HOLTEK

Enhanced A/D Type 8-bit OTP MCU

HT46R068B/HT46R069B

Revision: V1.10 Date: May 02, 2012

www.holtek.com

Table of Contents

Features

CPU Features

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

Peripheral Features

- Up to 62 bidirectional I/O lines
- Up to 16 channel 12-bit ADC
- Up to 4 channel 8-bit PWM
- Single channel 12-bit DAC
- Serial Interfaces Module with Dual SPI and I^2C interfaces
- Single Serial SPI Interface
- Software controlled 4-SCOM lines LCD COM driver with 1/2 bias
- External interrupt input shared with an I/O line
- Two 8-bit programmable Timer/Event Counter with overflow interrupt and prescaler
- Single 16-bit programmable Timer/Event Counter with overflow interrupt
- Time-Base function
- Programmable Frequency Divider – PFD

General Description

The Enhanced A/D MCUs are a series of 8-bit high performance, RISC architecture microcontrollers specifically designed for a wide range of applications. The usual Holtek microcontroller features of low power consumption, I/O flexibility, timer functions, oscillator options, power down and wake-up functions, watchdog timer and low voltage reset, combine to provide devices with a huge range of functional options while still maintaining a high level of cost effectiveness. The fully integrated system oscillator HIRC, which requires no external components and which has three frequency selections, opens up a huge range of new application possibilities for these devices, some of which may include industrial control, consumer products, household appliances subsystem controllers, etc.

Selection Table

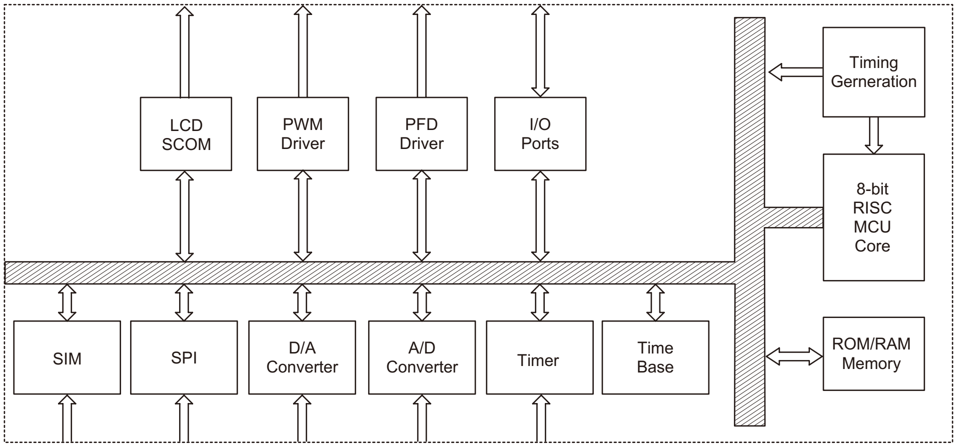
Part No.	Program Memory	Data Memory	I/O	8-bit Timer	16-bit Timer	Time Base	HIRC (MHz)	RTC (LXT)	LCD SCOM
HT46R068B	16Kx16	512x8	50	2	1	1	4/8/12	√(*)	4
HT46R069B	32Kx16	1024x8	62	2	1	1	4/8/12	√(*)	4

Part No.	A/D	PWM	D/A	Interface	PFD	Stack	Package
HT46R068B	12-bitx16	8-bitx4	12-bitx1	SPI/I^{2}C, SPI	√	8	28SKDIP/SOP/SSOP 44/52QFP
HT46R069B	12-bitx16	8-bitx4	12-bitx1	SPI/I^{2}C, SPI	√	8	44/52QFP 64LQFP

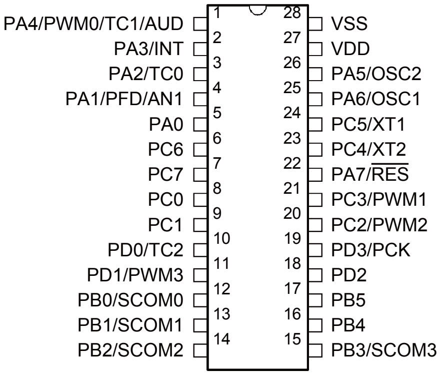
Note: "*" the oscillator is connected to the XT1/XT2 pins with TinyPower^{TM} design.

