Technical Document

- Application Note
  - HA0075E MCU Reset and Oscillator Circuits Application Note

Features

CPU Features

- Operating voltage:
  f_{SYS}= 32768Hz: 2.2V~5.5V
  f_{SYS}= 4MHz: 2.2V~5.5V
  f_{SYS}= 8MHz: 3.0V~5.5V
  f_{SYS}= 12MHz: 4.5V~5.5V
- Up to 0.33μs instruction cycle with 12MHz system clock at V_{DD}= 5V
- Idle/Sleep mode and wake-up functions to reduce power consumption
- Oscillator types:
  External 32768Hz Crystal -- LXT
  External RC -- ERC
  Internal 4/8/12MHz RC -- HIRC
  External high frequency crystal -- HXT
  Internal 32kHz RC -- LIRC
- Four operational modes: Normal, Slow, Idle, Sleep
- Fully integrated internal 4MHz, 8MHz and 12MHz oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- Up to 12-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- OTP Program Memory: 2K×14 ~ 32K×16
- RAM Data Memory: 128×8 ~ 2304×8 Bits
- Watchdog Timer function
- Up to 50 bidirectional I/O lines
- 8 channel 12-bit ADC
- Up to 4 channel 12-bit PWM
- Software controlled 4-SCOM lines LCD driver with 1/2 bias
- Multiple pin-shared external interrupts
- Up to three 8-bit programmable Timer/Event Counter with overflow interrupt and prescaler
- Up to one 16-bit programmable Timer/Event Counter with overflow interrupt
- Serial Interfaces Module - SIM for SPI or I^2C
- Time-Base functions
- Low voltage reset function
- Low voltage detect function
- PFD/Buzzer for audio frequency generation
- 12-bit Audio DAC output
- Wide range of available package types

General Description

These TinyPower™ A/D Type 8-bit high performance RISC architecture microcontrollers are specifically, designed for applications that interface directly to analog signals. The devices include an integrated multi-channel Analog to Digital Converter, Pulse Width Modulation and and DAC outputs.

With their fully integrated SPI and I^2C functions, designers are provided with a means of easy communication with external peripheral hardware. The benefits of integrated A/D, PWM and DAC functions, in addition to low power consumption, high performance, I/O flexibility and low-cost, provides the device with the versatility for a wide range of products in the home appliance and industrial application areas. Some of these products could include electronic metering, environmental monitoring, handheld instruments, electronically controlled tools, motor driving in addition to many others.

The unique Holtek TinyPower technology also gives the devices extremely low current consumption characteristics, an extremely important consideration in the present trend for low power battery powered applications. The usual Holtek MCU features such as power down and wake-up functions, oscillator options, programmable frequency divider, etc. combine to ensure user applications require a minimum of external components.

Selection Table

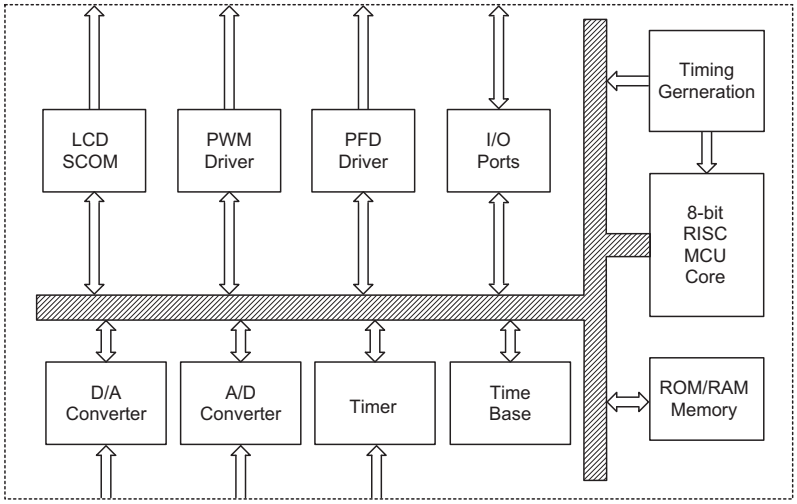
Part No.	Program Memory	Data Memory	I/O	Timer		Time Base	HIRC (MHz)	RTC (LXT)	LCD SCOM	A/D	D/A	PWM	Stack	Package
				8-bit	16-bit									
HT56R22	2K×14	128×8	22	2	—	1	4/8/12	√	4	12-bit×8	12-bit×1	12-bit×3	6	16DIP/NSOP/SSOP 20DIP/SOP/SSOP 24SKDIP/SOP/SSOP
HT56R23	4K×15	256×8	42	2	1	1	4/8/12	√	4	12-bit×8	12-bit×1	12-bit×4	12	28SKDIP/SOP/SSOP 44QFP
HT56R24	8K×16	640×8	42	2	1	1	4/8/12	√	4	12-bit×8	12-bit×1	12-bit×4	12	28SKDIP/SOP/SSOP 44QFP
HT56R25	16K×16	1152×8	50	3	1	1	4/8/12	√	4	12-bit×8	12-bit×1	12-bit×4	12	28SKDIP/SOP 28SSOP(209mil) 44/52QFP
HT56R26	32K×16	2304×8	50	3	1	1	4/8/12	√	4	12-bit×8	12-bit×1	12-bit×4	12	28SKDIP/SOP 28SSOP(209ml) 44/52QFP

Note: 1. The devices are only available in OTP versions.

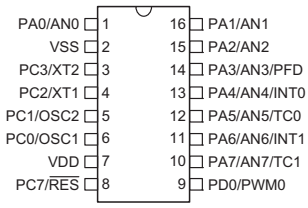
2. For devices that exist in more than one package formats, the table reflects the situation for the larger package.

Block Diagram

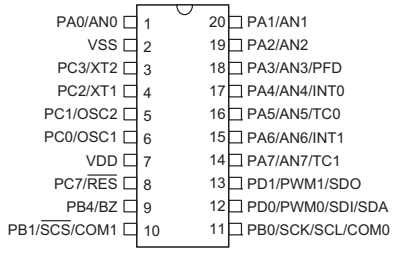
The following block diagram illustrates the main functional blocks.



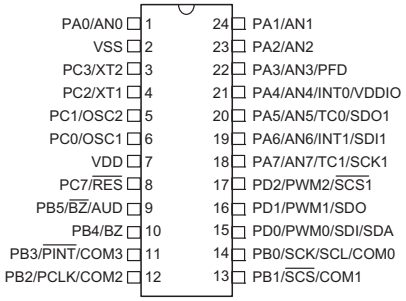
Pin Assignment



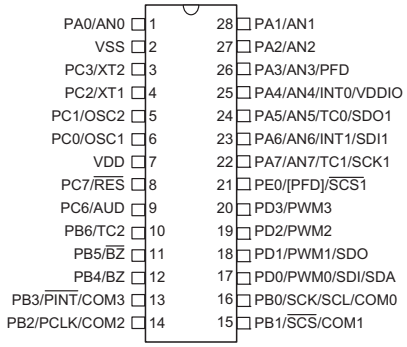
HT56R22
16 DIP-A/NSOP-A



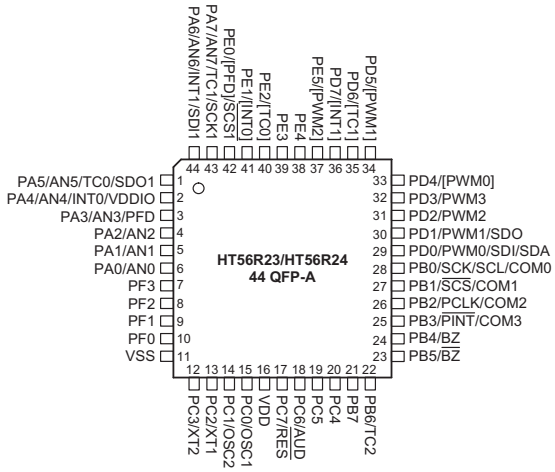
HT56R22
20 DIP-A/SOP-A/SSOP-A



HT56R22
24 SKDIP-A/SOP-A/SSOP-A

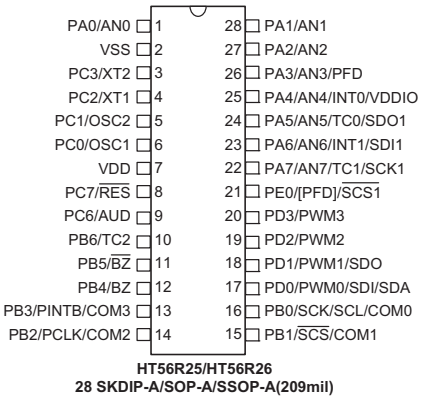


HT56R23/HT56R24
28 SKDIP-A/SOP-A/SSOP-A

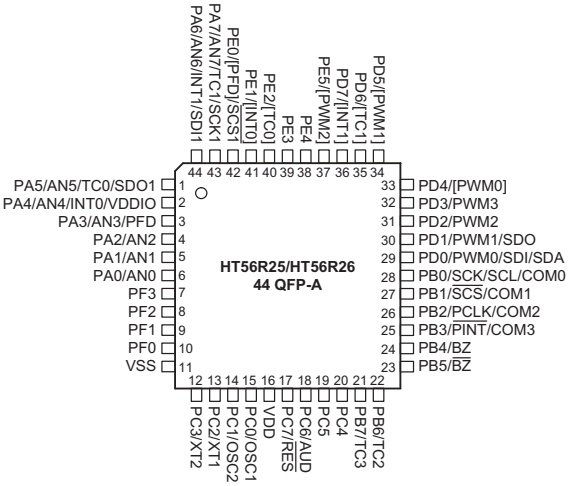


HT56R23/HT56R24
44 QFP-A

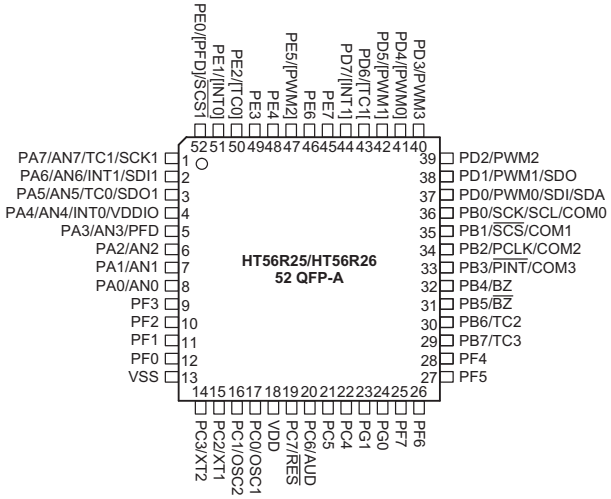
Note: Bracketed pin names indicate non-default pinout remapping locations.



HT56R25/HT56R26
28 SKDIP-A/SOP-A/SSOP-A(209mil)



HT56R25/HT56R26
44 QFP-A



HT56R25/HT56R26
52 QFP-A

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

Pin Description

HT56R22

Pin Name	Function	OPT	I/T	O/T	Description
PA0/AN0	PA0	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN0	ADCR	AN	—	A/D channel 0
PA1/AN1	PA1	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN1	ADCR	AN	—	A/D channel 1
PA2/AN2	PA2	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN2	ADCR	AN	—	A/D channel 2
PA3/PFD/AN3	PA3	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PFD	CO PINMAP	ST	—	PFD output
	AN3	ADCR	AN	—	A/D channel 3
PA4/INT0/AN4/VDDIO	PA4	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT0	PINMAP	ST	—	External interrupt 0 input
	AN4	ADCR	AN	—	A/D channel 4
	VDDIO	CO	PWR	—	VDDIO power input
PA5/TC0/SDO1/AN5	PA5	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TC0	PINMAP	ST	—	External Timer 0 clock input
	SDO1	CO SPICTL0	—	CMOS	SPI1 serial data output
	AN5	ADCR	AN	—	A/D channel 5
PA6/INT1/SDI1/AN6	PA6	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT1	PINMAP	ST	—	External interrupt 1 input
	SDI1	CO SPICTL0	—	CMOS	SPI1 serial data input
	AN6	ADCR	AN	—	A/D channel 6
PA7/TC1/SCK1/AN7	PA7	PAWK	ST	CMOS	General purpose I/O. Register enabled wake-up.
	TC1	PINMAP	ST	—	External Timer 1 clock input
	SCK1	CO SPICTL0	ST	CMOS	SPI1 serial clock input or output
	AN7	ADCR	AN	—	A/D channel 7
PB0/SCK0/SCL/SCOM0	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SCK0	CO SIMCTL0	ST	CMOS	SPI1 serial clock input or output
	SCL	CO SIMCTL0	ST	—	I^2C serial clock input
	SCOM0	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM

Pin Name	Function	OPT	I/T	O/T	Description
PB1/$\overline{SCS0}$/SCOM1	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{SCS0}$	CO SIMCTL0	ST	CMOS	SPI0 select control pin
	SCOM1	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB2/PCLK/SCOM2	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PCLK	CO SIMCTL0	—	CMOS	Peripheral clock output
	SCOM2	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB3/$\overline{PINT}$/SCOM3	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{PINT}$	—	ST	—	Peripheral interrupt input, falling edge trigger
	SCOM3	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB4/BZ	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	BZ	CO	—	—	Buzzer output
PB5/$\overline{BZ}$/AUD	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{BZ}$	CO	—	—	Buzzer bar output
	AUD	CO DACTRL		AO	Audio output
PC0/OSC1	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OSC1	CO	AN	—	Oscillator pin
PC1/OSC2	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OSC2	CO	—	CMOS	Oscillator pin
PC2/XT1	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	XT1	CO	—	CMOS	Oscillator pin
PC3/XT2	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	XT2	CO	—	LXT	Oscillator pin
PC7/$\overline{RES}$	PC7	PCPU	ST	NMOS	General purpose I/O. Register enabled pull-up.
	$\overline{RES}$	CO	ST	—	Reset input
PD0/PWM0/SDI0/SDA	PD0	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM0	PINMAP	—	CMOS	PWM0 output
	SDI0	CO SIMCTL0	ST	—	SPI0 serial data input
	SDA	CO SIMCTL0	ST	OD	I^2C data input or output
PD1/PWM1/SDO0	PD1	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM1	PINMAP	—	CMOS	PWM1 output
	SDO0	CO SIMCTL0	—	CMOS	SPI0 serial data output
PD2/PWM2/$\overline{SCS1}$	PD2	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM2	PINMAP	—	CMOS	PWM2 output
	$\overline{SCS1}$	CO SPICTL0	ST	CMOS	SPI1 chip select pin
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground

Note: I/T: Input type; O/T: Output type
OPT: Optional by configuration option (CO) or register option
PWR: Power; CO: Configuration option
ST: Schmitt Trigger input; CMOS: CMOS output; AN: analog input
SCOM= software controlled LCD COM
HXT: High frequency crystal oscillator
LXT: Low frequency crystal oscillator

HT56R23/HT56R24

Pin Name	Function	OPT	I/T	O/T	Description
PA0/AN0	PA0	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN0	ADCR	AN	—	A/D channel 0
PA1/AN1	PA1	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN1	ADCR	AN	—	A/D channel 1
PA2/AN2	PA2	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN2	ADCR	AN	—	A/D channel 2
PA3/PFD/AN3	PA3	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PFD	CO PINMAP	ST	—	PFD output
	AN3	ADCR	AN	—	A/D channel 3
PA4/INT0/AN4/VDDIO	PA4	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT0	PINMAP	ST	—	External interrupt 0 input
	AN4	ADCR	AN	—	A/D channel 4
	VDDIO	CO	PWR	—	VDDIO power input
PA5/TC0/SDO1/AN5	PA5	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TC0	PINMAP	ST	—	External Timer 0 clock input
	SDO1	CO SPICTL0	—	CMOS	SPI1 serial data output
	AN5	ADCR	AN	—	A/D channel 5
PA6/INT1/SDI1/AN6	PA6	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT1	PINMAP	ST	—	External interrupt 1 input
	SDI1	CO SPICTL0	—	CMOS	SPI1 serial data input
	AN6	ADCR	AN	—	A/D channel 6
PA7/TC1/SCK1/AN7	PA7	PAWK	ST	CMOS	General purpose I/O. Register enabled wake-up.
	TC1	PINMAP	ST	—	External Timer 1 clock input
	SCK1	CO SPICTL0	ST	CMOS	SPI1 serial clock input or output
	AN7	ADCR	AN	—	A/D channel 7

Pin Name	Function	OPT	I/T	O/T	Description
PB0/SCK0/SCL/SCOM0	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SCK0	CO SIMCTL0	ST	CMOS	SPI1 serial clock input or output
	SCL	CO SIMCTL0	ST	—	I^2C serial clock input
	SCOM0	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB1/$\overline{SCS0}$/SCOM1	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{SCS0}$	CO SIMCTL0	ST	CMOS	SPI0 select control pin
	SCOM1	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB2/PCLK/SCOM2	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PCLK	CO SIMCTL0	—	CMOS	Peripheral clock output
	SCOM2	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB3/$\overline{PINT}$/SCOM3	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{PINT}$	—	ST	—	Peripheral interrupt input, falling edge trigger
	SCOM3	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB4/BZ	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	BZ	CO	—	—	Buzzer output
PB5/$\overline{BZ}$	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{BZ}$	CO	—	—	Buzzer bar output
PB6/TC2	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TC2	—	ST	—	External Timer 2 clock input
PB7	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
PC0/OSC1	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OSC1	CO	AN	—	Oscillator pin
PC1/OSC2	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OSC2	CO	—	CMOS	Oscillator pin
PC2/XT1	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	XT1	CO	—	CMOS	Oscillator pin
PC3/XT2	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	XT2	CO	—	LXT	Oscillator pin
PC4~PC5	PCn	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
PC6/AUD	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AUD	CO DACTRL	—	AO	Audio output
PC7/$\overline{RES}$	PC7	PCPU	ST	NMOS	General purpose I/O. Register enabled pull-up.
	$\overline{RES}$	CO	ST	—	Reset input
PD0/PWM0/SDI0/SDA	PD0	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM0	PINMAP	—	CMOS	PWM0 output
	SDI0	CO SIMCTL0	ST	—	SPI0 serial data input
	SDA	CO SIMCTL0	ST	OD	I^2C data input or output

Pin Name	Function	OPT	I/T	O/T	Description
PD1/PWM1/SDO0	PD1	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM1	PINMAP	—	CMOS	PWM1 output
	SDO0	CO SIMCTL0	—	CMOS	SPI0 serial data output
PD2/PWM2	PD2	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM2	PINMAP	—	CMOS	PWM2 output
PD3/PWM3	PD3	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM3	—	—	CMOS	PWM3 output
PD4/PWM0	PD4	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM0	PINMAP	—	CMOS	PWM0 output
PD5/PWM1	PD5	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM1	PINMAP	—	CMOS	PWM1 output
PD6/TC1	PD6	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TC1	PINMAP	ST	—	External Timer 1 clock input
PD7/INT1	PD7	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	INT1	PINMAP	ST	—	External interrupt 1 input
PE0/PFD/$\overline{SCS1}$	PE0	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PFD	CO PINMAP	—	CMOS	PFD output
	$\overline{SCS1}$	CO SPICTL0	ST	CMOS	SPI1 chip select pin
PE1/INT0	PE1	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	INT0	PINMAP	ST	—	External interrupt 0 input
PE2/TC0	PE2	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TC0	PINMAP	ST	—	External Timer 0 clock input
PE3~PE4	PEn	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
PE5/PWM2	PE5	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM2	PINMAP	—	CMOS	PWM2 output
PF0~PF3	PFn	PFPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
PG0~PG1	PGn	PGPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground

Note: I/T: Input type; O/T: Output type
OPT: Optional by configuration option (CO) or register option
PWR: Power; CO: Configuration option
ST: Schmitt Trigger input; CMOS: CMOS output; AN: analog input
SCOM: Software controlled LCD COM
HXT: High frequency crystal oscillator
LXT: Low frequency crystal oscillator

HT56R25/HT56R26

Pin Name	Function	OPT	I/T	O/T	Description
PA0/AN0	PA0	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN0	ADCR	AN	—	A/D channel 0
PA1/AN1	PA1	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN1	ADCR	AN	—	A/D channel 1
PA2/AN2	PA2	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN2	ADCR	AN	—	A/D channel 2
PA3/PFD/AN3	PA3	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PFD	CO PINMAP	ST	—	PFD output
	AN3	ADCR	AN	—	A/D channel 3
PA4/INT0/AN4/VDDIO	PA4	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT0	PINMAP	ST	—	External interrupt 0 input
	AN4	ADCR	AN	—	A/D channel 4
	VDDIO	CO	PWR	—	VDDIO power input
PA5/TC0/SDO1/AN5	PA5	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TC0	PINMAP	ST	—	External Timer 0 clock input
	SDO1	CO SPICTL0	—	CMOS	SPI1 serial data output
	AN5	ADCR	AN	—	A/D channel 5
PA6/INT1/SDI1/AN6	PA6	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT1	PINMAP	ST	—	External interrupt 1 input
	SDI1	CO SPICTL0	—	CMOS	SPI1 serial data input
	AN6	ADCR	AN	—	A/D channel 6
PA7/TC1/SCK1/AN7	PA7	PAWK	ST	CMOS	General purpose I/O. Register enabled wake-up.
	TC1	PINMAP	ST	—	External Timer 1 clock input
	SCK1	CO SPICTL0	ST	CMOS	SPI1 serial clock input or output
	AN7	ADCR	AN	—	A/D channel 7
PB0/SCK0/SCL/SCOM0	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SCK0	CO SIMCTL0	ST	CMOS	SPI1 serial clock input or output
	SCL	CO SIMCTL0	ST	—	I^2C serial clock input
	SCOM0	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM

Pin Name	Function	OPT	I/T	O/T	Description
PB1/$\overline{SCS0}$/SCOM1	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{SCS0}$	CO SIMCTL0	ST	CMOS	SPI0 select control pin
	SCOM1	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB2/PCLK/SCOM2	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PCLK	CO SIMCTL0	—	CMOS	Peripheral clock output
	SCOM2	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB3/$\overline{PINT}$/SCOM3	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{PINT}$	—	ST	—	Peripheral interrupt input, falling edge trigger
	SCOM3	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB4/BZ	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	BZ	CO	—	—	Buzzer output
PB5/$\overline{BZ}$	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{BZ}$	CO	—	—	Buzzer bar output
PB6/TC2	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TC2	—	ST	—	External Timer 2 clock input
PB7/TC3	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TC3	—	ST	—	External Timer 3 clock input
PC0/OSC1	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OSC1	CO	AN	—	Oscillator pin
PC1/OSC2	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OSC2	CO	—	CMOS	Oscillator pin
PC2/XT1	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	XT1	CO	—	CMOS	Oscillator pin
PC3/XT2	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	XT2	CO	—	LXT	Oscillator pin
PC4~PC5	PCn	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
PC6/AUD	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AUD	CO DACTRL	—	AO	Audio output
PC7/$\overline{RES}$	PC7	PCPU	ST	NMOS	General purpose I/O. Register enabled pull-up.
	$\overline{RES}$	CO	ST	—	Reset input
PD0/PWM0/SDI0/SDA	PD0	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM0	PINMAP	—	CMOS	PWM0 output
	SDI0	CO SIMCTL0	ST	—	SPI0 serial data input
	SDA	CO SIMCTL0	ST	OD	I^2C data input or output
PD1/PWM1/SDO0	PD1	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM1	PINMAP	—	CMOS	PWM1 output
	SDO0	CO SIMCTL0	—	CMOS	SPI0 serial data output

Pin Name	Function	OPT	I/T	O/T	Description
PD2/PWM2	PD2	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM2	PINMAP	—	CMOS	PWM2 output
PD3/PWM3	PD3	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM3	—	—	CMOS	PWM3 output
PD4/PWM0	PD4	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM0	PINMAP	—	CMOS	PWM0 output
PD5/PWM1	PD5	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM1	PINMAP	—	CMOS	PWM1 output
PD6/TC1	PD6	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TC1	PINMAP	ST	—	External Timer 1 clock input
PD7/INT1	PD7	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	INT1	PINMAP	ST	—	External interrupt 1 input
PE0/PFD/$\overline{SCS1}$	PE0	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PFD	CO PINMAP	—	CMOS	PFD output
	$\overline{SCS1}$	CO SPICTL0	ST	CMOS	SPI1 chip select pin
PE1/INT0	PE1	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	INT0	PINMAP	ST	—	External interrupt 0 input
PE2/TC0	PE2	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TC0	PINMAP	ST	—	External Timer 0 clock input
PE3~PE4	PEn	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
PE5/PWM2	PE5	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM2	PINMAP	—	CMOS	PWM2 output
PE6~PE7	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.
PG0~PG1	PGn	PGPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground

Note: I/T: Input type; O/T: Output type
OPT: Optional by configuration option (CO) or register option
PWR: Power; CO: Configuration option
ST: Schmitt Trigger input; CMOS: CMOS output; AN: analog input
SCOM: Software controlled LCD COM
HXT: High frequency crystal oscillator
LXT: Low frequency crystal oscillator

Absolute Maximum Ratings

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

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
			f_{SYS}=8MHz	3.0	—	5.5	V
			f_{SYS}=12MHz	4.5	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC, RC OSC)	3V	No load, $f_{SYS}=f_M$=1MHz	—	170	250	μA
		5V		—	380	570	μA
I_{DD2}	Operating Current (Crystal OSC, RC OSC)	3V	No load, $f_{SYS}=f_M$=2MHz	—	240	360	μA
		5V		—	490	730	μA
I_{DD3}	Operating Current (Crystal OSC, RC OSC)	3V	No load, $f_{SYS}=f_M$=4MHz (note 4)	—	440	660	μA
		5V		—	900	1350	μA
I_{DD4}	Operating Current (EC Mode, Filter On)	3V	No load, $f_{SYS}=f_M$=4MHz	—	380	570	μA
		5V		—	720	1080	μA
I_{DD5}	Operating Current (EC Mode, Filter Off)	3V	No load, $f_{SYS}=f_M$=4MHz	—	370	550	μA
		5V		—	680	1020	μA
I_{DD6}	Operating Current (Crystal OSC, RC OSC)	5V	No load, $f_{SYS}=f_M$=8MHz	—	1.8	2.7	mA
I_{DD7}	Operating Current (Crystal OSC, RC OSC)	5V	No load, $f_{SYS}=f_M$=12MHz	—	2.6	4.0	mA
I_{DD8}	Operating Current (Slow Mode, f_M=4MHz) (Crystal OSC, RC OSC)	3V	No load, $f_{SYS}=f_{SLOW}$=500kHz	—	150	220	μA
		5V		—	340	510	μA
I_{DD9}	Operating Current (Slow Mode, f_M=4MHz) (Crystal OSC, RC OSC)	3V	No load, $f_{SYS}=f_{SLOW}$=1MHz	—	180	270	μA
		5V		—	400	600	μA
I_{DD10}	Operating Current (Slow Mode, f_M=4MHz) (Crystal OSC, RC OSC)	3V	No load, $f_{SYS}=f_{SLOW}$=2MHz	—	270	400	μA
		5V		—	560	840	μA
I_{DD11}	Operating Current (Slow Mode, f_M=8MHz) (Crystal OSC, RC OSC)	3V	No load, $f_{SYS}=f_{SLOW}$=1MHz	—	240	360	μA
		5V		—	540	810	μA
I_{DD12}	Operating Current (Slow Mode, f_M=8MHz) (Crystal OSC, RC OSC)	3V	No load, $f_{SYS}=f_{SLOW}$=2MHz	—	320	480	μA
		5V		—	680	1020	μA
I_{DD13}	Operating Current (Slow Mode, f_M=8MHz) (Crystal OSC, RC OSC)	3V	No load, $f_{SYS}=f_{SLOW}$=4MHz	—	500	750	μA
		5V		—	1000	1500	μA
I_{DD14}	Operating Current f_{SYS}= LXT or LIRC (note 1)	3V	No load, WDT off	—	6	9	μA
		5V		—	10	15	μA
I_{STB1}	Standby Current (Sleep) (f_{SYS}, f_{SUB}, f_S, f_{WDT}=off)	3V	No load, system HALT, WDT off	—	0.2	1.0	μA
		5V		—	0.3	2.0	μA
I_{STB2}	Standby Current (Sleep) (f_{SYS}, $f_{WDT}=f_{SUB}$= LXT or LIRC	3V	No load, system HALT, WDT on	—	2	4	μA
		5V		—	3	5	μA

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB3}	Standby Current (Idle) (f_{SYS}=on, f_{SYS}=f_M=4MHz, f_{WDT}=off, f_S (note 2)=f_{SUB}=LXT or LIRC	3V	No load, system HALT, WDT off, SPI or I^2C on, PCLK on, PCLK=f_{SYS}/8	—	150	250	μA
		5V		—	350	550	μA
V_{IL1}	Input Low Voltage for I/O Ports, TC0/1/2/3 and INT0/1	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TC0/1/2/3 and INT0/1	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
V_{LVR}	Low Voltage Reset Voltage	—	Configuration option: 2.1V	1.98	2.1	2.22	V
		—	Configuration option: 3.15V	2.98	3.15	3.32	V
		—	Configuration option: 4.2V	3.98	4.2	4.42	V
V_{LVD}	Low Voltage Detector Voltage	—	Configuration option: 2.2V	2.08	2.2	2.32	V
		—	Configuration option: 3.3V	3.12	3.3	3.50	V
		—	Configuration option: 4.4V	4.12	4.4	4.70	V
I_{OL1}	I/O Port Sink Current	3V	V_{OL}=$0.1V_{DD}$	6	12	—	mA
		5V		10	25	—	mA
I_{OH1}	I/O Port Source Current	3V	V_{OH}=$0.9V_{DD}$	–2	–4	—	mA
		5V		–5	–8	—	mA
R_{PH}	Pull-high Resistance for I/O Ports	3V	—	20	60	100	kΩ
		5V		10	30	50	kΩ
I_{SCOM}	SCOM Operating Current	5V	SCOMC, ISEL=0	17.5	25.0	32.5	μA
			SCOMC, ISEL=1	35	50	65	μA
V_{SCOM}	V_{DD}/2 Voltage for LCD COM	5V	No load	0.475	0.500	0.525	V_{DD}

Note: 1. LXT is in slow start mode (RTCC.4=QOSC=1) for the D.C. current measurement.

