Write Function Enable Procedure
Set FMOD [2:0] =110 & FWPEN=1 → Select FWEN mode & Start Flash write Hardware activate a counter
Wrtie the following pattern to Flash Data registers FD1L= 00h , FD1H = 04h FD2L= 0Dh , FD2H = 09h FD3L= C3h , FD3H = 40h
Is counter overflow ?
No
Yes
FWPEN=0
Is pattern correct ?
No
Yes
CFWEN = 1
CFWEN=0
Success
Failed
END

Flash Memory Write Function Enable Procedure

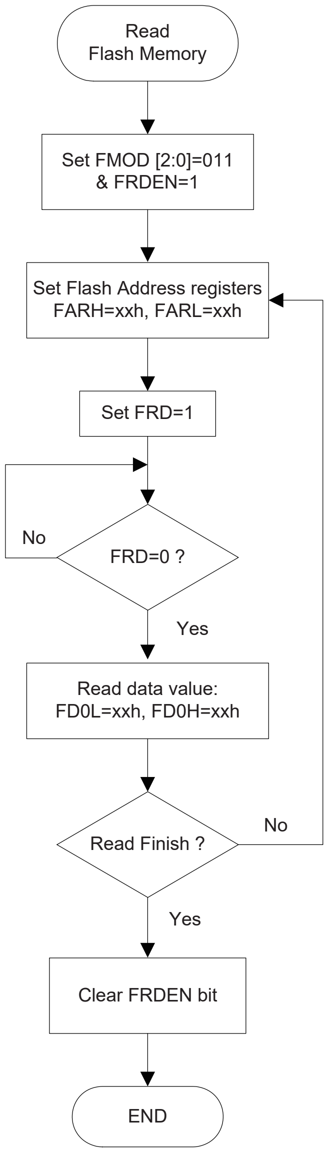
Flash Memory Read/Write Procedure

After the Flash Memory write function is successfully enabled through the preceding IAP procedure, users must first erase the corresponding Flash Memory page and then initiate the Flash Memory write operation. For the device the number of the page erase operation is 64 words per page, the available page erase address is only specified by FARH register and the content of the FARL[7:6] bit field.

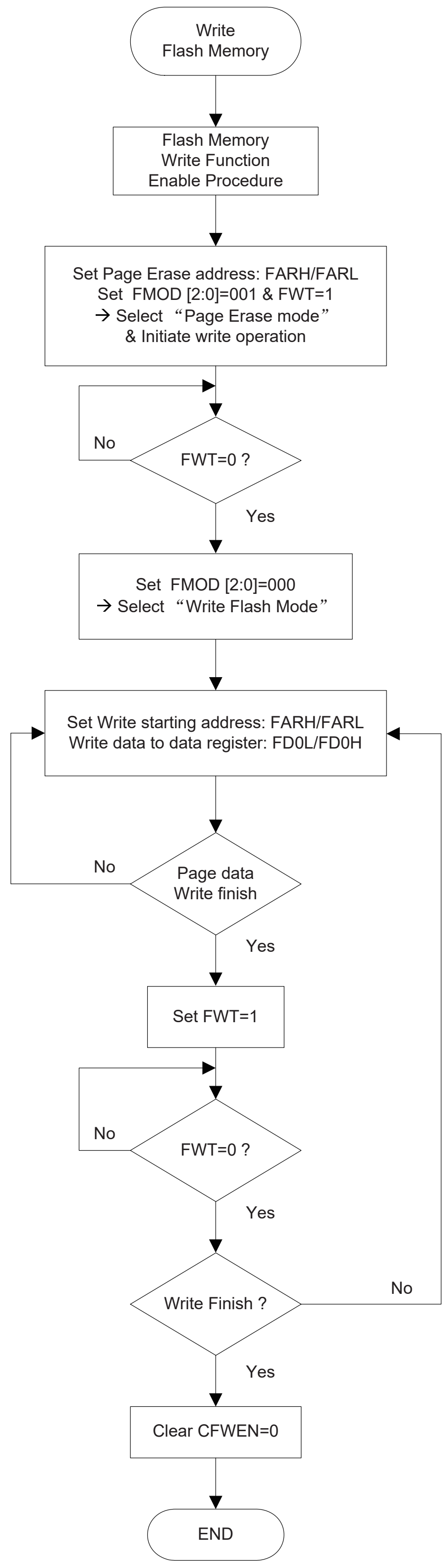
Erase Page	FARH	FARL[7:6]	FARL[5:0]
0	0000 0000	00	xx xxxx
1	0000 0000	01	xx xxxx
2	0000 0000	10	xx xxxx
3	0000 0000	11	xx xxxx
4	0000 0001	00	xx xxxx
5	0000 0001	01	xx xxxx
: :	: :	: :	: :
254	0011 1111	10	xx xxxx
255	0011 1111	11	xx xxxx
: :	: :	: :	: :
382	0101 1111	10	xx xxxx
383	0101 1111	11	xx xxxx

"x": don't care

Erase Page Number and Selection



Read Flash Memory Procedure


Write
Flash Memory
Flash Memory
Write Function
Enable Procedure
Set Page Erase address: FARH/FARL
Set FMOD [2:0]=001 & FWT=1
→ Select "Page Erase mode"
& Initiate write operation
No
FWT=0 ?
Yes
Set FMOD [2:0]=000
→ Select "Write Flash Mode"
Set Write starting address: FARH/FARL
Write data to data register: FD0L/FD0H
No
Page data
Write finish
Yes
Set FWT=1
No
FWT=0 ?
Yes
Write Finish ?
No
Yes
Clear CFWEN=0
END

Write Flash Memory Procedure

Note: When the FWT or FRD bit is set to 1, the MCU is stopped.

RAM Data Memory

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

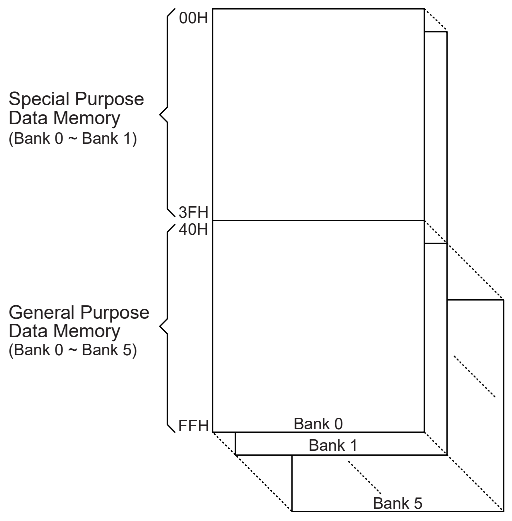
Structure

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

The Data Memory is subdivided into six banks, all of which are implemented in 8-bit wide RAM. The address range of the Special Purpose Data Memory for the device is from 00H to 3FH while the General Purpose Data Memory address range is from 40H to FFH.

Special Purpose Data Memory	General Purpose Data Memory	
Located Banks	Capacity	Bank: Address
0, 1	1152×8	0: 40H~FFH 1: 40H~FFH : : 4: 40H~FFH 5: 40H~FFH

Data Memory Summary



Data Memory Structure

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Bank 0	Bank 1
00H	IAR0	▨
01H	MP0	▨
02H	IAR1	▨
03H	MP1	▨
04H	BP	▨
05H	ACC	▨
06H	PCL	▨
07H	TBLP	▨
08H	TBLH	▨
09H	▨	▨
0AH	STATUS	▨
0BH	INTC0	▨
0CH	TMR0	▨
0DH	TMR0C	▨
0EH	TMR1	▨
0FH	TMR1C	▨
10H	PA	▨
11H	PAC	▨
12H	PAPU	▨
13H	PAWK	▨
14H	PB	▨
15H	PBC	▨
16H	PBPU	▨
17H	PC	▨
18H	PCC	▨
19H	PCPU	▨
1AH	CTRL0	▨
1BH	CTRL1	▨
1CH	▨	▨
1DH	▨	▨
1EH	INTC1	▨
1FH	ANCSR	▨
20H	ADRL	▨
21H	ADRH	▨
22H	ADCR	▨
23H	ACSR	▨
24H	TBHP	▨
25H	PD	▨
26H	PDC	▨
27H	PDPU	▨
28H	PE	▨
29H	PEC	▨
2AH	PEPU	▨
2BH	PF	▨
2CH	PFC	▨
2DH	PFPU	▨
2EH	TMR2L	▨
2FH	TMR2H	▨
30H	▨	▨
31H	CTRL2	FC0
32H	TMR2C	FC1
33H	DACR	FC2
34H	DA0R	FARL
35H	DA1R	FARH
36H	DA2R	FD0L
37H	DA3R	FD0H
38H	SPICR	FD1L
39H	BDR	FD1H
3AH	LVDC	FD2L
3BH	WDTC	FD2H
3CH	LVRC	PD3L
3DH	▨	FD3H
3EH	▨	▨
3FH	▨	▨

▨ : Unused, read as 00H

Special Purpose Data Memory

Special Function Registers

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

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation is using these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data only from Bank 0 while the IAR1 register together with the MP1 register can access data from any Data Memory 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. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address which the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank 0, while MP1 and IAR1 are used to access data from all banks according to the BP register. Direct Addressing can only be used with Bank 0, all other banks must be addressed indirectly using MP1 and IAR1.

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

Indirect Addressing Program Example

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

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

Bank Pointer – BP

For this device, the Program and Data Memory are divided into several banks. Selecting the required Program and Data Memory area is achieved using the Bank Pointer. Bits 6~5 of the Bank Pointer is used to select Program Memory Bank 0, 1 or 2, while bits 2~0 are used to select Data Memory Banks 0~5.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the Power Down Mode, in which case, the Data Memory bank remains unaffected. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from other banks 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	—	PMBP1	PMBP0	—	—	DMBP2	DMBP1	DMBP0
R/W	—	R/W	R/W	—	—	R/W	R/W	R/W
POR	—	0	0	—	—	0	0	0

Bit 7	Unimplemented, read as "0"
Bit 6~5	**PMBP1~PMBP0**: Program Memory Bank Pointer 00: Bank 0 01: Bank 1 10: Bank 2 11: Undefined
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 11x: Undefined

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

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

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

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

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

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

STATUS Register

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

"x": unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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

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

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

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

System Control Registers – CTRL0, CTRL1, CTRL2

These registers are used to provide control for various internal functions. These internal functions include the PFD function, Audio Processor control, certain system clock options, LXT oscillator low power control, external interrupt edge trigger type selection, Time Base function division ratio and the LXT oscillator on/off control after execution of HALT instruction.

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

Bit 7 **PCFG**: I/O configuration
0: INT/TMR0/PFD pin-shared with PA3/PA2/PA1
1: INT/TMR0/PFD pin-shared with PB4/PB5/PB6

Bit 6 **PFDCS**: PFD clock source
0: Timer 0
1: Timer 1

Bit 5~3 Unimplemented, read as "0"

Bit 2 **PFDC**: I/O or PFD selection
0: I/O
1: PFD

Bit 1 **LXTLP**: LXT oscillator low power control
0: Quick start-up mode
1: Low power mode

Bit 0 **CLKMOD**: System clock mode selection
0: High system clock – PLL mode
1: Low system clock – LXT is used as the system clock

CTRL1 Register

Bit	7	6	5	4	3	2	1	0
Name	INTEG1	INTEG0	TBSEL1	TBSEL0	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	1	0	0	0	—	—	—	—

Bit 7~6 **INTEG1~INTEG0**: External interrupt edge type selection
00: Disable
01: Rising edge trigger
10: Falling edge trigger
11: Dual edge trigger

Bit 5~4 **TBSEL1~TBSEL0**: Time Base period selection
00: $2^{10}/f_{TP0}$
01: $2^{11}/f_{TP0}$
10: $2^{12}/f_{TP0}$
11: $2^{13}/f_{TP0}$

Bit 3~2 Unimplemented, read as "0"

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

Bit 7~6 **M1~M0**: PLL output frequency selection
00: 24.576MHz
01: 32.768MHz
10: 40.96MHz
11: 49.152MHz
When the PLLEN bit is 0, PLL output clock is logic low. When the PLLEN bit is 1, PLL output clock is determined by M1~M0 bits.

Bit 5 **PLLD2**: PLL output frequency divider selection (for Audio Processor)
0: PLL output frequency divided by 2
1: PLL output frequency divided by 4
The detailed output frequencies are shown in the following table.

Bit 4 **AUPRST**: Audio Processor hardware reset bit
1→0→1: Audio Processor hardware reset

Bit 3 **PLLEN**: PLL enable control
0: Disable
1: Enable

Bit 2~1 **PLLD1~PLLD0**: PLL output frequency divider selection (for MCU)
00: PLL output frequency divided by 2
01: PLL output frequency divided by 4
10: PLL output frequency divided by 8
11: PLL output frequency divided by 16
Note that when the PLL frequency is 40.96/49.152MHz, setting these bits to "00" is useless, the hardware will force them to "01".

Bit 0 **LXTEN**: LXT oscillator on/off control after execution of HALT instruction
0: LXT off – Sleep mode
1: LXT on – Idle mode

PLL output frequencies corresponding to the M1 and M0 settings are shown below.

PLLEN	M1	M0	Clock Output
0	x	x	Low (Logic state)
1	0	0	24.576MHz
1	0	1	32.768MHz
1	1	0	40.96MHz
1	1	1	49.152MHz

x: Don't care.

Divided PLL output frequency for MCU system clock.

		PLLD1, PLLD0	0,0	0,1	1,0	1,1
		Divide	2	4	8	16
PLL freq. (MHz)	24.576	f_{SYS} (MHz)	12.288	6.144	3.072	1.536
	32.768		16.384	8.192	4.096	2.048
	40.96		10.24^(Note)	10.24	5.12	2.56
	49.152		12.288^(Note)	12.288	6.144	3.072

Note: When the PLL frequency is 40.96/49.152MHz, PLLD1~PLLD0 = 00 is useless, they will be forced to "01" by hardware.

Divided PLL output frequency for Audio Processor system clock.

		PLLD2	0	1
		Divide	2	4
PLL freq. (MHz)	24.576	f_{AP} (MHz)	12.288	6.144
	32.768		16.384	8.192
	40.96		20.48	10.24
	49.152		24.576	12.288

Oscillators

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

System Oscillator Overview

In addition to being the source of the main system clock the external oscillator LXT also provides clock source for the Watchdog Timer function, Timer/Event counters and Time Base. The external oscillator requires some external components while the integrated internal oscillator, requires no external components. The system clock can be provided from a choice of two oscillator modes, LXT crystal oscillator mode and LXT crystal oscillator together with PLL mode, which is selected by application program. The LIRC oscillator is mainly used for the Watchdog Timer function.

Type	Name	Frequency	Pins
External Low Speed Crystal	LXT	32.768kHz	XIN/XOUT
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

Internal PLL Frequency Generator

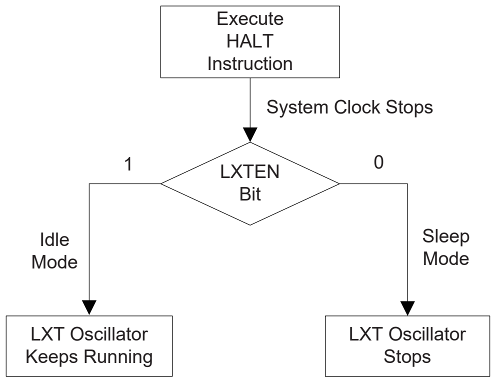
The internal PLL frequency generator is used to generate frequency for the MCU normal mode system clock and Audio Processor system clock. This PLL generator can be enabled or disabled by the PLL control bit PLLEN in the CTRL2 register. After a power on reset, the PLL control bit will be cleared to 0 to disable the PLL generator. To select the PLL output to be the high speed system clock, the CLKMOD bit in the CTRL0 register should be cleared to zero. The PLL generator will provide a range of frequencies, the selection of which is implemented using the M1~M0 bits in the CTRL2 register. And the PLL output frequency division for the MCU and Audio Processor are determined by the PLLD1~PLLD0 bits and PLLD2 bit respectively.