Block Diagram

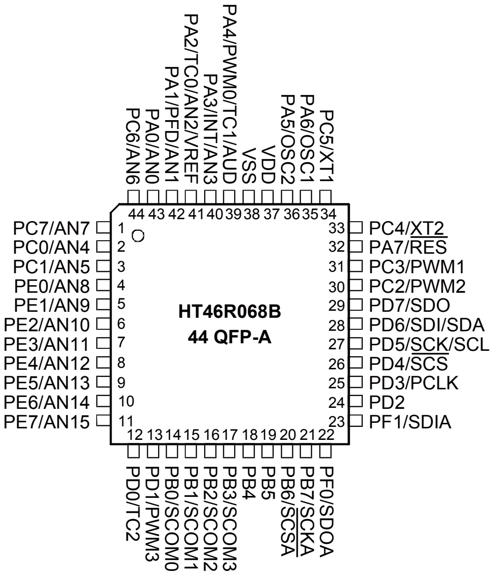
The following block diagram illustrates the main functional blocks.

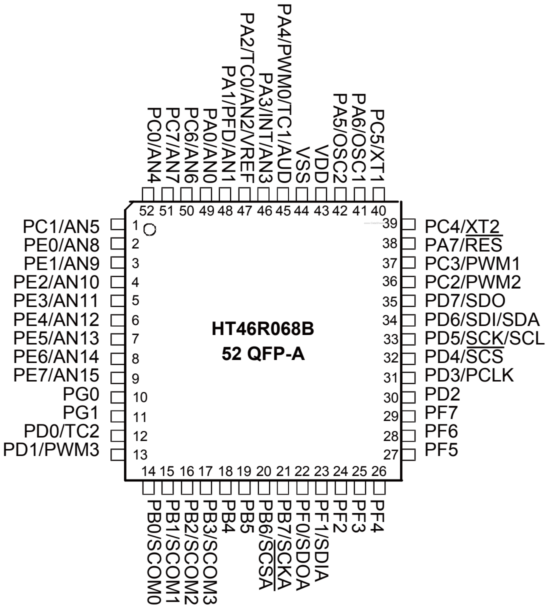


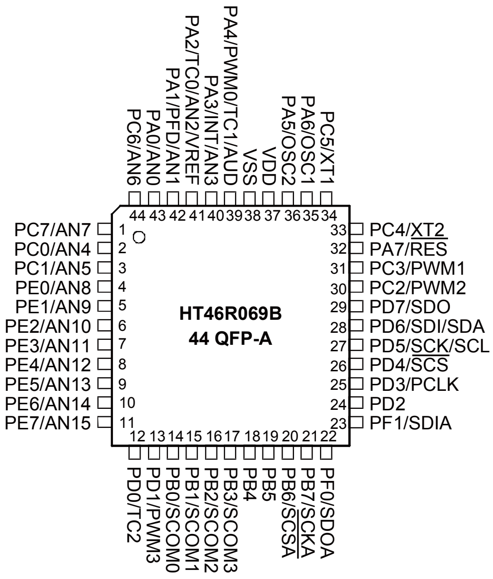
Pin Assignment

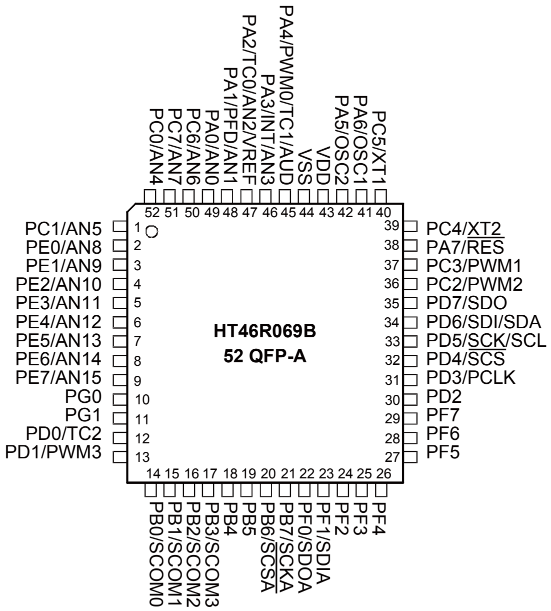


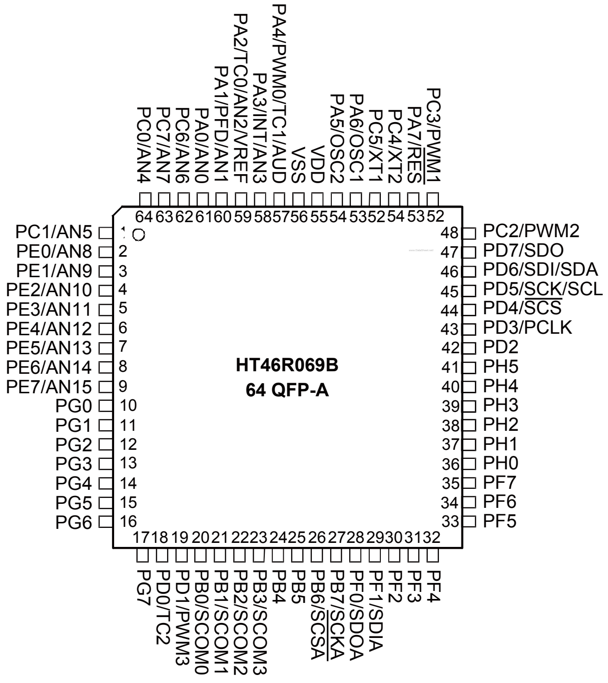
HT46R068B
28 SKDIP-A/SSOP-A/SOP-A






HT46R069B
44 QFP-A
1 PC7/AN7
2 PC0/AN4
3 PC1/AN5
4 PE0/AN8
5 PE1/AN9
6 PE2/AN10
7 PE3/AN11
8 PE4/AN12
9 PE5/AN13
10 PE6/AN14
11 PE7/AN15
12 PD0/TC2
13 PD1/PWM3
14 PB0/SCOM0
15 PB1/SCOM1
16 PB2/SCOM2
17 PB3/SCOM3
18 PB4
19 PB5
20 PB6/SCSA
21 PB7/SCKA
22 PF0/SDOA
23 PF1/SDIA
24 PD2
25 PD3/PCLK
26 PD4/SCS
27 PD5/SCK/SCL
28 PD6/SDI/SDA
29 PD7/SDO
30 PC2/PWM2
31 PC3/PWM1
32 PA7/RES
33 PC4/XT2
34 PC5/XT1
35 PA6/OSC1
36 PA5/OSC2
37 VDD
38 VSS
39 PA4/PWM0/TC1/AUD
40 PA3/INT/AN3
41 PA2/TC0/AN2/VREF
42 PA1/PFD/AN1
43 PA0/AN0
44 PC6/AN6