2. f_S is the internal clock for the Buzzer, RTC Interrupt, Time Base Interrupt and the WDT.

3. Both Timer/Event Counters are off. Timer filter is disabled for all test conditions.

4. All peripherals are in OFF condition if not mentioned at I_{DD}, I_{STB} tests.

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (Crystal OSC, RC OSC)	—	2.2V~5.5V	32	—	4000	kHz
			3.0V~5.5V	32	—	8000	kHz
			4.5V~5.5V	32	—	12000	kHz
f_{4MERC}	System clock (ERC)	5V	R=150kΩ, Ta=25°C*	–2%	4	+2%	MHz
		5V	R=150kΩ, Ta=–40°C~+85°C	–8%	4	+8%	MHz
		2.7V~5.5V	R=150kΩ, Ta=–40°C~+85°C	–15%	4	+15%	MHz
f_{4MIRC}	System clock (HIRC)	5V	Ta=25°C	–2%	4	+2%	MHz
		5V	Ta=–40°C~85°C	–5%	4	+5%	MHz
		2.7V~5.5V	Ta=–40°C~85°C	–10%	4	+10%	MHz
f_{LXT}	System clock LXT	—	—	—	32768	—	Hz
f_{TIMER}	Timer I/P Frequency (TMR0/TMR1)	—	2.2V~5.5V	0	—	4000	kHz
			3.0V~5.5V	0	—	8000	kHz
			4.5V~5.5V	0	—	12000	kHz
f_{LIRC}	LIRC Oscillator	—	2.2V~5.5V, After Trim	28.8	32.0	35.2	kHz
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{LVR}	Low Voltage Reset Time	—	—	0.1	0.4	0.6	ms
t_{SST1}	System Start-up Timer Period	—	Power-on	—	1024	—	t_{SYS}*
t_{SST2}	System Start-up Timer Period for XTAL or RTC oscillator	—	Wake-up from Power Down Mode	—	1024	—	t_{SYS}*
t_{SST3}	System Start-up Timer Period for External RC or External Clock	—	Wake-up from Power Down Mode	—	1	2	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs

Note: 1. $t_{SYS}=1/f_{SYS1}$ or $1/f_{SYS2}$

2. * For f_{4MERC}, as the resistor tolerance will influence the frequency a precision resistor is recommended.

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

ADC Characteristics

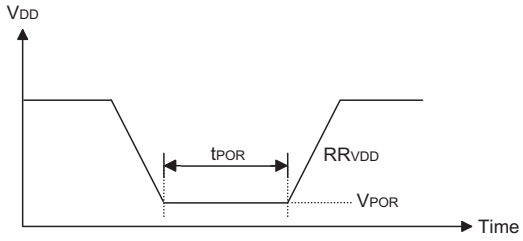
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
DNL	A/C Differential Non-Linearity	5V	t_{AD}=0.5μs	–2	—	2	LSB
INL	ADC Integral Non-Linearity	5V	t_{AD}=0.5μs	–4	—	4	LSB
I_{ADC}	Additional Power Consumption if A/D Converter is Used	3V	—	—	0.50	0.75	mA
		5V		—	1.00	1.50	mA
t_{AD}	A/D Clock Period	—	—	0.5	—	—	μs
t_{ADC}	A/D Conversion Time	—	—	—	16	—	t_{AD}

Power-on Reset Characteristics

Ta=25°C

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



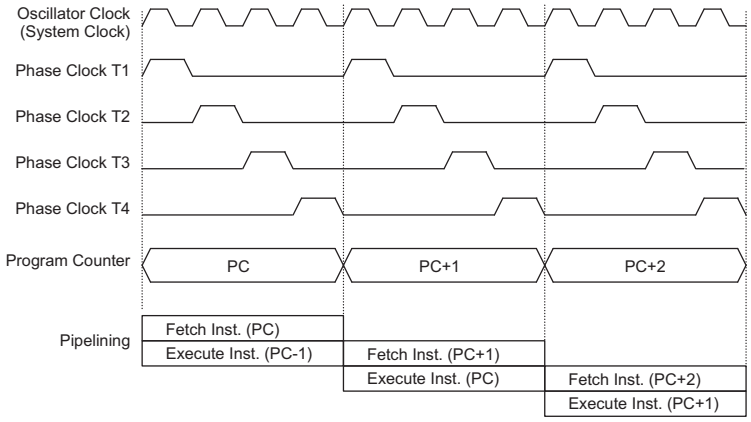
System Architecture

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

Clocking and Pipelining

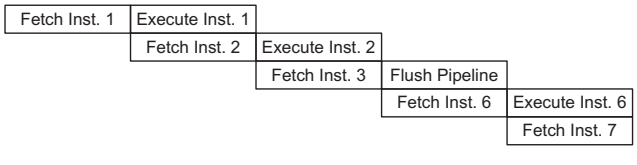
The main system clock, derived from either a Crystal/Resonator or RC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry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

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



System Clocking and Pipelining

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



Instruction Fetching

Program Counter

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. Note that the Program Counter width varies with the Program Memory capacity depending upon which device is selected. However, it must be noted that only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by user.

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

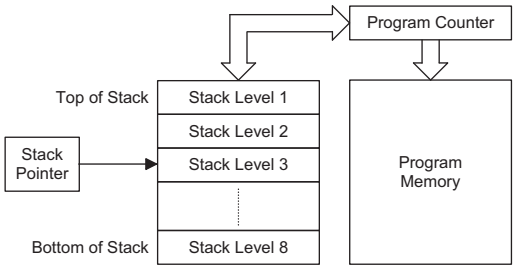
Device	Program Counter	
	Program Counter High Byte	PCL Register
HT56R22	PC10~PC8	PCL7~PCL0
HT56R23	PC11~PC8	
HT56R24	PC12~PC8	
HT56R25	PC13~PC8	
HT56R26	PC14~PC8	

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

The lower byte of the Program Counter is fully accessible under program control. Manipulating the PCL might cause program branching, so an extra cycle is needed to pre-fetch. Further information on the PCL register can be found in the Special Function Register section.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is neither part of the Data or Program Memory space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, SP, 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.



Device	Stack Levels
HT56R22	6
HT56R23 HT56R24 HT56R25 HT56R26	12

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

Program Memory

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

Structure

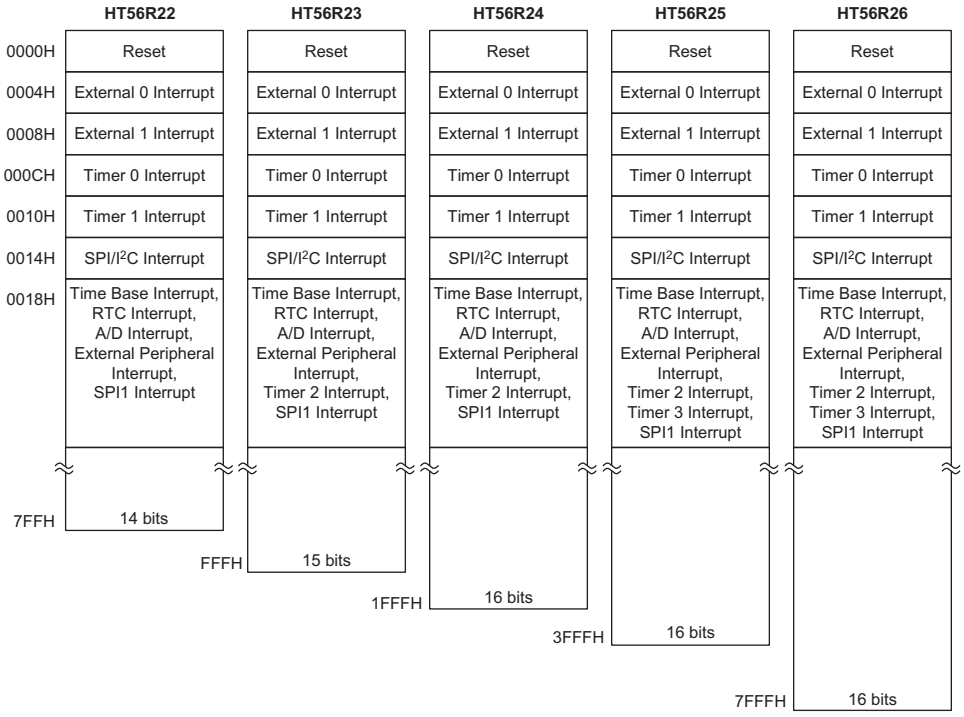
The Program Memory has a capacity of 2K×14 to 32K×16. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by separate table pointer registers.

Device	Capacity
HT56R22	2K×14
HT56R23	4K×15
HT56R24	8K×16
HT56R25	16K×16
HT56R26	32K×16

Special Vectors

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

- Reset Vector
  This vector is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.
- External interrupt 0/1 vector
  This vector is used by the external interrupt. If the external interrupt pin on the device receives an edge transition, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full. The external interrupt active edge transition type, whether high to low, low to high or both is specified in the INTEDGE register.
- Timer/Event 0/1 counter interrupt vector
  This internal vector is used by the Timer/Event Counters. If a Timer/Event Counter overflow occurs, the program will jump to its respective location and begin execution if the associated Timer/Event Counter interrupt is enabled and the stack is not full.
- SPI/I^2C interrupt vector
  This internal vector is used by the SPI/I^2C interrupt. When either an SPI or I^2C bus, dependent upon which one is selected, requires data transfer, the program will jump to this location and begin execution if the SPI/I^2C interrupt is enabled and the stack is not full.



Program Memory Structure

- Multifunction Interrupt vector
  The Multi-function Interrupt vector is shared by several internal functions such as a Time Base overflow, a Real Time Clock overflow, an A/D converter conversion completion, a falling edge appearing on the External Peripheral interrupt pin, a Timer/Event Counter 2 or a Timer/Event Counter 3 overflow, a SPI data transfer completion. The program will jump to this location and begin execution if the relevant interrupt is enabled and the stack is not full.

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the lower order address of the look up data to be retrieved in the table pointer register, TBLP. This register defines the lower 8-bit address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the current Program Memory page or last Program Memory page using the ″TABRDC[m]″ or ″TABRDL [m]″ instructions, respectively. When these instructions are 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 following diagram illustrates the addressing/data flow of the look-up table:

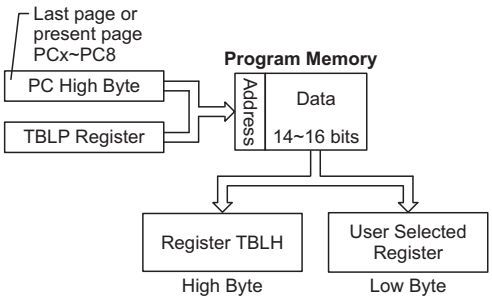


Table Program Example

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 last page which is stored there using the ORG statement. The value at this ORG statement is ″700H″ which refers to the start address of the last page within the 2K Program Memory of the HT56R22 microcontrollers. The table pointer is setup here to have an initial value of ″06H″. This will ensure that the first data read from the data table will be at the Program Memory address ″706H″ or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the ″TABRDC [m]″ instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the ″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.

Instruction	Table Location Bits														
	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	PC14	PC13	PC12	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: PC14~PC8: Current Program Counter bits
@7~@0: Table Pointer TBLP bits

For the HT56R22, the Table address location is 11 bits, i.e. from b10~b0.
For the HT56R23, the Table address location is 12 bits, i.e. from b11~b0.
For the HT56R24, the Table address location is 13 bits, i.e. from b12~b0.
For the HT56R25, the Table address location is 14 bits, i.e. from b13~b0.
For the HT56R26, the Table address location is 15 bits, i.e. from b14~b0.

Table Read Program Example:

```
tempreg1 db ?          ; temporary register #1
tempreg2 db ?          ; temporary register #2
     :
     :

mov a,06h              ; initialise table pointer - note that this address
                       ; is referenced

mov tblp,a             ; to the last page or present page
     :
     :

tabrdl   tempreg1      ; transfers value in table referenced by table pointer
                       ; to tempreg1
                       ; data at prog. memory address "706H" transferred to
                       ; tempreg1 and TBLH

dec tblp               ; reduce value of table pointer by one

tabrdl   tempreg2      ; transfers value in table referenced by table pointer
                       ; to tempreg2
                       ; data at prog.memory address "705H" transferred to
                       ; tempreg2 and TBLH
                       ; in this example the data "1AH" is transferred to
                       ; tempreg1 and data "0FH" to register tempreg2
                       ; the value "00H" will be transferred to the high byte
                       ; register TBLH
     :
     :

org 700h               ; sets initial address of last page

dc  00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
     :
     :
```

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

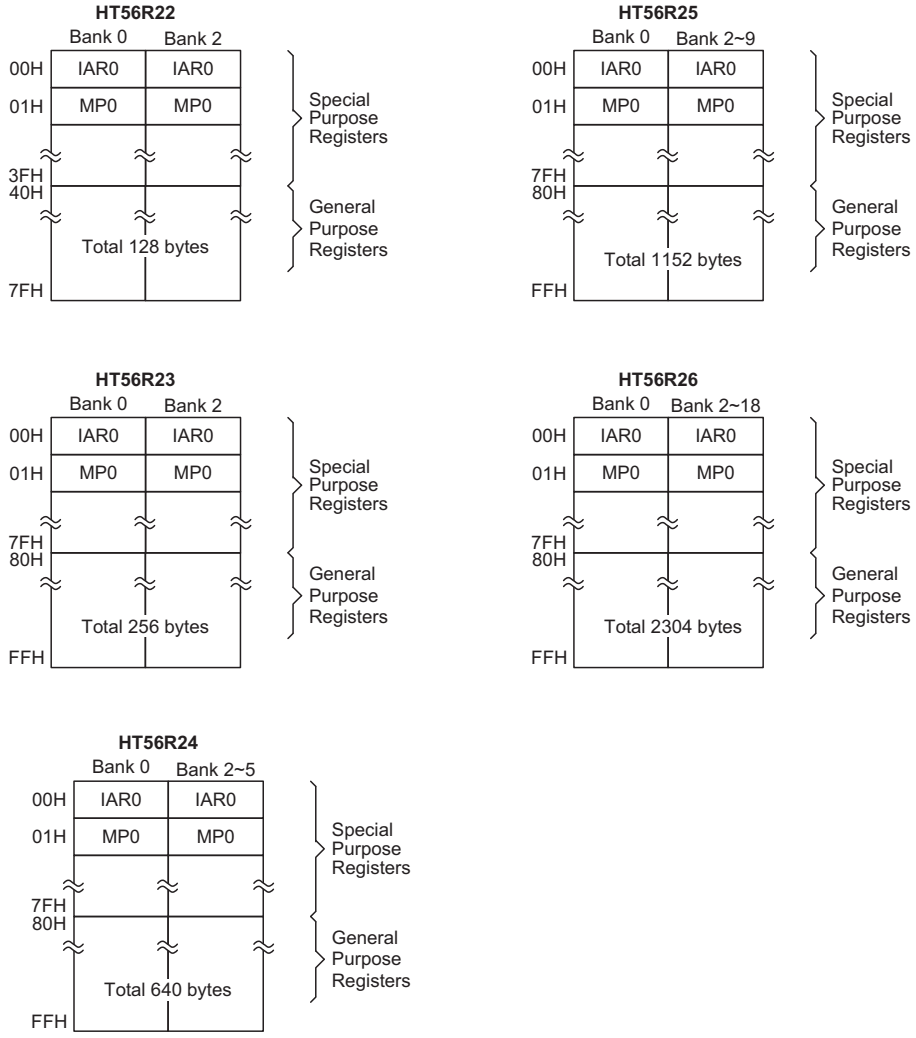
Divided into two sections, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some 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.

Device	Capacity	Banks
HT56R22	128×8	0, 2
HT56R23	256×8	0, 2
HT56R24	640×8	0, 2~5
HT56R25	1152×8	0, 2~9
HT56R26	2304×8	0, 2~18

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

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

For some devices, the Data Memory is subdivided into banks, which are selected using a Bank Pointer. Only data in Bank 0 can be directly addressed, data in Bank 2~n must be indirectly addressed.



Data Memory Structure

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

Special Purpose Data Memory

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

Special Function Registers

To ensure successful operation of the microcontroller, certain internal registers are implemented in the Data Memory area. These registers ensure correct operation of internal functions such as timers, interrupts, etc., as well as external functions such as I/O data control. The location of these registers within the Data Memory begins at the address ″00H″ and are mapped into both Bank 0 and Bank 1. Any unused Data Memory locations between these special function registers and the point where the General Purpose Memory begins is reserved and attempting to read data from these locations will return a value of ″00H″.

Indirect Addressing Registers – IAR0, IAR1

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

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to indirectly address and track data. MP0 can only be used to indirectly address data in Bank 0 while MP1 can be used to address data in Bank 0 and Bank1. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, is the address specified by the related Memory Pointer. Note that for the HT56R22 device, bit 7 of the Memory Pointers is not required to address the full memory space. When bit 7 of the Memory Pointers for this device is read, a value of ″1″ will be returned. Note that indirect addressing using MP1 and IAR1 must be used to access any data in Bank 1. The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.

- **Indirect Addressing Program Example**

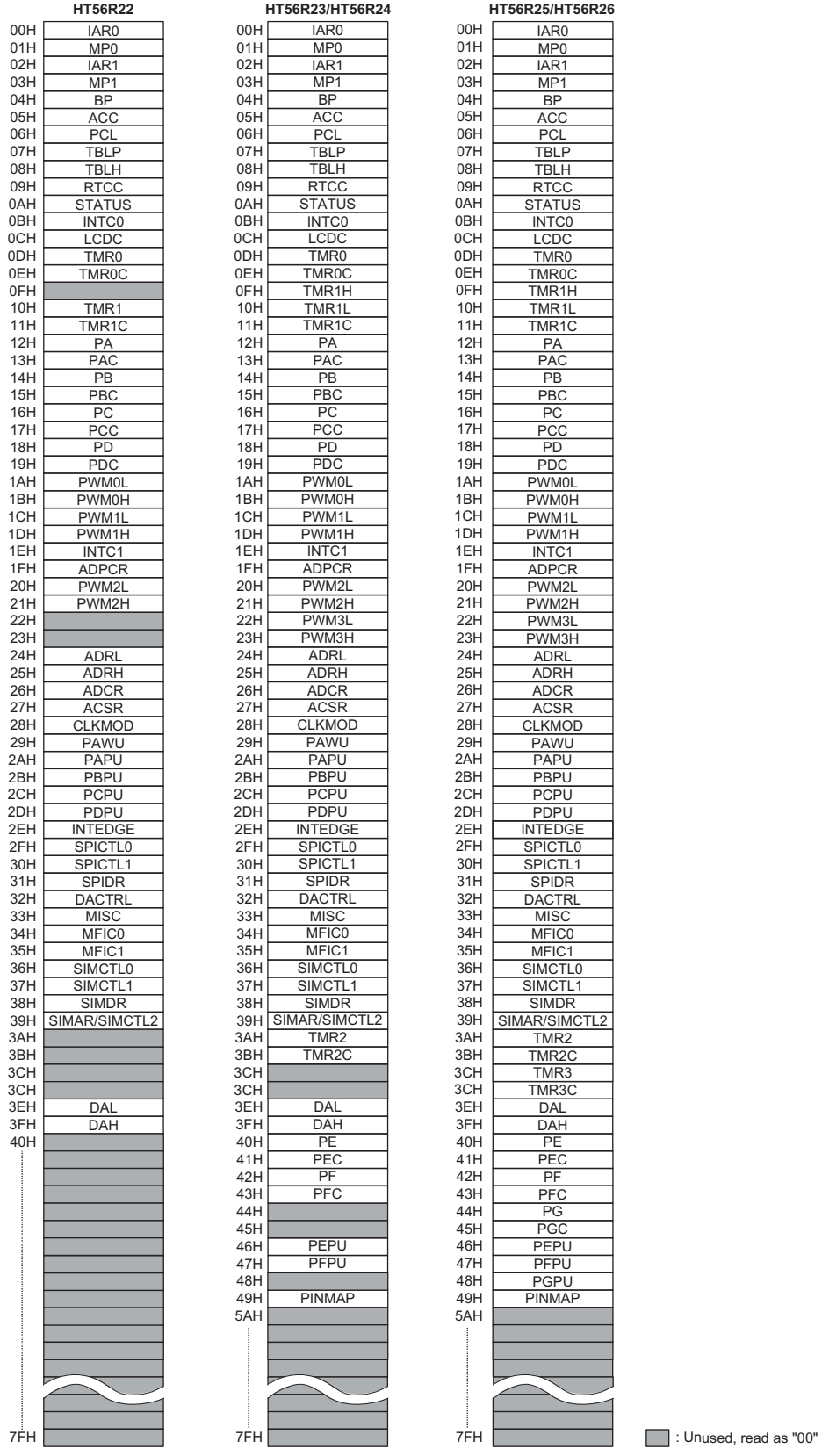
```
data .section  'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 code
org   00h

start:
     mov a,04h                  ; setup size of block
     mov block,a
     mov a,offset adres1        ; Accumulator loaded with first RAM address
     mov mp0,a                  ; setup memory pointer with first RAM address

loop:
     clr IAR0                   ; clear the data at address defined by MP0
     inc mp0                    ; increment memory pointer
     sdz block                  ; check if last memory location has been cleared
     jmp loop

continue:
```

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



Special Purpose Data Memory

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Bank Pointer – BP

Depending upon which device is used, the Program and Data Memory is divided into several banks. Selecting the required Program and Data Memory area is achieved using the Bank Pointer.

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

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

Device	Bit							
	7	6	5	4	3	2	1	0
HT56R22 HT56R23	—	—	—	—	—	—	DMBP1	DMBP0
HT56R24	—	—	—	—	—	DMBP2	DMBP1	DMBP0
HT56R25	—	—	PMBP0	—	DMBP3	DMBP2	DMBP1	DMBP0
HT56R26	—	PMBP1	PMBP0	DMBP4	DMBP3	DMBP2	DMBP1	DMBP0

BP Registers List

- **BP Register**
  - HT56R22/HT56R23

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

Bit 7 ~ 2 Unimplemented, read as "0"

Bit 1 ~ 0 **DMBP1 ~ DMBP0**: Select Data Memory Banks
00: Bank 0
01: Reserved
10: Bank 2
11: Undefined

• HT56R24

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

Bit 7 ~ 3 Unimplemented, read as "0"

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

• HT56R25

Bit	7	6	5	4	3	2	1	0
Name	—	—	PMBP0	—	DMBP3	DMBP2	DMBP1	DMBP0
R/W	—	—	R/W	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7 ~ 2 Unimplemented, read as "0"

Bit5 **PMBP0**: Select Program Memory Banks
0: Bank 0, Program Memory Address is from 0000H ~ 1FFFH
1: Bank 1, Program Memory Address is from 2000H ~ 3FFFH

Bit4 Unimplemented, read as "0"

Bit3 ~ 0 **DMBP3 ~ DMBP0**: Select Data Memory Banks
0000: Bank 0
0001: Reserved
0010: Bank 2
0011: Bank 3
:
:
1001: Bank 9
1010~1111: Undefined

• HT56R26

Bit	7	6	5	4	3	2	1	0
Name	—	PMBP1	PMBP0	DMBP4	DMBP3	DMBP2	DMBP1	DMBP0
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 ~ 2 Unimplemented, read as "0"

Bit6~5 **PMBP1, PMBP0**: Select Program Memory Banks
00: Bank 0, Program Memory Address is from 0000H ~ 1FFFH
01: Bank 1, Program Memory Address is from 2000H ~ 3FFFH
10: Bank 2, Program Memory Address is from 4000H ~ 5FFFH
11: Bank 3, Program Memory Address is from 6000H ~ 7FFFH

Bit4 ~ 0 **DMBP4 ~ DMBP0**: Select Data Memory Banks
00000: Bank 0
00001: Reserved
00010: Bank 2
00011: Bank 3
:
:
10010: Bank 18
10011~11111: Undefined

Status Register – STATUS

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

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

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

In addition, on entering an interrupt sequence or executing a subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status registers are important and if the interrupt routine can change the status register, precautions must be taken to correctly save it. Note that bits 0~3 of the STATUS register are both readable and writeable bits.

Input/Output Ports and Control Registers

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

- **STATUS Register**

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

"x" unknown

Bit 7, 6 Unimplemented, read as "0"

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

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

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

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

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

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

Wake-up Function Register – PAWK

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

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

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

Software COM Register – SCOMC

The pins PB0~PB3 on Port B can be used as SCOM lines to drive an external LCD panel. To implement this function, the SCOMC register is used to setup the correct bias voltages on these pins.

Oscillator

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

System Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base functions. External oscillators requiring some external components as well as a two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators.

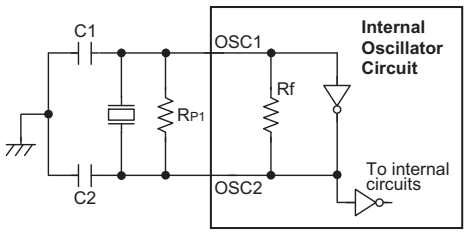
Type	Name	Freq.	Pins
External Crystal	HXT	400kHz~ 12MHz	OSC1/ OSC2
External RC	ERC	400kHz~ 12MHz	OSC1
Internal High Speed RC	HIRC	4, 8 or 12MHz	—
External Low Speed Crystal	LXT	32768Hz	— XT1/ XT2*
Internal Low Speed RC	LIRC	32kHz	—

System Clock Configurations

There are five system oscillators. Three high speed oscillators and two low speed oscillators. The high speed oscillators are the external crystal/ceramic oscillator - HXT, the external - ERC, and the internal RC oscillator - HIRC. The two low speed oscillator are the external 32768Hz oscillator - LXT and the internal 32kHz oscillator - LIRC.

External Crystal/Resonator Oscillator – HXT

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



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

Crystal/Resonator Oscillator – HXT

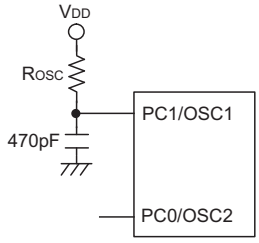
Crystal Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
12MHz	—	—
8MHz	—	—
4MHz	—	—
1MHz	—	—
455kHz (see Note 2)	10pF	10pF

Note: 1. C1 and C2 values are for guidance only.
2. XTAL mode configuration option: 455kHz.
3. R_{P1}=5MΩ~10MΩ is recommended.

Crystal Recommended Capacitor Values

External RC Oscillator – ERC

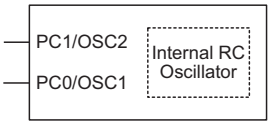
Using the ERC oscillator only requires that a resistor, with a value between 47kΩ and 1.5MΩ, is connected between OSC1 and VDD, and a capacitor is connected between OSC and ground, providing a low cost oscillator configuration. It is only the external resistor that determines the oscillation frequency; the external capacitor has no influence over the frequency and is connected for stability purposes only. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a resistance/frequency reference point, it can be noted that with an external 150kΩ resistor connected and with a 5V voltage power supply and temperature of 25 degrees, the oscillator will have a frequency of 4MHz within a tolerance of 2%. Here only the OSC1 pin is used, which is shared with I/O pin PC0, leaving pin PC1 free for use as a normal I/O pin.