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.

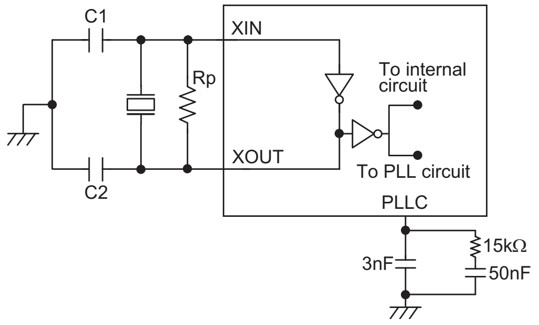
To select the LXT oscillator to be the low speed system oscillator, the CLKMOD bit in the CTRL0 register should be set high. When a HALT instruction is executed, the system clock is stopped, but the LXTEN bit in the CTRL2 register determines if the LXT oscillator continues running when the microcontroller 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.

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.



LXTEN Bit Function

For oscillator stability and to minimise the effects of noise and crosstalk, it is important to ensure that the crystal and any associated resistors and capacitors along with interconnecting lines are all located as close to the MCU as possible.



LXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
32.768kHz	10pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. 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.

Internal 32kHz Oscillator – LIRC

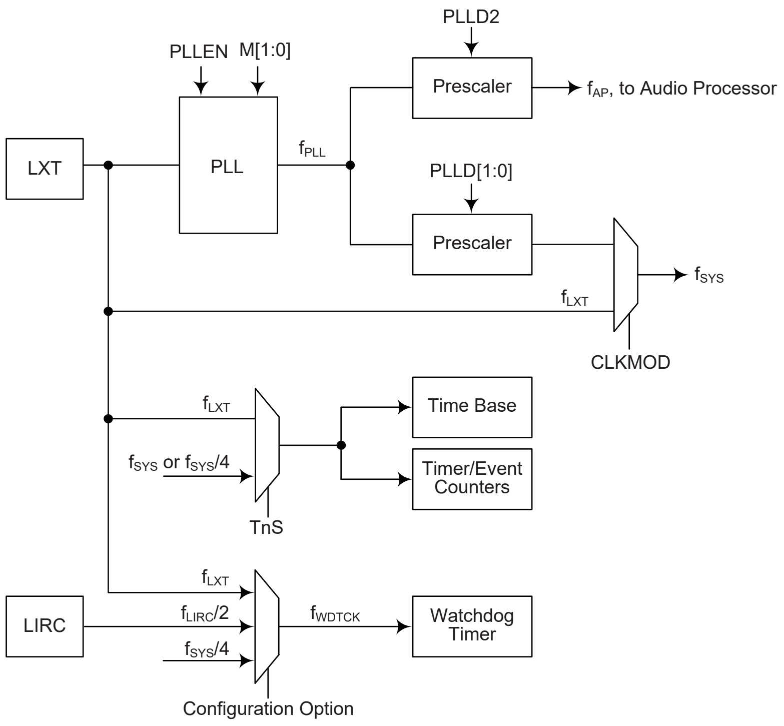
The Internal 32kHz oscillator is served as one of Watchdog Timer clock source. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. The LIRC is enabled once the Watchdog Timer or the LVR function is enabled, and it is disabled if both the Watchdog Timer and the LVR function are disabled.

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.



Note: The Time Base clock source is selected using the T0S bit. Refer to the associated section for Time Base and Timer/Event Counters clock source selection details.

Device Clock Configurations

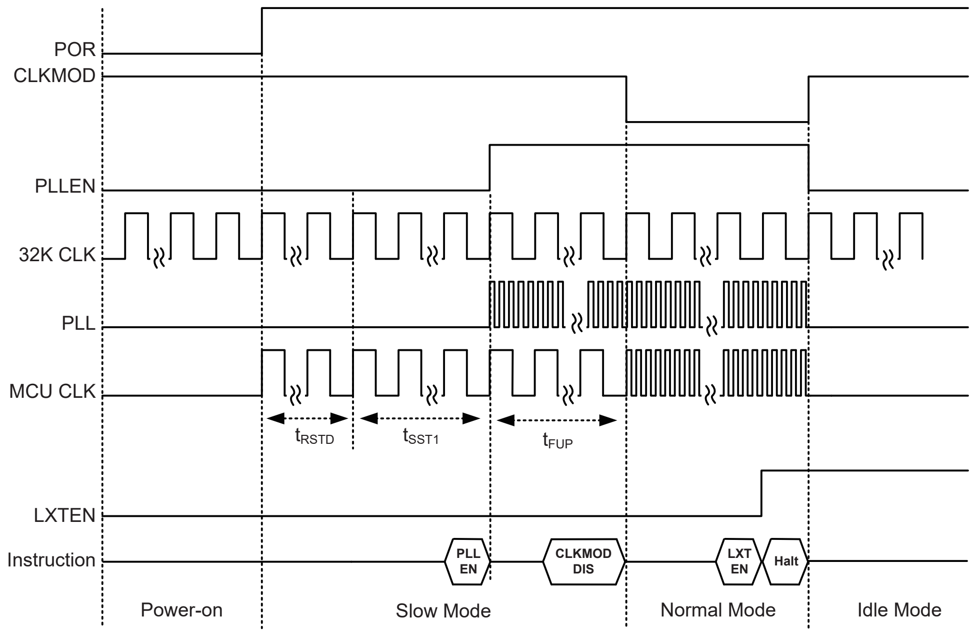
MCU/Audio Processor (AP) Operating Mode Table

MCU		Audio Processor		Register Setting			LXT Oscillator	HALT Instruction	IDLE COMMAND for AP
Mode	System Clock	Mode	System Clock	PLLEN bit	CLKMOD bit	LXTEN bit			
Normal	PLL	Normal	PLL_{AP}	1	0	x	On	Non executed	Non executed
	PLL	Idle1	HALT	1	0	x	On		executed
	PLL	Idle2	PLL_{AP}	1	0	x	On		executed
Slow	LXT	Normal	PLL_{AP}	1	1	x	On		Non executed
	LXT	Idle1	HALT	1	1	x	On		Executed
	LXT	Idle2	PLL_{AP}	1	1	x	On		Executed
Idle	HALT	Idle1	HALT	0	1	1	On	Executed	x
Sleep	HALT	Idle1	HALT	0	1	0	Off		

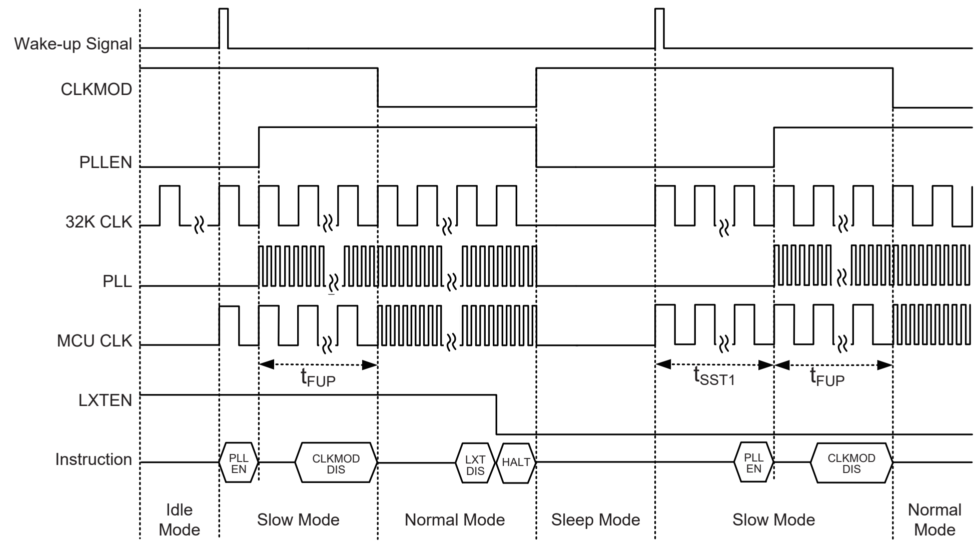
"x": Don't care

Note: 1. The LXT oscillator is the 32768Hz oscillator.
2. t_{SST1} for the LXT oscillator is 1024 clocks.
3. If PLLEN=0, the CLKMOD bit will be set.
4.To disable PLL (PLLEN bit = 0) before MCU executing the HALT instruction.
5.Clear bit AUPRST of CTRL2 register of internal MCU to enter Idle1 and clear I/O and CLI command register value of audio processor.
6. Make 80F00 of I/O command to enter idle2 and clear CLI command register value of audio processor.
7. f_{AP} : PLL divided frequency to Audio processor. $PLL_{AP} = 1/ f_{AP}$.

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

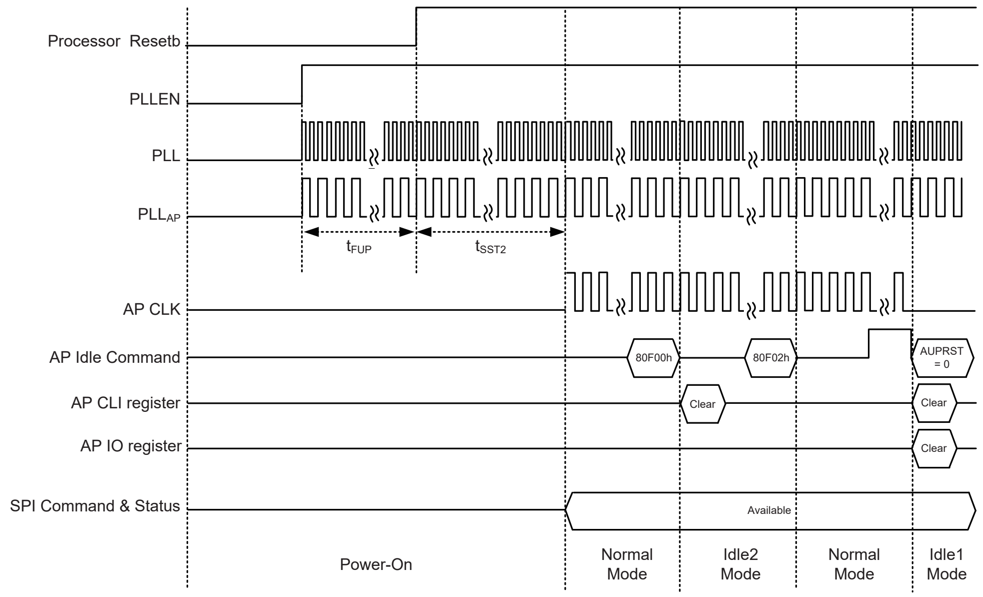


MCU Power-on Sequency



Wake-up Sequency

The Audio Processor operating mode switch timing are shown in the following figures:



Audio Processor Power-on Sequency

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 and 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 LIRC 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 option or the related register has enabled the Watchdog Timer then the internal LIRC oscillator 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 oscillator, 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 = 50ms
3. t_{SST} = 1024 t_{SYS}

Wake-up Delay Time

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal LIRC oscillator clock division, $f_{LIRC}/2$, the system clock division, $f_{SYS}/4$, or the LXT oscillator, which is selected by the configuration option. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register. The LIRC oscillator has an approximate period frequency of 32kHz at a supply voltage of 5V. However, it should be noted that this specified internal clock period can vary with V_{DD}, temperature and process variations. The LXT oscillator is supplied by an external 32.768kHz crystal.

Watchdog Timer Control Registers

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. This register together with several configuration options control the overall operation of the Watchdog Timer.

WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT enable/disable control
10101: Disable
01010: Enable
Other values: MCU reset
When these bits are changed by the environmental noise to reset the microcontroller, the reset operation will be activated after 2~3 WDTCK clock cycles.

Bit 2~0 **WS2~WS0**: WDT Time-out period selection
000: $2^8/f_{WDTCK}$
001: $2^{10}/f_{WDTCK}$
010: $2^{12}/f_{WDTCK}$
011: $2^{14}/f_{WDTCK}$
100: $2^{15}/f_{WDTCK}$
101: $2^{16}/f_{WDTCK}$
110: $2^{17}/f_{WDTCK}$
111: $2^{18}/f_{WDTCK}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, this clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. Note that if the Watchdog Timer function is not enabled, then any instruction related to the Watchdog Timer will result in no operation.

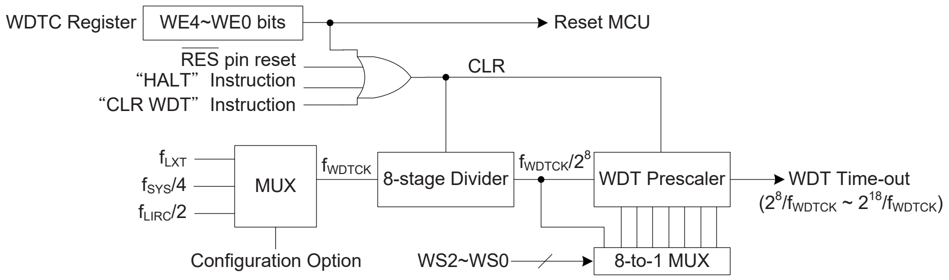
Setting the various Watchdog Timer options are controlled via the configuration options and the internal register WDTC. Enabling the Watchdog Timer can be controlled by both a configuration option and the WE4~WE0 bits in the internal WDTC register in the Data Memory. The Watchdog Timer will be disabled if bits WE4~WE0 in the WDTC register are written with the binary value 10101B while the WDT Timer will be enabled if these bits are written with the binary value 01010B. If these bits are written with the other values except 10101B and 01010B, the MCU will be reset.

WDT Configuration Option	WE4~WE0 Bits	WDT Function
WDT always enable	xxxxx	Enable
By software control	10101B	Disable
	01010B	Enable
	Any other value	Reset MCU

Watchdog Timer Enable/Disable Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the 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. Four 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 $\overline{\text{RES}}$ reset pin, the second is a WDT software reset, which means a certain value except 10101B and 01010B written into the WE4~WE0 bits, the third is using the Clear Watchdog Timer software instruction and the fourth is via a "HALT" instruction.

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



Watchdog Timer

Reset and Initialisation

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

In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ 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 deal with normal operation after the reset line is allowed to return high.

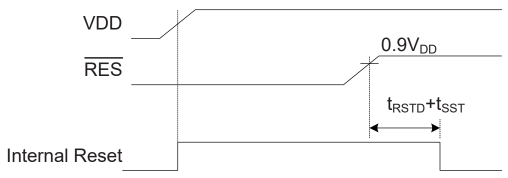
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being set. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset is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

Reset Functions

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

Power-on Reset

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



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

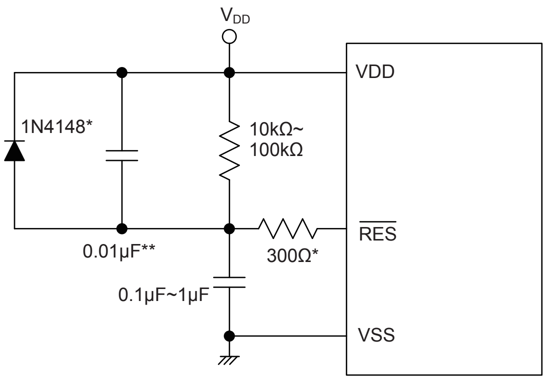
Power-On Reset Timing Chart

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

As the reset pin is shared with PA7, the reset function must be selected using a configuration option. Although the microcontroller has an internal RC reset function, if the VDD power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{\text{RES}}$ pin, whose additional time delay will ensure that the $\overline{\text{RES}}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{\text{RES}}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.

For most applications a resistor connected between VDD and the $\overline{\text{RES}}$ pin and a capacitor connected between VSS and the $\overline{\text{RES}}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{\text{RES}}$ pin should be kept as short as possible to minimise any stray noise interference.