HT46R069B
52 QFP-A
1 PC1/AN5
2 PE0/AN8
3 PE1/AN9
4 PE2/AN10
5 PE3/AN11
6 PE4/AN12
7 PE5/AN13
8 PE6/AN14
9 PE7/AN15
10 PG0
11 PG1
12 PD0/TC2
13 PD1/PWM3
14 PB0/SCOM0
15 PB1/SCOM1
16 PB2/SCOM2
17 PB3/SCOM3
18 PB4
19 PB5
20 PB6/SCSA
21 PB7/SCKA
22 PF0/SDOA
23 PF1/SDIA
24 PF2
25 PF3
26 PF4
27 PF5
28 PF6
29 PF7
30 PD2
31 PD3/PCLK
32 PD4/SCS
33 PD5/SCK/SCL
34 PD6/SDI/SDA
35 PD7/SDO
36 PC2/PWM2
37 PC3/PWM1
38 PA7/RES
39 PC4/XT2
40 PC5/XT1
41 PA6/OSC1
42 PA5/OSC2
43 VDD
44 VSS
45 PA4/PWM0/TC1/AUD
46 PA3/INT/AN3
47 PA2/TC0/AN2/VREF
48 PA1/PFD/AN1
49 PA0/AN0
50 PC6/AN6
51 PC7/AN7
52 PC0/AN4


HT46R069B
64 QFP-A
1 PC1/AN5
2 PE0/AN8
3 PE1/AN9
4 PE2/AN10
5 PE3/AN11
6 PE4/AN12
7 PE5/AN13
8 PE6/AN14
9 PE7/AN15
10 PG0
11 PG1
12 PG2
13 PG3
14 PG4
15 PG5
16 PG6
17 PG7
18 PD0/TC2
19 PD1/PWM3
20 PB0/SCOM0
21 PB1/SCOM1
22 PB2/SCOM2
23 PB3/SCOM3
24 PB4
25 PB5
26 PB6/SCSA
27 PB7/SCKA
28 PF0/SDOA
29 PF1/SDIA
30 PF2
31 PF3
32 PF4
33 PF5
34 PF6
35 PF7
36 PH0
37 PH1
38 PH2
39 PH3
40 PH4
41 PH5
42 PD2
43 PD3/PCLK
44 PD4/SCS
45 PD5/SCK/SCL
46 PD6/SDI/SDA
47 PD7/SDO
48 PC2/PWM2
64 PC0/AN4
63 PC7/AN7
62 PC6/AN6
61 PA0/AN0
60 PA1/PFD/AN1
59 PA2/TC0/AN2/VREF
58 PA3/INT/AN3
57 PA4/PWM0/TC1/AUD
56 VSS
55 VDD
54 PA5/OSC2
53 PA6/OSC1
52 PC5/XT1
54 PC4/XT2
53 PA7/RES
52 PC3/PWM1

Pin Description

Pin Name	Function	OPT	I/T	O/T	Descriptions
PA0/AN0	PA0	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN0	ANCSR0	AN	—	A/D channel 0
PA1/PFD/AN1	PA1	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PFD	CTRL0	—	CMOS	PFD output
	AN1	ANCSR0	AN	—	A/D channel 1
PA2/TC0/AN2/VREF	PA2	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TC0	—	ST	—	External Timer 0 clock input
	AN2	ANCSR0	AN	—	A/D channel 2
	VREF	ACSR	AN	—	ADC reference input
PA3/INTB/AN3	PA3	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INTB	—	ST	—	External Interrupt input
	AN3	ANCSR0	AN	—	A/D channel 3
PA4/PWM0/TC1/AUD	PA4	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PWM0	CTRL0	—	CMOS	PWM output
	TC1	—	ST	—	External Timer 1 clock input
	AUD	—	—	AN	DAC output
PA5/OSC2	PA5	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OSC2	CO	—	OSC	Oscillator pin
PA6/OSC1	PA6	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OSC1	CO	OSC	—	Oscillator pin
PA7/$\overline{\text{RES}}$	PA7	PAWK	ST	NMOS	General purpose I/O. Register enabled wake-up.
	$\overline{\text{RES}}$	CO	ST	—	Reset input
PB0/SCOM0	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SCOM0	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB1/SCOM1	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SCOM1	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB2/SCOM2	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SCOM2	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB3/SCOM3	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SCOM3	SCOMC	—	SCOM	Software controlled 1/2 bias LCD COM
PB4,PB5	PB4,PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
PB6/$\overline{\text{SCSA}}$	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	$\overline{\text{SCSA}}$	—	ST	—	SPI Slave Select
PB7/SCKA	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SCKA	—	ST	CMOS	SPI Serial Clock
PC0/AN4	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN4	ANCSR0	AN	—	A/D channel 4
PC1/AN5	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN5	ANCSR0	AN	—	A/D channel 5
PC2/PWM2	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM2	CTRL2	—	CMOS	PWM output
PC3/PWM1	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM1	CTRL0	—	CMOS	PWM output
PC4/XT2	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	XT2	CO	—	LXT	Low frequency crystal pin
PC5/XT1	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	XT1	CO	—	LXT	Low frequency crystal pin