External RC Oscillator – ERC

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of either 4MHz, 8MHz or 12MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of either 3V or 5V and at a temperature of 25 degrees, the fixed oscillation frequency of 4MHz, 8MHz or 12MHz will have a tolerance within 2%. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PC1 and PC0 are free for use as normal I/O pins.

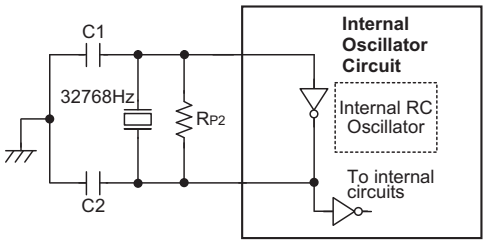


Note: PC0/PC1 used as normal I/Os

Internal RC Oscillator – HIRC

External 32768Hz Crystal Oscillator – LXT

When the microcontroller enters the Idle/Sleep Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the Power-down Mode. To do this, another clock, independent of the system clock, must be provided. To do this a configuration option exists to allow a low speed oscillator, known as the LXT oscillator to be used. The LXT oscillator is implemented using a 32768Hz crystal connected to pins. However, for some crystals, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. The external parallel feedback resistor, R_{P2}, is required.



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

External LXT Oscillator – HXT

LXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
32768Hz	8pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. R_{P2}=5M~10MΩ is recommended.		

32768 Hz Crystal Recommended Capacitor Values

For the devices, a configuration option determines if the XT1/XT2 pins are used for the LXT oscillator or as I/O pins.

- If the I/O option is selected then the XT1/XT2 pins can be used as normal I/O pins.
- If the ″LXT oscillator″is selected then the 32768Hz crystal should be connected to the XT1/ XT2 pins.

LXT Oscillator Low Power Function

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

QOSC Bit	LXT Mode
0	Quick Start
1	Low-power

After power on the QOSC 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 QOSC 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 QOSC bit high about 2 seconds after power-on.

It should be noted that, no matter what condition the QOSC 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 Low Speed Oscillator – LIRC

When microcontrollers enter a power down condition, their internal clocks are normally switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep some internal functions operational, such as timers, even when the microcontroller is in the Power-down mode. To do this, the device has a LIRC oscillator, which is a fully integrated free running RC oscillator with a typical period of 31.2 s at 5V, requiring no external components. It is selected via configuration option. When the device enters the Power Down Mode, the system clock will stop running, however the LIRC oscillator will continue to run if selected to keep various internal functions operational.

System Operating Modes

The devices have the ability to operate in several different modes. This range of operating modes, known as Normal Mode, Slow Mode, Idle Mode and Sleep Mode, allow the devices to run using a wide range of different slow and fast clock sources. The devices also possess the ability to dynamically switch between different clocks and operating modes. With this choice of operating functions users are provided with the flexibility to ensure they obtain optimal performance from the device according to their application specific requirements.

Clock Sources

In discussing the system clocks for the devices, they can be seen as having a dual clock mode. These dual clocks are what are known as a High Oscillator and the other as a Low Oscillator. The High and Low Oscillator are the system clock sources and can be selected dynamically using the HLCLK bit in the CLKMOD register.

The High Oscillator has the internal name f_M whose source is selected using a configuration option from a choice of either an external crystal/resonator, external RC oscillator or external clock source.

The Low Oscillator clock source, has the internal name f_{SL}, whose source is also selected by configuration option. This internal f_{SL}, f_M clock, is further modified by the SLOWC0~SLOWC2 bits in the CLKMOD register to provide the low frequency clock source f_{SLOW}.

An additional sub internal clock, with the internal name f_{SUB}, is a 32kHz clock source which can be sourced from either LXT or LIRC, selected by configuration option. Together with $f_{SYS}/4$, it is used as a clock source for certain internal functions such as the LCD driver, Watchdog Timer, Buzzer, RTC Interrupt and Time Base Interrupt. The LCD clock source is the f_{SUB} clock source divided by 8, giving a frequency of 4kHz. The internal clock f_S, is simply a choice of either f_{SUB} or $f_{SYS}/4$, using a configuration option.

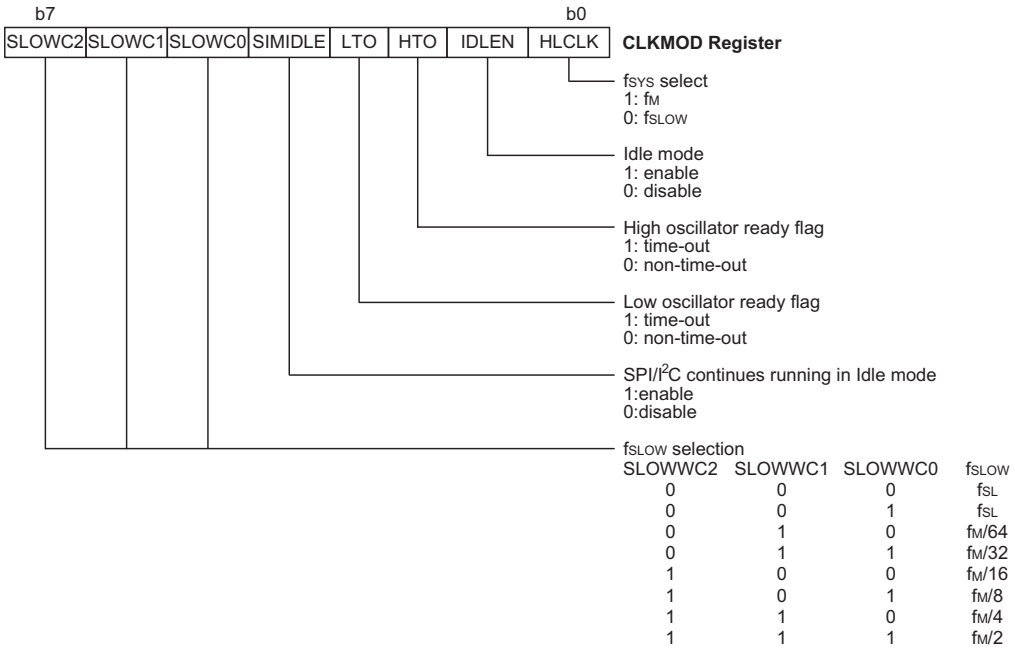
Operating Modes

After the correct clock source configuration selections are made, overall operation of the chosen clock is achieved using the CLKMOD register. A combination of the HLCLK and IDLEN bits in the CLKMOD register and use of the HALT instruction determine in which mode the device will be run. The devices can operate in the following Modes.

- Normal Mode
  f_M on, f_{SLOW} on, $f_{SYS}=f_M$, CPU on, f_S on, f_{WDT} on/off depending upon the WDT configuration option and WDT control register.
- Slow Mode0
  f_M off, f_{SLOW}=LXT or LIRC, $f_{SYS}=f_{SLOW}$, CPU on, f_S on, f_{WDT} on/off depending upon the WDT configuration option and WDT control register.
- Slow Mode1
  f_M on, $f_{SLOW}=f_M/2$~$f_M/64$, $f_{SYS}=f_{SLOW}$, CPU on, f_S on, f_{WDT} on/off depending upon the WDT configuration option and WDT control register.
- Idle Mode
  f_M, f_{SLOW}, f_{SYS} off, CPU off; f_{SUB} on, f_S on/off by selecting f_{SUB} or $f_{SYS}/4$, f_{WDT} on/off depending upon the WDT configuration option and WDT control register.

- Sleep Mode
  f_M, f_{SLOW}, f_{SYS}, f_S, CPU off; f_{SUB}, f_{WDT} on/off depending upon the WDT configuration option and WDT control register.

For all devices, when the system enters the Sleep or Idle Mode, the high frequency system clock will always stop running. The accompanying tables shows the relationship between the CLKMOD bit, the HALT instruction and the high/low frequency oscillators. The CLMOD bit can change normal or Slow Mode.

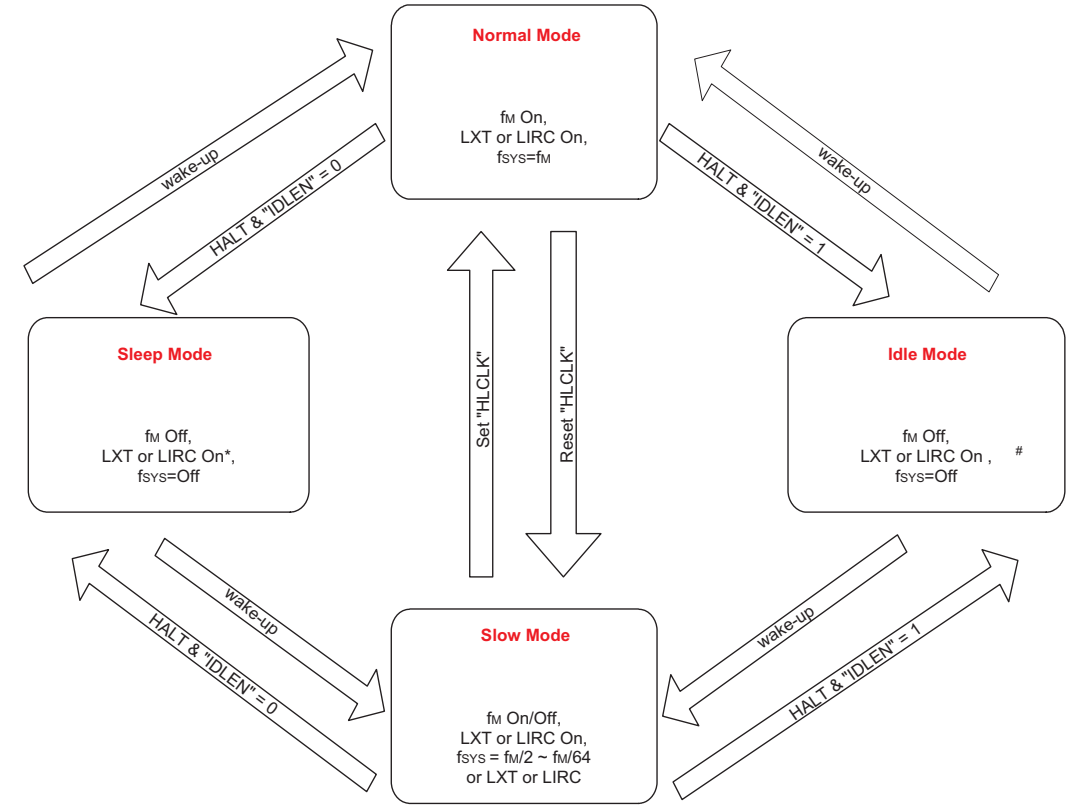


SLOWWC2	SLOWWC1	SLOWWC0	f_{SLOW}
0	0	0	f_{SL}
0	0	1	f_{SL}
0	1	0	$f_M/64$
0	1	1	$f_M/32$
1	0	0	$f_M/16$
1	0	1	$f_M/8$
1	1	0	$f_M/4$
1	1	1	$f_M/2$

Clock Control Register – CLKMOD

- **Operating Mode Control**

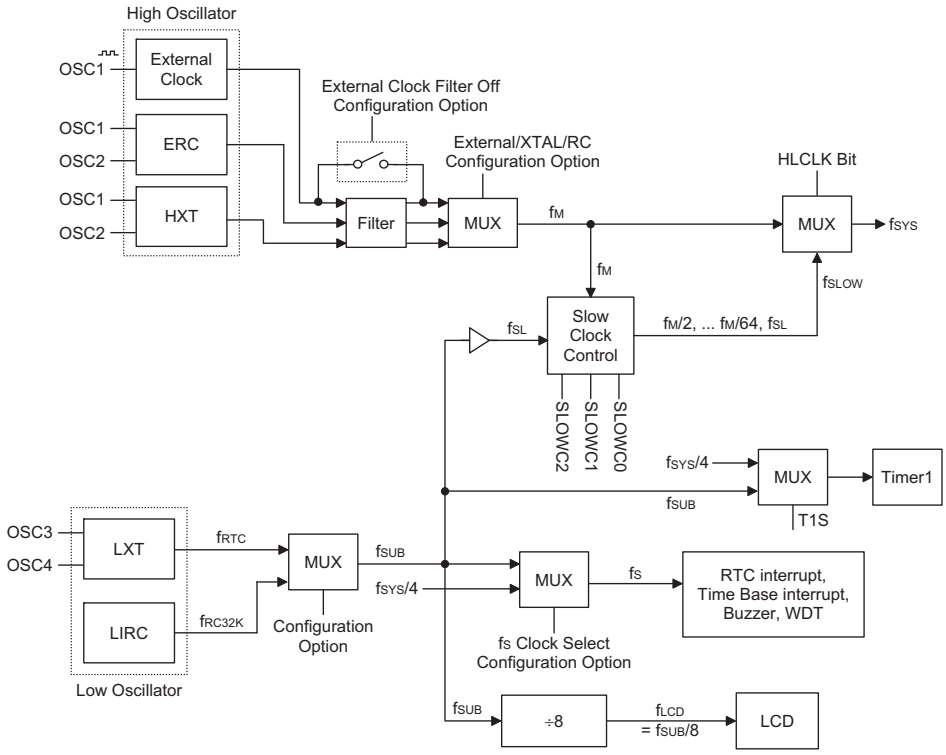
Operation Mode	Description		
	CPU	f_{SYS}	f_{SUB}
NORMAL Mode	On	f_M	On
SLOW0 Mode	On	f_{SL}	On
SLOW1 Mode	On	$f_M/2 \sim f_M/64$	On
IDLE Mode	Off	Off	On
SLEEP Mode	Off	Off	On/Off



"*" Depends the WDT enable/disable condition.

"#" Either the 32768Hz or 32K_INT must be ON.

Dual Clock Mode Operation



Dual Clock Mode Structure

Mode Switching

The devices are switched between one mode and another using a combination of the HLCLK bit in the CLKMOD register and the HALT instruction. The HLCLK bit chooses whether the system runs in either the Normal or Slow Mode by selecting the system clock to be sourced from either a high or low frequency oscillator. The HALT instruction forces the system into either the Idle or Sleep Mode, depending upon whether the IDLEN bit in CLKMOD register is set or not.

When a HALT instruction is executed and the IDLEN bit is not set. The system enters the Sleep mode the following conditions exist:

- The system oscillator will stop running and the application program will stop at the ″HALT″ instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT is enabled and clock source is selected from f_{SUB}. The WDT will stop if its clock source originates from the system clock or the WDT is disabled.
- 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.

Power Down Mode and Wake-up

Power Down Mode

All of the Holtek microcontrollers have the ability to enter a Power Down Mode. When the device enters this mode, the normal operating current, will be reduced to an extremely low standby current level. This occurs because when the device enters the Power Down Mode, the system oscillator is stopped which reduces the power consumption to extremely low levels, however, as the device maintains its present internal condition, it can be woken up at a later stage and continue running, without requiring a full reset. This feature is extremely important in application areas where the MCU must have its power supply constantly maintained to keep the device in a known condition but where the power supply capacity is limited such as in battery applications.

Entering the Power Down Mode

There is only one way for the device to enter the Power Down Mode and that is to execute the ″HALT″ instruction in the application program. When this instruction is executed, the following will occur:

- The system oscillator will stop running and the application program will stop at the ″HALT″ instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the WDT 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 Power Down 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 minimized. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to devices which have different package types, as there may be undonbed pins, which must either be setup as outputs or if setup as inputs must have pull-high resistors connected. Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the configuration options have enabled the Watchdog Timer internal oscillator.

Wake-up

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

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

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

Each pin on Port A can be setup via an individual configuration option to permit a negative transition on the pin to wake-up the system. When a Port A pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction.

If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set to "1" before entering the Power Down Mode, the wake-up function of the related interrupt will be disabled.

No matter what the source of the wake-up event is, once a wake-up situation occurs, a time period equal to 1024 system clock periods will be required before normal system operation resumes. However, if the wake-up has originated due to an interrupt, the actual interrupt subroutine execution will be delayed by an additional one or more cycles. If the wake-up results in the execution of the next instruction following the "HALT" instruction, this will be executed immediately after the 1024 system clock period delay has ended.

Watchdog Timer

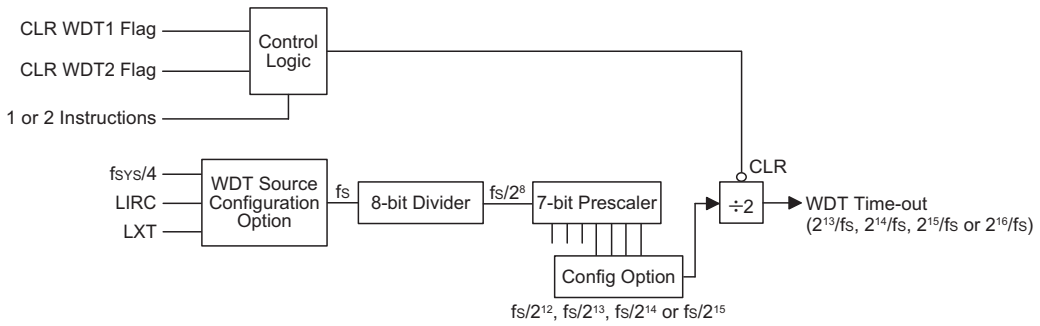
The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise. It operates by providing a device reset when the Watchdog Timer counter overflows.

Watchdog Timer Operation

The Watchdog Timer clock source is provided by the internal clock, f_S, which is in turn supplied by one of two sources selected by configuration option: f_{SUB} or $f_{SYS}/4$. Note that if the Watchdog Timer configuration option has been disabled, then any instruction relating to its operation will result in no operation.

Most of the Watchdog Timer options, such as enable/disable, Watchdog Timer clock source and clear instruction type are selected using configuration options. In addition to a configuration option to enable the Watchdog Timer, there are four bits, WDTEN3~ WDTEN0, in the MISC register to offer an additional enable control of the Watchdog Timer. These bits must be set to a specific value of 1010 to disable the Watchdog Timer. Any other values for these bits will keep the Watchdog Timer enabled. After power on these bits will have the disabled value of 1010.

One of the WDT clock sources is the internal f_{SUB}, which can be sourced from either the LXT or LIRC. The LIRC has an approximate period of 31.2μs at a supply voltage of 5V. However, it should be noted that this specified internal clock period can vary with VDD, temperature and process variations. The LXT is supplied by an external 32768Hz crystal. The other Watchdog Timer clock source option is the $f_{SYS}/4$ clock. Whether the Watchdog Timer clock source is LIRC, LXT or $f_{SYS}/4$, it is divided by 2^{13}~2^{16}, using configuration option to obtain the required Watchdog Timer time-out period. The max time out period is when the 2^{16} option is selected. This time-out period may vary with temperature, VDD and process variations. As the clear instruction only resets the last stage of the divider chain, for this reason the actual division ratio and corresponding Watchdog Timer time-out can vary by a factor of two. The exact division ratio depends upon the residual value in the Watchdog Timer counter before the clear instruction is executed.



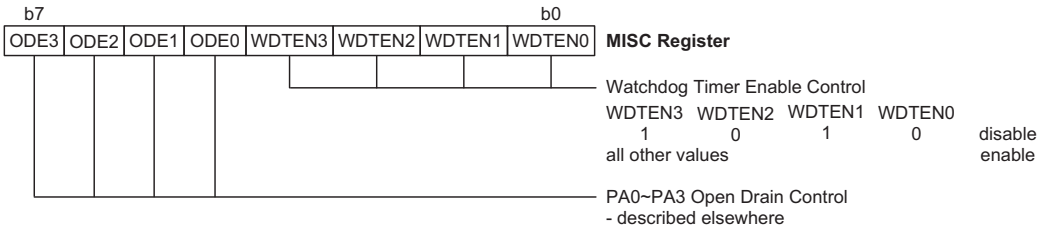
Watchdog Timer

If the $f_{SYS}/4$ clock is used as the Watchdog Timer clock source, it should be noted that when the system enters the Power Down Mode, then the instruction clock is stopped and the Watchdog Timer will lose its protecting purposes. For systems that operate in noisy environments, using the LIRC oscillator is strongly recommended.

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 Power Down Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is an external hardware reset, which means a low level on the $\overline{RES}$ pin, the second is using the watchdog software instructions and the third is via a ″HALT″ instruction.

Clearing the Watchdog Timer

There are two methods of using software instructions to clear the Watchdog Timer, one of which must be chosen by configuration option. The first option is to use the single ″CLR WDT″ instruction while the second is to use the two commands ″CLR WDT1″ and ″CLR WDT2″. For the first option, a simple execution of ″CLR WDT″ will clear the WDT while for the second option, both ″CLR WDT1″ and ″CLR WDT2″ must both be executed to successfully clear the Watchdog Timer. Note that for this second option, if ″CLR WDT1″ is used to clear the Watchdog Timer, successive executions of this instruction will have no effect, only the execution of a ″CLR WDT2″ instruction will clear the Watchdog Timer. Similarly after the ″CLR WDT2″ instruction has been executed, only a successive ″CLR WDT1″ instruction can clear the Watchdog Timer.



Watchdog Timer Software Control – MISC

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.

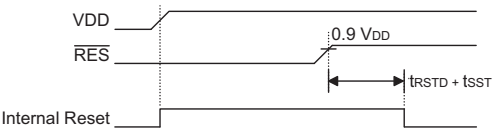
In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup.

Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset is implemented in situations where the power supply voltage falls below a certain threshold.

Reset Functions

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

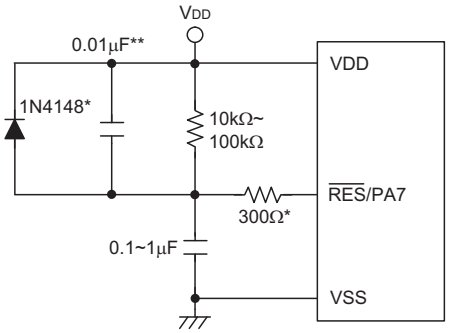
- Power-on Reset
  The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to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



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

Power-On Reset Timing Chart

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 Enhanced Reset Circuit shown is recommended.

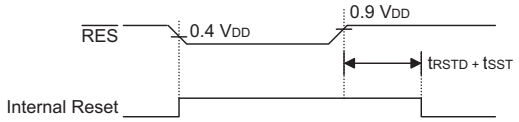


Note: "*" It is recommended that this component is added for added ESD protection
"**" It is recommended that this component is added in environments where power line noise is significant

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

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

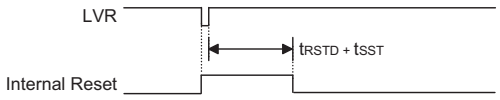
- $\overline{RES}$ Pin Reset
  This type of reset occurs when the microcontroller is already running and the $\overline{RES}$ pin is forcefully pulled low by external hardware such as an external switch. In this case as in the case of other reset, the Program Counter will reset to zero and program execution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



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

$\overline{RES}$ Reset Timing Chart

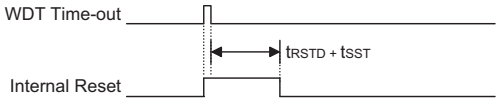
- Low Voltage Reset – LVR
  The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is selected via a configuration option. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the A.C. characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected via configuration options.



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

Low Voltage Reset Timing Chart

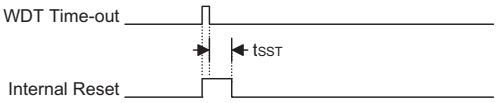
- Watchdog Time-out Reset during Normal Operation
  The Watchdog time-out Reset during normal operation is the same as a hardware $\overline{RES}$ pin reset except that the Watchdog time-out flag TO will be set to "1".



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

WDT Time-out Reset during Normal Operation Timing Chart

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



WDT Time-out Reset during Idle/Sleep Timing Chart

Note: The t_{SST} can be chosen to be either 1024 or 2 clock cycles via configuration option if the system clock source is provided by ERC or HIRC. The SST is 1024 for HXT or LXT.

Reset Initial Conditions

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

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

Note: "u" stands for unchanged

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

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

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

Register	HT56R22	HT56R23	HT56R24	HT56R25	HT56R26	Power-on Reset	$\overline{RES}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (Idle/Sleep)
MP0	•					–xxx xxxx	–uuu uuuu	–uuu uuuu	–uuu uuuu
MP1	•					–xxx xxxx	–uuu uuuu	–uuu uuuu	–uuu uuuu
MP0		•	•	•	•	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1		•	•	•	•	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	•	•				----- --00	----- --00	----- --00	----- --uu
			•			----- - 000	----- - 000	----- - 000	----- - uuu
				•		--00 0000	--00 0000	--00 0000	--uu uuuu
					•	-000 0000	-000 0000	-000 0000	-uuu uuuu
ACC	•	•	•	•	•	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	•	•	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	•	•	•	•	•	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	•					--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
		•				-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
			•	•	•	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RTCC	•	•	•	•	•	--00 0111	--00 0111	--00 0111	--uu uuuu
STATUS	•	•	•	•	•	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC0	•	•	•	•	•	-000 0000	-000 0000	-000 0000	-uuu uuuu
LCDC	•	•	•	•	•	--00 0000	--00 0000	--00 0000	--uu uuuu
TMR0	•	•	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	•	•	•	•	•	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1	•					xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	•					00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
		•	•	•	•	0000 1---	0000 1---	0000 1---	uuuu u---
TMR1L		•	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1H		•	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2		•	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2C		•	•	•	•	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR3				•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR3C				•	•	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	•	•	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	•	•	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAWK	•	•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	•	•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	HT56R22	HT56R23	HT56R24	HT56R25	HT56R26	Power-on Reset	$\overline{RES}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (Idle/Sleep)
PB	•					--11 1111	--11 1111	--11 1111	--uu uuuu
		•	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	•					--11 1111	--11 1111	--11 1111	--uu uuuu
		•	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	•					--00 0000	--00 0000	--00 0000	--uu uuuu
		•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	•					1--- 1111	1--- 1111	1--- 1111	u--- uuuu
		•	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	•					1--- 1111	1--- 1111	1--- 1111	u--- uuuu
		•	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	•					---- 0000	---- 0000	---- 0000	---- uuuu
		•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	•					---- -111	---- -111	---- -111	---- -uuu
		•	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	•					---- -111	---- -111	---- -111	---- -uuu
		•	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPU	•					---- -000	---- -000	---- -000	---- -uuu
		•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWM0L	•	•	•	•	•	0000 ---0	0000 ---0	0000 ---0	uuuu ---u
PWM0H	•	•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWM1L	•	•	•	•	•	0000 ---0	0000 ---0	0000 ---0	uuuu ---u
PWM1H	•	•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC1	•	•				---0 ---0	---0 ---0	---0 ---0	---u ---u
		•	•	•		--00 --00	--00 --00	--00 --00	--uu --uu
	•	•	•	•	•	-000 -000	-000 -000	-000 -000	-uuu -uuu
ADPCR	•	•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWM2L	•	•	•	•	•	0000 ---0	0000 ---0	0000 ---0	uuuu ---u
PWM2H	•	•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWM3L		•	•	•	•	0000 ---0	0000 ---0	0000 ---0	uuuu ---u
PWM3H		•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADRL	•	•	•	•	•	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	•	•	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	•	•	•	•	•	01-- -000	01-- -000	01-- -000	uuu- --uu
ACSR	•	•	•	•	•	11-- -000	11-- -000	11-- -000	uu-- -uuu
CLKMOD	•	•	•	•	•	0000 0x11	0000 0x11	0000 0x11	uuuu uuuu
INTEDGE	•	•	•	•	•	---- 0000	---- 0000	---- 0000	---- uuuu
SPICTL0	•	•	•	•	•	111- --0-	111- --0-	111- --0-	uuu- --u-

Register	HT56R22	HT56R23	HT56R24	HT56R25	HT56R26	Power-on Reset	$\overline{RES}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (Idle/Sleep)
SPICTL1	•	•	•	•	•	--00 0000	--00 0000	--00 0000	--uu uuuu
SPIDR	•	•	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
DACTRL	•	•	•	•	•	xxx- ---0	xxx- ---0	xxx- ---0	uuu- ---u
MISC	•	•	•	•	•	0000 1010	0000 1010	0000 1010	uuuu uuuu
MFIC0	•	•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFIC1	•					---0 ---0	---0 ---0	---0 ---0	---u ---u
		•	•			--00 --00	--00 --00	--00 --00	--uu --uu
				•	•	-000 -000	-000 -000	-000 -000	-uuu -uuu
SIMCTRL0	•	•	•	•	•	1110 000-	1110 000-	1110 000-	uuuu uuu-
SIMCTRL1	•	•	•	•	•	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMDR	•	•	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMAR/ SIMCTL2	•	•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR2		•	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	uu-u uuuu
TMR2C		•	•	•	•	00-0 1000	00-0 1000	00-0 1000	uuuu uuuu
TMR3				•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	uu-u uuuu
TMR3C					•	00-0 1000	00-0 1000	00-0 1000	uuuu uuuu
DAL	•	•	•	•	•	0000 ----	0000 ----	0000 ----	uuuu ----
DAH	•	•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PE		•	•			--11 1111	--11 1111	--11 1111	--uu uuuu
				•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC		•	•			--11 1111	--11 1111	--11 1111	--uu uuuu
				•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEPU		•	•			--00 0000	--00 0000	--00 0000	--uu uuuu
				•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PF		•	•			---- 1111	---- 1111	---- 1111	---- uuuu
				•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFC		•	•			---- 1111	---- 1111	---- 1111	---- uuuu
				•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFPU		•	•			---- 0000	---- 0000	---- 0000	---- uuuu
				•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PG				•	•	----- --11	----- --11	----- --11	----- --uu
PGC				•	•	----- --11	----- --11	----- --11	----- --uu
PGPU				•	•	----- --00	----- --00	----- --00	----- --uu
PINMAP		•	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu

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

Input/Output Ports

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

For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]", where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selectable via a register known as PAPU, PBPU, PCPU, PDPU, PEPU, PFPU and PGPU located in the Data Memory. The pull-high resistors are implemented using weak PMOS transistors. Note that pin PC7 does not have a pull-high resistor selection.