For applications that operate within an environment where more noise is present the reset circuit shown is recommended.

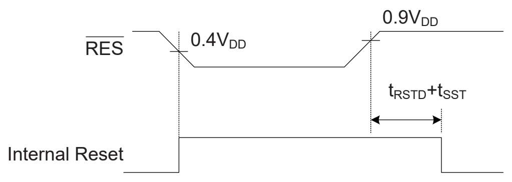


Note: "*" It is recommended that this component is added for added ESD protection.
"**" It is recommended that this component is added in environments where power line noise is significant.

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

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

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

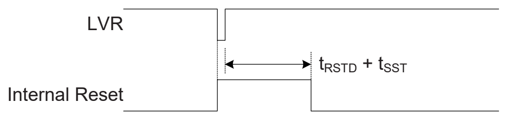


Note: t_{RSTD} is power-on delay, typical time=16.7ms

$\overline{\text{RES}}$ 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 and provides an MCU reset should the value fall below a certain predefined level. The LVR function is always enabled with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing a battery, the LVR will automatically reset the device internally. For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for greater than the value t_{LVR} specified in the LVR/LVD characteristics. If the low voltage state does not exceed t_{LVR}, the LVR will ignore it and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after 2~3 f_{LIRC} clock cycles. Note that the LVR function will automatically be disabled when the MCU enters the Power Down Mode.



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

Low Voltage Reset Timing Chart

- **LVRC Register**

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

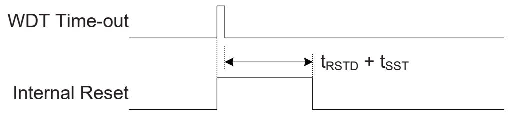
Bit 7~0 **LVS7~LVS0**: LVR Voltage Select control
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Any other value: Generates MCU reset – register is reset to POR value

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

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

Watchdog Time-out Reset during Normal Operation

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

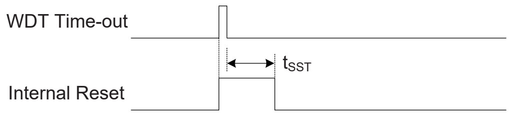


Note: t_{RSTD} is power-on delay, typical time=16.7ms

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.

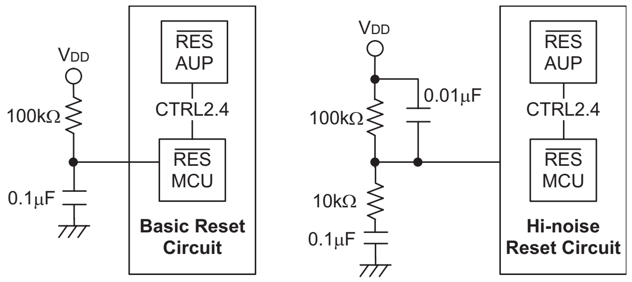


Note: t_{SST} is 1024 clock cycles for the system clock source provided by LXT.

WDT Time-out Reset during Idle/Sleep Timing Chart

Audio Processor Reset

The Audio Processor function is reset by the AUPRST bit in the CTRL2 control register.



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

Audio Processor 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 Mode 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 Mode operation
1	u	WDT time-out reset during NORMAL 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
Precaler	The Timer Counter Prescaler will be cleared
Input/Output Ports	I/O ports will be set as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

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

Register	Power On Reset	$\overline{\text{RES}}$ or LVR Reset (Normal Operation)	$\overline{\text{RES}}$ or LVR Reset (Idle/Sleep)	WDT Time-out (Normal Operation)	WDT Time-out (Idle/Sleep)
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	-00- -000	-00- -000	-00- -000	-00- -000	-uu- -uuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
STATUS	--00 xxxx	--uu uuuu	--01 uuuu	--1u 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
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	0000 1000	0000 1000	0000 1000	0000 1000	uuuu 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	0000 0000	0000 0000	0000 0000	0000 0000	uuuu 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
PF	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTRL0	0000 0001	0000 0001	0000 0001	0000 0001	uuuu uuuu
CTRL1	1000 ----	1000 ----	1000 ----	1000 ----	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	00-- 0000	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
ACSR	11-- -000	11-- -000	11-- -000	11-- -000	uu-- -uuu

Register	Power On Reset	$\overline{\text{RES}}$ or LVR Reset (Normal Operation)	$\overline{\text{RES}}$ or LVR Reset (Idle/Sleep)	WDT Time-out (Normal Operation)	WDT Time-out (Idle/Sleep)
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 uuxx
LVDC	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu
WDTC	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu
LVRC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
FC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC2	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
FARL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
FD0L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

Note: "-" not implement
"u" means "unchanged"
"x" means "unknown"

Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA~PF. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A, [m]", where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWK	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PE	—	—	—	—	—	—	PE1	PE0
PEC	—	—	—	—	—	—	PEC1	PEC0
PEPU	—	—	—	—	—	—	PEPU1	PEPU0
PF	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0

"—": Unimplemented, read as "0"

I/O Basic Function Registers List

Pull-high Resistors

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

PxPU Register

Bit	7	6	5	4	3	2	1	0
Name	PxPU7	PxPU6	PxPU5	PxPU4	PxPU3	PxPU2	PxPU1	PxPU0
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

PxPUn: I/O Port x Pin pull-high function control
0: Disable
1: Enable

The PxPUn bit is used to control the pin pull-high function. Here the "x" can be A, B, C, D, E and F. However, the actual available bits for each I/O Port may be different.

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~PA7 can be selected individually to have this wake-up feature using an internal register known as PAWK, located in the Data Memory.

PAWK Register

Bit	7	6	5	4	3	2	1	0
Name	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
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

PAWKn: Port A Pin wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

Each I/O port has its own control register known as PAC~PFC, to control 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 set as a CMOS output. If the pin is currently set as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

PxC Register

Bit	7	6	5	4	3	2	1	0
Name	PxC7	PxC6	PxC5	PxC4	PxC3	PxC2	PxC1	PxC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

PxCn: I/O Port x Pin type selection
0: Output
1: Input

The PxCn bit is used to control the pin type selection. Here the "x" can be A, B, C, D, E and F. However, the actual available bits for each I/O Port may be different.

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 Reset Pin

The external reset pin, $\overline{\text{RES}}$, is pin-shared with PA7. Whether to use the pin as an external reset pin or not is determined by the associated configuration option.

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 Inputs

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 ANCSR register, 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 options remain, however if used as A/D inputs then any pull-high resistor configurations associated with these pins will be automatically disconnected.

D/A Outputs

The device has 4 outputs from the D/A converter. All of these analog outputs are pin-shared with I/O pins. If these pins are to be used as D/A outputs and not as I/O pins then the corresponding ENn bits in the DACR register, must be properly setup. There are no configuration options associated with the D/A converter.

SPI Pins

The SPI associated pins are pin-shared with I/O pins. This function is controlled using the IEMC bit in the SPICR register.

Audio processor Pins

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

Pin Remapping Configuration

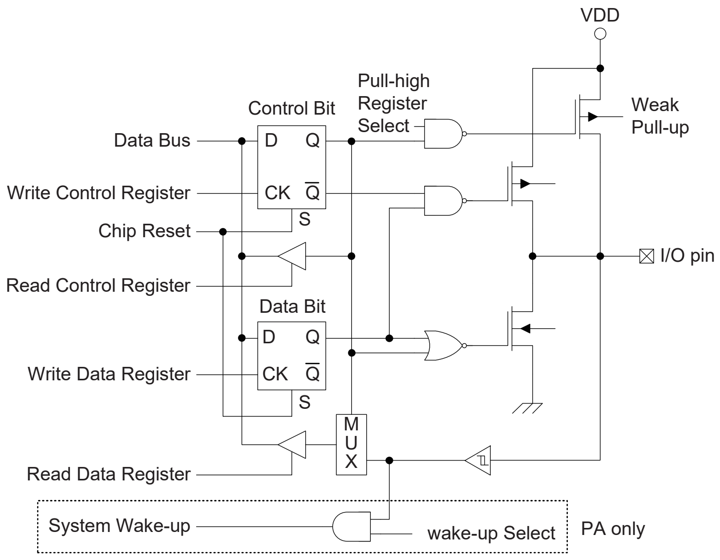
The pin remapping function enables the function pins INT, TMR0 and PFD to be located on different port pins. It is important not to confuse the Pin Remapping function with the Pin-shared function, these two functions have no interdependence.

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.

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

I/O Pin Structure

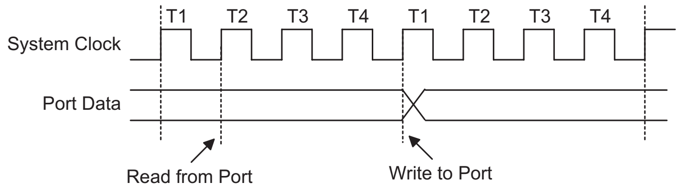
The accompanying diagram illustrates the internal structure of the I/O logic function. As the exact logical construction of the I/O pin may differ from this diagram, it is supplied as a guide only to assist with the functional understanding of the I/O logic function. The wide range of pin-shared structures does not permit all types to be shown.



Logic Function Input/Output Structure

Programming Considerations

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



Read Modify Write Timing

Pins PA0~PA7 each have wake-up functions, 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 pins. Single or multiple pins on Port A can be set to have this function.

Timer/Event Counters

The provision of timer form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contains from two 8-bit and one 16-bit count-up timers. 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.

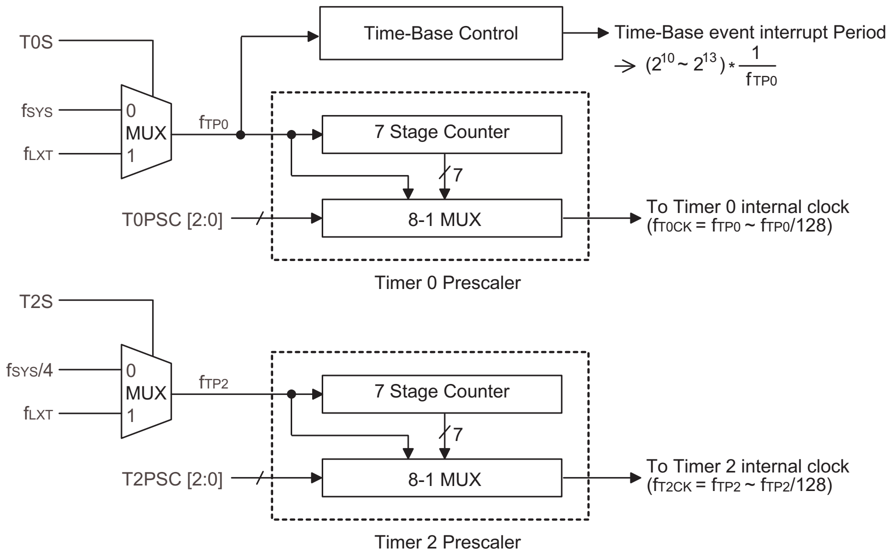
There are two types of registers related to the Timer/Event Counters. The first is the register that contains the actual value of the timer and into which an initial value can be preloaded. Reading from this register retrieves the contents of the Timer/Event Counter. The second type of associated register 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.

Name	Bits	Data Register	Control Register
Timer/Event Counter 0	8	TMR0	TMR0C
Timer/Event Counter 1	8	TMR1	TMR1C
Timer/Event Counter 2	16	TMR2H/TMR2L	TMR2C

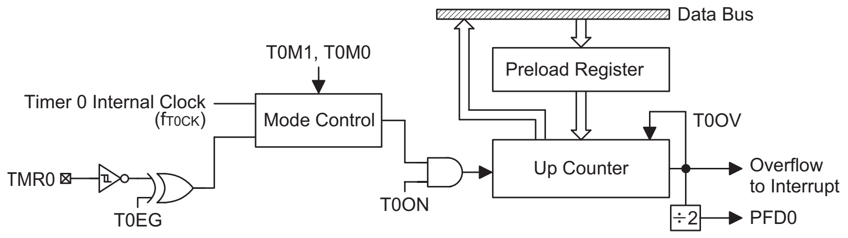
Configuring the Timer/Event Counter Input Clock Source

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. The internal clock source can be f_{SYS}, $f_{SYS}/4$ or the external low speed oscillator LXT, selected by the TnS bit in the register TMRnC register. 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 TnPSC2~TnPSC0.

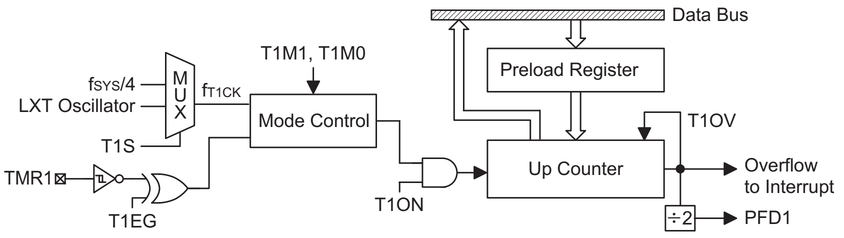
An external clock source is used when the Timer/Event Counter is in the event counting mode, the clock source being provided on an external timer pin TMRn. Depending upon the condition of the TnEG bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.



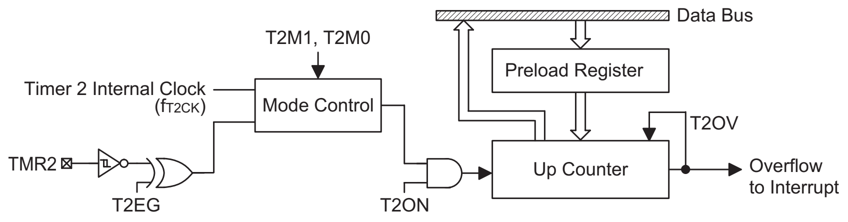
Clock Source for Timer 0/Timer 2/Time Base



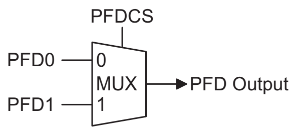
8-bit Timer/Event Counter 0 Structure



8-bit Timer/Event Counter 1 Structure



16-bit Timer/Event Counter 2 Structure



PFD Option

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 Counters or FFFFH for the 16-bit Timer/Event Counter, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then 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. It should be noted that after power-on, the preload registers will be in an unknown condition. Note that if the Timer/Event Counter is in an OFF condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter the next time an overflow occurs.

Timer Control Registers – TMR0C, TMR1C, 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 controls 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 select 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 to 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**: Timer 0 operation mode selection
00: No mode available
01: Event counter mode
10: Timer mode
11: Pulse width capture mode

Bit 5 **T0S**: Timer 0 and Time Base clock source selection
0: f_{SYS}
1: LXT oscillator

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

Bit 3 **T0EG**: Timer/Event Counter 0 active edge selection
In event counter mode (T0M1~T0M0 = 01)
0: Count on rising edge
1: Count on falling edge
In pulse width measurement mode (T0M1~T0M0 = 11)
0: Start counting on falling edge, stop on the rising edge
1: Start counting on rising edge, stop on the falling edge

Bit 2~0 **T0PSC2~ T0PSC0**: Timer 0 prescalar rate selection
000: f_{TP0}
001: $f_{TP0}/2$
010: $f_{TP0}/4$
011: $f_{TP0}/8$
100: $f_{TP0}/16$
101: $f_{TP0}/32$
110: $f_{TP0}/64$
111: $f_{TP0}/128$

TMR1C Register

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

Bit 7~6 **T1M1~T1M0**: Timer 1 operation mode selection
00: No mode available
01: Event counter mode
10: Timer mode
11: Pulse width capture mode

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

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

Bit 3 **T1EG**: Timer/Event Counter 1 active edge selection
In event counter mode (T1M1~T1M0 = 01)
0: Count on rising edge
1: Count on falling edge
In pulse width measurement mode (T1M1~T1M0 = 11)
0: Start counting on falling edge, stop on the rising edge
1: Start counting on rising edge, stop on the 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**: Timer 2 operation mode selection
00: No mode available
01: Event counter mode
10: Timer mode
11: Pulse width capture mode

Bit 5 **T2S**: Timer 2 clock source selection
0: $f_{SYS}/4$
1: LXT oscillator

Bit 4 **T2ON**: Timer/event counter 2 counting enable
0: Disable
1: Enable

Bit 3 **T2EG**: Timer/Event Counter 2 active edge selection
In event counter mode (T2M1~T2M0 = 01)
0: Count on rising edge
1: Count on falling edge
In pulse width measurement mode (T2M1~T2M0 = 11)
0: Start counting on falling edge, stop on the rising edge
1: Start counting on rising edge, stop on the falling edge.