Pin Name	Function	OPT	I/T	O/T	Descriptions
PC6/AN6	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	AN6	ANCSR0	AN	—	A/D channel 6
PC7/AN7	PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN7	ANCSR0	AN	—	A/D channel 7
PD0/TC2	PD0	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	TC2	—	ST	—	External Timer 2 clock input
PD1/PWM3	PD1	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	PWM3	CTRL2	—	CMOS	PWM output
PD2	PD2	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
PD3/PCLK	PD3	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	PCLK	—	—	CMOS	Peripheral Clock output
PD4/$\overline{SCS}$	PD4	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	$\overline{SCS}$	—	ST	CMOS	SPI Slave Select
PD5/SCK/SCL	PD5	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	SCK	—	ST	CMOS	SPI Serial Clock
	SCL	—	ST	NMOS	I^{2}C Clock
PD6/SDI/SDA	PD6	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	SDI	—	ST	—	SPI Data input
	SDA	—	ST	NMOS	I^{2}C Data
PD7/SDO	PD7	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	SDO	—	—	CMOS	SPI Data output
PE0/AN8	PE0	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	AN8	ANCSR1	AN	—	A/D channel 8
PE1/AN9	PE1	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	AN9	ANCSR1	AN	—	A/D channel 9
PE2/AN10	PE2	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	AN10	ANCSR1	AN	—	A/D channel 10
PE3/AN11	PE3	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	AN11	ANCSR1	AN	—	A/D channel 11
PE4/AN12	PE4	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	AN12	ANCSR1	AN	—	A/D channel 12
PE5/AN13	PE5	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	AN13	ANCSR1	AN	—	A/D channel 13
PE6/AN14	PE6	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	AN14	ANCSR1	AN	—	A/D channel 14
PE7/AN15	PE7	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	AN15	ANCSR1	AN	—	A/D channel 15
PF0/SDOA	PF0	PFPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	SDOA	—	—	CMOS	SPI Data output
PF1/SDIA	PF1	PFPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
	SDIA	—	ST	—	SPI Data input
PF2~PF7	PFn	PFPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
PG0~PG7	PGn	PGPU	ST	CMOS	General purpose I/O. Register enabled pull-up .
PH0~PH5	PHn	PHPU	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 or output

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	100mA
Total Power Dissipation	500mW
Storage Temperature	-50°C to 125°C
Operating Temperature	-40°C to 85°C
I_{OH} Total	-100mA