Port A Wake-up

If the HALT instruction is executed, the device will enter the Idle/Sleep Mode, where the system clock will stop resulting in power being conserved, a feature that is important for battery and other low-power applications. Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the PA0~PA7 pins from high to low. After a HALT instruction forces the microcontroller into entering the Idle/Sleep Mode, the processor will remain idle or in a low-power state until the logic condition of the selected wake-up pin on Port A changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Note that pins PA0 to PA7 can be selected individually to have this wake-up feature using an internal register known as PAWK, located in the Data Memory.

- **PAWK, PAC, PAPU, PBC, PBPU, PCC, PCPU, PDC, PDPU Register**
  - HT56R22

Register Name	POR	Bit							
		7	6	5	4	3	2	1	0
PAWK	00H	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
PAC	FFH	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	00H	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PBC	3FH	—	—	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	00H	—	—	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PCC	8FH	PCC7	—	—	—	PCC3	PCC2	PCC1	PCC0
PCPU	00H	—	—	—	—	PCPU3	PCPU2	PCPU1	PCPU0
PDC	07H	—	—	—	—	—	PDC2	PDC1	PDC0
PDPU	00H	—	—	—	—	—	PDPU2	PDPU1	PDPU0

"—" Unimplemented, read as "0"

PAWKn: PA wake-up function enable
0: disable
1: enable

PACn/PBCn/PCCn/PDCn: I/O type selection
0: output
1: input

PAPUn/PBPUn/PCPUn/PDPUn: Pull-high function enable
0: disable
1: enable

- **PAWK, PAC, PAPU, PBC, PBPU, PCC, PCPU, PDC, PDPU, PEC, PEPU, PFC, PFPU**
  - HT56R23/HT56R24

Register Name	POR	Bit							
		7	6	5	4	3	2	1	0
PAWK	00H	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
PAC	FFH	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	00H	PAPU\7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PBC	FFH	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	00H	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PCC	FFH	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	00H	—	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PDC	FFH	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	00H	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PEC	3FH	—	—	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	00H	—	—	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PFC	0FH	—	—	—	—	PFC3	PFC2	PFC1	PFC0
PFPU	00H	—	—	—	—	PFPU3	PFPU2	PFPU1	PFPU0

"—" Unimplemented, read as "0"

PAWKn: PA wake-up function enable
0: disable
1: enable

PACn/PBCn/PCCn/PDCn/PECn/PFCn: I/O type selection
0: output
1: input

PAPUn/PBPUn/PCPUn/PDPUn/PEPUn/PFPUn: Pull-high function enable
0: disable
1: enable

- **PAWK, PAC, PAPU, PBC, PBPU, PCC, PCPU, PDC, PDPU, PEC, PEPU, PFC, PFPU, PGC, PGPU**
  - HT56R25/HT56R26

Register Name	POR	Bit							
		7	6	5	4	3	2	1	0
PAWK	00H	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
PAC	FFH	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	00H	PAPU\7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PBC	FFH	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	00H	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PCC	FFH	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	00H	—	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PDC	FFH	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	00H	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PEC	FFH	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	00H	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PFC	FFH	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
PFPU	00H	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0
PGC	03H	—	—	—	—	—	—	PGC1	PGC0
PGPU	00H	—	—	—	—	—	—	PGPU1	PGPU0

"—" Unimplemented, read as "0"

PAWKn: PA wake-up function enable
0: disable
1: enable

PACn/PBCn/PCCn/PDCn/PECn/PFCn/PGCn: I/O type selection
0: output
1: input

PAPUn/PBPUn/PCPUn/PDPUn/PEPUn/PFPUn/PGPUn: Pull-high function enable
0: disable
1: enable

I/O Port Control Registers

Each Port has its own control register, known as PAC, PBC, PCC, PDC, PEC, PFC and PGC which controls the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

Pin-shared Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For some pins, the chosen function of the multi-function I/O pins is set by configuration options while for others the function is set by application program control.

- External Interrupt Input
  The external interrupt pin, INT0/INT1, are pin-shared with an I/O pins. To use the pins as external interrupt inputs the correct bits in the INTC0 register must be programmed. The pin must also be setup as an input by setting bit in the Port Control Register. A pull-high resistor can also be selected via the appropriate port pull-high resistor register. Note that even if the pin is setup as an external interrupt input the I/O function still remains.
- External Timer/Event Counter Input
  The Timer/Event Counter pins, TC0, TC1, TC2 and TC3 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 Configuration option. Note that the corresponding bit of the port control register, must setup the pin as an output to enable the PFD output. If the port control register has setup the pin as an input, then the pin will function as a normal logic input with the usual pull-high selection, even if the PFD function has been selected.
- PWM Outputs
  The PWM function whose outputs are pin-shared with I/O pins. The PWM output functions are chosen using the PWMnL and PWMnH (n=0~3) registers. Note that the corresponding bit of the port control registers, for the output pin, must setup the pin as an output to enable the PWM output. If the pins are setup as inputs, then the pin will function as a normal logic input with the usual pull-high selections, even if the PWM registers have enabled the PWM function.
- SCOM Driver Pins
  Pins PB0~PB3 on Port B can be used as LCD COM driver pins. This function is controlled using the SCOMC register which will generate the necessary 1/2 bias signals on these four pins.
- A/D Inputs
  Each device in this series has eight inputs to the A/D converter. All of these analog inputs are pin-shared with I/O pins. If these pins are to be used as A/D inputs and not as I/O pins then the corresponding PCRn bits in the A/D converter control register, ADPCR, must be properly setup. There are no configuration options associated with the A/D converter. If chosen as I/O pins, then full pull-high resistor configuration options remain, however if used as A/D inputs then any pull-high resistor configuration options associated with these pins will be automatically disconnected.

Pin Remapping Configuration

The pin remapping function enables the function pins INT0/1, TC0/1, PFD, PWM0/1/2 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.

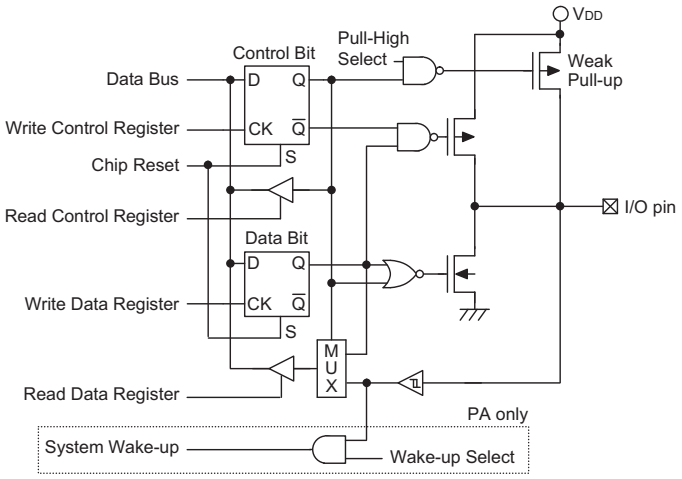
The PMAP0~7 bit in the PINMAP register allows the three function pins INT0/1, TC0/1, PFD, PWM0/1/2 to be remapped to different port pins. After power up, these bits will be reset to zero, which will define the default port pins to which these three functions will be mapped. Changing this bit will move the functions to other port pins.

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

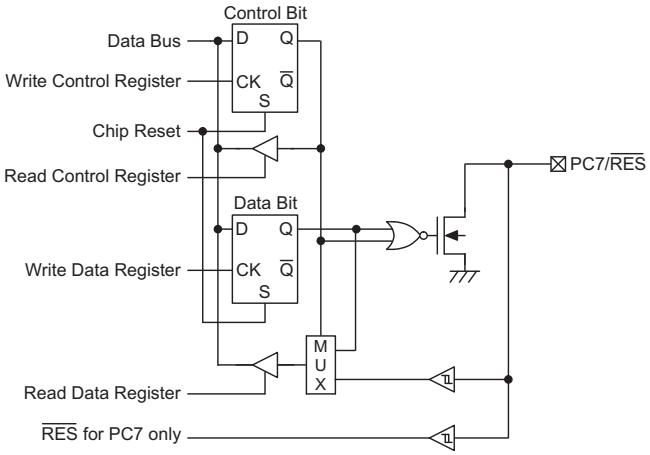
• **HT56R23/HT56R24/HT56R25/HT56R26**

Register Name	POR	Bit							
		7	6	5	4	3	2	1	0
PINMAP	00H	PMAP7	PMAP6	PMAP5	PMAP4	PMAP3	PMAP2	PMAP1	PMAP0
		0: PD2/PWM2 1: PE5/[PWM2]	0 : PD1/PWM1 1: PD5/[PWM1]	0: PD0/PWM0 1: PD4/[PWM0]	0: PA7/TC1 1: PD6/[TC1]	0: PA6/INT1 1: PD7/[INT1]	0: PA5/TC0 1: PE2/[TC0]	0: PA4/INT0 1: PE1/[INT0]	0: PA3/PFD 1: PE0/[PFD]

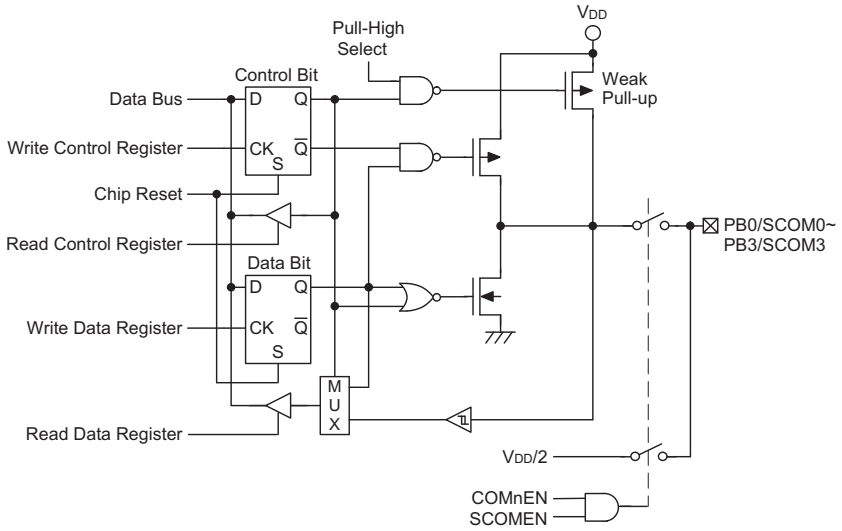
PINMAP Register



Generic Input/Output Ports



PC7 NMOS Input/Output Port



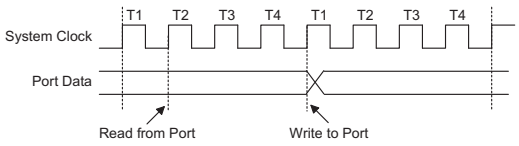
PB Input/Output Port

I/O Pin Structures

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

Programming Considerations

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



Read Modify Write Timing

Pins PA0 to PA7 each have a wake-up 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 of the these pins. Single or multiple pins on Port A can be setup to have this function.

Timer/Event Counters

The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The devices contain several 8-bit and 16-bit count-up timers. As each timer has three different operating modes, they can be configured to operate as a general timer, an external event counter or as a pulse width measurement device. The provision of a prescaler to the clock circuitry of the 8-bit Timer/Event Counter also gives added range to this timer.

There are two types of registers related to the Timer/Event Counters. The first are the registers that contain the actual value of the Timer/Event Counter and into which an initial value can be preloaded. Reading from these registers 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/Event Counter is to be used. The Timer/Event Counters can have the their clock configured to come from an internal clock source. In addition, their clock source can also be configured to come from an external timer pin.

Configuring the Timer/Event Counter Input Clock Source

The internal timer's clock can originate from various sources. The system clock source is used when the Timer/Event Counter is in the timer mode or in the pulse width measurement mode. For the 8-bit Timer/Event Counter this internal clock source is f_{SYS} which is also divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register, TMRnC, bits TnPSC0~TnPSC2. For the 16-bit Timer/Event Counter this internal clock source can be chosen from a combination of internal clocks using a configuration option and the TnS bit in the TMRnC register.

An external clock source is used when the timer is in the event counting mode, the clock source being provided on an external timer pin TMR0, TMR1, TMR2 or TMR3 depending upon which timer is used. Depending upon the condition of the TnE bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.

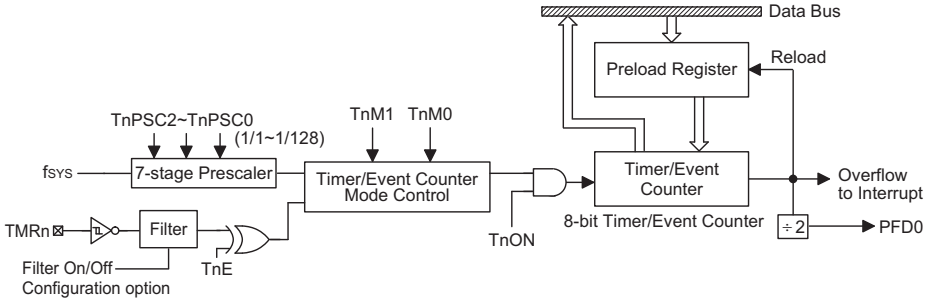
Device	All Devices
No. of 8-bit Timers	3
Timer Name	Timer/Event Counter 0 Timer/Event Counter 2 Timer/Event Counter 3
Timer Register Name	TMR0 TMR2 TMR3
Control Register Name	TMR0C TMR2C TMR3C
No. of 16-bit Timers	1
Timer Name	Timer/Event Counter 1
Timer Register Name	TMR1L/TMR1H
Control Register Name	TMR1C

Timer Registers – TMR0, TMR1L/TMR1H, TMR2, TMR3

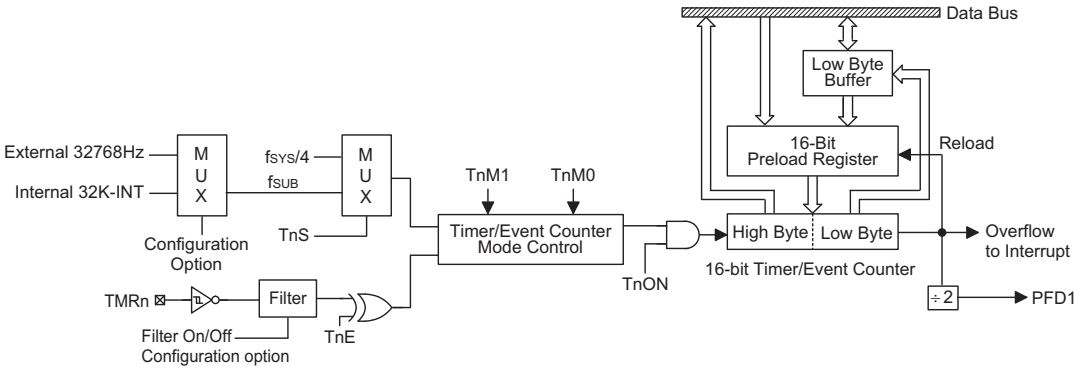
The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. For the 8-bit Timer/Event Counters, these registers are known as TMR0, TMR2 or TMR3. For the 16-bit Timer/Event Counter, a pair of registers are required and are known as TMR1L/TMR1H. 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 or FFFFH for the 16-bit timer at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting.

To achieve a maximum full range count of FFH for the 8-bit timer or FFFFH for the 16-bit timer, the preload registers must first be cleared to all zeros. It should be noted that after power-on, the preload register will be in an unknown condition. Note that if the Timer/Event Counter is switched off and data is written to its preload registers, this data will be immediately written into the actual timer registers. However, if the Timer/Event Counter is enabled and counting, any new data written into the preload data registers during this period will remain in the preload registers and will only be written into the timer registers the next time an overflow occurs.

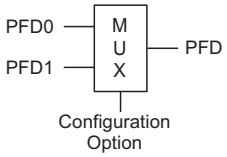
For the 16-bit Timer/Event Counter which has both low byte and high byte timer registers, accessing these registers is carried out in a specific way. It must be noted when using instructions to preload data into the low byte timer register, the data will only be placed in a low byte buffer and not directly into the low byte timer register. The actual transfer of the data into the low byte timer register is only carried out when a write to its associated high byte timer register, namely TMR1H, is executed. On the other hand, using instructions to preload data into the high byte timer register will result in the data being directly written to the high byte timer register. At the same time the data in the low byte buffer will be transferred into its associated low byte timer register. For this reason, the low byte timer register should be written first when preloading data into the 16-bit timer registers. It must also be noted that to read the contents of the low byte timer register, a read to the high byte timer register must be executed first to latch the contents of the low byte timer register into its associated low byte buffer. After this has been done, the low byte timer register can be read in the normal way. Note that reading the low byte timer register will result in reading the previously latched contents of the low byte buffer and not the actual contents of the low byte timer register.



8-bit Timer/Event Counter Structure



16-bit Timer/Event Counter Structure



b7 b0

TnM1	TnM0	—	TnON	TnE	TnPSC2	TnPSC1	TnPSC0

TMRnC Register (n=0, 2, 3)

Timer prescaler rate select

TnPSC2	TnPSC1	TnPSC0	Timer Rate
0	0	0	1:1
0	0	1	1:2
0	1	0	1:4
0	1	1	1:8
1	0	0	1:16
1	0	1	1:32
1	1	0	1:64
1	1	1	1:128

Event Counter active edge select
1: count on falling edge
0: count on rising edge

Pulse Width Measurement active edge select
1: start counting on rising edge, stop on falling edge
0: start counting on falling edge, stop on rising edge

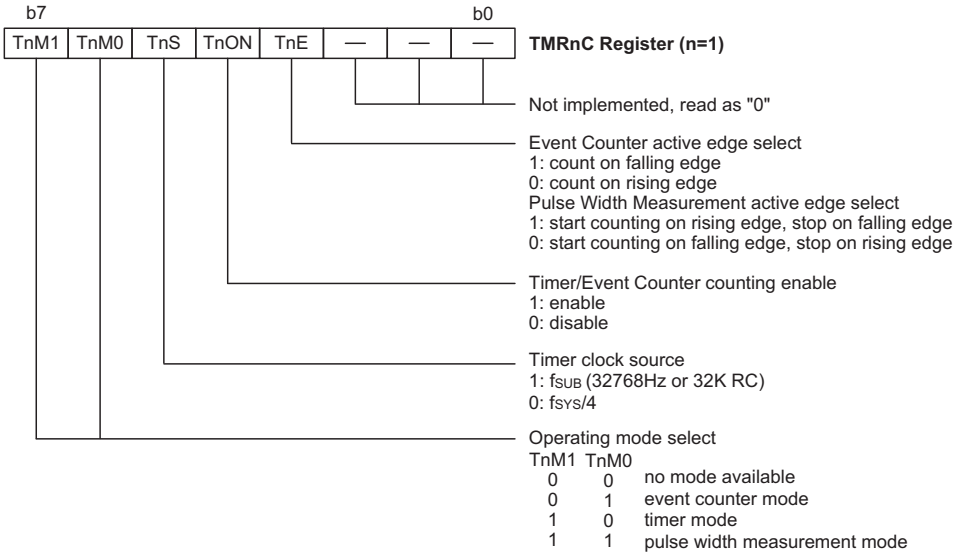
Timer/Event Counter counting enable
1: enable
0: disable

Not implemented, read as "0"

Operating mode select

TnM1	TnM0	
0	0	no mode available
0	1	event counter mode
1	0	timer mode
1	1	pulse width measurement mode

Timer/Event Counter Control Register – TMRnC



Timer/Event Counter Control Register – TMRnC

Timer Control Registers –
TMR0C, TMR1C, TMR2C, TMR3C

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.

It is the Timer Control Register together with its corresponding timer registers that control the full operation of the Timer/Event Counters. Before the timers can be used, it is essential that the appropriate Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width measurement mode, bits 7 and 6 of the corresponding 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, depending upon which timer is used, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. For timers that have prescalers, 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 measurement 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 TnE. An additional T1S bit in the 16-bit Timer/Event Counter control register is used to determine the clock source for the Timer/Event Counter.

• **TMR0C Register**

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

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

Bit 5 Not implemented, read as ″0″

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

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

Bit 2~0 **T0PSC2, T0PSC1, T0PSC0**: Timer prescaler rate selection
Timer internal clock=
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

- **TMR1C Register**
  - HT56R22

Bit	7	6	5	4	3	2	1	0
Name	T1M1	T1M0	—	T1ON	T1EG	T1PSC2	T1PSC1	T1PSC0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	1	0	0	0

Bit 7,6 **T1M1, T1M0**: Timer1 operation mode selection
00: no mode available
01: event counter mode
10: timer mode
11: pulse width capture mode

Bit 5 Not implemented, read as ″0″

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

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

Bit 2~0 **T1PSC2, T1PSC1, T1PSC0**: Timer prescaler rate selection
Timer internal clock=
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

- **TMR1C Register**

  ♦ HT56R23/HT56R24/HT56R25/HT56R26

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

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

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

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

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

Bit 2~0 unimplemented, read as ″0″

- **TMR2C Register**

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

Bit 7, 6 **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 unimplemented, read as ″0″

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

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

Bit 2~0 **T2PSC2, T2PSC1, T2PSC0**: Timer prescaler rate selection
Timer internal clock=
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

- **TMR3C Register**
  - ♦ HT56R25/26

Bit	7	6	5	4	3	2	1	0
Name	T3M1	T3M0	—	T3ON	T3EG	T3PSC2	T3PSC1	T3PSC0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	1	0	0	0

Bit 7, 6 **T3M1, T3M0**: Timer 3 Operation mode selection
00: no mode available
01: event counter mode
10: timer mode
11: pulse width capture mode

Bit 5 unimplemented, read as ″0″

Bit 4 **T3ON**: Timer/event counter counting enable
0: disable
1: enable

Bit 3 **T3EG**:
Event counter active edge selection
0: count on raising edge
1: count on falling edge
Pulse Width Capture active edge selection
0: start counting on falling edge, stop on rasing edge
1: start counting on raising edge, stop on falling edge

Bit 2~0 **T3PSC2, T3PSC1, T3PSC0**: Timer prescaler rate selection
Timer internal clock=
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

Timer Mode

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

Control Register Operating Mode Select Bits for the Timer Mode	Bit7	Bit6
	1	0

In this mode the internal clock is used as the timer clock. The timer input clock source is either f_{SYS} , $f_{SYS}/4$ or the LXT oscillator. However, this timer clock source is further divided by a prescaler, the value of which is determined by the bits TnPSC2~TnPSC0 in the Timer Control Register. The timer-on bit, TnON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one; when the timer is full and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupt is one of the wake-up sources, however, the internal interrupts can be disabled by ensuring that the TnE bits of the INTCn register are reset to zero.

Event Counter Mode

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

Control Register Operating Mode Select Bits for the Event Counter Mode	Bit7	Bit6
	0	1

In this mode, the external timer TCn pin, is used as the Timer/Event Counter clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit TnON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. If the Active Edge Select bit, TnEG, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the TnEG is high, the counter will increment each time the external timer pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register, is reset to zero.

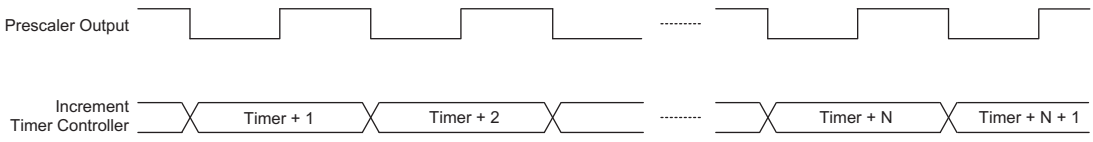
As the external timer pin is shared with an I/O pin, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Event Counting Mode,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

Pulse Width Capture Mode

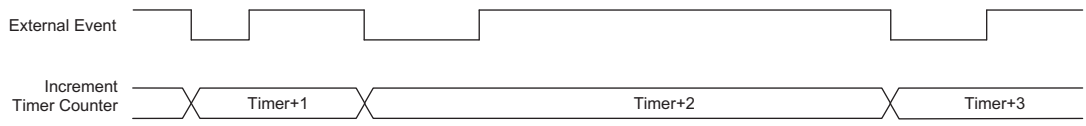
In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Pulse Width Capture Mode	Bit7	Bit6
	1	1

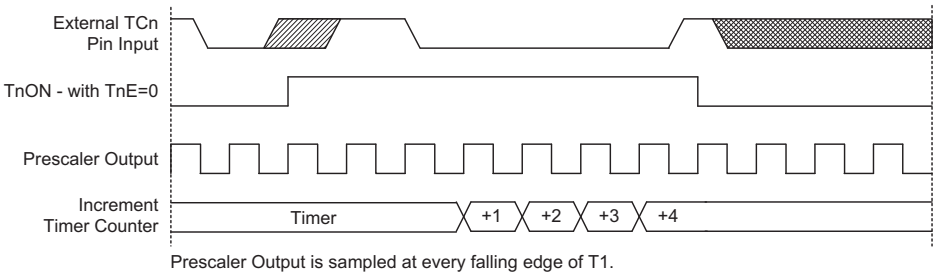
In this mode the internal clock, f_{SYS} , $f_{SYS}/4$ or the LXT, is used as the internal clock for the 8-bit Timer/Event Counter. However, the clock source, f_{SYS}, for the 8-bit timer is further divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits



Timer Mode Timing Chart



Event Counter Mode Timing Chart (TnEG=1)



Pulse Width Capture Mode Timing Chart (TnE=0)

TnPSC2~TnPSC0, which are bits 2~0 in the Timer Control Register. After the other bits in the Timer Control Register have been setup, the enable bit TnON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the external timer pin.

If the Active Edge Select bit TnEG, which is bit 3 of the Timer Control Register, is low, once a high to low transition has been received on the external timer pin, the Timer/Event Counter will start counting until the external timer pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. If the Active Edge Select bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. It is important to note that in the pulse width capture Mode, the enable bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under program control.

The residual value in the Timer/Event Counter, which can now be read by the program, therefore represents the length of the pulse received on the TCn pin. As the enable bit has now been reset, any further transitions on the external timer pin will be ignored. The timer cannot begin further pulse width capture until the enable bit is set high again by the program. In this way, single shot pulse measurements can be easily made.

It should be noted that in this mode the Timer/Event Counter is controlled by logical transitions on the external timer pin and not by the logic level. When the Timer/Event Counter is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register, is reset to zero.

As the TCn pin is shared with an I/O pin, to ensure that the pin is configured to operate as a pulse width capture pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the pulse width capture Mode, the second is to ensure that the port control register configures the pin as an input.

Prescaler

Bits TnPSC0~TnPSC2 of the TMRnC register can be used to define a division ratio for the internal clock source of the Timer/Event Counter enabling longer time out periods to be setup.

Programmable Frequency Divider – PFD

The Programmable Frequency Divider provides a means of producing a variable frequency output suitable for applications requiring a precise frequency generator.

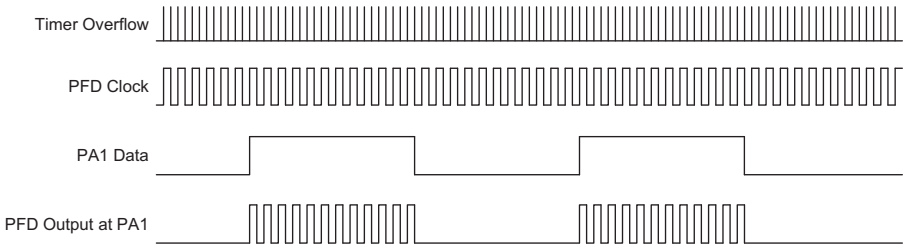
The PFD output is pin-shared with the I/O pin PA3. The PFD function is selected via configuration option, however, if not selected, the pin can operate as a normal I/O pin.