Bit 2~0 **T2PSC2~ T2PSC0**: Timer 2 prescalar rate selection
000: f_{TP2}
001: $f_{TP2}/2$
010: $f_{TP2}/4$
011: $f_{TP2}/8$
100: $f_{TP2}/16$
101: $f_{TP2}/32$
110: $f_{TP2}/64$
111: $f_{TP2}/128$

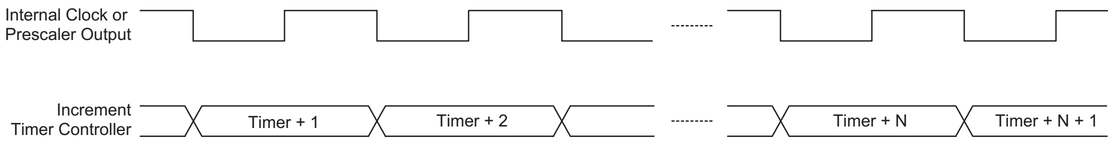
Timer Mode

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.

Bit7	Bit6
1	0

Control Register Operating Mode Select Bits for the Timer Mode

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, for some timers, 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 are one of the wake-up sources. However, the internal interrupt can be disabled by ensuring that the TnE bits of the INTCn register are reset to zero.


Internal Clock or Prescaler Output
Increment Timer Controller
Timer + 1
Timer + 2
Timer + N
Timer + N + 1

Timer Mode Timing Chart

Event Counter Mode

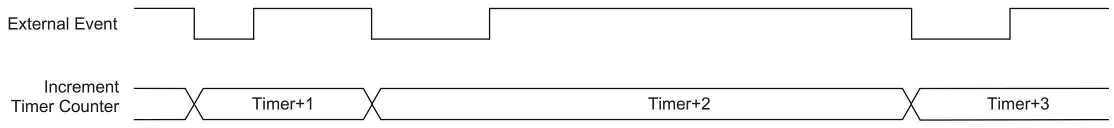
In this mode, a number of externally changing logic events, occurring on the external timer TMRn pin, can be recorded by the Timer/Event Counter. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

Bit7	Bit6
0	1

Control Register Operating Mode Select Bits for the Event Counter Mode

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 set, the enable bit TnON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. If the Active Edge Select bit, TnEG, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the TnEG is high, the counter will increment each time the external timer pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register. It is reset to zero.

As the external timer pin is shared with an I/O pin, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Event Counting Mode. The second is to ensure that the port control register configures the pin as an input. It should be noted that in the event counting mode, even if the microcontroller is in the Idle/Sleep Mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input TMRn pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (TnEG=1)

Pulse Width Capture Mode

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

Bit7	Bit6
1	1

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

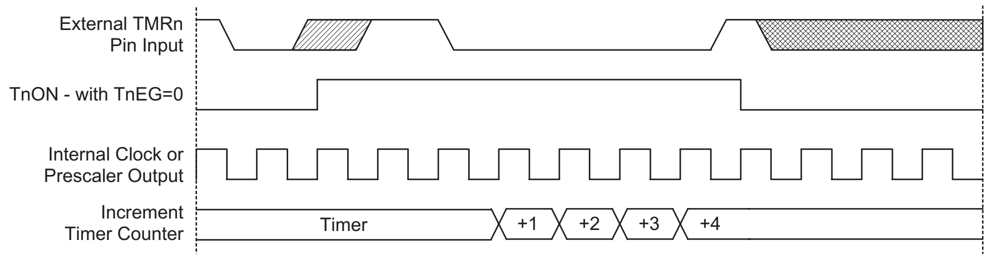
In this mode the internal clock, f_{SYS}, $f_{SYS}/4$ or LXT, is used as the internal clock for the 8-bit and 16-bit Timer/Event Counters. However, the clock source, f_{SYS}, $f_{SYS}/4$ or LXT, for the 8-bit timer 0 and 16-bit timer 2 is furtherly divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits TnPSC2~TnPSC0, which are bit 2~0 of the Timer Control Register, After other bits in the Timer Control Register have been set, 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 be implemented. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the pulse width capture mode, the second is to ensure that the port control register configure the pin as an input.



Pulse Width Capture Mode Timing Chart (TnEG=0)

Prescaler

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

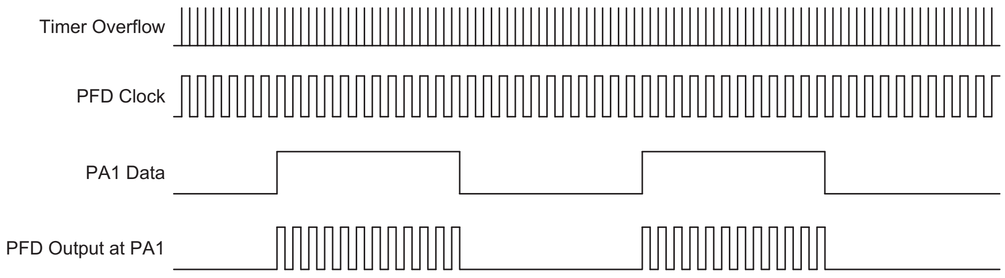
PFD Function

The Programmable Frequency Divider provides a means of producing a variable frequency output suitable for application, 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 PFDC bit in CTRL0. For this device the clock source can come from either Timer/Event Counter 0 or Timer/Event Counter 1. The output frequency is controlled by loading the required values into the timer prescaler and timer registers to give the required division ratio. The counter will begin to count-up from this preload register value until full, at which point an overflow signal is generated, causing the internal PFDn output to change state. Then the counter will be automatically reloaded with the preload register value and continue counting-up. If the CTRL0 register has selected the PFD function, then for PFD output to operate, it is essential for the Port A control register PAC to set the PFD pin as output. PA1 must be set high to activate the PFD. The output data bit can be used as the on/off control bit for the PFD output. Note that the PFD output will be low if the output data bit is cleared to zero.

When using this method of frequency generation, if the crystal oscillator is used for the system clock, very precise values of frequencies can be generated.

The PDF pin function, which is pin-shared with PA1 by default option, can be remapped to PB6 by setting the PCFG bit in the CTRL0 register to "1".



PFD Function (PCFG=0)

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 set for use as a Timer/Event Counter input pin. This is achieved by ensuring that the mode selects bits in the Timer/Event Counter control register, either the event counter or pulse width capture mode. Additionally the corresponding Port Control Register bit must be set high to ensure that the pin is set 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 the Timer/Event Counter is 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 can be 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 interrupt 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 set along with how the interrupts are enabled and managed. Note that 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/Event 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:
                  ; set Timer 0 registers
mov  a,09bh       ; set Timer 0 preload value
mov  tmr0,a
mov  a,081h       ; set Timer 0 control register
mov  tmr0c,a      ; timer mode, fSYS and prescaler set to /2
                  ; set interrupt register
mov  a,009h       ; enable both master interrupt and timer 0 interrupt
mov  intc0,a
:       :
set  tmr0c.4      ; start Timer
:       :
```

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. The Time Base function is enabled/disabled once its clock source is enabled/disabled.

When the Time Base time out occurs, 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.

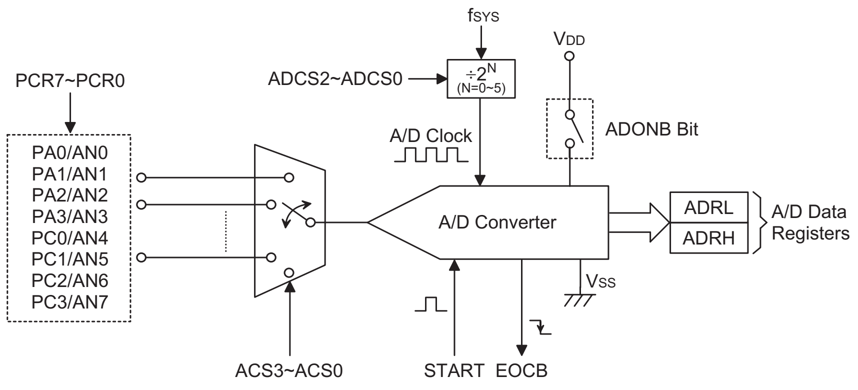
Analog to Digital Converter

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.

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



A/D Converter Structure

A/D Converter 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 bits 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.

ADRH, ADRL Register

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

"x" unknown

"—": Unimplemented, read as "0"

D11~D0: A/D conversion data

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

To control the function and operation of the A/D converter, three control registers known as ADCR, ACSR and ANCSR are provided. These 8-bit registers define functions such as the on/off function, 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 PCR7~PCR0 bits contained in the ANCSR register determine which pins on PA0~PA3 and 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 PCRn bit has a value of 1, then the corresponding pin, namely one of the AN0~AN7 analog inputs, will be set as analog inputs. Note that if the PCRn bit is set to zero, then the corresponding pin on PA0~PA3 or PC0~PC3 will be set as a normal I/O pin or other pin-shared function, the analog input channels will be all disabled and the A/D converter circuitry will be powered off.

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	0	—	—	0	0	0	0

Bit 7 **START**: Start the A/D conversion
0→1→0: Start
0→1: Reset the A/D converter and set EOCB to "1"
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process. When the bit is set high the A/D converter will be reset.

Bit 6 **EOCB**: End of A/D conversion flag
0: A/D conversion ended
1: A/D conversion in progress
This read only flag is used to indicate when an A/D conversion process has completed. When the conversion process is running, the bit will be high.

Bit 5~4 Unimplemented, read as "0"

Bit 3~0 **ACS3~ACS0**: A/D channel selection
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**: A/D Converter module on/off control bit
0: A/D Converter module is on
1: A/D Converter module is 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. It is recommended to set ADONB bit high 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: $f_{SYS}/2$
001: $f_{SYS}/8$
010: $f_{SYS}/32$
011: Undefined, can't be used
100: f_{SYS}
101: $f_{SYS}/4$
110: $f_{SYS}/16$
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

A/D Operation

The START bit in the register is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the EOCB bit in the ADCR register will be set to 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 value of permissible A/D clock period, t_{AD}, is 0.5μs~10μs, care must be taken for system clock frequencies. For example, as the system clock speeds at a frequency of 4MHz, the ADCS2, ADCS1 and ADCS0 bits should not be set to "100". 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 or larger than the specified A/D Clock Period range.

f_{SYS}	A/D Clock Period (t_{AD})						
	ADCS2, ADCS1, ADCS0 = 000 (f_{SYS}/2)	ADCS2, ADCS1, ADCS0 = 001 (f_{SYS}/8)	ADCS2, ADCS1, ADCS0 = 010 (f_{SYS}/32)	ADCS2, ADCS1, ADCS0 =100 (f_{SYS})	ADCS2, ADCS1, ADCS0 = 101 (f_{SYS}/4)	ADCS2, ADCS1, ADCS0 = 110 (f_{SYS}/16)	ADCS2, ADCS1, ADCS0 = 011,111
1MHz	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 set as normal input/output pins or whether they are set 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 set through register programming, apply to the input pins only when they are used as normal I/O pins, if set as A/D inputs the pull-high resistors will be automatically disconnected. Note that it is not necessary to first set the A/D pin as an input in the related I/O 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

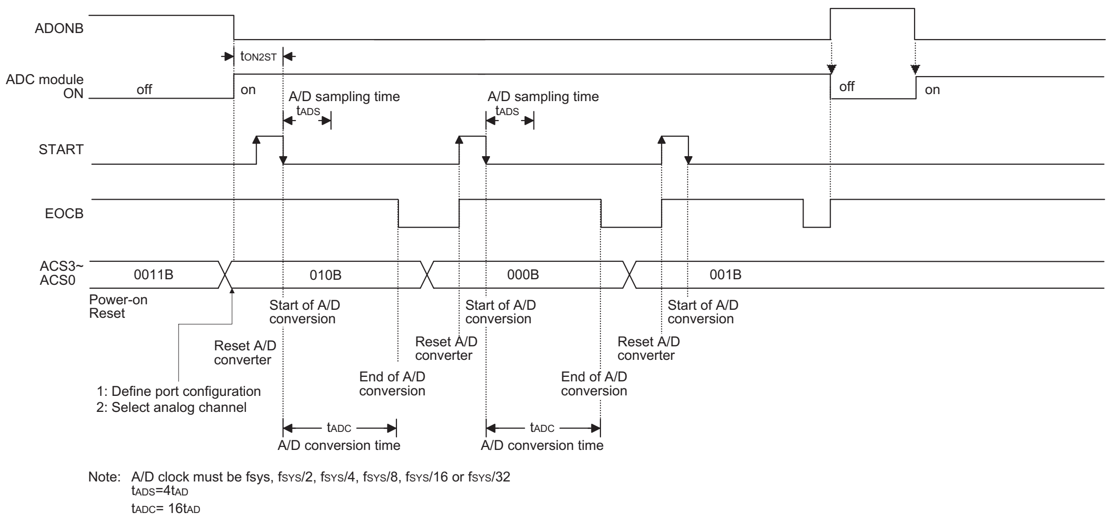
The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

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

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

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

The setup 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.



A/D Conversion Timing

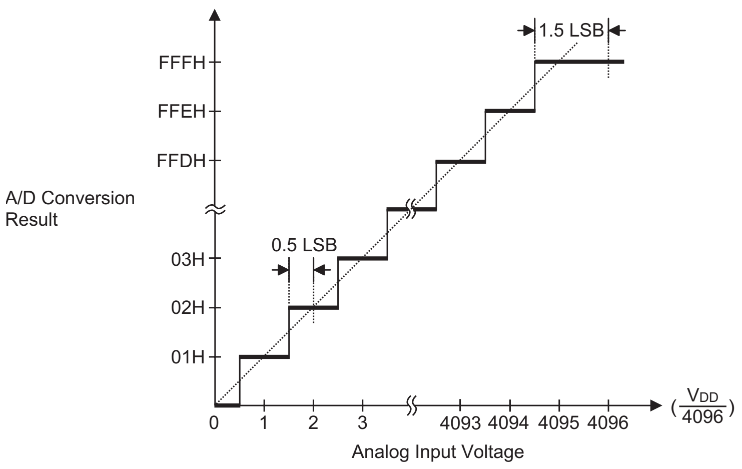
Programming Considerations

When programming, the 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} divided by 4096. The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter.

Note that to reduce the quantisation error, a 0.5 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 digitized value will change at a point 1.5 LSB below the V_{DD} level.



Ideal A/D Transfer Function

A/D Programming Example

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

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

```
clr  ADE               ; disable A/D Converter interrupt
mov  a,01h
mov  ACSR,a            ; select fSYS/8 as A/D clock and ADONB=0
mov  a,01h
mov  ANCSR,a           ; setup ANCSR register to configure I/O Port PA0 as A/D input
mov  a,00h
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 A/D Converter module, it is necessary to set ADONB as "1".

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

```
clr  ADE              ; disable A/D Converter interrupt
mov  a,01h
mov  ACSR,a           ; select fSYS/8 as A/D clock and ADONB=0
mov  a,01h
mov  ANCSR,a          ; setup ANCSR register to configure I/O Port PA0 as A/D input
mov  a,00h
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 A/D Converter interrupt request flag
set  ADE              ; enable A/D Converter interrupt
set  EMI              ; enable global interrupt
:
:
                      ; A/D Converter interrupt service routine
ADC_ISR:
mov  acc_stack,a      ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a   ; save STATUS to user defined memory
:
:
mov  a,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 A/D Converter interrupt flag
reti
```

Note: To power off A/D Converter 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 requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs.

The device contains only one external interrupt and multiple internal interrupts. The external interrupt is controlled by the action of the external interrupt pin, while the internal interrupt is controlled by the Timer/Event Counter, the A/D converter interrupt, Timer Base interrupt and Audio Processor.

Interrupt Register

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

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

Function	Enable Bit	Request Flag	Note
Global	EMI	—	—
INT Pin	EIF	EEI	—
Timer/Event Counter	TnE	TnF	n=0~2
A/D Converter	ADE	ADF	—
Timer Base	TBE	TBF	—
Audio Processor	AUPE	AUPF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTC0	—	T0F	AUPF	EIF	T0E	AUPE	EEI	EMI
INTC1	TBF	ADF	T2F	T1F	TBE	ADE	T2E	T1E

Interrupt Registers List

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: No request
1: Interrupt request

Bit 5 **AUPF**: Audio Processor interrupt request flag
0: No request
1: Interrupt request

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

Bit 3 **T0E**: Timer/Event Counter 0 interrupt control
0: Disable
1: Enable

Bit 2 **AUPE**: Audio Processor interrupt control
0: Disable
1: Enable

Bit 1 **EEI**: INT pin interrupt control
0: Disable
1: Enable

Bit 0 **EMI**: Global interrupt control
0: Disable
1: Enable

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	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 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **ADF**: A/D converter interrrupt request flag
0: No request
1: Interrupt request

Bit 5 **T2F**: Timer/Event Counter 2 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T1F**: Timer/Event Counter 1 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TBE**: Time Base interrupt control
0: Disable
1: Enable

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

Bit 1 **T2E**: Timer/Event Counter 2 interrupt control
0: Disable
1: Enable

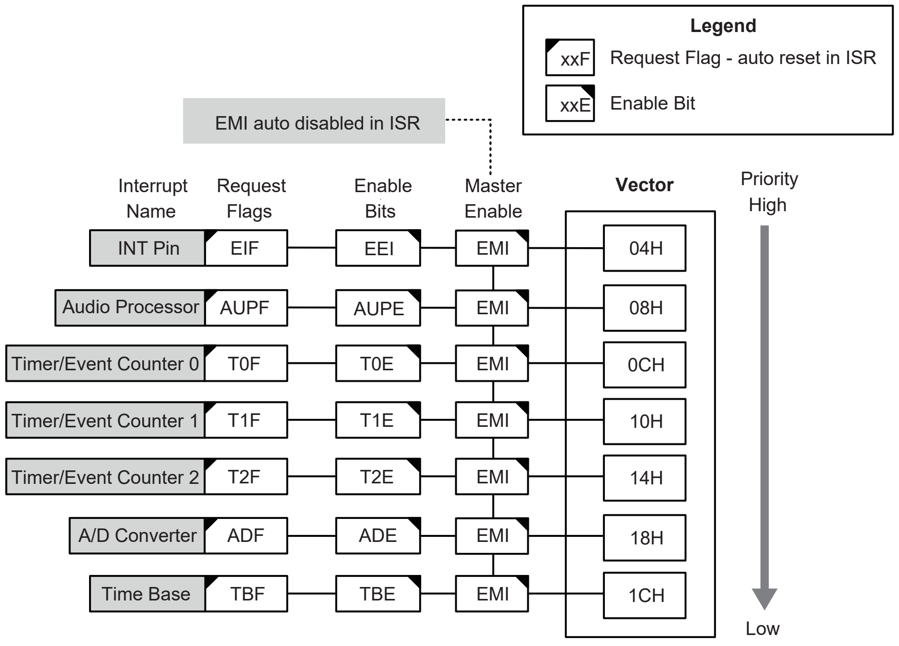
Bit 0 **T1E**: Timer/Event Counter 1 interrupt control
0: Disable
1: Enable

Interrupt Operation

A Timer/Event Counter overflow, a completion of A/D conversion or an active edge on the external interrupt pin, etc., will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector.

The instruction at this vector will usually be a JMP statement which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI instruction, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

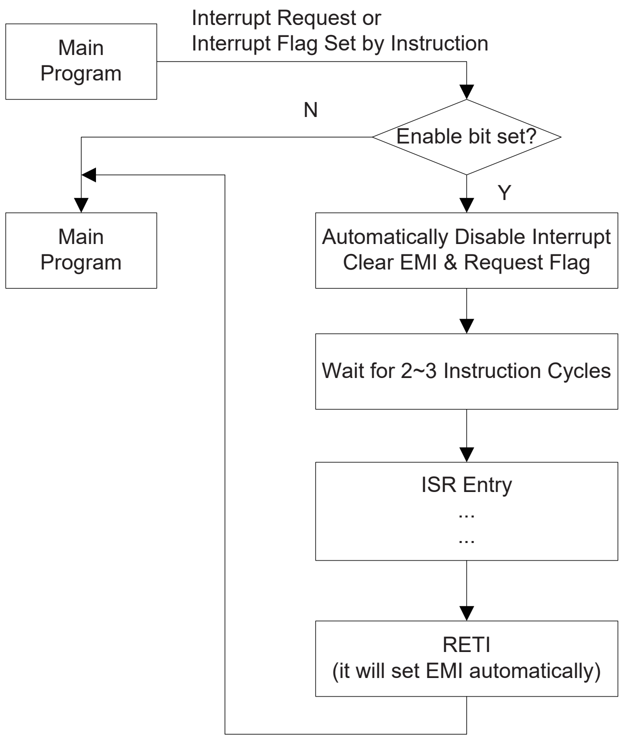
The various interrupt enable bits, together with their associated request flags, are shown in the following diagram with their order of priority.



Interrupt Structure

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded. If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full.

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


Main Program
Interrupt Request or
Interrupt Flag Set by Instruction
N
Enable bit set?
Y
Main Program
Automatically Disable Interrupt
Clear EMI & Request Flag
Wait for 2~3 Instruction Cycles
ISR Entry
...
...
RETI
(it will set EMI automatically)

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 converter complete	6	18H
Time Base Overflow	7	1CH

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

The INT pin function, which is pin-shared with PA3 by default option, can be remapped to PB4 by setting the PCFG bit in the CTRL0 register to "1".

For an external interrupt to occur, the global interrupt enable bit, EMI, and external interrupt enable bit, EEI, must first be set. An actual external interrupt will take place when the external interrupt request flag, EIF is set, a situation that will occur when an edge transition appears on the external INT pin. 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	Dual edge trigger

The external interrupt pin is pin-shared with the I/O pin and can only be used 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 set as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and a transition appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, 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 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.

A/D Converter Interrupt

For an A/D interrupt to occur, the global interrupt enable bit EMI and the corresponding interrupt enable bit ADE must be first set. An actual A/D interrupt will take place when the A/D converter request flag ADF is set, a situation that will occur when an A/D conversion process has completed. When the interrupt is enabled, the stack is not full and an A/D conversion process finishes execution, a subroutine call to the relevant A/D interrupt vector, will take place. When the interrupt is serviced, the A/D interrupt request flag ADF will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Time Base Interrupt

For a Time Base interrupt to occur, the global interrupt enable bit EMI and the corresponding interrupt enable bit TBE, must first be set. An actual Time Base interrupt will take place when the Time Base request flag TBF is set, a situation that will occur when the Time Base overflows. When the interrupt is enabled, the stack is not full and a Time Base overflow occurs a subroutine call to Time Base interrupt vector will take place. When the interrupt is serviced, the Time Base interrupt flag TBF will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Audio Processor Interrupt

The Audio Processor interrupt is initialised by setting the Audio Processor request flag, AUPF. When the Audio Processor returns a data, the SPIRQ bit in the SPICR register transits from "1" to "0", which will trigger an interrupt request. If the interrupt is enabled, the stack is not full and the AUPF is set, a subroutine call to the Audio Processor interrupt vector will occur. When the interrupt is serviced, the related interrupt request flag AUPF will be reset and the EMI bit will be automatically cleared to disable further interrupts.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the Sleep or Idle Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the Sleep or Idle Mode and its system oscillator is stopped, situations such as external edge transitions on the external interrupt pins may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the Sleep Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

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

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

Every interrupt has the capability of waking up the microcontroller when it is in 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 if the contents of the accumulator, status register or other registers are altered by the interrupt service program, which may corrupt the desired control sequence, then the contents should be saved in advance.

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

Low Voltage Detector – LVD

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

LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, V_{LVD2}~V_{LVD0}, are used to select one of eight fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

LVDC Register

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **LVDO**: LVD Output flag
0: No Low Voltage Detected
1: Low Voltage Detected

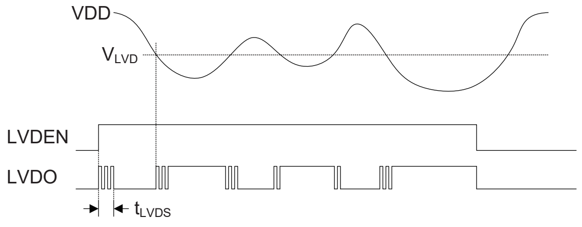
Bit 4 **LVDEN**: Low Voltage Detector Enable control
0: Disable
1: Enable

Bit 3 Unimplemented, read as "0"

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

LVD Operation

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



LVD Operation

SPI Function

The SPI interface is one method to communicate with Audio Processor. The command group types can be referred in chapter of MCU Interfacing.

SPI format and control register are shown below.

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

"x": unknown

Bit 7 **IEMC**: Mode control bit
0: External MCU mode (pad control) – PA5/PC4/PC5/PC6/PC7 pins are used as MISO/MOSI/ SPIRQ/SPICK/SPISS pin functions respectively
1: Internal mode (MCU control)
In the external MCU mode, the Audio Processor can be controlled by an external MCU via the SPI interface. SPISS/SPICK/MOSI cannot be read or written and MISO/ SPIRQ is read only when an SPIRQ falling edge occurs, it will not affect AUPF flag to generate an interrupt.

Bit 6 Unimplemented, read as "0"

Bit 5 **ERAM**: PB4/PB5/PB6/PB7 pins are used as Audio Processor GPIO0/1/2/3 respectively
0: Disable
1: Enable

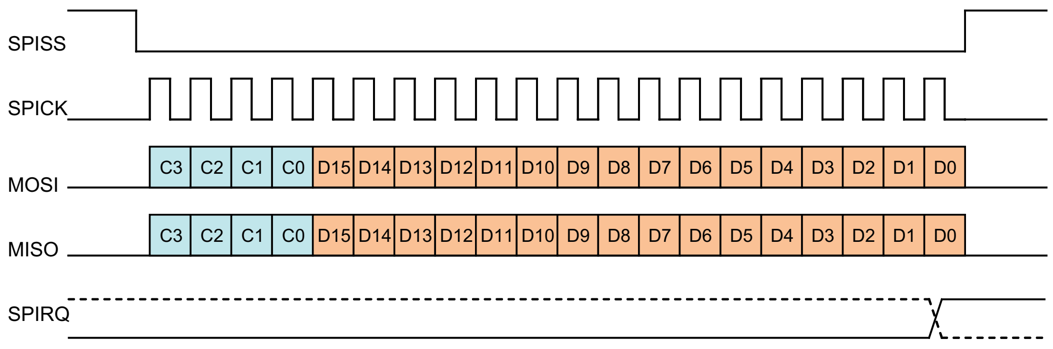
Bit 4 **SPISS**: Select Audio Processor
0: Select
1: Not select

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**: This bit is cleared to 0 by Audio processor when the Audio processor interrupt occurs. This bit will be set to 1 by Audio processor after the MCU has read command/ data successfully.


SPISS
SPICK
MOSI
MISO
SPIRQ
C3 C2 C1 C0 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

BEEP Function

This function uses application program to build a square wave on speaker 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

Bit 7~2 Unimplemented, read as "0"

Bit 1 **BEEP1**: output signal
0: Low
1: High
When the BEEP1 function is useful and is selected as Audio Processor source, the BEEP 1 pad is connected to Audio Processor.

Bit 0 **BEEP0**: output signal
0: Low
1: High
When the BEEP0 function is useful and is selected as Audio Processor source, the BEEP 0 pad is connected to speaker.
These two bits are output pin signals, setting or clearing BDR register data can modify the BEEP0/BEEP1 status. If the two functions are not useful, modifying the BEEP0/BEEP1 data will not affect the BEEP0/BEEP1 pad status.

Digital to Analog Converter – DAC

The device includes a 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 pins DAO0~DAO3 which are pin-shared with I/O pins 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

Bit 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

Bit 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

Bit 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

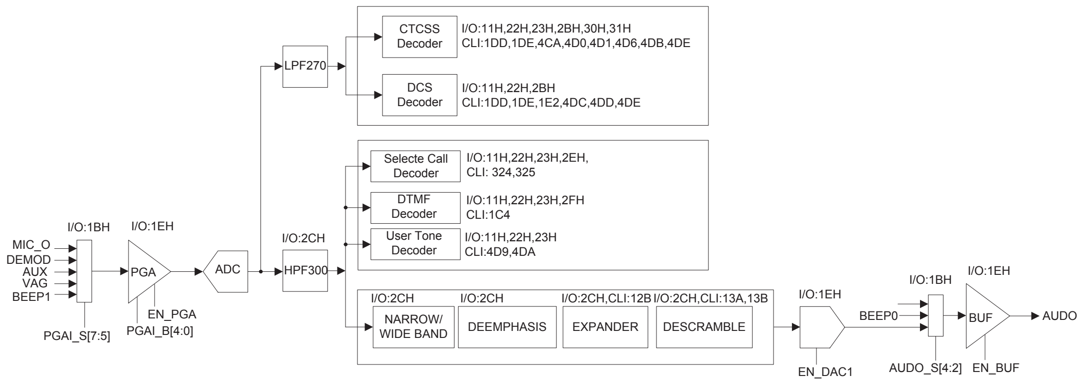
Bit 7~0 **DA07~DA00**: DAC0 output data

Audio Processor

The device includes an audio processor function for processing of signals received on the RF carrier. Aiming 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.



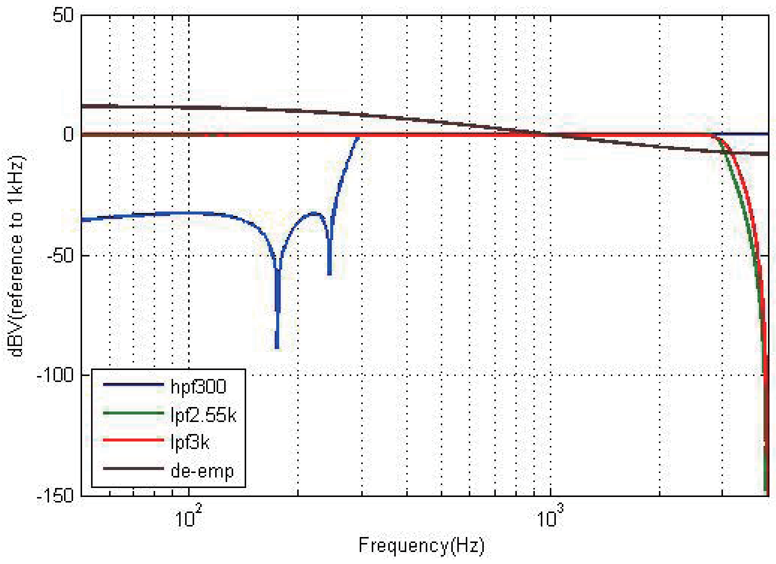
Note: “I/O: XXH” stands for I/O command, “CLI: xxx” stands for CLI command.