The clock source for the PFD circuit can originate from either Timer/Event Counter 0 or Timer/Event Counter 1 overflow signal selected via configuration option. The output frequency is controlled by loading the required values into the timer registers and prescaler registers to give the required division ratio. The timer will begin to count-up from this preload register value until full, at which point an overflow signal is generated, causing the PFD output to change state. The timer will then be automatically reloaded with the preload register value and continue counting-up.

For the PFD output to function, it is essential that the corresponding bit of the Port A control register PAC bit 3 is setup as an output. If setup as an input the PFD output will not function, however, the pin can still be used as a normal input pin. The PFD output will only be activated if bit PA3 is set to ″1″. This output data bit is used as the on/off control bit for the PFD output. Note that the PFD output will be low if the PA3 output data bit is cleared to ″0″.

Using this method of frequency generation, and if a crystal oscillator is used for the system clock, very precise values of frequency can be generated.

Bits TnPSC0~TnPSC2 of the control register can be used to define the pre-scaling stages of the internal clock source of the Timer/Event Counter. The



PFD Function

Timer/Event Counter overflow signal can be used to generate signals for the PFD and Timer Interrupt.

I/O Interfacing

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

Programming Considerations

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

When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, this should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialised before using them for the first time. The associated timer enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The edge select, timer mode and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application. It is also important to ensure that an initial value is first loaded into the timer registers before the timer is switched on; this is because after power-on the initial values of the timer registers are unknown. After the timer has been initialised the timer can be turned on and off by controlling the enable bit in the timer control register.

When the Timer/Event Counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the Timer/Event Counter interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in a Power-down condition. This situation may occur if the Timer/Event Counter is in the Event Counting Mode and if the external signal continues to change state. In such a case, the Timer/Event Counter will continue to count these external events and if an overflow occurs the device will be woken up from its Power-down condition. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the ″HALT″ instruction to enter the Idle/Sleep Mode.

Timer Program Example

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

• **PFD Programming Example**

```
org 04h             ; external interrupt vector

org 08h             ; Timer Counter 0 interrupt vector
jmp tmr0int         ; jump here when Timer 0 overflows
:         :
org 20h             ; main program
:         :
;internal Timer 0 interrupt routine
tmr0int:
:
; Timer 0 main program placed here
:

:
begin:
;setup Timer 0 registers
mov  a,09bh         ; setup Timer 0 preload value
mov tmr0,a
mov a,081h          ; setup Timer 0 control register
mov tmr0c,a         ; timer mode and prescaler set to /2
;setup interrupt register
mov a,00dh          ; enable master interrupt and both timer interrupts
mov intc0,a
:         :
set tmr0c.4         ; start Timer 0
:         :
```

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 12~15 stage counter which sets the division ratio of the clock source. This division ratio is controlled by the time base divider configuration option. The clock source is selected using a peripheral clock configuration option.

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

Pulse Width Modulator

The devices contains a series of Pulse Width Modulation, PWM, outputs. Useful for such applications such as motor speed control, the PWM function provides an output with a fixed frequency but with a duty cycle that can be varied by setting particular values into the corresponding PWM register.

Part No.	Channels	PWM Mode	Output Pin	Register Names
All devices	4	8+4	PD0~PD3	PWM0L~PWM3L PWM0H~PWM3H

PWM Overview

A register pair, located in the Data Memory is assigned to each Pulse Width Modulator output and are known as the PWM registers. It is in each register pair that the 12-bit value, which represents the overall duty cycle of one modulation cycle of the output waveform, should be placed. The PWM registers also contain the enable/disable control bit for the PWM outputs. To increase the PWM modulation frequency, each modulation cycle is modulated into sixteen individual modulation sub-sections, known as the 8+4 mode. Note that it is only necessary to write the required modulation value into the corresponding PWM register as the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware. The PWM clock source is the system clock f_{SYS}.

This method of dividing the original modulation cycle into a further 16 sub-cycles enables the generation of higher PWM frequencies, which allow a wider range of applications to be served. As long as the periods of the generated PWM pulses are less than the time constants of the load, the PWM output will be suitable as such long time constant loads will average out the pulses of the PWM output. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood. As the PWM clock is the system clock, f_{SYS}, and as the PWM value is 12-bits wide, the overall PWM cycle frequency is $f_{SYS}/4096$. However, when in the 8+4 mode of operation, the PWM modulation frequency will be $f_{SYS}/256$.

PWM Modulation Frequency	PWM Cycle Frequency	PWM Cycle Duty
$f_{SYS}/256$	$f_{SYS}/4096$	(PWM register value)/4096

8+4 PWM Mode Modulation

Each full PWM cycle, as it is 12-bits wide, has 4096 clock periods. However, in the 8+4 PWM mode, each PWM cycle is subdivided into sixteen individual sub-cycles known as modulation cycle 0 ~ modulation cycle 15, denoted as ″i″ in the table. Each one of these sixteen sub-cycles contains 256 clock cycles. In this mode, a modulation frequency increase of sixteen is achieved. The 12-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit4~bit11 is denoted here as the DC value. The second group which consists of bit0~bit3 is known as the AC value. In the 8+4 PWM mode, the duty cycle value of each of the two modulation sub-cycles is shown in the following table.

Parameter	AC (0~15)	DC (Duty Cycle)
Modulation cycle i (i=0~15)	i<AC	$\frac{DC+1}{256}$
	i≥AC	$\frac{DC}{256}$

8+4 Mode Modulation Cycle Values

The accompanying diagram illustrates the waveforms associated with the 8+4 mode of PWM operation. It is important to note how the single PWM cycle is subdivided into 16 individual modulation cycles, numbered 0~15 and how the AC value is related to the PWM value.

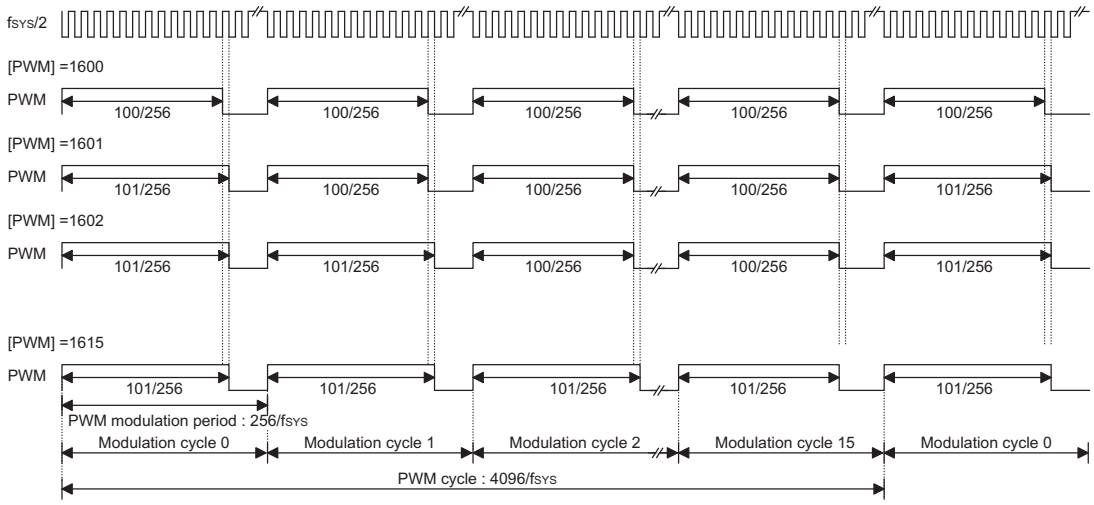
PWM Output Control

The four PWM0~PWM3 outputs are shared with pins PD0~PD3. To operate as a PWM output and not as an I/O pin, bit 0 of the relevant PWM register bit must be set high. A zero must also be written to the corresponding bit in the PDC port control register, to ensure that the PWM0 output pin is setup as an output. After these two initial steps have been carried out, and of course after the required PWM 12-bit value has been written into the PWM register pair register, writing a ″1″ to the corresponding PD data register will enable the PWM data to appear on the pin. Writing a ″0″ to the bit will disable the PWM output function and force the output low. In this way, the Port D data output register bits, can also be used as an on/off control for the PWM function. Note that if the enable bit in the PWM register is set high to enable the PWM function, but a ″1″ has been written to its corresponding bit in the PDC control register to configure the pin as an input, then the pin can still function as a normal input line, with pull-high resistor selections.

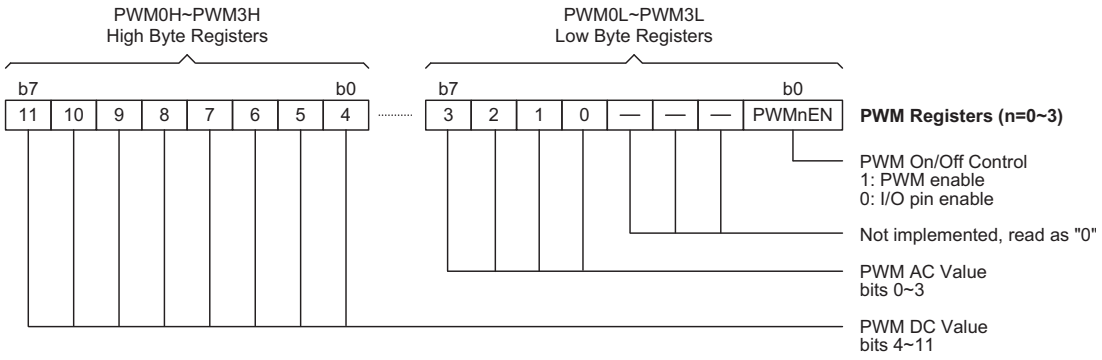
PWM Programming Example

The following sample program shows how the PWM output is setup and controlled.

```
mov a,64h       ; setup PWM0 value to 1600 decimal which is 640H
mov pwm0h,a     ; setup PWM0H register value
clr pwm0l       ; setup PWM0L register value
clr pdc.0       ; setup pin PD0 as an output
set pwm0en      ; set the PWM0 enable bit
set pd.0        ; Enable the PWM0 output
 :   :
 :   :
clr pd.0        ; PWM0 output disabled – PD0 will remain low
```



8+4 PWM Mode



PWM Register Pairs

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 Data Registers – ADRL, ADRH

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

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

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

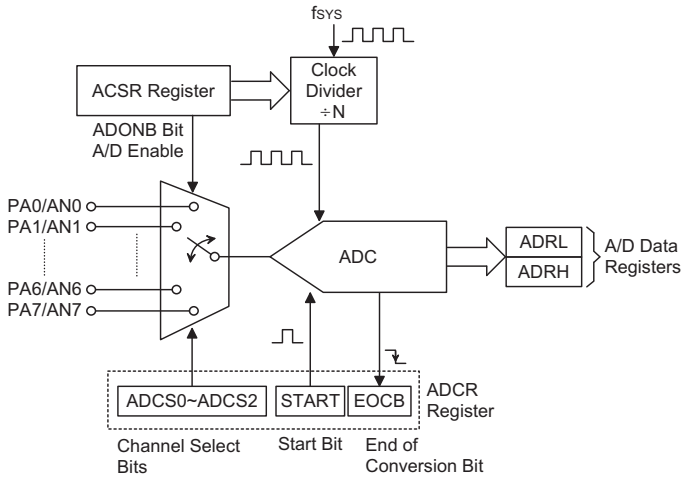
A/D Data Registers

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

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

The ACS2~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 ACS2~ACS0 bits in the ADCR register to determine which analog channel is actually connected to the internal A/D converter.

The ADPCR control register contains the PCR7~PCR0 bits which determine which pins on PA7~PA0 are used as analog inputs for the A/D converter and which pins are to be used as normal I/O pins. If PCR7~PCR0 has a value of "11111111", then all eight pins, namely AN7~AN0 will all be set as analog inputs. Note that if the PCR7~PCR0 bits are all set to zero, then all the PA7~PA0 pins will be setup as normal I/Os.



A/D Converter Structure

- **ADRH, ADRL Register**

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

″x″ unknown

unimplemented, read as ″0″

D11~D0: ADC conversion data

- **ADCR Register**

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

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

Bit 5~3 unimplemented, read as ″0″

Bit 2~0 **ACS2~ACS0**: Select A/D channel
000 AN0
001 AN1
010 AN2
011 AN3
100 AN4
101 AN5
110 AN6
111 AN7

• **ADPCR 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	x	0	0	0	0	0	0	0

″x″ unknown

Port PA - A/D converter input pin selection

Bit 7 **PCR7**: PA7 or AN7
0: PA7 I/O pin or other pin-shared function
1: AN7 A/D converter input

Bit 6 **PCR6**: PA6 or AN6
0: PA6 I/O pin or other pin-shared function
1: AN6 A/D converter input

Bit 5 **PCR5**: PA5 or AN5
0: PA5 I/O pin or other pin-shared function
1: AN5 A/D converter input

Bit 4 **PCR4**: PA4 or AN4
0: PA4 I/O pin or other pin-shared function
1: AN4 A/D converter input

Bit 3 **PCR3**: PA3 or AN3
0: PA3 I/O pin or other pin-shared function
1: AN3 A/D converter input

Bit 2 **PCR2**: PA2 or AN2
0: PA2 I/O pin or other pin-shared function
1: AN2 A/D converter input

Bit 1 **PCR1**: PA1 or AN1
0: PA1 I/O pin or other pin-shared function
1: AN1 A/D converter input

Bit 0 **PCR0**: PA0 or AN0
0: PA0 I/O pin or other pin-shared function
1: AN0 A/D converter input

• **ACSR Register**

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

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

Bit 6 **ADONB**: ADC module power on/off control bit
0: ADC module power on
1: ADC module power off
Note: 1. it is recommended to set ADONB=1 before entering idle/sleep to reduce power consumption
2. ADONB=1 will power down the ADC module.

Bit 5~3 unimplemented, read as ″0″

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

The START bit in the register is used to start and reset the A/D converter. When themicrocontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the EOCB bit in the ADCR register will be set to a "1" and the analog to digital converter will be reset. It is the START bit that is used to control the overall start operation of the internal analog to digital converter.

The EOCB bit in the ADCR register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to "0" by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the EOCB bit in the ADCR register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock f_{SYS}, is first divided by a division ratio, the value of which is determined by the ADCS2, ADCS1 and ADCS0 bits in the ACSR register.

Controlling the power on/off function of the A/D converter circuitry is implemented using the value of the ADONB bit.

Although the A/D clock source is determined by the system clock f_{SYS}, and by bits ADCS2, ADCS1 and ADCS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the minimum value of permissible A/D clock period, t_{AD}, is 0.5μs, care must be taken for system clock speeds in excess of 4MHz. For system clock speeds in excess of 4MHz, the ADCS2, ADCS1 and ADCS0 bits should not be set to "000". Doing so will give A/D clock periods that are less than the minimum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

A/D Input Pins

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

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

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, ADCS1 and ADCS0 in the register.
- Step 2
  Select which pins are to be used as A/D inputs and configure them as A/D input pins by correctly programming the PCR7~PCR0 bits in the ADPCR 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 ACS2~ACS0 bits which are also contained in the register.
- Step 5
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, in the INTC0 interrupt control register must be set to ″1″, the multi-function interrupt enable bit, EMFI, in the INTC1 register and the A/D converter interrupt bit, EADI, in the INTC1 register must also be set to ″1″.
- Step 6
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR register from ″0″ to ″1″ and then to ″0″ again. Note that this bit should have been originally set to ″0″.
- Step 7
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value. As an alternative method if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

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

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

The setting up and operation of the A/D converter function is fully under the control of the application program as there are no configuration options associated with the A/D converter. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is $16t_{AD}$ where t_{AD} is equal to the A/D clock period.

Programming Considerations

When programming, special attention must be given to the PCR7~PCR0 bits in the ADPCR 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. 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. The ADONB bit should be set high before entering any of the low power operating modes or before a HALT instruction is executed to reduce power consumption.

A/D Transfer Function

As the device contain 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 VDD voltage, this gives a single bit analog input value of $V_{DD}/4096$. The diagram show the ideal transfer function between the analog input value and the digitised output value for the A/D converter.

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

A/D Programming Example

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

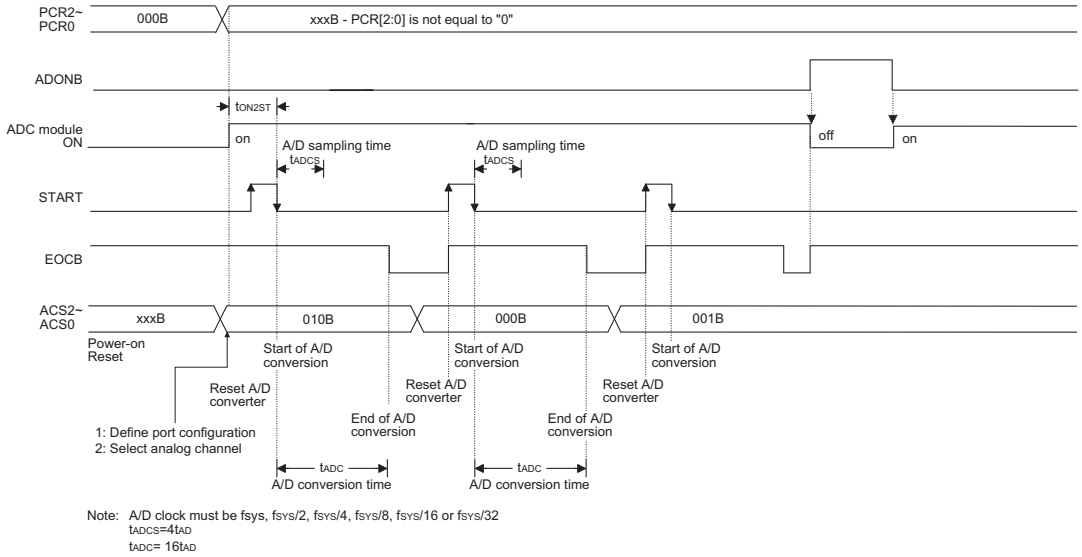
Example: using an EOCB polling method to detect the end of conversion

```
    clr  EADI                  ; disable ADC interrupt
    mov  a,00000001B
    mov  ACSR,a                ; select fSYS/8 as A/D clock and turn on ADONB bit
    mov  a,00100000B           ; setup ADCR register to configure Port PB0~PB3
                               ; as A/D inputs
    mov  ADCR,a                ; and select AN0 to be connected to the A/D converter
         :
         :                     ; As the Port B channel bits have changed the
                               ; following START
                               ; signal (0-1-0) must be issued
                               ; instruction cycles
         :
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
```

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

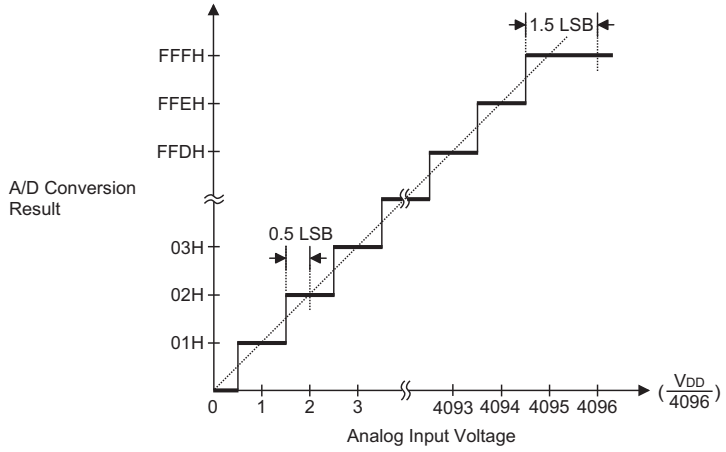
```
    clr  EADI                  ; disable ADC interrupt
    mov  a,00000001B
    mov  ACSR,a                ; select fSYS/8 as A/D clock and turn on ADONB bit

    mov  a,00100000B           ; setup ADCR register to configure Port PB0~PB3
                               ; as A/D inputs
    mov  ADCR,a                ; and select AN0 to be connected to the A/D
         :
                               ; As the Port B channel bits have changed the
                               ; following START signal(0-1-0) must be issued
                               ;
         :
Start_conversion:
    clr  START
    set  START                 ; reset A/D
    clr  START                 ; start A/D
    clr  ADF                   ; clear ADC interrupt request flag
    set  EADI                  ; enable ADC interrupt
    set  EMFI                  ; enable multi-function interrupt
    set  EMI                   ; enable global interrupt
         :
         :
         :
; ADC interrupt service routine
ADC_:
    mov  acc_stack,a           ; save ACC to user defined memory
    a,STATUS
    mov  status_stack,a        ; save STATUS to user defined memory
         :
         :
    mov  a,ADRL                ; read low byte conversion result value
    mov  adrl_buffer,a         ; save result to user defined register
    mov  a,ADRH                ; read high byte conversion result value
    mov  adrh_buffer,a         ; save result to user defined register
         :
         :
EXIT__ISR:
    mov  a,status_stack
    mov  STATUS,a              ; restore STATUS from user defined memory
    mov  a,acc_stack           ; restore ACC from user defined memory
    clr  ADF                   ; clear ADC interrupt flag
    reti
```



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

A/D Conversion Timing



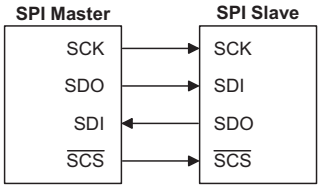
Ideal A/D Transfer Function

Serial Interface Function

The device contains a Serial Interface Function, which includes both the four line SPI interface and the two line I^2C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I^2C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins therefore the SIM interface function must first be selected using a configuration option. As both interface types share the same pins and registers, the choice of whether the SPI or I^2C type is used is made using a bit in an internal register.

SPI Interface

The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the programming requirements when communicating with external hardware devices.

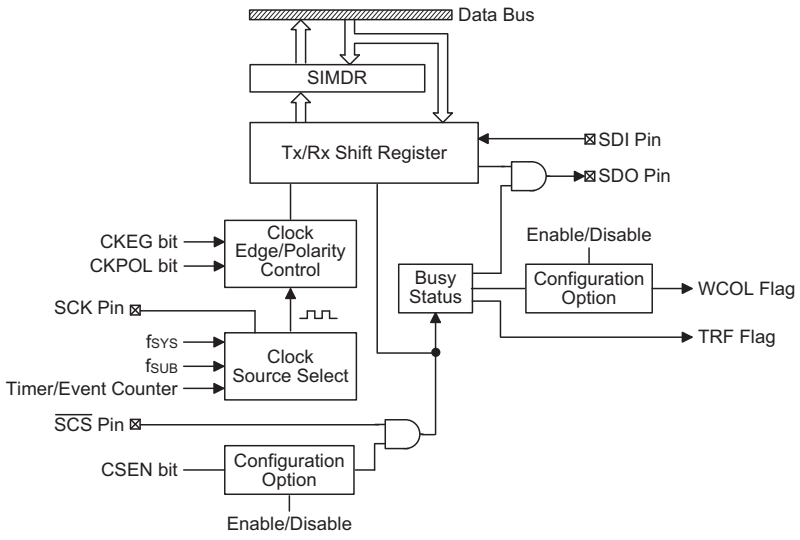


SPI Master/Slave Connection

The communication is full duplex and operates as a slave/master type, where the MCU can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, here, as only a single select pin, $\overline{SCS}$, is provided only one slave device can be connected to the SPI bus.

- SPI Interface Operation
  The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, SCK is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I^2C function pins, the SPI interface must first be enabled by selecting the SIM enable configuration option and setting the correct bits in the SIMCTL0/SIMCTL2 register. After the SPI configuration option has been configured it can also be additionally disabled or enabled using the SIMEN bit in the SIMCTL0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ pin only one slave device can be utilised.

  The SPI function in this device offers the following features:
  - Full duplex synchronous data transfer
  - Both Master and Slave modes
  - LSB first or MSB first data transmission modes
  - Transmission complete flag
  - Rising or falling active clock edge
  - WCOL and CSEN bit enabled or disable select



SPI Block Diagram

The status of the SPI interface pins is determined by a number of factors such as whether the device is in the master or slave mode and upon the condition of certain control bits such as CSEN, SIMEN and $\overline{SCS}$. In the table I, Z represents an input floating condition.
There are several configuration options associated with the SPI interface. One of these is to enable the SIM function which selects the SIM pins rather than normal I/O pins. Note that if the configuration option does not select the SIM function then the SIMEN bit in the SIMCTL0 register will have no effect. Another two SIM configuration options determine if the CSEN and WCOL bits are to be used.

Configuration Option	Function
SIM Function	SIM interface or I/O pins
SPI CSEN bit	Enable/Disable
SPI WCOL bit	Enable/Disable

SPI Interface Configuration Options

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMDR data register and two control registers SIMCTL0 and SIMCTL2. Note that the SIMCTL1 register is only used by the I^2C interface.

The SIMDR register is used to store the data being transmitted and received. The same register is used by both the SPI and I^2C functions. Before the microcontroller writes data to the SPI bus, the actual data to be transmitted must be placed in the SIMDR register. After the data is received from the SPI bus, the microcontroller can read it from the SIMDR register. Any transmission or reception of data from the SPI bus must be made via the SIMDR register.

Bit	7	6	5	4	3	2	1	0
Label	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
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

There are also two control registers for the SPI interface, SIMCTL0 and SIMCTL2. Note that the SIMCTL2 register also has the name SIMAR which is used by the I^2C function. The SIMCTL1 register is not used by the SPI function, only by the I^2C function. Register SIMCTL0 is used to control the enable/disable function and to set the data transmission clock frequency. Although not connected with the SPI function, the SIMCTL0 register is also used to control the Peripheral Clock prescaler. Register SIMCTL2 is used for other control functions such as LSB/MSB selection, write collision flag etc.

The following gives further explanation of each SIMCTL1 register bit:

- SIMIDLE
  The SIMIDLE bit is used to select if the SPI interface continues running when the device is in the IDLE mode. Setting the bit high allows the SPI interface to maintain operation when the device is in the Idle mode. Clearing the bit to zero disables any SPI operations when in the Idle mode.
  This SPI/I^2C idle mode control bit is located at CLKMOD register bit4.

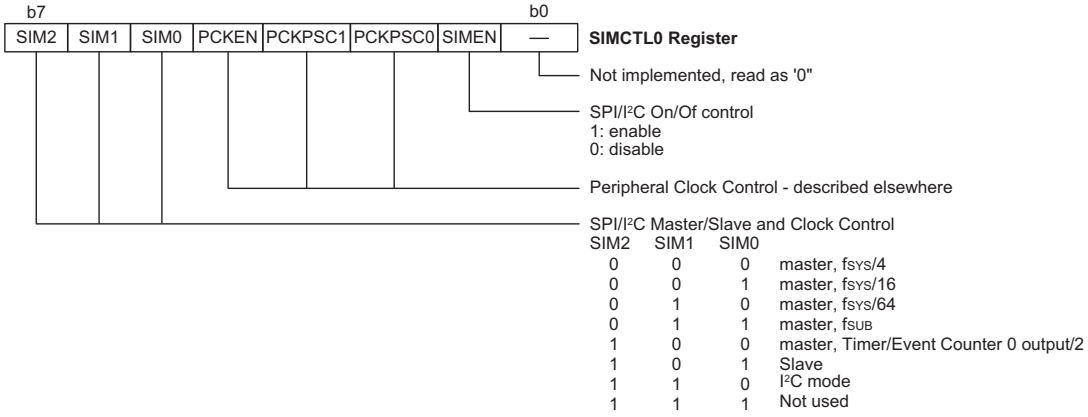
- SIMEN
  The bit is the overall on/off control for the SPI interface. When the SIMEN bit is cleared to zero to disable the SPI interface, the SDI, SDO, SCK and $\overline{SCS}$ lines will be in a floating condition and the SPI operating current will be reduced to a minimum value. When the bit is high the SPI interface is enabled. The SIMconfiguration option must have first enabled the SIM interface for this bit to be effective. Note that when the SIMEN bit changes from low to high the contents of the SPI control registers will be in an unknown condition and should therefore be first initialised by the application program.