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 pin, AUDO. Individual filters are available for the following:

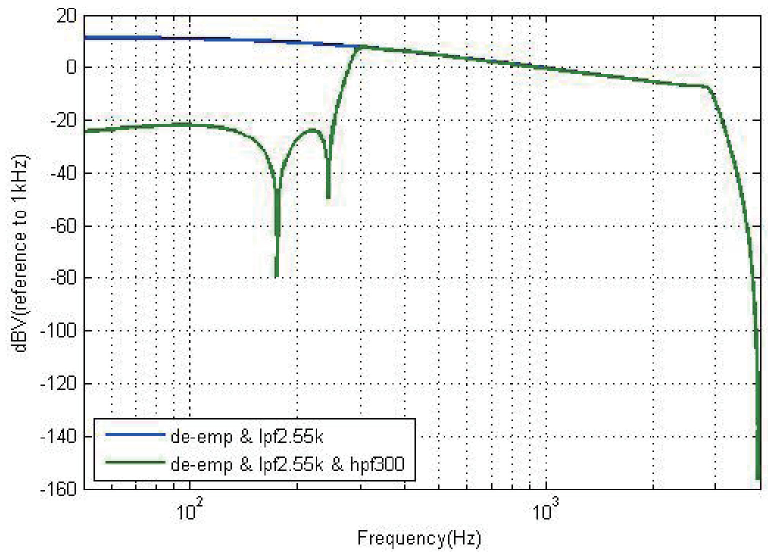
- 300Hz High Pass for rejection of sub audio signals
- 2.55kHz Low Pass for Narrow Bandwidth channel operation
- 3.0kHz Low Pass for Wide Bandwidth channel operation



Receiver Audio Filter Frequency Response

Receiver De-emphasis

The device includes a selectable de-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 Scrambler Inversion Frequency registers. 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 increase the signal dynamic range greatly.

Receiving and Decoding CTCSS or DCS Codes

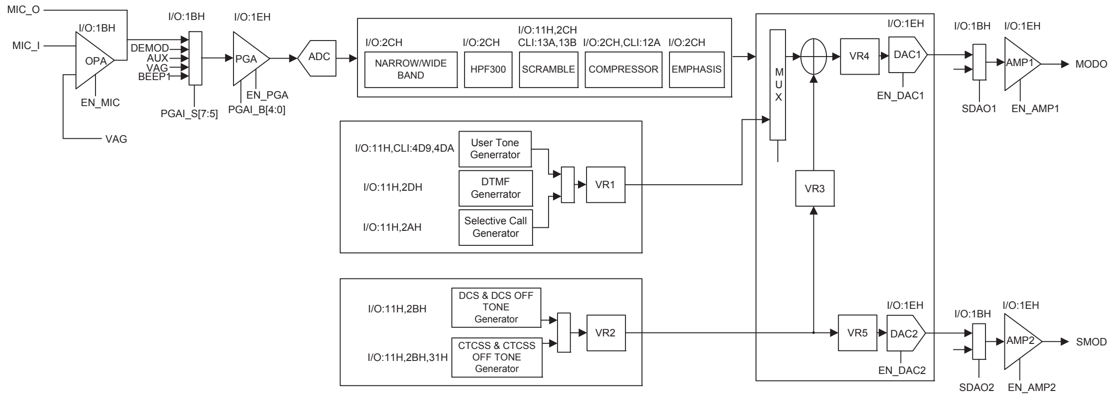
The device can accurately detect valid CTCSS tones rapidly which prevents losing the start of audio or data transmissions. It can also continuously monitor the detected tones to ensure low risk of false drop out. The DCS code is in NRZ format and is transmitted at a rate of 134.4±0.4bps. The device is 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 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 summarised into Selective call, DTMF and User-tone categories. For Selective call, the device 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 the related registers. This ensures that the device can remain compatible with all available tone systems in use. The device 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.

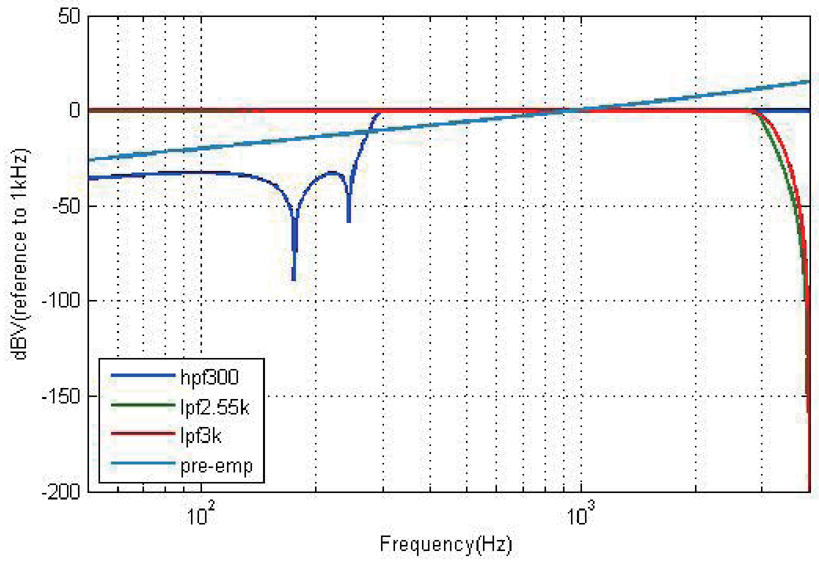


Note: "I/O: XXH" stands for I/O command, "CLI: xxx" stands for CLI command.

Transmitter Block Diagram

Transmitter Audio Filters

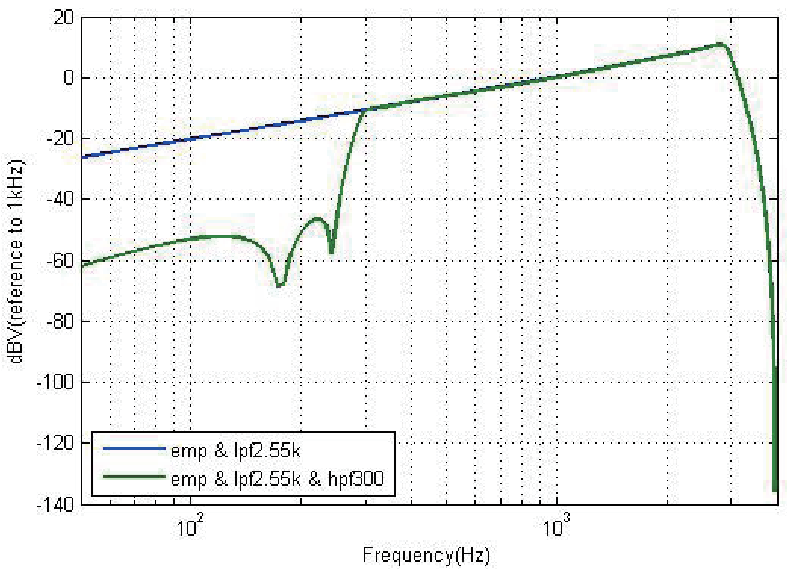
The transmitter audio filters include a 25kHz wide-band and a 12.5kHz narrow-band channel filter. The filter frequency responses are shown in the following figure.



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.



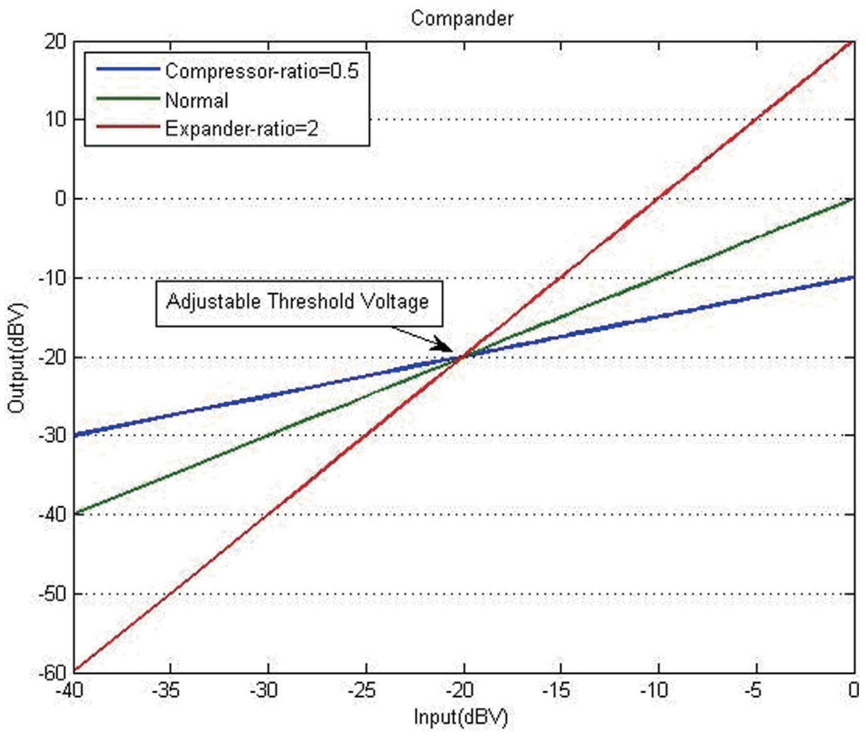
Transmitter Audio Frequency Pre-emphasis

Transmitter Scrambler

The device includes an optional frequency inversion scrambler for both transmit and receive modes. It 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 transmit and receive mode. It 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. In addition to 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. In addition to 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

Receiver

Voice Band Processing	Sub Tone	Audio Tone (DTMF/Selective Call/User-tone)
Basic Voice Filter	inv DCS, DCS/ User-defined DCS Decoder	DTMF Decoder
	CTCSS/ User-defined CTCSS Decoder	DTMF Decoder
	inv DCS, DCS/ User-defined DCS Decoder	Selective Call / User-tone Decoder
	CTCSS/ User-defined CTCSS Decoder	Selective Call / User-tone Decoder
Basic Voice Filter + De-emphasis	inv DCS, DCS/ User-defined DCS Decoder	DTMF Decoder
	CTCSS/ User-defined CTCSS Decoder	DTMF Decoder
	inv DCS, DCS/ User-defined DCS Decoder	Selective Call Decoder/ User-tone
	CTCSS/ User-defined CTCSS Decoder	Selective Call Decoder/ User-tone
Basic Voice Filter + Expander	inv DCS, DCS/ User-defined DCS Decoder	DTMF Decoder
	CTCSS/ User-defined CTCSS Decoder	DTMF Decoder
	inv DCS, DCS/ User-defined DCS Decoder	Selective Call Decoder/ User-tone
	CTCSS/ User-defined CTCSS Decoder	Selective Call Decoder/ User-tone
Basic Voice Filter + De-emphasis + Expander	inv DCS, DCS/ User-defined DCS Decoder	DTMF Decoder
	CTCSS/ User-defined CTCSS Decoder	DTMF Decoder
	inv DCS, DCS/ User-defined DCS Decoder	Selective Call Decoder/ User-tone
	CTCSS/ User-defined CTCSS Decoder	Selective Call Decoder/ User-tone
Basic Voice Filter + De-emphasis + Expander + De-Scramble	inv DCS, DCS/ User-defined DCS Decoder	DTMF Decoder
	CTCSS/ User-defined CTCSS Decoder	DTMF Decoder

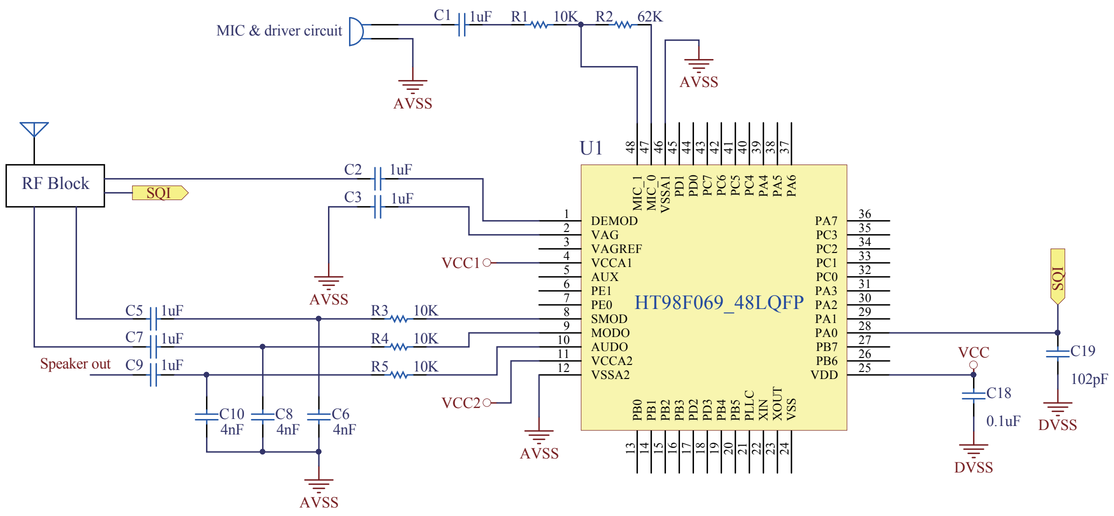
Transmitter

Voice Band Processing	Sub Tone	Audio Tone (DTMF/Selective Call/User-tone)
Basic Voice Filter	inv DCS, DCS/ User-defined DCS Encoder	DTMF Encoder
	CTCSS/ User-defined CTCSS Encoder	DTMF Encoder
	inv DCS, DCS/ User-defined DCS Encoder	Selective Call / User-tone Encoder
	CTCSS/ User-defined CTCSS Encoder	Selective Call / User-tone Encoder
Basic Voice Filter + Emphasis	inv DCS, DCS/ User-defined DCS Encoder	DTMF Encoder
	CTCSS/ User-defined CTCSS Encoder	DTMF Encoder
	inv DCS, DCS/ User-defined DCS Encoder	Selective Call / User-tone Encoder
	CTCSS/ User-defined CTCSS Encoder	Selective Call / User-tone Encoder
Basic Voice Filter + Compress	inv DCS, DCS/ User-defined DCS Encoder	DTMF Encoder
	CTCSS/ User-defined CTCSS Encoder	DTMF Encoder
	inv DCS, DCS/ User-defined DCS Encoder	Selective Call / User-tone Encoder
	CTCSS/ User-defined CTCSS Encoder	Selective Call / User-tone Encoder
Basic Voice Filter + Emphasis + Compress	inv DCS, DCS/ User-defined DCS Encoder	DTMF Encoder
	CTCSS/ User-defined CTCSS Encoder	DTMF Encoder
	inv DCS, DCS/ User-defined DCS Encoder	Selective Call / User-tone Encoder
	CTCSS/ User-defined CTCSS Encoder	Selective Call / User-tone Encoder
Basic Voice Filter + Emphasis + Compress + Scramble	inv DCS, DCS/ User-defined DCS Encoder	DTMF Encoder
	CTCSS/ User-defined CTCSS Encoder	DTMF Encoder

Note: The voice band and audio tone must be processed separately in the transmitter.