- SIM0~SIM2
  These bits setup the overall operating mode of the SIM function. As well as selecting if the I^2C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from the Timer/Event Counter. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

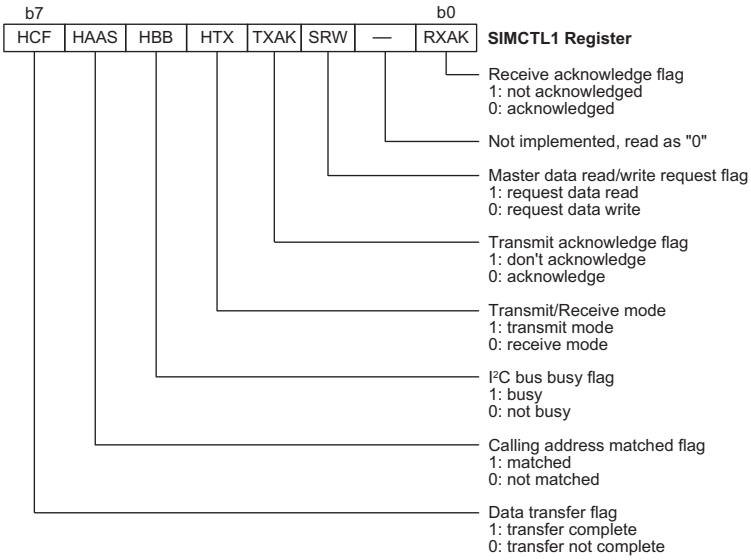
Pin	Master/Salve SIMEN=0	Master – SIMEN=1		Slave – SIMEN=1		
		CSEN=0	CSEN=1	CSEN=0	CSEN=1 SCS=0	CSEN=1 SCS=1
$\overline{SCS}$	Z	Z	L	Z	I, Z	I, Z
SDO	Z	O	O	O	O	Z
SDI	Z	I, Z	I, Z	I, Z	I, Z	Z
SCK	Z	H: CKPOL=0 L: CKPOL=1	H: CKPOL=0 L: CKPOL=1	I, Z	I, Z	Z

Note: "Z" floating, "H" output high, "L" output low, "I" Input, "O"output level, "I,Z" input floating (no pull-high)

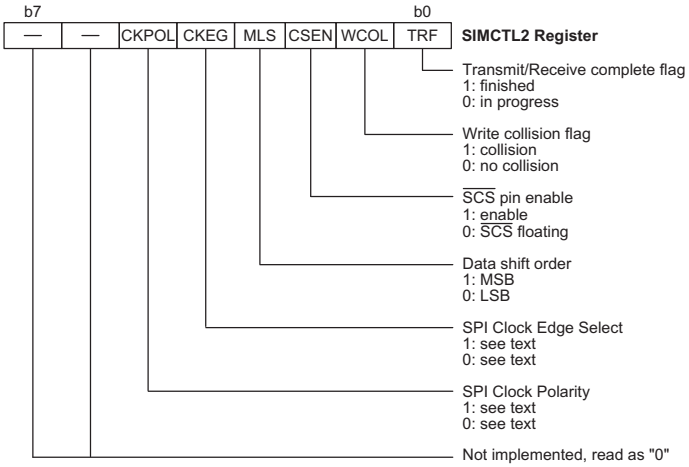
SPI Interface Pin Status



SPI/I²C Control Register – SIMCTL0



I²C Control Register – SIMCTL1



SPI Control Register – SIMCTL2

SIM0	SIM1	SIM2	SPI Master/Slave Clock Control and I2C Enable
0	0	0	SPI Master, $f_{SYS}/4$
0	0	1	SPI Master, $f_{SYS}/16$
0	1	0	SPI Master, $f_{SYS}/64$
0	1	1	SPI Master, f_{SUB}
1	0	0	SPI Master Timer/Event Counter 0 output/2
1	0	1	SPI Slave
1	1	0	I^2C mode
1	1	0	Not used

SPI Control Register – SIMCTL2

The SIMCTL2 register is also used by the I^2C interface but has the name SIMAR

- TRF
  The TRF bit is the Transmit/Receive Complete flag and is set high automatically when an SPI data transmission is completed, but must be cleared by the application program. It can be used to generate an interrupt.
- WCOL
  The WCOL bit is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMDR register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program. Note that using the WCOL bit can be disabled or enabled via configuration option.
- CSEN
  The CSEN bit is used as an on/off control for the $\overline{SCS}$ pin. If this bit is low then the $\overline{SCS}$ pin will be disabled and placed into a floating condition. If the bit is high the $\overline{SCS}$ pin will be enabled and used as a select pin. Note that using the CSEN bit can be disabled or enabled via configuration option.
- MLS
  This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.
- CKEG and CKPOL
  These two bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOL bit determines the base condition of the clock line, if the bit is high then the SCK line will be low when the clock is inactive. When the CKPOL bit is low then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOL.

CKPOL	CKEG	SCK Clock Signal
0	0	High Base Level Active Rising Edge
0	1	High Base Level Active Falling Edge
1	0	Low Base Level Active Falling Edge
1	1	Low Base Level Active Rising Edge

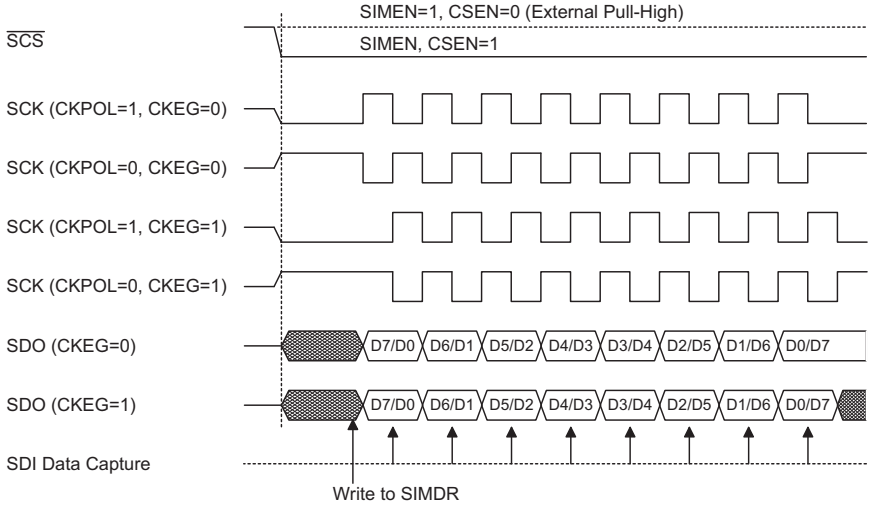
SPI Communication

After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMDR register, transmission/reception will begin simultaneously. When the data transfer is complete, the TRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMDR register will be transmitted and any data on the SDI pin will be shifted into the SIMDR register. The master should output an $\overline{SCS}$ signal to enable the slave device before a clock signal is provided and slave data transfers should be enabled/disabled before/after an $\overline{SCS}$ signal is received.
The SPI will continue to function even after a HALT instruction has been executed.

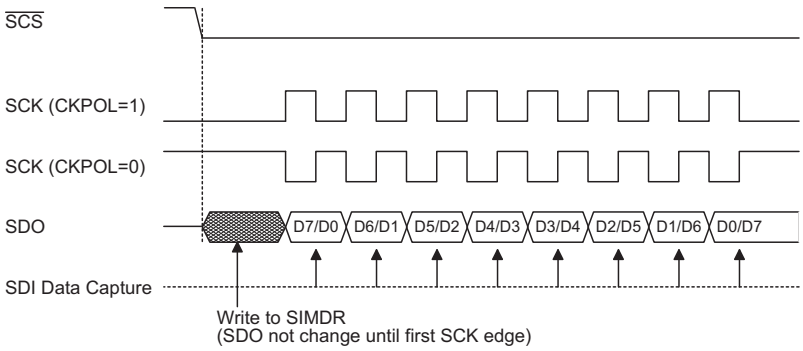
I^2C Interface

The I^2C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication protocol and the ability to accommodate multiple devices on the same bus has made it an extremely popular interface type for many applications.

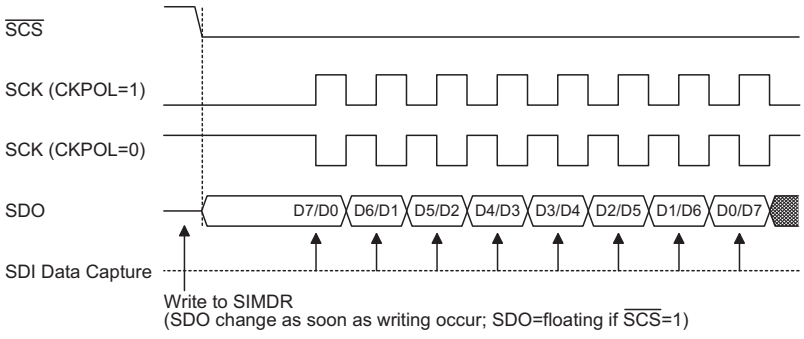
- I^2C Interface Operation
  The I^2C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I^2C bus is identified by a unique address which will be transmitted and received on the I^2C bus.
  When two devices communicate with each other on the bidirectional I^2C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For these devices, which only operates in slave mode, there are two methods of transferring data on the I^2C bus, the slave transmit mode and the slave receive mode.



SPI Master Mode Timing

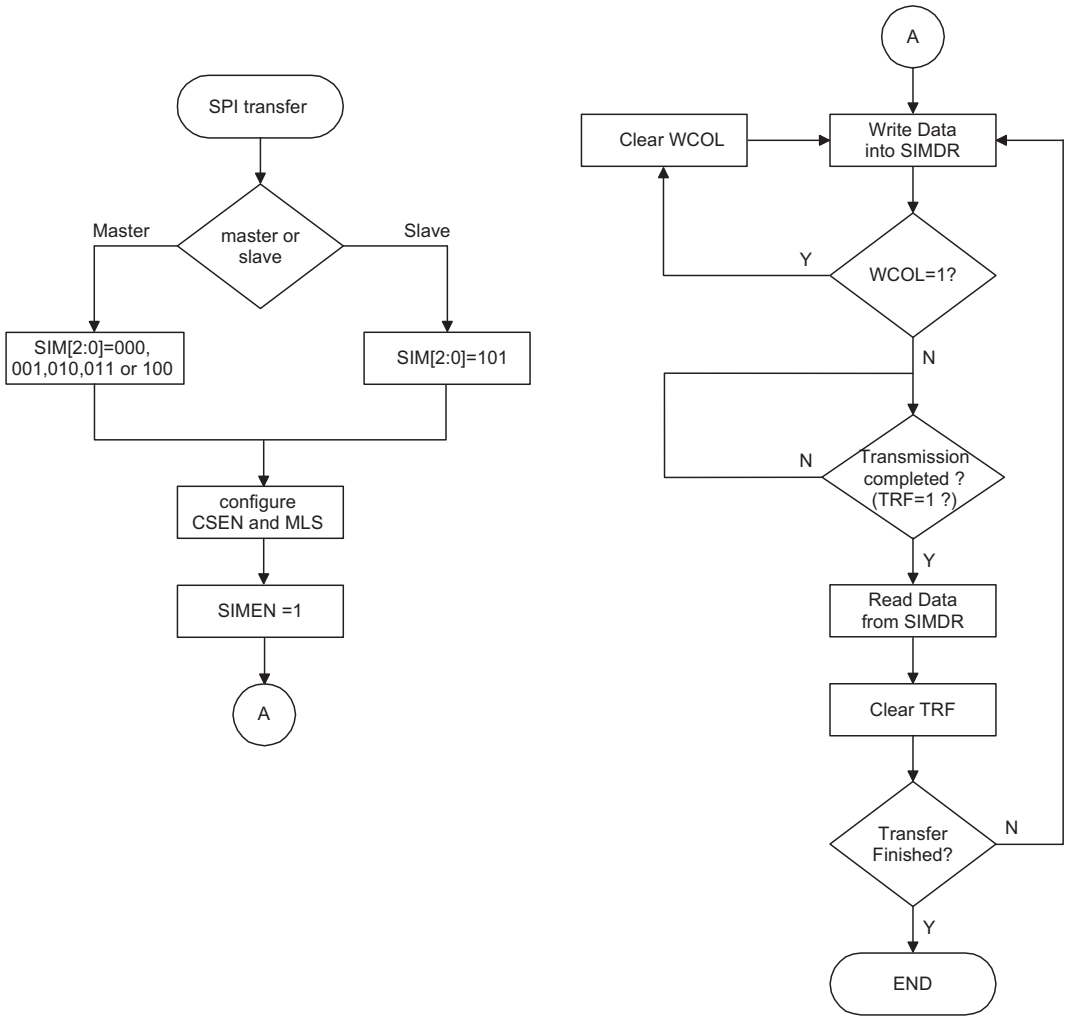


SPI Slave Mode Timing (CKEG=0)

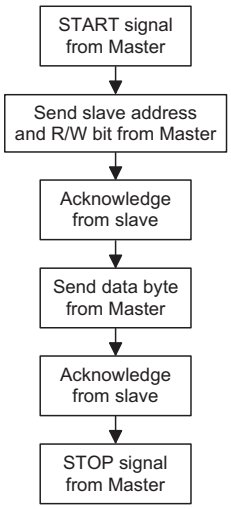


Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignore the $\overline{SCS}$ level.

SPI Slave Mode Timing (CKEG=1)



SPI Transfer Control Flowchart



There are several configuration options associated with the I²C interface. One of these is to enable the function which selects the SIM pins rather than normal I/O pins. Note that if the configuration option does not select the SIM function then the SIMEN bit in the SIMCTL0 register will have no effect. A configuration option exists to allow a clock other than the system clock to drive the I²C interface. Another configuration option determines the debounce time of the I²C interface. This uses the internal clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 1 or 2 system clocks.

SIM	Function
SIM function	SIM interface or SEG pins
I²C clock	I²C runs without internal clock Disable/Enable
I²C debounce	No debounce, 1 system clock; 2 system clocks

I²C Interface Configuration Options

- I²C Registers

  There are three control registers associated with the I²C bus, SIMCTL0, SIMCTL1 and SIMAR and one data register, SIMDR. The SIMDR register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I²C bus. Before the microcontroller writes data to the I²C bus, the actual data to be transmitted must be placed in the SIMDR register. After the data is received from the I²C bus, the microcontroller can read it from the SIMDR register. Any transmission or reception of data from the I²C bus must be made via the SIMDR register.

  Note that the SIMAR register also has the name SIMCTL2 which is used by the SPI function. Bits SIMIDLE , SIMEN and bits SIM0~SIM2 in register SIMCTL0 are used by the I²C interface. The SIMCTL0 register is shown in the above SPI section.

  - SIMIDLE

    The SIMIDLE bit is used to select if the I²C interface continues running when the device is in the IDLE mode. Setting the bit high allows the I²C interface to maintain operation when the device is in the Idle mode. Clearing the bit to zero disables any I²C operations when in the Idle mode.

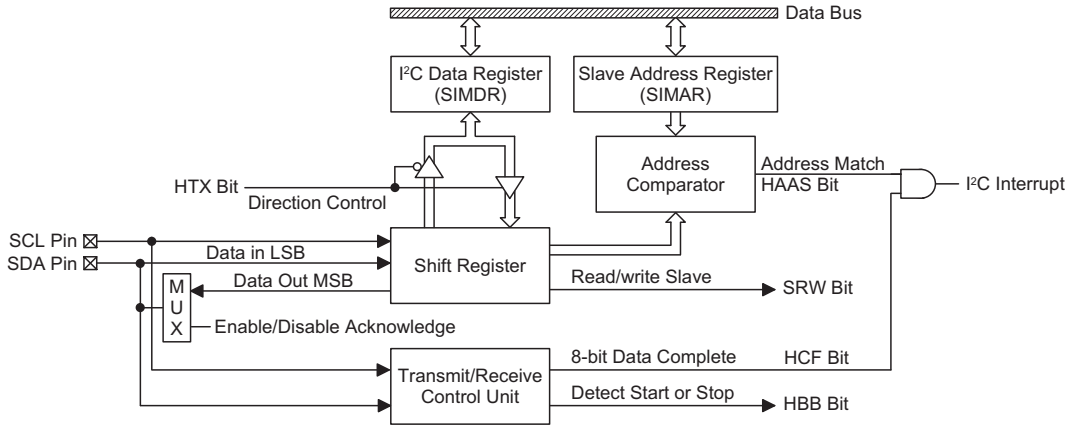
    This SPI/I²C idle mode control bit is located at CLKMOD register bit4.

  - SIMEN

    The SIMEN bit is the overall on/off control for the I²C interface. When the SIMEN bit is cleared to zero to disable the I²C interface, the SDA and SCL lines will be in a floating condition and the I²C operating current will be reduced to a minimum value. In this condition the pins can be used as SEG functions. When the bit is high the I²C interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. Note that when the SIMENbit changes from low to high the contents of the I²C control registers will be in an unknown condition and should therefore be first initialised by the application program.

  - SIM0~SIM2

    These bits setup the overall operating mode of the SIM function. To select the I²C function, bits SIM2~SIM0 should be set to the value 110.



I²C Block Diagram

- RXAK
  The RXAK flag is the receive acknowledge flag. When the RXAK bit has been reset to zero it means that a correct acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When in the transmit mode, the transmitter checks the RXAK bit to determine if the receiver wishes to receive the next byte. The transmitter will therefore continue sending out data until the RXAK bit is set high. When this occurs, the transmitter will release the SDA line to allow the master to send a STOP signal to release the bus.
- SRW
  The SRW bit is the Slave Read/Write bit. This bit determines whether the master device wishes to transmit or receive data from the I^2C bus. When the transmitted address and slave address match, that is when the HAAS bit is set high, the device will check the SRW bit to determine whether it should be in transmit mode or receive mode. If the SRW bit is high, the master is requesting to read data from the bus, so the device should be in transmit mode. When the SRW bit is zero, the master will write data to the bus, therefore the device should be in receive mode to read this data.
- TXAK
  The TXAK flag is the transmit acknowledge flag. After the receipt of 8-bits of data, this bit will be transmitted to the bus on the 9th clock. To continue receiving more data, this bit has to be reset to zero before further data is received.
- HTX
  The HTX flag is the transmit/receive mode bit. This flag should be set high to set the transmit mode and low for the receive mode.
- HBB
  The HBB flag is the I^2C busy flag. This flag will be high when the I^2C bus is busy which will occur when a START signal is detected. The flag will be reset to zero when the bus is free which will occur when a STOP signal is detected.
- HASS
  The HASS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.
- HCF
  The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

I^2C Control Register – SIMAR

The SIMAR register is also used by the SPI interface but has the name SIMCTL2.

The SIMAR register is the location where the 7-bit slave address of the microcontroller is stored. Bits 1~7 of the SIMAR register define the microcontroller slave address. Bit 0 is not defined. When a master device, which is connected to the I^2C bus, sends out an address, which matches the slave address in the SIMAR register, the microcontroller slave device will be selected. Note that the SIMAR register is the same register as SIMCTL2 which is used by the SPI interface.

I^2C Bus Communication

Communication on the I^2C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I^2C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the microcontroller matches that of the transmitted address, the HAAS bit in the SIMCTL1 register will be set and an I^2C interrupt will be generated. After entering the interrupt service routine, the microcontroller slave device must first check the condition of the HAAS bit to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the microcontroller to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I^2C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

Step 1

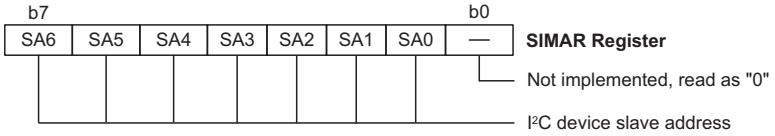
Write the slave address of the microcontroller to the I^2C bus address register SIMAR.

Step 2

Set the SIMEN bit in the SIMCTL0 register to "1" to enable the I^2C bus.

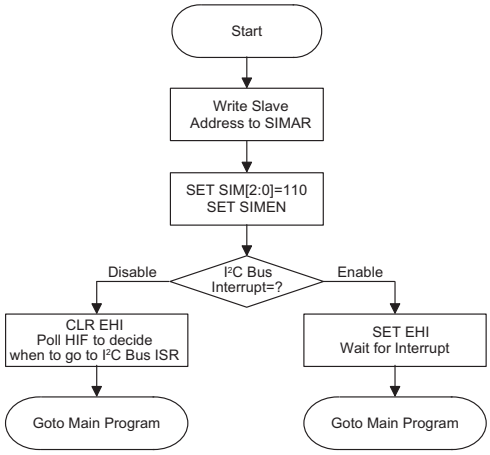
Step 3

Set the ESIM bit of the interrupt control register to enable the I^2C bus interrupt.



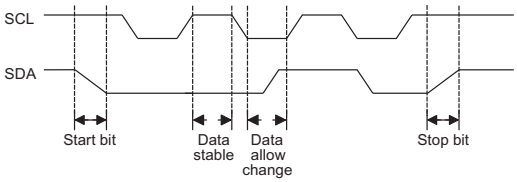
I^2C Slave Address Register – SIMAR

- Start Signal
  The START signal can only be generated by the master device connected to the I²C bus and not by the microcontroller, which is only a slave device. This START signal will be detected by all devices connected to the I²C bus. When detected, this indicates that the I²C bus is busy and therefore the HBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.
- Slave Address
  The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the SIMCTL1 register. The device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The microcontroller slave device will also set the status flag HAAS when the addresses match. As an I²C bus interrupt can come from two sources, when the program enters the interrupt subroutine, the HAAS bit should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMDR register, or in the receive mode where it must implement a dummy read from the SIMDR register to release the SCL line.
- SRW Bit
  The SRW bit in the SIMCTL1 register defines whether the microcontroller slave device wishes to read data from the I²C bus or write data to the I²C bus. The microcontroller should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW bit is set to "1" then this indicates that the master wishes to read data from the I²C bus, therefore the microcontroller slave device must be setup to send data to the I²C bus as a transmitter. If the SRW bit is "0" then this indicates that the master wishes to send data to the I²C bus, therefore the microcontroller slave device must be setup to read data from the I²C bus as a receiver.
- Acknowledge Bit
  After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. This acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS bit is high, the addresses have matched and the microcontroller slave device must check the SRW bit to determine if it is to be a transmitter or a receiver. If the SRW bit is high, the microcontroller slave device should be setup to be a transmitter so the HTX bit in the SIMCTL1 register should be set to "1" if the SRW bit is low then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMCTL1 register should be set to "0".



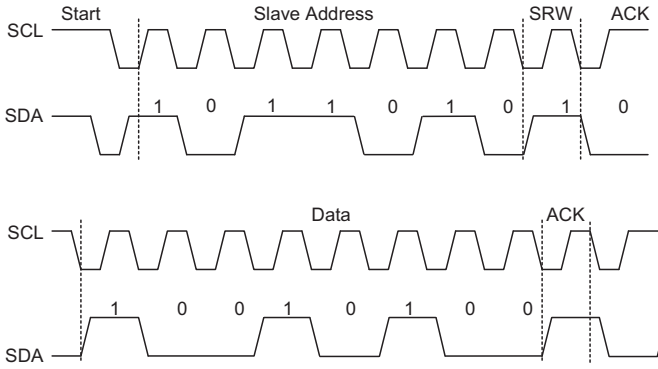
I²C Bus Initialisation Flow Chart

- Data Byte
  The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the transmitter does not receive an acknowledge bit signal from the receiver, then it will release the SDA line and the master will send out a STOP signal to release control of the I²C bus. The corresponding data will be stored in the SIMDR register. If setup as a transmitter, the microcontroller slave device must first write the data to be transmitted into the SIMDR register. If setup as a receiver, the microcontroller slave device must read the transmitted data from the SIMDR register.



Data Timing Diagram

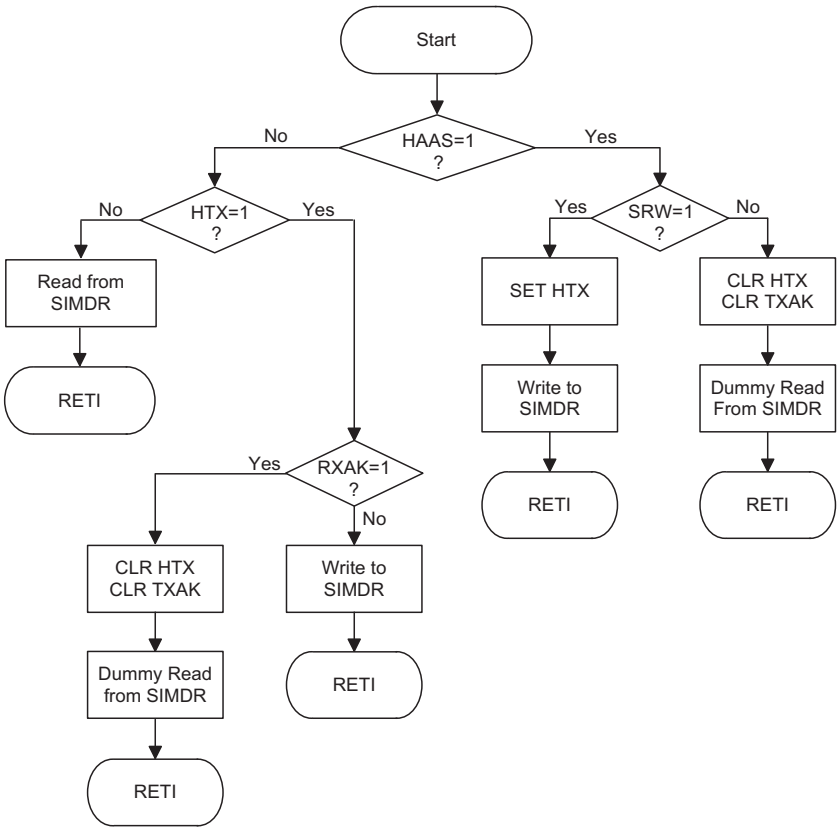
- Receive Acknowledge Bit
  When the receiver wishes to continue to receive the next data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The microcontroller slave device, which is setup as a transmitter will check the RXAK bit in the SIMCTL1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.



S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	--------	S	SA	SR	M	D	A	D	A	--------	P

I²C Communication Timing Diagram



I²C Bus ISR Flow Chart

SPI Interface

The devices contain an independent SPI function. It is important not to confuse this independent SPI function with the additional one contained within the combined SIM function, which is described in another section of this datasheet.

The SPI interface is a full duplex serial data link, originally designed by Motorola, which allows multiple devices connected to the same SPI bus to communicate with each other. The devices communicate using a master/slave technique where only the single master device can initiate a data transfer. A simple four line signal bus is used for all communication.

SPI Interface Communication

Four lines are used for each function. These are, SDI1 Serial Data Input, SDO1 Serial Data Output, SCK1 Serial Clock and SCS1 Slave Select. Note that the condition of the Slave Select line is conditioned by the CSEN1 bit in the SPICTL1 control register. If the CSEN1 bit is high then the SCS1 line is active while if the bit is low then the SCS line will be in a floating condition. The accompanying timing diagram depicts the basic timing protocol of the SPI bus.

SPI Registers

There are three registers for control of the SPI Interface. These are the two control registers SPICTL0 and SPICTL1 and the SBDR data register. The SPICTL0 register is used for the overall SPI enable/disable, master/slave selection and clock selection. The SPICTL1 register is used for SPI setup including, clock polarity, edge selection as well as certain status flags. The SBDR register is used for data storage. After Power on, the contents of the SBDR register will be in an unknown condition. Note that data written to the SBDR register will only be written to the TXRX buffer, whereas data read from the SBDR register will actual be read from the register.

- **SPIDR Register**

Bit	7	6	5	4	3	2	1	0
Name	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0
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 **SPD7~SPD0**: SPI data

- **SPICTL0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SP12	SP11	SP10	—	—	—	SPIEN	—
R/W	R/W	R/W	R/W	—	—	—	—	—
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SPI2~SPI0**: Master/Slave Clock Select
000: SPI master, $f_{SYS}/4$
001: SPI master, $f_{SYS}/16$
010: SPI master, $f_{SYS}/64$
011: SPI master, f_{SUB}
100: SPI master, timer 0 output/2 (PFD0)
101: SPI slave

Bit 4~2 unimplemented, read as ″0″

Bit 1 **SPIEN**: SPI Enable/Disable
0: disable
1: enable

Bit 0 unimplemented, read as ″0″

• **SPICTL1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CKPOL1	CKEG1	MLS1	CSEN1	WCOL1	TRF1
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 unimplemented, read as ″0″

Bit 5 **CKPOL1**: Determines the base condition of the clock line
0: SCK1 line high when the clock is inactive
1: SCK1 line low when the clock is inactive
The CKPOL1 bit determines the base condition of the clock line, if the bit is high, then the SCK1 line will be low when the clock is inactive. When the CKPOL1 bit is low, then the SCK1 line will be high when the clock is inactive.

Bit 4 **CKEG1**: Determines the SPI1 SCK1 active clock edge type
CKPOL1=0:
0: SCK1 has high base level with data capture on SCK1 rising edge
1: SCK1 has high base level with data capture on SCK1 falling edge
CKPOL1=1:
0: SCK1 has low base level with data capture on SCK1 falling edge
1: SCK1 has low base level with data capture on SCK1 rising edge
The CKEG1 and CKPOL1 bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before a data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOL1 bit determines the base condition of the clock line, if the bit is high, then the SCK1 line will be low when the clock is inactive. When the CKPOL1 bit is low, then the SCK1 line will be high when the clock is inactive. The CKEG1 bit determines active clock edge type which depends upon the condition of CKPOL1 bit.

Bit 3 **MLS1**: Determines the data shift order - MSB or LSB
0: LSB transmitted first
1: MSB transmitted first

Bit 2 **CSEN1**: SPI1 bus select
0: Disable - SPI1 bus is floating
1: Enable

Bit 1 **WCOL1**: Write collision flag
0: Collision free
1: Collision detected
This flag is set by the by the SPI1 bus and cleared by the application program. The flag will be set to 1 if data is written to the SPIDR register (TXRX buffer) when a data is still being transferred. Any such data write actions will be ignored in such cases.

Bit 0 **TRF1**: Transmit/Receive completion flag
0: Not complete
1: Data Transmission/Reception Complete
This flag will be set high when a data reception or transmission has completed. It must be cleared using the application program and can be used to generate an interrupt.

SPI Bus Enable/Disable

To enable the SPI bus, the SBEN bit should be set high, then wait for data to be written to the SBDR (TXRX buffer) register. For the Master Mode, after data has been written to the SBDR (TXRX buffer) register then transmission or reception will start automatically. When all the data has been transferred, the TRF1 bit should be set. For the Slave Mode, when clock pulses are received on SCK1, data in the TXRX buffer will be shifted out or data on SDI1 will be shifted in. When the SPI bus is disabled, SCK1, SDI1, SDO1 and SCS1 will be setup as I/O pins.

SPI Operation

The SPI is selected using the application program. All communication is carried out using the 4-line interface for both Master or Slave Mode. The CSEN1 bit in the SPICTL1 register controls the SCSB line of the SPI interface. Setting this bit high, will enable the SPI interface by allowing the SCS1 line to be active, which can then be used to control the SPI interface. If the CSEN1 bit is low, the SCS1 line will be in a floating condition and can therefore not be used for control of the SPI interface. When the CSEN1 bit is set high then SDI1 line will be placed in a floating condition and the SDO1 line will be high. If in the Master Mode, the SCK1 line will be either high or low depending upon the clock polarity configuration option. If in the Slave Mode the SCK1 line will be in a floating condition. If CSEN1 is low then the bus will be disabled and SCS1, SDI1, SDO1 and SCK1 will all be in a floating condition. The SPI function keeps running in the IDLE mode - the SPI module can still operate after a HALT instruction is executed. The CKEG1 and CKPOL1 bits must be setup before the SPI is enabled; otherwise undesired clock edge may be generated.

In the Master Mode, the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written to the SBDR register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission or reception. The following sequences show the order to be followed for data transfer in both Master and Slave Modes:

- Master Mode
  - Step 1
    Setup the SPI2~SPI0 bits in the **SPICTL0** control register to select the Master Mode and the required clock speed. Values of 000~101 can be selected.
  - Step 2
    Setup the SPIEN bit and setup the MLS1 bit to choose if the data is MSB or LSB first, this must be same as the Slave device.
  - Step 3
    Setup the CSEN1 bit in the **SPICTL1** control register to enable the SPI interface.
  - Step 4
    For write operations: write the data to the SBDR register, which will actually place the data into the TXRX buffer. Then use the SCK1 and SCS1 lines to output the data. Then goto to step 5. For read operations: the data transferred in on the SDI1 line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SBDR register.
  - Step 5
    Check the WCOL1 bit, if set high then a collision error has occurred so return to Step 4. If zero then go to the following step.
  - Step 6
    Check the TRF1 bit or wait for an SPI serial bus interrupt.
  - Step 7
    Read data from the SBDR register.
  - Step 8
    Clear flag TRF1.
  - Step 9
    Goto step 4.
- Slave Mode
  - Step 1
    Setup the **SPI2~SPI0** bits to 101 to select the Slave Mode.
  - Step 2
    Setup the SPIEN bit and setup the MLS1 bit to choose if the data is MSB or LSB first, this must be same as the Master device.
  - Step 3
    Setup the CSEN1 bit in the **SPICTL1** control register to enable the SPI interface.
  - Step 4
    For write operations: write data to the SBDR register, which will actually place the data into the TXRX register, then wait for the master clock and SCS1 signal. After this goto Step 5. For read operations: the data transferred in on the SDI1 line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SBDR register.
  - Step 5
    Check the WCOL1 bit, if set high then a collision error has occurred so return to step 4. If equal to zero then goto the following step.
  - Step 6
    Check the TRF1 bit or wait for an SPI interrupt.
  - Step 7
    Read data from the SBDR register.
  - Step 8
    Clear TRF1
  - Step 9
    Goto step 4

SPI Configuration Options

A configuration option is provided for an overall on/off control for the SPI bus. Additional configuration options are provided to enable operation of the WCOL1 bit which is the write collision bit and the CSEN1 bus select bit.

Peripheral Clock Output

The Peripheral Clock Output allows the device to supply external hardware with a clock signal synchronised to the microcontroller clock.

Peripheral Clock Operation

As the peripheral clock output pin, PCK, is shared with an I/O line, the required pin function is chosen via PCKEN in the SIMC0 register. The Peripheral Clock function is controlled using the SIMC0 register. The clock source for the Peripheral Clock Output can originate from either the Timer/Event Counter 0 or a divided ratio of the internal f_{SYS} clock. The PCKEN bit in the SIMC0 register is the overall on/off control, setting PCKEN bit to 1 enables the Peripheral Clock, setting PCKEN bit to 0 disables it. The required division ratio of the system clock is selected using the PCKP1 and PCKP0 bits in the same register. If the device enters the SLEEP Mode this will disable the Peripheral Clock output.

- **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	1	1	1	0	0	0	0	—

Bit 7~5 **SIM2, SIM1, SIM0**: SIM operating mode control described in SIM section

Bit 4 **PCKEN**: PCK output pin control
0: Disable
1: Enable

Bit 3~2 **PCKP1, PCKP0**: select PCK output pin frequency
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/8$
11: TM0 CCRP match frequency/2

Bit 1 **SIMEN**: SIM control described in SIM section

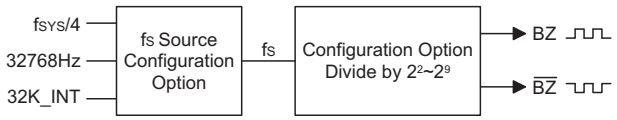
Bit 0 unimplemented, read as ″0″

Buzzer

Operating in a similar way to the Programmable Frequency Divider, the Buzzer function provides a means of producing a variable frequency output, suitable for applications such as Piezo-buzzer driving or other external circuits that require a precise frequency generator. The BZ and $\overline{BZ}$ pins form a complementary pair, and are pin-shared with I/O pins, PB4 and PB5. A configuration option is used to select from one of three buzzer options. The first option is for both pins PB4 and PB5 to be used as normal I/Os, the second option is for both pins to be configured as BZ and $\overline{BZ}$ buzzer pins, the third option selects only the PB4 pin to be used as a BZ buzzer pin with the PB5 pin retaining its normal I/O pin function. Note that the $\overline{BZ}$ pin is the inverse of the BZ pin which together generate a differential output which can supply more power to connected interfaces such as buzzers.

The buzzer is driven by the internal clock source, , which then passes through a divider, the division ratio of which is selected by configuration options to provide a range of buzzer frequencies from $f_S/2^2$ to $f_S/2^9$. The clock source that generates f_S, which in turn controls the buzzer frequency, can originate from three different sources, the 32768Hz oscillator, the 32K_INT oscillator or the System oscillator/4, the choice of which is determined by the f_S clock source configuration option. Note that the buzzer frequency is controlled by configuration options, which select both the source clock for the internal clock f_S and the internal division ratio. There are no internal registers associated with the buzzer frequency.

If the configuration options have selected both pins PB4 and PB5 to function as a BZ and $\overline{BZ}$ complementary pair of buzzer outputs, then for correct buzzer operation it is essential that both pins must be setup as outputs by setting bits PBC4 and PBC5 of the PBC port control register to zero. The PB4 data bit in the PB data register must also be set high to enable the buzzer outputs, if set low, both pins PB4 and PB5 will remain low. In this way the single bit PB4 of the PB register can be used as an on/off control for both the BZ and $\overline{BZ}$ buzzer pin outputs. Note that the PB5 data bit in the PB register has no control over the $\overline{BZ}$ buzzer pin PB5.



Buzzer Function

PB4/PB5 Pin Function Control

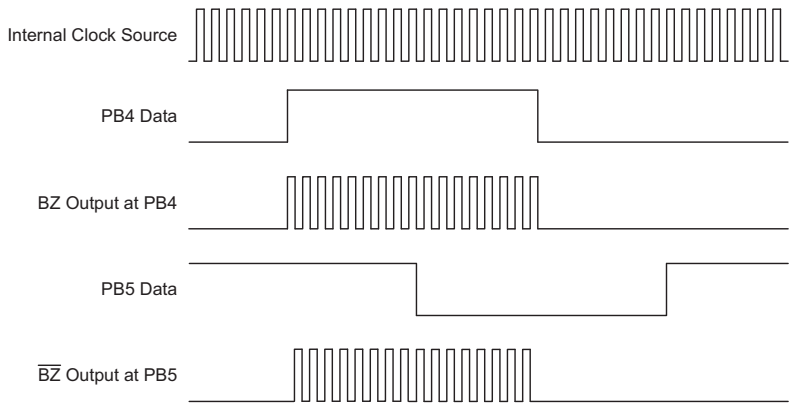
PBC Register PBC4	PBC Register PBC5	PB Data Register PB4	PB Data Register PB5	Output Function
0	0	1	x	PB4=BZ PB5=$\overline{BZ}$
0	0	0	x	PB4="0" PB5="0"
0	1	1	x	PB4=BZ PB5=input line
0	1	0	x	PB4="0" PB5=input line
1	0	x	D	PB4=input line PB5=D
1	1	x	x	PB4=input line PB4=input line

"x" stands for don't care

"D" stands for Data "0" or "1"

If configuration options have selected that only the PB4 pin is to function as a BZ buzzer pin, then the PB5 pin can be used as a normal I/O pin. For the PB4 pin to function as a BZ buzzer pin, PB4 must be setup as an output by setting bit PBC4 of the PBC port control register to zero. The PB4 data bit in the PB data register must also be set high to enable the buzzer output, if set low pin PB4 will remain low. In this way the PB4 bit can be used as an on/off control for the BZ buzzer pin PB4. If the PBC4 bit of the PBC port control register is set high, then pin PB4 can still be used as an input even though the configuration option has configured it as a BZ buzzer output.

Note that no matter what configuration option is chosen for the buzzer, if the port control register has setup the pin to function as an input, then this will override the configuration option selection and force the pin to always behave as an input pin. This arrangement enables the pin to be used as both a buzzer pin and as an input pin, so regardless of the configuration option chosen; the actual function of the pin can be changed dynamically by the application program by programming the appropriate port control register bit.



Buzzer Output Pin Control

Note: The above drawing shows the situation where both pins PB4 and PB5 are selected by configuration option to be BZ and $\overline{BZ}$ buzzer pin outputs. The Port Control Register of both pins must have already been setup as output. The data setup on pin PB5 has no effect on the buzzer outputs.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter or Time Base requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs.

The devices contain a single external interrupt and multiple internal interrupts.

Interrupt Register

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by using several registers, INTC0, INTC1, MFIC0 and MFIC1. By controlling the appropriate enable bits in this registers each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request flag will be set by the microcontroller. The global enable flag if cleared to zero will disable all interrupts.

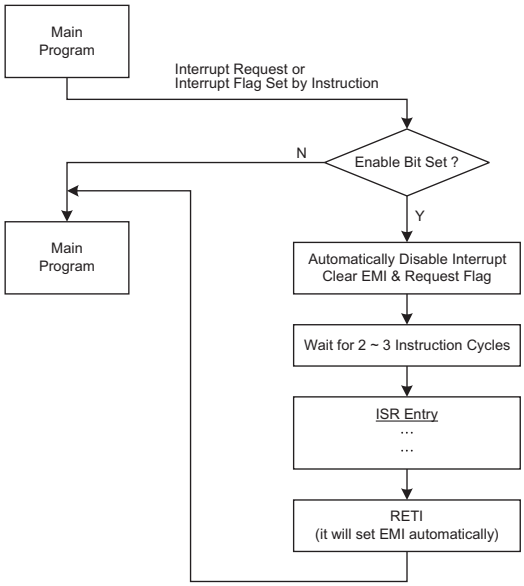
Interrupt Operation

A range of internal and external events can all generate an interrupt, by setting their corresppponding 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.

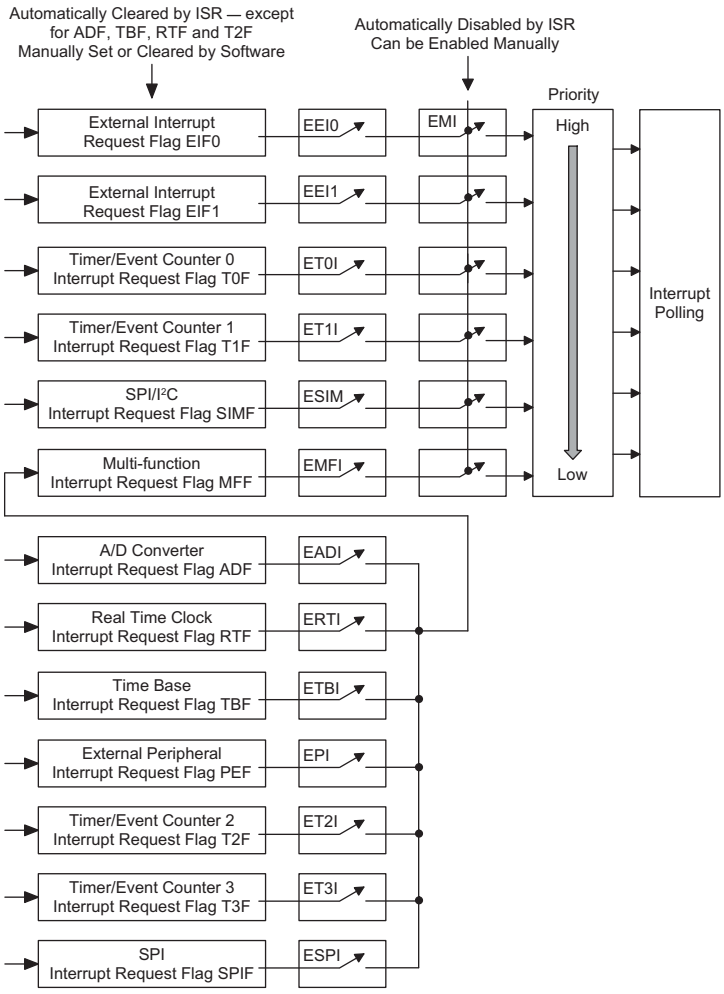
The various interrupt enable bits, together with their associated request flags, are shown in the following diagram with their order of priority.

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

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



Interrupt Flow



Interrupt Structure

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.

HT56R22		
Interrupt Source	**Priority**	**Vector**
External Interrupt 0	1	04H
External Interrupt 1	2	08H
Timer/Event Counter 0 Overflow	3	0CH
Timer/Event Counter 1 Overflow	4	10H
SPI/I^2C Interrupt	5	14H
Time Base Interrupt	6	18H
RTC Interrupt	6	18H
A/D Converter Interrupt	6	18H
External Peripheral Interrupt	6	18H
SPI Interrupt	6	18H

HT56R23/HT56R24		
Interrupt Source	**Priority**	**Vector**
External Interrupt 0	1	04H
External Interrupt 1	2	08H
Timer/Event Counter 0 Overflow	3	0CH
Timer/Event Counter 1 Overflow	4	10H
SPI/I^2C Interrupt	5	14H
Time Base Interrupt	6	18H
RTC Interrupt	6	18H
A/D Converter Interrupt	6	18H
External Peripheral Interrupt	6	18H
SPI1	6	18H
Timer/Event Counter 2 Overflow	6	18H

HT56R25/HT56R26		
Interrupt Source	**Priority**	**Vector**
External Interrupt 0	1	04H
External Interrupt 1	2	08H
Timer/Event Counter 0 Overflow	3	0CH
Timer/Event Counter 1 Overflow	4	10H
SPI/I^2C Interrupt	5	14H
Time Base Interrupt	6	18H
RTC Interrupt	6	18H
A/D Converter Interrupt	6	18H
External Peripheral Interrupt	6	18H
SPI1	6	18H
Timer/Event Counter 2 Overflow	6	18H
Timer/Event Counter 3 Overflow	6	18H

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

Interrupt Operation

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

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred. The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

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

External Interrupt

For an external interrupt to occur, the global interrupt enable bit, EMI, and external interrupt enable bits, EEI0 and EEI1, must first be set. Additionally the correct interrupt edge type must be selected using the INTEDGE register to enable the external interrupt function and to choose the trigger edge type. An actual external interrupt will take place when the external interrupt request flag, EIF0 or EIF1, is set, a situation that will occur when a transition, whose type is chosen by the edge select bit, appears on the INT0 or INT1 pin. The external interrupt pins are pin-shared with the I/O pins PA4 and PA6 and can only be configured as external interrupt pins if their corresponding external interrupt enable bit in the INTC0 register has been set. The pin must also be setup as an input by setting the corresponding PAC.4 and PAC.6 bits in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector at location 04H or 08H, will take place. When the interrupt is serviced, the external interrupt request flags, EIF0 or EIF1, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on this pin will remain valid even if the pin is used as an external interrupt input.

The INTEDGE register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising and falling edge types can be chosen along with an option to allow both edge types to trigger an external interrupt. Note that the INTEDGE register can also be used to disable the external interrupt function.

- **INTEDGE Register - All Devices**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 unimplemented, read as ″0″

Bit 3~2 **INT1S1, INT1S0**: interrupt edge control for INT1 pin
00: disable
01: rising edge
10: falling edge
11: rising and falling edges

Bit 1~0 **INT0S1, INT0S0**: interrupt edge control for INT0 pin
00: disable
01: rising edge
10: falling edge
11: rising and falling edges

• **INTC0 Register - All Devices**

Bit	7	6	5	4	3	2	1	0
Name	—	T0F	INT1F	INT0F	T0E	INT1E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 unimplemented, read as ″0″

Bit 6 **T0F**: Timer/Event Counter 0 interrupt request flag
0: inactive
1: active

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

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

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

Bit 2 **INT1E**: external interrupt 1 enable
0: disable
1: enable

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

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

• **INTC1 Register - All Devices**

Bit	7	6	5	4	3	2	1	0
Name	—	MFF	SIMF	T1F	—	MFE	SIME	T1E
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 unimplemented, read as ″0″

Bit 6 **MFF**: Multi-function interrupt request flag
1: active
0: inactive

Bit 5 **SIMF**: SPI/I^2C interrupt request flag
1: active
0: inactive

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

Bit 3 unimplemented, read as ″0″

Bit 2 **MFE**: Multi-function interrupt enable
0: disable
1: enable

Bit 1 **SIME**: Serial Interface Module interrupt enable
0: disable
1: enable

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

• **MFIC0 Register - All devices**

Bit	7	6	5	4	3	2	1	0
Name	XPF	TBF	RTF	ADF	XPE	TBE	RTE	ADE
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 **XPF**: External Peripheral Interrupt Request Flag
0: inactive
1: active

Bit 6 **TBF**: Time Base Interrupt Request Flag
0: inactive
1: active

Bit 5 **RTF**: Real Time Clock Interrupt Request Flag
0: inactive
1: active

Bit 4 **ADF**: A/D Converter Interrupt Request Flag
0: inactive
1: active

Bit 3 **XPE**: External Peripheral Interrupt Enable
0: enable
1: disable

Bit 2 **TBE**: Time Base Interrupt Enable
0: enable
1: disable

Bit 1 **RTE**: Real Time Clock Interrupt Control
0: enable
1: disable

Bit 0 **ADE**: A/D Converter Interrupt Control
0: enable
1: disable

• **MFIC1 Register - HT56R22**

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

Bit 7~5 unimplemented, read as ″0″

Bit 4 **SPIF**: SPI Interface Interrupt Request Flag
0: inactive
1: active

Bit 3~1 unimplemented, read as ″0″

Bit 0 **SPIE**: SPI Interface Interrupt Control
0: enable
1: disable

• **MFIC1 Register - HT56R23/HT56R24**

Bit	7	6	5	4	3	2	1	0
Name	—	—	T2F	SPIF	—	—		SPIE
R/W	—	—	R/W	R/W	—	—		R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 unimplemented, read as ″0″

Bit 5 **T2F**: Timer/Event Counter 2 Interrupt Request Flag
0: inactive
1: active

Bit 4 **SPIF**: SPI Interface Interrupt Request Flag
0: inactive
1: active

Bit 3~2 unimplemented, read as ″0″

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

Bit 0 **SPIE**: SPI Interface Interrupt Control
0: enable
1: disable

• **MFIC1 Register - HT56R25/HT56R26**

Bit	7	6	5	4	3	2	1	0
Name	—	T3F	T2F	SSPIFPIF	—	T3E	T2E	SPIE
R/W	—		R/W	R/W	—			R/W
POR	—		0	0	—			0

Bit 7 unimplemented, read as ″0″

Bit 6 **T3F**: Timer/Event Counter 2 Interrupt Request Flag
0: inactive
1: active

Bit 5 **T2F**: Timer/Event Counter 2 Interrupt Request Flag
0: inactive
1: active

Bit 4 **SPIF**: SPI Interface Interrupt Request Flag
0: inactive
1: active

Bit 3 unimplemented, read as ″0″

Bit 2 **T3E**: Timer/Event Counter 2 Interrupt Control
0: enable
1: disable

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

Bit 0 **SPIE**: SPI Interface Interrupt Control
0: enable
1: disable

Timer/Event Counter Interrupt

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

Time Base Interrupt

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 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.

A/D Converter Interrupt

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

Serial Interface Module Interrupt

The Serial Interface Module Interrupt, also known as the SIM interrupt, is contained within the Multi-function Interrupt. A SIM Interrupt request will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, SIME, and Multi-function interrupt enable bits, must first be set. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SIM interface, a subroutine call to the Multi-function Interrupt vector, will take place. When the Serial Interface Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the SIMF flag will not be automatically cleared, it has to be cleared by the application program.

External Peripheral Interrupt

The External Peripheral Interrupt operates in a similar way to the external interrupt and is contained within the Multi-function Interrupt. A Peripheral Interrupt request will take place when the External Peripheral Interrupt re quest flag, XPF, is set, which occurs when a negative edge transition appears on the PINT pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, external peripheral interrupt enable bit, XPE, and Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a negative transition appears on the External Peripheral Interrupt pin, a subroutine call to the Multi-function Interrupt, will take place. When the External Peripheral Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the XPF flag will not be automatically cleared, it has to be cleared by the application program. The external peripheral interrupt pin is pin-shared with several other pins with different functions. It must therefore be properly configured to enable it to operate as an External Peripheral Interrupt pin.

Multi-function Interrupt

Within these devices there is a Multi-function interrupt. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources.

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

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

Programming Considerations

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

It is recommended that programs do not use the "CALL subroutine" instruction within the interrupt subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications. 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.

All of these interrupts have the capability of waking up the processor when in the Idle/Sleep Mode.

Only the Program Counter is pushed onto the stack. If the contents of the register or status register are altered by the interrupt service program, which may corrupt the desired control sequence, then the contents should be saved in advance.

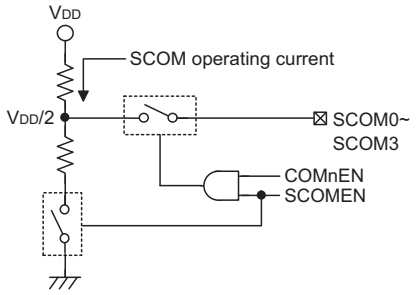
LCD SCOM Function

The devices have the capability of driving external LCD panels. The common pins for LCD driving, SCOM0~SCOM3, are pin shared with certain pin on the PB0~PB3 port. The LCD signals (COM and SEG) are generated using the application program.

LCD Operation

An external LCD panel can be driven using this device by configuring the PB0~PB3 pins as common pins and using other output ports lines as segment pins. The LCD driver function is controlled using the SCOMC register which in addition to controlling the overall on/off function also controls the bias voltage setup function. This enables the LCD COM driver to generate the necessary $V_{DD}/2$ voltage levels for LCD 1/2 bias operation.

The SCOMEN bit in the SCOMC register is the overall master control for the LCD Driver, however this bit is used in conjunction with the COMnEN bits to select which Port B pins are used for LCD driving. Note that the Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation.



LCD COM Bias

SCOMEN	COMnEN	Pin Function	O/P Level
0	X	I/O	0 or 1
1	0	I/O	0 or 1
1	1	SCOMN	$V_{DD}/2$

Output Control

LCD Bias Control

The LCD COM driver enables a range of selections to be provided to suit the requirement of the LCD panel which is being used. The bias resistor choice is implemented using the ISEL1 and ISEL0 bits in the SCOMC register.

- **SCOMC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	ISEL	SCOMEN	COM3EN	COM2EN	COM1EN	COM0EN
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7, 6 unimplemented, read as "0"

Bit 5 **ISEL**: SCOM operating current selection (V_{DD}=5V)
0: 25μA
1: 50μA

Bit 4 **SCOMEN**: SCOM module on/off control
0: disable
1: enable
SCOMn can be enable by COMnEN if SCOMEN=1

Bit 3 **COM3EN**: PB3 or SCOM3 selection
0: GPIO
1: SCOM3

Bit 2 **COM2EN**: PB2 or SCOM2 selection
0: GPIO
1: SCOM2

Bit 1 **COM1EN**: PB1 or SCOM1 selection
0: GPIO
1: SCOM1

Bit 0 **COM0EN**: PB0 or SCOM0 selection
0: GPIO
1: SCOM0

Digital to Analog Converter – DAC

All devices include a 12-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 two registers DAL and DAH. The DAH register stores the highest 8-bits, DA4~DA11, while DAL stores the lowest 4-bits, DA0~DA3. An additional control register, DACTRL, provides overall DAC on/off control in addition to a 3-bit 8-level volume control. The DAC output is channeled to pin AUD which is pin-shared with I/O pin PB5. When the DAC is enabled by setting the DACEN pin high, then the original I/O function will be disabled, along with any pull-high resistor options.

- **DACH Register**

Bit	7	6	5	4	3	2	1	0
Name	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4
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

Bits 7~0 DA11~DA0: Audio Output DAC high byte bits

- **DACL Register**

Bit	7	6	5	4	3	2	1	0
Name	DA3	DA2	DA1	DA0	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	0	0	0	0

Bits 7~4 DA3~DA0: Audio Output DAC low bits

Bits 3~0 unimplemented, read as "0"

- **DACTL Register**

Bit	7	6	5	4	3	2	1	0
Name	VOL2	VOL1	VOL0	—	—	—	—	DACEN
R/W	R/W	R/W	R/W	—	—	—	—	R/W
POR	0	0	0	0	0	0	0	0

Bits 7~5 **VOL2~VOL0**: Audio Volume Control

Bits 4~1 unimplemented, read as "0"

Bit 0 **DACEN**: DAC On/Off Control
0: Off
1: On

The DAC output is channeled to pin AUD which is pin-shared with I/O pin PB5. When the DAC is enabled by setting the DACEN pin 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 VDD.