Audio Signal Routing

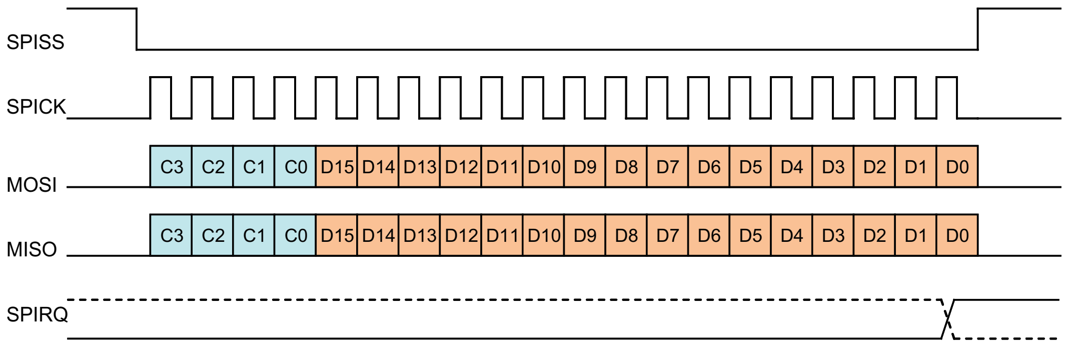
The device has a flexible audio signal routing input and output structure to route signals to and from the audio processor. The routing setup path is controlled using the path selection register which is described in a different section. The gain of the internal Programmable Gain Amplifier shown in the 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
SPICK
MOSI
C3 C2 C1 C0 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
MISO
C3 C2 C1 C0 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
SPIRQ

SPISS from the MCU: This pin is an input pin and is 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 of data D15~D0.

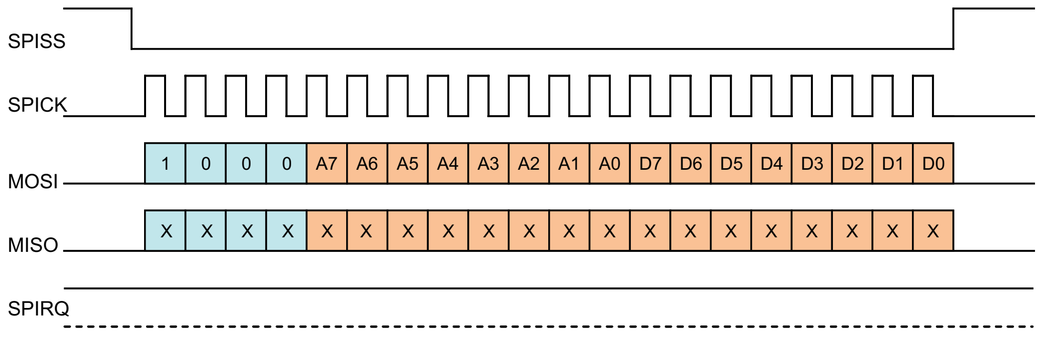
Command Groups

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.

I/O Command Group

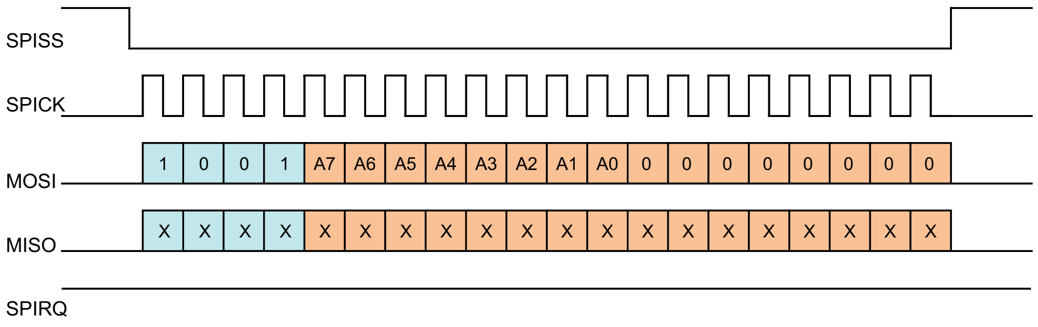
- **I/O Command Group Write – C [3:0] ="1000"**

The MCU will send 8 bits of I/O command address and 8 bits of I/O command data to Audio Processor. The host device sends I/O register write command to Audio Processor. The waveform is shown as follows.

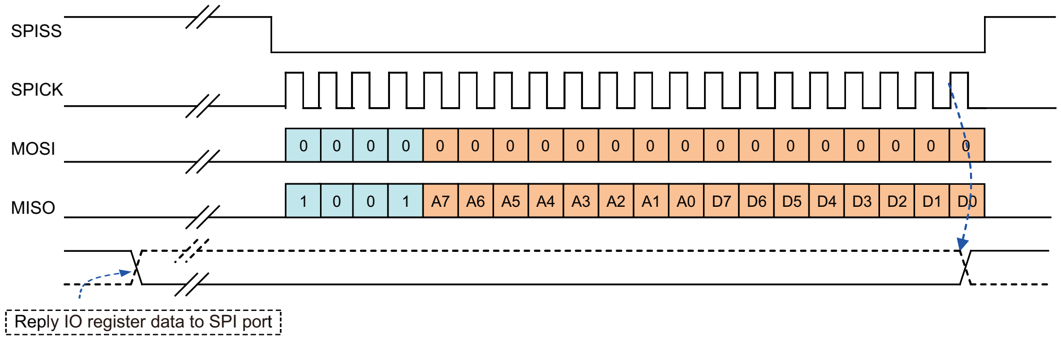

SPISS
SPICK
MOSI
1 0 0 0 A7 A6 A5 A4 A3 A2 A1 A0 D7 D6 D5 D4 D3 D2 D1 D0
MISO
X X X X X X X X X X X X X X X X X X X X
SPIRQ

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

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


SPISS
SPICK
MOSI
1 0 0 1 A7 A6 A5 A4 A3 A2 A1 A0 0 0 0 0 0 0 0 0
MISO
X X X X X X X X X X X X X X X X X X X X
SPIRQ

During the first cycle, the I/O command (I/O address) is sent to the Audio Processor

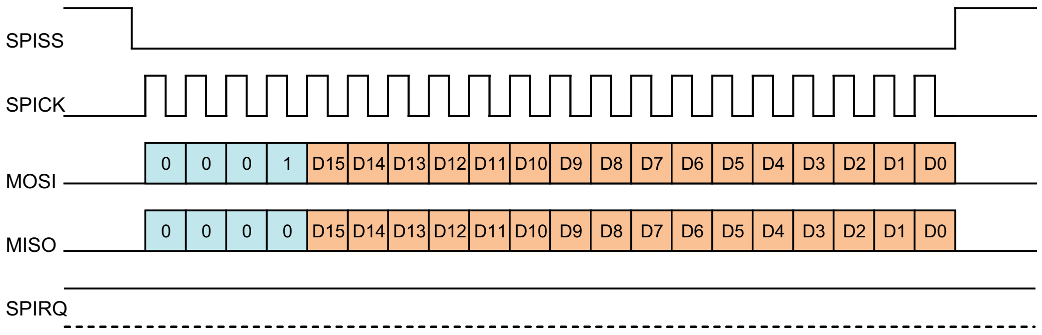

SPISS
SPICK
MOSI
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
MISO
1 0 0 1 A7 A6 A5 A4 A3 A2 A1 A0 D7 D6 D5 D4 D3 D2 D1 D0
Reply IO register data to SPI port

During the second cycle, the Audio Processor returns its I/O command data to the host

During I/O command R/W, user can issue SPI control waveform to Read/Write, without checking the SPIRQ status.

CLI Command Group

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


SPISS
SPICK
MOSI
0 0 0 1 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
MISO
0 0 0 0 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
SPIRQ

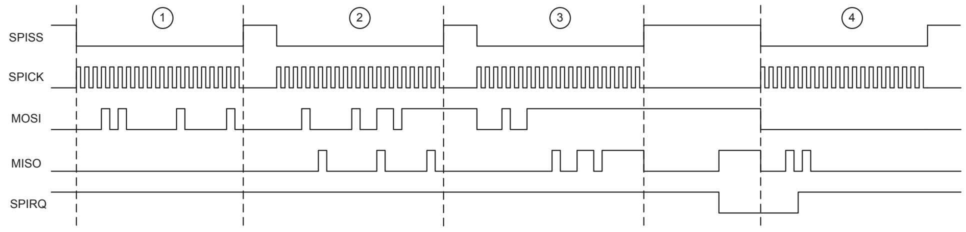
- **Write CMD Details**

Phase	CLI_CMD	Major	Minor	Multi	Length
①	4'b0001	4'b0100	4'b0000	4'b1000	4'b0010
②	4'b0001	Address[15:0]			
③	4'b0001	Data[15:0]			

Reply:

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

Note: When the audio processor replies with "14000", this means that the write data condition has been met.



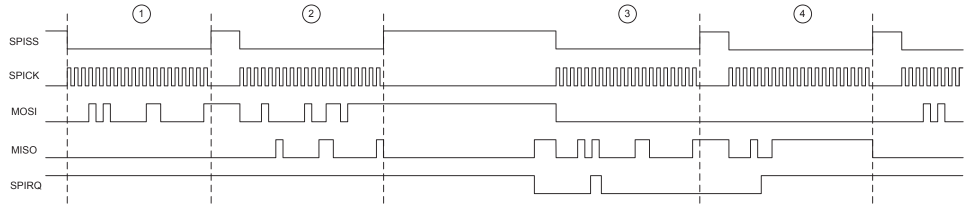
After CLI Command SPI write command phase 3, the SPIRQ will be low. User must check SPIRQ status to issue SPI read phase 4.The return data will be '14000'.

- **Read CMD Details**

Phase	CLI_CMD	Major	Minor	Multi	Length
①	4'b0001	4'b0100	4'b0001	4'b1000	4'b0001
②	4'b0001	Address[15:0]			

Reply:

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



After CLI Command SPI read command phase 1 and 2, the SPIRQ will be low. User must check SPIRQ status to issue SPI read phase 3 and 4. In phase 3, the HT98F069 will return '14181. In phase 4, the requested data will be returned.

I/O Command Group Summary

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

Address / Bit	7	6	5	4	3	2	1	0
00~0E	Reserved							
0F	Bit7~Bit0							
10	Reserved							
11	—	TRX 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
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

I/O Command Group Detail

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

Software Internal Reset Control – 0Fh Address

Bit	7	6	5	4	3	2	1	0
Name	Bit7~Bit0							

Bit 7~0 **00000000**: Software reset audio processor, the corresponding I/O command is 80F00
00000010: Software start audio processor, the corresponding I/O command is 80F02

Mode Control – 11h Address

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

Bit 7 Unimplemented, ignore this value

Bit 6~5 **TRX Operation Mode**: Audio processor operation mode selection
10: TX mode
11: RX mode

Bit 4~2 **In-band Tone Mode**: In-band tone signal selection
001: User Tones (to DA1)
010: Selective Call signal
100: DTMF
Others : All disable

Bit 1~0 **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				

Bit 7~5 Unimplemented, ignore this value

Bit 4~0 **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

Bit 7~5 **PGAI_S**: PGA input source selection
000: MICO
001: DEMOD
010: AUX
011: BEEP1
1xx: VAG

Bit 4~2 **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

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

Bit 0 **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	/EN_DAC2	/EN_DAC1	EN_AMP2	EN_AMP1	EN_BUF	EN_MIC	EN_PGA	1'b1

Bit 7 **/EN_DAC2**: DAC2 on/off control
0: Enable
1: Disable

Bit 6 **/EN_DAC1**: DAC1 on/off control
0: Enable
1: Disable

Bit 5 **EN_AMP2**: AMP2 on/off control
0: Disable
1: Enable

Bit 4 **EN_AMP1**: AMP1 on/off control
0: Disable
1: Enable

Bit 3 **EN_BUF**: Buffer on/off control
0: Disable
1: Enable

Bit 2 **EN_MIC**: MIC on/off control
0: Disable
1: Enable

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

Bit 0 **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

Bit 7 Unimplemented, ignore this value

Bit 6 **IRQ**: Interrupt request – (DTMF, Selective call, CTC, DCS, VOX) issued on the MCU PE
0: Disable
1: Enable

Bit 5 **DTMF INT**: DTMF event interrupt issue enable/disable selection
0: Disable
1: Enable

Bit 4 **Selective call INT**: Selective Call event interrupt issue enable/disable selection
0: Disable
1: Enable

Bit 3 **CTCSS INT**: CTCSS event interrupt issue enable/disable selection
0: Disable
1: Enable

Bit 2 **DCS INT**: DCS event interrupt issue enable/disable selection
0: Disable
1: Enable

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

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

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

Bit 7~6 Unimplemented, ignore this value

Bit 5 **DTMF Event**: DTMF code detection.
0: No DTMF event change detected
1: DTMF event status change has been detected

Bit 4 **Selective Call Event**: Selective Call detection.
0: No Selective call event change detected
1: Selective call event status cange has been detected

Bit 3 **CTCSS Event**: CTCSS code detection
0: No CTCSS event change detected
1: CTCSS event status change has been detected

Bit 2 **DCS Event**: DCS code detection
0: No DCS event change detected
1: DCS event status change has been detected

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

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

VOX Threshold Status – 29h Address

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

Bit 7~2 Unimplemented, ignore this value

Bit 1~0 **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 Call			

Bit 7~4 Unimplemented, ignore this value

Bit 3~0 **Selective Call**: Selective call tone number selection
0000 ~ 1111 are assigned as selective call number 0 ~ F respectively, 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						

Bit 7 **Sub_Inv**: DCS inverted select bit
In DCS mode:
0: Non-inverted DCS code
1: Inverted DCS code
In CTCSS mode: Don't care

Bit 6~0 **Sub Audio Tone**: Sub Audio Tone Channel Number
These bits are used to select DCS or CTCSS tone number, refer to 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

Bit 7 **EN_Scram**: Audio Scrambling on/off control
0: Disable
1: Enable

Bit 6 **EN_Comp**: Audio Compandor on/off control
0: Disable
1: Enable

Bit 5 **EN_Emp**: Audio Pre/De-emphasis on/off control
0: Disable
1: Enable

Bit 4 **EN_NBW**: Narrow band-width channel on/off control
0: Disable
1: Enable

Bit 3 **EN_WBW**: Wide band-width channel on/off control
0: Disable
1: Enable

Bit 2 **EN_HPF300**: Audio 300Hz HPF on/off control
0: Disable
1: Enable

Bit 1 **EN_VOX**: VOX detected on/off control
0: Disable
1: Enable (in TX mode)

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

DTMF Tone – 2Dh Address

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

Bit 7~4 Unimplemented, ignore this value

Bit 3~0 **DTMF Tone**: DTMF number selection
0000 ~ 1111 are assigned as DTMF number, refer to 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
852	1336	8	1000
852	1477	9	1001
941	1209	*	1010
941	1336	0	1011
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			

Bit 7~5 Unimplemented, ignore this value

Bit 4~0 **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 to check whether a tone exists or not. When a tone is detected, the calculation result will be automatically stored in the selective call finder register.

DTMF Finder – 2Fh Address

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

Bit 7~5 Unimplemented, ignore this value

Bit 4~0 **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 to check whether a tone exists or not. When a tone is detected, the calculation result will be automatically stored in the DTMF finder register.

Event2 Status – 30h Address

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

Bit 7~1 Unimplemented, ignore this value

Bit 0 **CTC_Anti-tone Event**: CTCSS Anti-tone detection
0: No CTCSS Anti-tone event change detected
1: CTCSS Anti-tone event status change has been detected

Event2 Control – 31h Address

Bit	7	6	5	4	3	2	1	0
Name			—				EN_CTC_Rx_Anti-tone	EN_CTC_Tx_Anti-tone

Bit 7~2 Unimplemented, ignore this value

Bit 1 **EN_CTC_Rx_Anti-tone**: CTCSS Rx Anti-tone on/off control
0: Disable
1: Enable

Bit 0 **EN_CTC_Tx_Anti-tone**: CTCSS Tx Anti-tone on/off control
When the CTCSS is in the Tx mode, the EN_CTC_Anti bit can enable the CTCSS anti-tone signal. This control bit changing from 0 to 1 or from 1 to 0 both have anti-tone effect (phase reversal 180^o).

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. – parameter 1															
013b	Scrambler Inv. Freq. – parameter 2															
013c ~ 0c3	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							
04cc	Reserved															
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															

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

Note: 1. When the different operation mode changes such as an Rx or Tx mode change, a soft_reset command "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 the RSSI signal detection, the RF module must take this RSSI detection response time into account.