Configuration Options

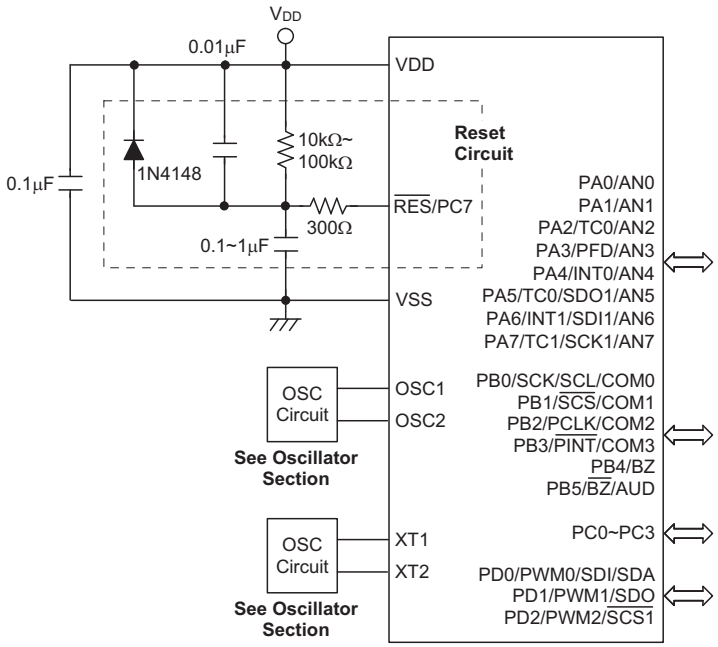
Configuration options refer to certain options within the MCU that are programmed into the OTP Program Memory device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device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

No.	Options
Oscillator Options	
1	High Oscillator type selection - f_M 1. External Crystal Oscillator 2. External RC Oscillator 3. Externally supplied clock - internal filter on 4. Externally supplied clock - internal filter off
2	f_{SUB} clock selection: 1. 32768Hz External Oscillator 2. 32K_INT Internal Oscillator
3	f_S clock selection: f_{SUB} or $f_{SYS}/4$
4	XTAL mode selection: 455KHz or 1M~12MHz
5	32768Hz Crystal: enable or disable
PFD Options	
6	PA3: normal I/O or PFD output
7	PFD clock selection: Timer/Event Counter 0 or Timer/Event Counter 1
Buzzer Options	
8	PA0/PA1: normal I/O or BZ/$\overline{BZ}$ or PA0=BZ and PA1 as normal I/O
9	Buzzer frequency: $f_S/2^2$, $f_S/2^3$, $f_S/2^4$, $f_S/2^5$, $f_S/2^6$, $f_S/2^7$, $f_S/2^8$, $f_S/2^9$
Time Base Option	
10	Time base time-out period: $2^{12}/f_S$, $2^{13}/f_S$, $2^{14}/f_S$, $2^{15}/f_S$,
LCD Option	
11	LCD type: R or C - HT56R66 only
Watchdog Options	
12	Watchdog Timer function: enable or disable
13	CLRWDT instructions: 1 or 2 instructions
14	WDT time-out period: $2^{12}/f_S$~$2^{13}/f_S$, $2^{13}/f_S$~$2^{14}/f_S$, $2^{14}/f_S$~$2^{15}/f_S$, $2^{15}/f_S$~$2^{16}/f_S$
LVD/LVR Options	
15	LVD function: enable or disable
16	LVR function: enable or disable
17	LVR/LVD voltage: 2.1V/2.2V or 3.15V/3.3V or 4.2V/4.4V
SPI Options	
18	SIM pin enable/disable
19	SPI_WCOL: enable/disable
20	SPI_CSEN: enable/disable, used to enable/disable (1/0) software CSEN function
I^2C Option	
21	I^2C debounce Time: no debounce, 1 system clock debounce, 2 system clock debounce

No.	Options
PINTB Option	
22	External peripheral interrupt or Segment function
Timer/Event Counter and External Interrupt Pins Filter Option	
23	Interrupt and Timer/Event Counter input pins internal filter On/Off control – applies to all pins
Lock Options	
24	Lock All
25	Partial Lock

Application Circuits

HT56R22



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z

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

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

Instruction Definition

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

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

Operation: ACC ← ACC + [m] + C

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

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

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

Operation: [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation: ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation: ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation: [m] ← ACC + [m]

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

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

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

Operation: ACC ← ACC ″AND″ [m]

Affected flag(s): Z

AND A,x Logical AND immediate data to ACC

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

Operation: ACC ← ACC ″AND″ x

Affected flag(s): Z

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

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

Operation: [m] ← ACC ″AND″ [m]

Affected flag(s): Z

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None
CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	[m] ← $\overline{[m]}$
Affected flag(s)	Z
CPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC ← $\overline{[m]}$
Affected flag(s)	Z
DAA [m]	Decimal-Adjust ACC for addition with result in Data Memory
Description	Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.
Operation	[m] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← ACC + 66H
Affected flag(s)	C
DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [m] – 1
Affected flag(s)	Z
DECA [m]	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] – 1
Affected flag(s)	Z
HALT	Enter power down mode
Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.
Operation	TO ← 0 PDF ← 1
Affected flag(s)	TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z
INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z
JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None
MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None
MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None
NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ x
Affected flag(s)	Z
ORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″OR″ [m]
Affected flag(s)	Z
RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None
RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None
RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None
RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i = 0~6) [m].0 ← [m].7
Affected flag(s)	None
RLA [m]	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i = 0~6) ACC.0 ← [m].7
Affected flag(s)	None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

Operation $[m] \leftarrow [m] - 1$
Skip if $[m] = 0$

Affected flag(s) None

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

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

Operation $ACC \leftarrow [m] - 1$
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 $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation $[m].i \leftarrow 1$

Affected flag(s) None

SIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] + 1 Skip if [m] = 0
Affected flag(s)	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] + 1 Skip if ACC = 0
Affected flag(s)	None
SNZ [m].i	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None
SUB A,[m]	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC – [m]
Affected flag(s)	OV, Z, AC, C
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC – [m]
Affected flag(s)	OV, Z, AC, C
SUB A,x	Subtract immediate data from ACC
Description	The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC – x
Affected flag(s)	OV, Z, AC, C

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m] = 0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

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

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

Operation Skip if [m].i = 0

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Package Information

16-pin DIP (300mil) Outline Dimensions

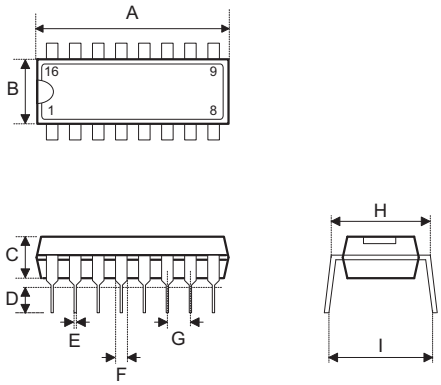


Fig1. Full Lead Packages

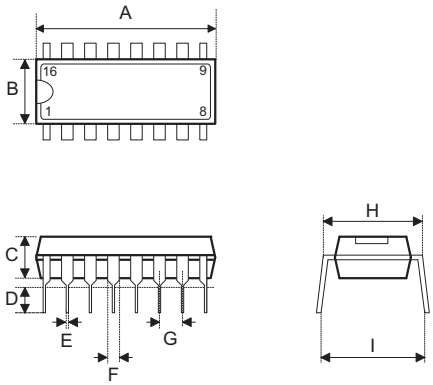


Fig2. 1/2 Lead Packages

• MS-001d (see fig1)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	780	—	880
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

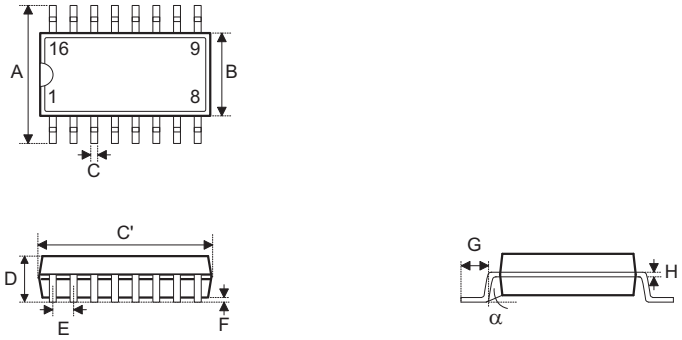
• MS-001d (see fig2)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	735	—	775
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

• MO-095a (see fig2)

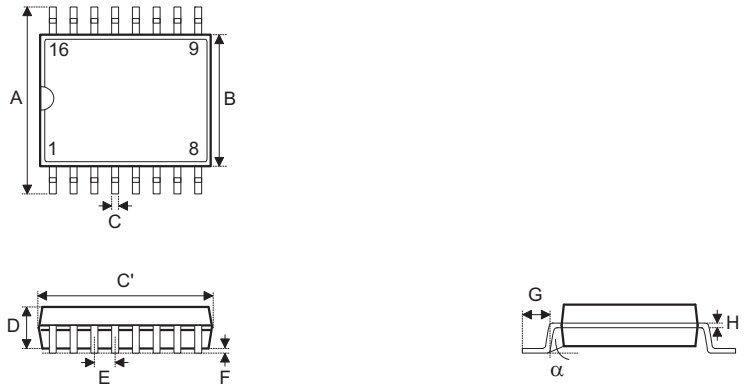
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	745	—	785
B	275	—	295
C	120	—	150
D	110	—	150
E	14	—	22
F	45	—	60
G	—	100	—
H	300	—	325
I	—	—	430

16-pin NSOP (150mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	228	—	244
B	149	—	157
C	14	—	20
C′	386	—	394
D	53	—	69
E	—	50	—
F	4	—	10
G	22	—	28
H	4	—	12
α	0°	—	10°

16-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.157
C	0.008	—	0.012
C′	0.189	—	0.197
D	0.054	—	0.060
E	—	0.025	—
F	0.004	—	0.010
G	0.022	—	0.028
H	0.007	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	3.99
C	0.20	—	0.30
C′	4.80	—	5.00
D	1.37	—	1.52
E	—	0.64	—
F	0.10	—	0.25
G	0.56	—	0.71
H	0.18	—	0.25
α	0°	—	8°

20-pin DIP (300mil) Outline Dimensions

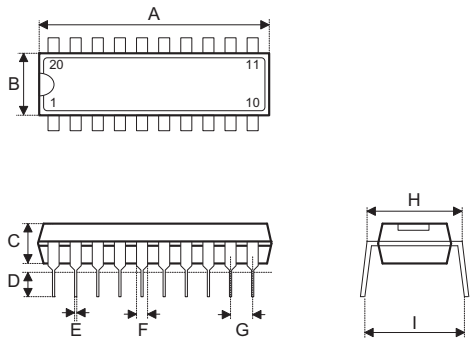


Fig1. Full Lead Packages

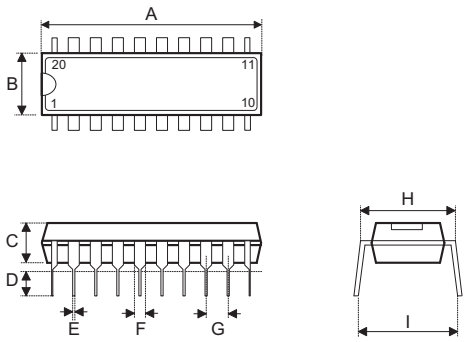


Fig2. 1/2 Lead Packages

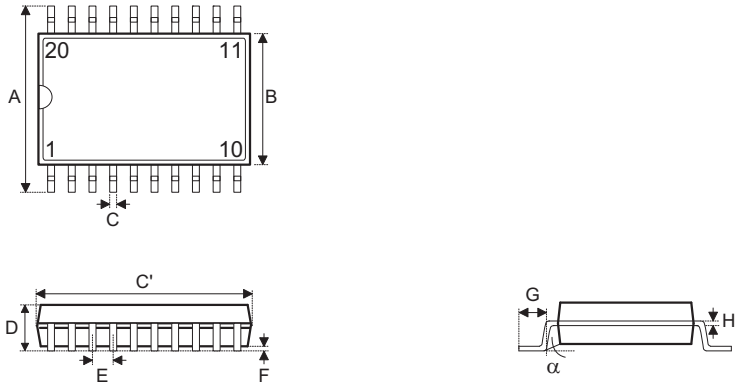
- MS-001d (see fig1)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	980	—	1060
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

- MO-095a (see fig2)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	945	—	985
B	275	—	295
C	120	—	150
D	110	—	150
E	14	—	22
F	45	—	60
G	—	100	—
H	300	—	325
I	—	—	430

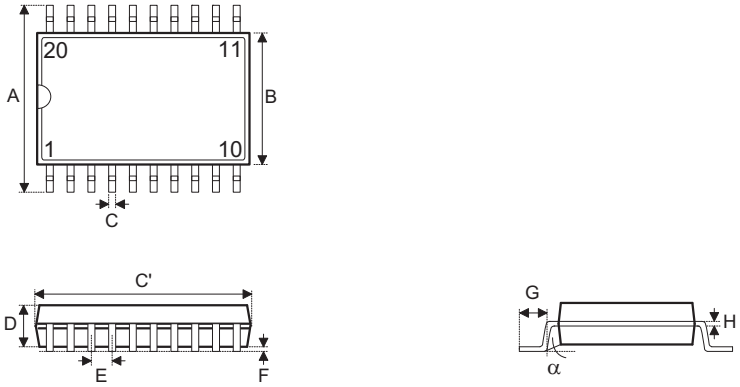
20-pin SOP (300mil) Outline Dimensions



• MS-013

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	393	—	419
B	256	—	300
C	12	—	20
C′	496	—	512
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

20-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.158
C	0.008	—	0.012
C′	0.335	—	0.347
D	0.049	—	0.065
E	—	0.025	—
F	0.004	—	0.010
G	0.015	—	0.050
H	0.007	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	4.01
C	0.20	—	0.30
C′	8.51	—	8.81
D	1.24	—	1.65
E	—	0.64	—
F	0.10	—	0.25
G	0.38	—	1.27
H	0.18	—	0.25
α	0°	—	8°

24-pin SKDIP (300mil) Outline Dimensions

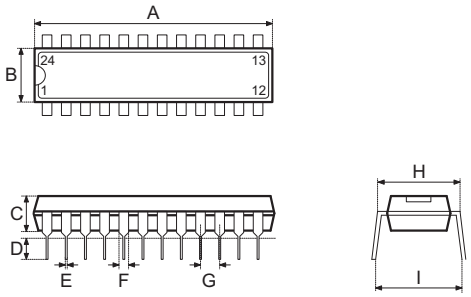


Fig1. Full Lead Packages

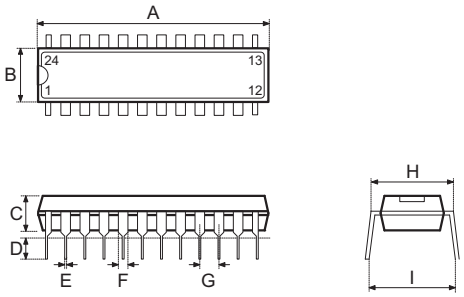


Fig2. 1/2 Lead Packages

• MS-001d (see fig1)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1230	—	1280
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

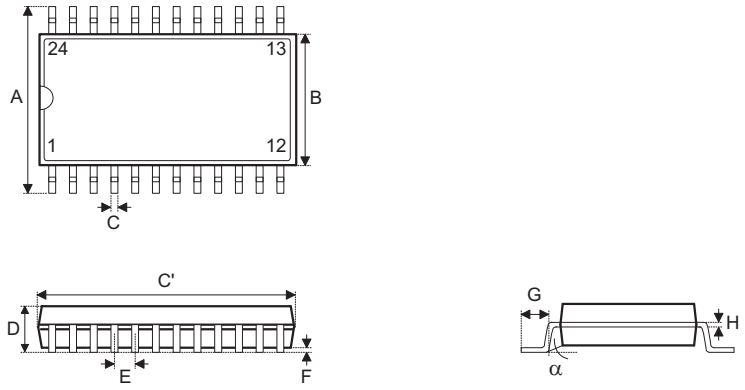
• MS-001d (see fig2)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1160	—	1195
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

• MO-095a (see fig2)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1145	—	1185
B	275	—	295
C	120	—	150
D	110	—	150
E	14	—	22
F	45	—	60
G	—	100	—
H	300	—	325
I	—	—	430

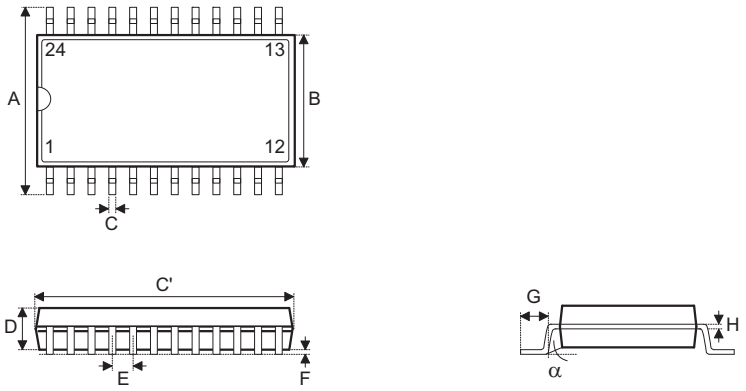
24-pin SOP (300mil) Outline Dimensions



• MS-013

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	393	—	419
B	256	—	300
C	12	—	20
C′	598	—	613
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

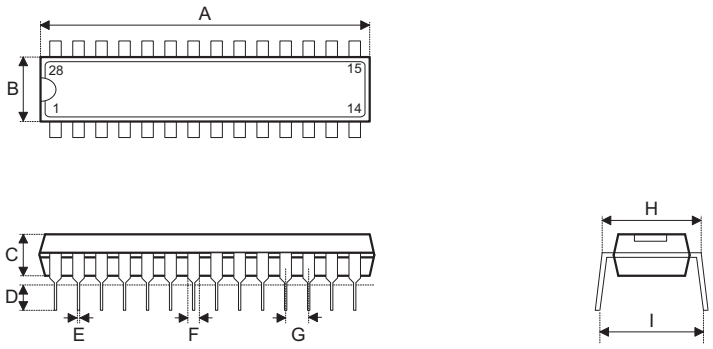
24-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.157
C	0.008	—	0.012
C′	0.335	—	0.346
D	0.054	—	0.060
E	—	0.025	—
F	0.004	—	0.010
G	0.022	—	0.028
H	0.007	—	0.010
α	0°	—	8°

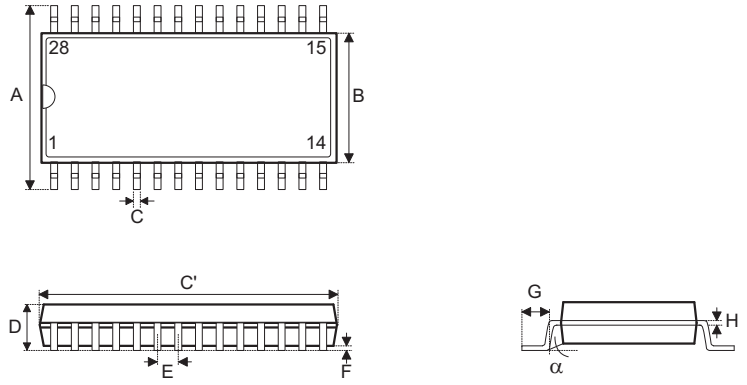
Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	3.99
C	0.20	—	0.30
C′	8.51	—	8.79
D	1.37	—	1.52
E	—	0.64	—
F	0.10	—	0.25
G	0.56	—	0.71
H	0.18	—	0.25
α	0°	—	8°

28-pin SKDIP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1375	—	1395
B	278	—	298
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	—	—	375

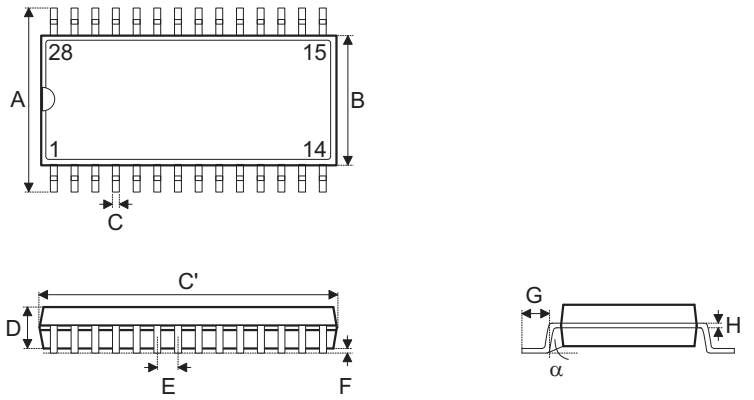
28-pin SOP (300mil) Outline Dimensions



• MS-013

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	393	—	419
B	256	—	300
C	12	—	20
C′	697	—	713
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

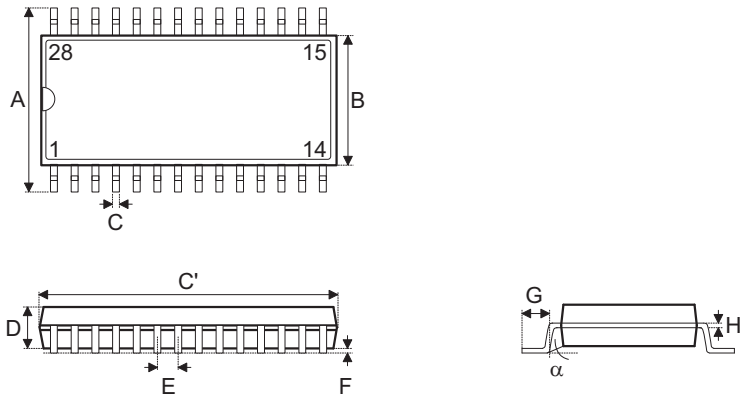
28-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.157
C	0.008	—	0.012
C′	0.386	—	0.394
D	0.054	—	0.060
E	—	0.025	—
F	0.004	—	0.010
G	0.022	—	0.028
H	0.007	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	3.99
C	0.20	—	0.30
C′	9.80	—	10.01
D	1.37	—	1.52
E	—	0.64	—
F	0.10	—	0.25
G	0.56	—	0.71
H	0.18	—	0.25
α	0°	—	8°

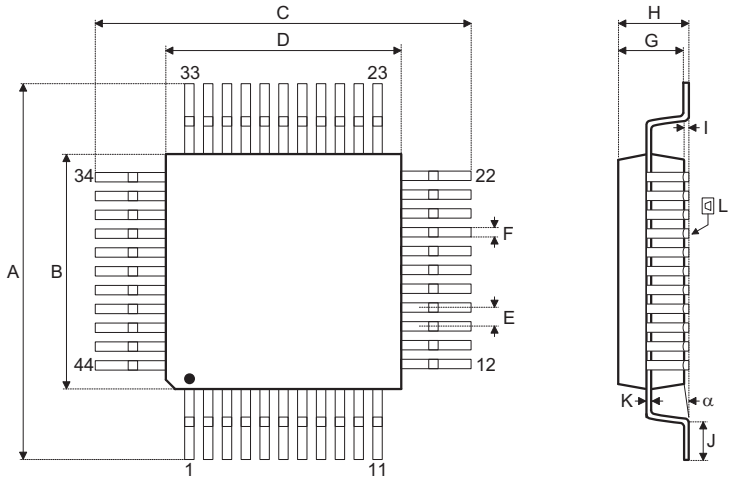
28-pin SSOP (209mil) Outline Dimensions



• MO-150

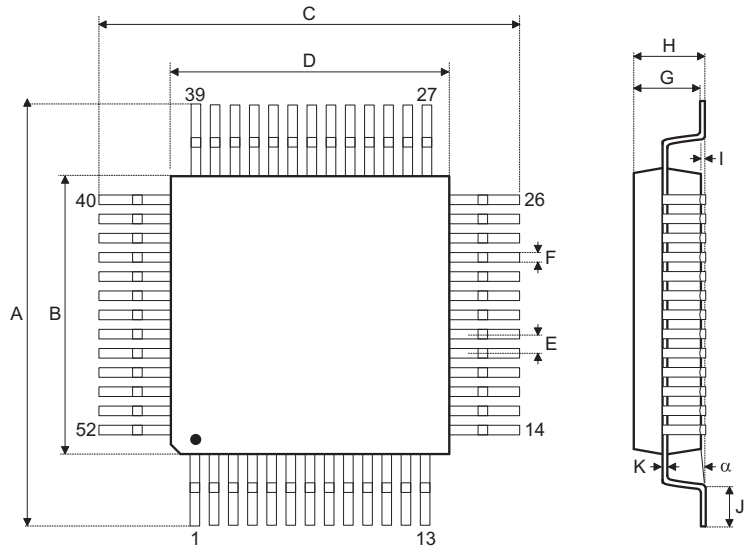
Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	7.40	—	8.20
B	5.00	—	5.60
C	0.22	—	0.33
C′	9.90	—	10.50
D	—		2.00
E	—	0.65	—
F	0.05	—	—
G	0.55	—	0.95
H	0.09	—	0.21
α	0°	—	8°

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



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

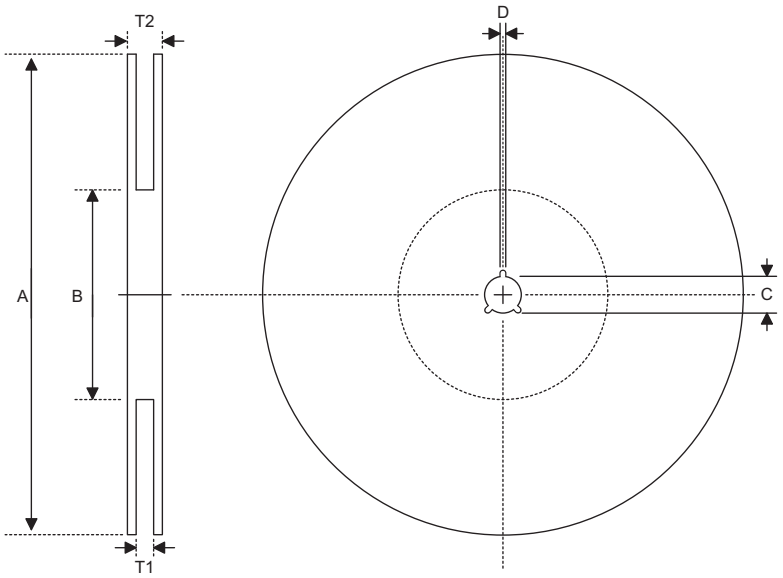
52-pin QFP (14mm×14mm) Outline Dimensions



Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	17.30	—	17.50
B	13.90	—	14.10
C	17.30	—	17.50
D	13.90	—	14.10
E	—	1.00	—
F	—	0.40	—
G	2.50	—	3.10
H	—	—	3.40
I	—	0.10	—
J	0.73	—	1.03
K	0.10	—	0.20
α	0°	—	7°

Product Tape and Reel Specifications

Reel Dimensions



SOP 16N (150mil), SSOP 20S (150mil), SSOP 24S (150mil), SSOP 28S (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 $^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	16.8 $^{+0.3/-0.2}$
T2	Reel Thickness	22.2±0.2

SSOP 16S

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 $^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	12.8 $^{+0.3/-0.2}$
T2	Reel Thickness	18.2±0.2

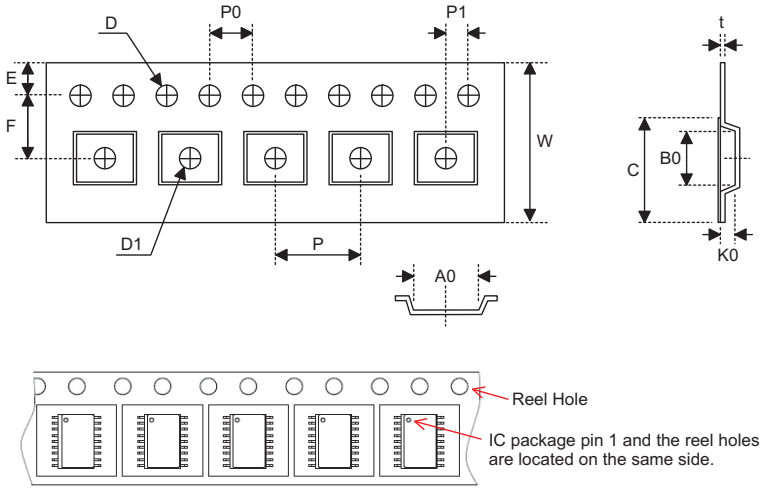
SOP 20W, SOP 24W, SOP 28W (300mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 $^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	24.8 $^{+0.3/-0.2}$
T2	Reel Thickness	30.2±0.2

SSOP 28S (209mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 +0.5/-0.2
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	28.4 +0.3/-0.2
T2	Reel Thickness	31.1 (max.)

Carrier Tape Dimensions



SOP 16N (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0±0.3
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	1.55 +0.10/-0.00
D1	Cavity Hole Diameter	1.50 +0.25/-0.00
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	10.3±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

SOP 20W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0 +0.3/-0.1
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5 +0.1/-0.0
D1	Cavity Hole Diameter	1.50 +0.25/-0.00
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.8±0.1
B0	Cavity Width	13.3±0.1
K0	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	21.3±0.1

SSOP 16S

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	12.0 +0.3/-0.1
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	5.5±0.1
D	Perforation Diameter	1.55±0.10
D1	Cavity Hole Diameter	1.50 +0.25/-0.00
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.4±0.1
B0	Cavity Width	5.2±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	9.3±0.1

SSOP 20S (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0 $^{+0.3/-0.1}$
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	1.5 $^{+0.1/-0.0}$
D1	Cavity Hole Diameter	1.50 $^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	9.0±0.1
K0	Cavity Depth	2.3±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

SSOP 24S (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0 $^{+0.3/-0.1}$
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	1.5 $^{+0.1/-0.0}$
D1	Cavity Hole Diameter	1.50 $^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	9.5±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

SSOP 28S (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0±0.3
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	1.55 +0.10/-0.00
D1	Cavity Hole Diameter	1.50 +0.25/-0.00
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	10.3±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

SSOP 28S (209mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0±0.3
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5 +0.1/-0.0
D1	Cavity Hole Diameter	1.50 +0.25/-0.00
P0	Perforation Pitch	4.0±0.2
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	8.4±0.1
B0	Cavity Width	10.65±0.10
K0	Cavity Depth	2.4±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	21.3±0.1

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

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

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

Holtek Semiconductor (USA), Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538, USA
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holtek.com