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 replies, the command 1FFFF is successfully accepted

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															

Bit 15~0 **Tx Compandor Threshold**: Tx Compandor threshold setting

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 (250mv@3.3V). When the input signal is larger than the threshold voltage, the signal will be compressed otherwise the signal will be expanded. When the input signal is equal to the 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.

Rx 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															

Bit 15~0 **Rx Compandor Threshold**: Rx Compandor threshold setting

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 (250mv@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 received signal is equal to the 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 Setup – 0134h Address

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

Bit 15~3 **0**: Zero, must be cleared to zero.

Bit 2~0 **AGC Setup**: Setting AGC attack/released time

The corresponding bits influence the AGC attack and release time. These bits should be set before enabling the AGC function.

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															

Bit 15~0 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															

Bit 15~0 Scrambler Inversion Frequency – parameter2:
Scrambler inversion central frequency parameter2
Scrambler inversion frequency range: 2.6kHz ~ 3.4kHz. Default: 3.3kHz
Note: if the user needs to self-define scrambler inversion central frequency, 013a and 013b addresses 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															

Bit 15~0 DTMF Power Threshold: DTMF power threshold detector
This register is used to setup the minimum detected DTMF signal level. The default value is Vth=0xff9b (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															

Bit 15~0 Off Tone Accepted Voltage parameter [15:0]:
User defines the acknowledged Off Tone power bit 15 ~ bit 0. The default value is 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															

Bit 15~0 Off Tone Released Voltage parameter [15:0]:
User defines the deceased Off Tone power bit 15 ~ bit 0. The default value is 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															

Bit 15~0 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 is 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															

Bit 15~0 Selective Call/User-defined Tone Accepted Voltage parameter [15:0]:
User defines the acknowledged selective call/user-defined tone power bit 15 ~ bit 0. The default value is 0xff60 (72mV-peak@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															

Bit 15~0 Selective Call/User-defined Tone Released Voltage parameter [15:0]:
User defines the deceased selective call/user-defined tone power bit 15 ~ bit 0. The default value is 0xff30 (25mV-amplitude@3.3V).

CTCSS Power Released 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															

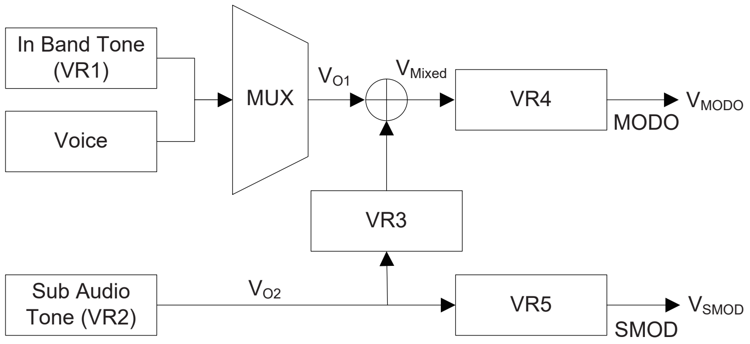
Bit 15~0 CTCSS Power Released Threshold Voltage
This register is used to set the minimum CTCSS signal level that is to be released. The default value is 0xff05 (15mV-peak @3.3V)

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							

Bit 15~8 **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: $V_{O1}= V_{voice} \times 1.1$ if bit2~bit4="000" in the I/O command address 11, otherwise $V_{O1}=0.75V_{DD} \times (VR1)/256$; where V_{voice} is the ADC input voltage.
Note: The actual output voltage will be a little degraded due to the DAC1 filter circuit.

Bit 7~0 **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: $V_{O2}= V_{DD} \times (VR2)/256$
Note: The actual output voltage will be a little degraded due to the DAC2 filter circuit.



Modulation Path Block Diagram

Mixed Gain – 04D2h Address

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

Bit 15~8 **0**: Zero, must be cleared to zero.

Bit 7~0 **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: $V_{mixed} = V_{O1} \times (512\text{-}VR3)/512 + V_{O2} \times (VR3/512)$
Note: V_{O1} can be output from the In-band tone tuned by VR1 or Voice

MODO Gain – 04D3h Address

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

Bit 15~10 **0**: Zero, must be cleared to zero.
Bit 9~0 **VR4**: MODO Gain Adjustment
This register is used to adjust MODO pin output gain levels.
Refer to the "Modulation Path Block Diagram" for optimal parameter details.
The VR4 setting is based on the formula: $V_{MODO} = V_{mixed} \times (VR4/1024)$
Note: The actual output voltage will be a little degraded due to the DAC1 filter circuit.

SMOD Gain – 04D5h Address

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

Bit 15~10 **0**: Zero, must be cleared to zero.
Bit 9~0 **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: $V_{SMOD} = V_{O2} \times VR5/1024$
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															

Bit 15~0 **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 bit 0 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															

Bit 15~0 **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, bit 0 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															

Bit 15~0 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															

Bit 15~0 User defined CTCSS tone parameter2
Refer to the application notes for detailed register setting.
User Defined CTCSS Tone frequency range: 67Hz ~ 275Hz.
Note: If it is necessary to define a CTCSS tone by the user, registers 04d0 and 04d1 must be used at the same time.

CTCSS Frequency Accept Variance – 04D6h Address

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

Bit 15~2 Unimplemented, ignore this value

Bit 1~0 **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															

Bit 15~0 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															

Bit 15~0 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 the user, 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															

Bit 15~0 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 the 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]															

Bit 15~0 **User DCS parameter[15:0]**: User defined 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]						

Bit 15~7 **0**: Zero, must be cleared to zero.

Bit 6~0 **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 a DCS tone by the user, registers 04dc and 04dd must be used at the same time. If 4dc and 4dd register bits are set to 1, the output is 0, if the bits are set to 0, the output is 1.

Note:
The method to fill out User DCS Codes (23 bits):
Step1: Reverse the bit order of User DCS Codes.
Step2: After step 1, do the 1's complement of reversed User DCS Codes.
Step3: Fill out the address 04DCH with the 1's complement of reversed User DCS Codes[15:0]
Fill out the address 04DDH with the 1's complement of reversed User DCS Codes[22:16]
For example:
DCS Codes Number = 036, the value is 0BE81EH (23 bits),
0BE81EH (Hex) = 000 1011 1110 1000 0001 1110B (Bin)
Step1: Reverse the bit order of User DCS Codes[22:0]
= 0111 1000 0001 0111 1101 000 B
Step2: Do the 1's complement of reversed User DCS Codes[22:0]
= 10000111111010000010111 B
= 4 3 F 4 1 7 H
Step3: Fill out the address 04DCH with the 1's complement of reversed User DCS Codes[15:0]
= F417 (Hex)
Fill out the address 04DDH with the 1's complement of reversed User DCS Codes[22:16]
= 43 (Hex)
Please refer to the application note for more details.

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															

Bit 15~0 **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).

MODO 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													

Bit 15~14	**0**: Zero, must be filled with b"0"
Bit 13~0	**Soft Limiter**: Tx amplitude limiter threshold to VCO In Tx mode, this limiter will constrain the MODO 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															
04E3	Selective_Call_1: parameter2															
04E4	Selective_Call_2: parameter1															
04E5	Selective_Call_2: parameter2															
04E6	Selective_Call_3: paramete1r															
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 allow the user to define parameters for selective call tones from tone 0 to 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, which are put in two registers in the audio processor. For details regarding the 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

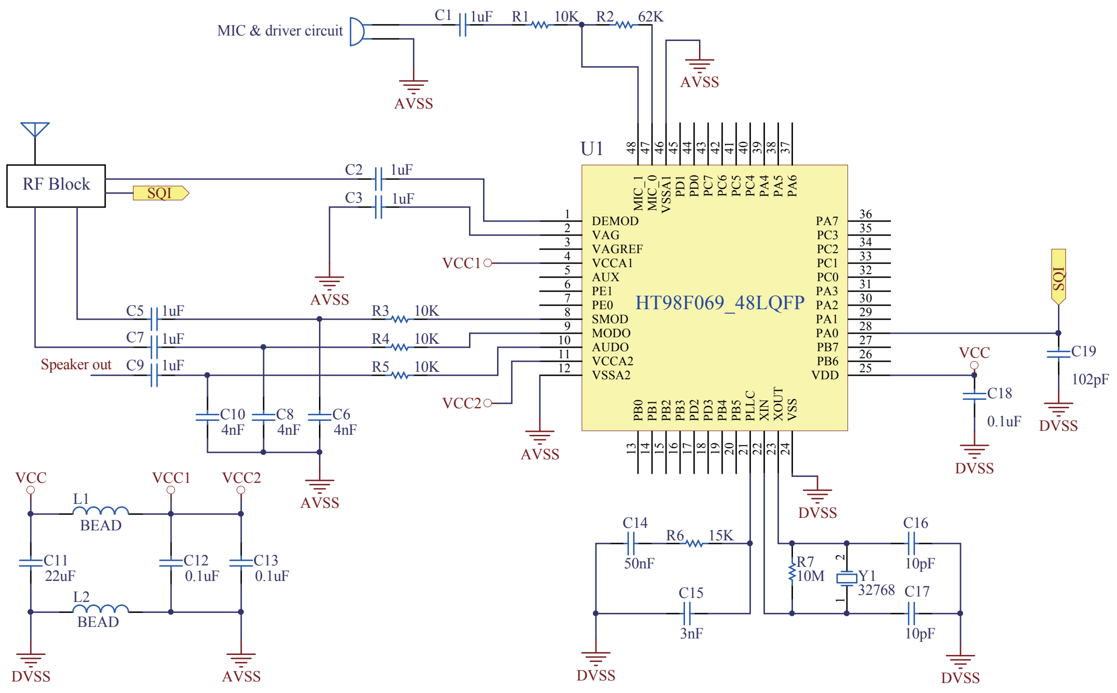
Selective Call Reference Sets Table

Configuration Options

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

No.	Options
Reset Pin Option	
1	PA7 / $\overline{RES}$ pin option: 1. $\overline{RES}$ pin 2. I/O pin
Watchdog Timer Option	
2	Watchdog Timer function: Always enable By software control
3	Watchdog Timer clock source: $f_{LIRC}/2$ $f_{SYS}/4$ f_{LXT}

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

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

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

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

Instruction Definition

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

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

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

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

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

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

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

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

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None

CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None

CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None

CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF

CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF

CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF

CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	$[m] \leftarrow \overline{[m]}$
Affected flag(s)	Z

CPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC ← $\overline{[m]}$
Affected flag(s)	Z

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

DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [m] − 1
Affected flag(s)	Z

DECA [m]	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] − 1
Affected flag(s)	Z

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

Operation	TO ← 0
	PDF ← 1

Affected flag(s)	TO, PDF

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

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

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

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

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

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

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

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

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

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

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

RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None

RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None

RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← [m].7
Affected flag(s)	None

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

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

Affected flag(s)	None

RLC [m]	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← C C ← [m].7
Affected flag(s)	C

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

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

Affected flag(s)	C

RR [m]	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← [m].0
Affected flag(s)	None

RRA [m]	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s)	None

RRC [m]	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← C C ← [m].0
Affected flag(s)	C

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

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

Affected flag(s)	C

SBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m] − C
Affected flag(s)	OV, Z, AC, C

SBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m] − C
Affected flag(s)	OV, Z, AC, C

SDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] − 1 Skip if [m]=0
Affected flag(s)	None

SDZA [m]	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	ACC ← [m] − 1 Skip if ACC=0
Affected flag(s)	None

SET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	[m] ← FFH
Affected flag(s)	None

SET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	[m].i ← 1
Affected flag(s)	None

SIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] + 1 Skip if [m]=0
Affected flag(s)	None

SIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] + 1 Skip if ACC=0
Affected flag(s)	None

SNZ [m].i	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None

SUB A,[m]	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m]
Affected flag(s)	OV, Z, AC, C

SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m]
Affected flag(s)	OV, Z, AC, C

SUB A,x	Subtract immediate data from ACC
Description	The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − x
Affected flag(s)	OV, Z, AC, C

SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0 ↔ [m].7~[m].4
Affected flag(s)	None

SWAPA [m]	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3~ACC.0 ← [m].7~[m].4 ACC.7~ACC.4 ← [m].3~[m].0
Affected flag(s)	None

SZ [m]	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m]=0
Affected flag(s)	None

SZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] Skip if [m]=0
Affected flag(s)	None

SZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if [m].i=0
Affected flag(s)	None

TABRD [m]	Read table (specific page) to TBLH and Data Memory
Description	The low byte of the program code (specific page) addressed by the table pointer pair (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

TABRDC [m]	Read table (current page) to TBLH and Data Memory
Description	The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

TABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

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

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

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

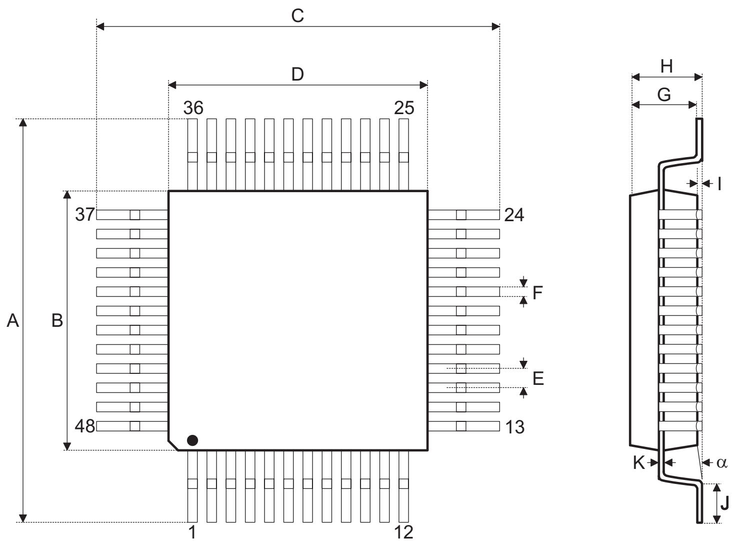
Package Information

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

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

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

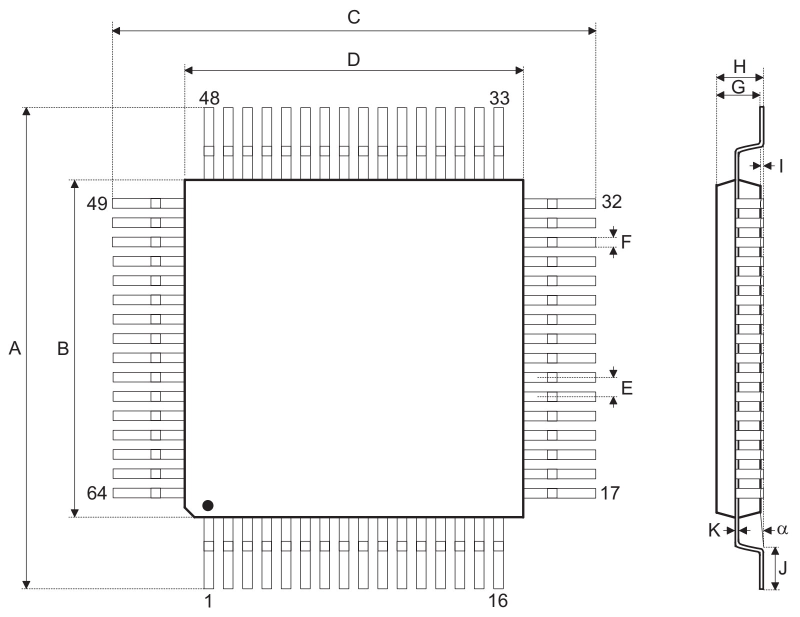
48-pin LQFP (7mm×7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.020 BSC	—
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.50 BSC	—
F	0.17	0.22	0.27
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

64-pin LQFP (7mm×7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.016 BSC	—
F	0.005	0.007	0.009
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.40 BSC	—
F	0.13	0.18	0.23
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°