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
			f_{SYS}=8MHz	3.0	—	5.5	V
			f_{SYS}=12MHz	4.5	—	5.5	V
I_{DD1}	Operating Current (HXT, HIRC, ERC)	3V	No load, f_{SYS}=4MHz	—	0.8	1.2	mA
		5V		—	1.5	2.25	mA
I_{DD2}	Operating Current (HXT, HIRC, ERC)	3V	No load, f_{SYS}=8MHz	—	1.4	2.1	mA
		5V		—	2.8	4.2	mA
I_{DD3}	Operating Current (HXT, HIRC, ERC)	5V	No load, f_{SYS}=12MHz	—	4	6	mA
I_{DD4}	Operating Current (HIRC + LXT, Slow Mode)	3V	No load, f_{SYS}=32768Hz (LXT on OSC1/OSC2, LVR disabled, LXTLP=1)	—	5	10	μA
		5V		—	12	24	μA
		3V	No load, f_{SYS}=32768Hz (LXT on XT1/XT2, LVR disabled, LXTLP=1)	—	5	10	μA
		5V		—	10	20	μA
I_{STB1}	Standby Current (LIRC On, LXT Off)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	Standby Current (LIRC Off, LXT Off)	3V	No load, system HALT	—	—	1	μA
		5V		—	—	2	μA
I_{STB3}	Standby Current (LIRC Off, LXT On, LXTLP=1)	3V	No load, system HALT (LXT on OSC1/OSC2)	—	—	5	μA
		5V		—	—	10	μA
		3V	No load, system HALT (LXT on XT1/XT2)	—	—	3	μA
		5V		—	—	5	μA
V_{IL1}	Input Low Voltage for I/O, TCn and INT	—	—	0	—	0.3V_{DD}	V
V_{IH1}	Input High Voltage for I/O, TCn and INT	—	—	0.7V_{DD}	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	0.4V_{DD}	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	0.9V_{DD}	—	V_{DD}	V
V_{LVR1}	Low Voltage Reset 1	—	V_{LVR}=4.2V	3.98	4.2	4.42	V
V_{LVR2}	Low Voltage Reset 2	—	V_{LVR}=3.15V	2.98	3.15	3.32	V
V_{LVR3}	Low Voltage Reset 3	—	V_{LVR}=2.1V	1.98	2.1	2.22	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVD1}	Low Voltage Detector Voltage 1	—	V_{LVD}= 4.4 V	4.12	4.4	4.70	V
V_{LVD2}	Low Voltage Detector Voltage 2	—	V_{LVD}= 3.3 V	3.12	3.3	3.50	V
V_{LVD2}	Low Voltage Detector Voltage 3	—	V_{LVD}= 2.2 V	2.08	2.2	2.32	V
I_{OL1}	I/O Port Sink Current (PA, PB, PC, PD, PE, PF, PG, PH)	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V		10	20	—	mA
I_{OH}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
		5V		-5	-10	—	mA
I_{OL2}	PA7 Sink Current	5V	V_{OL}=0.1V_{DD}	2	3	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
I_{SCOM}	SCOM Operating Current	5V	SCOMC, ISEL[1:0]=00	17.5	25.0	32.5	μA
			SCOMC, ISEL[1:0]=01	35	50	65	μA
			SCOMC, ISEL[1:0]=10	70	100	130	μA
			SCOMC, ISEL[1:0]=11	140	200	260	μA
V_{SCOM}	V_{DD}/2 Voltage for LCD COM	5V	No load	0.475	0.500	0.525	V_{DD}

Note: The standby current (I_{STB1}~I_{STB3}) and I_{DD4} are measured with all I/O pins in input mode and tied to V_{DD}.

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock	—	2.2V~5.5V	32	—	4000	kHz
			3.0V~5.5V	32	—	8000	kHz
			4.5V~5.5V	32	—	12000	kHz
f_{HIRC}	System Clock (HIRC)	3V/5V	Ta=25°C	-2%	4	+2%	MHz
		3V/5V	Ta=25°C	-2%	8	+2%	MHz
		5V	Ta=25°C	-2%	12	+2%	MHz
		3V/5V	Ta=0~70°C	-5%	4	+5%	MHz
		3V/5V	Ta=0~70°C	-5%	8	+5%	MHz
		5V	Ta=0~70°C	-5%	12	+5%	MHz
		2.2V~3.6V	Ta=0~70°C	-8%	4	+8%	MHz
		3.0V~5.5V	Ta=0~70°C	-8%	4	+8%	MHz
		3.0V~5.5V	Ta=0~70°C	-8%	8	+8%	MHz
		4.5V~5.5V	Ta=0~70°C	-8%	12	+8%	MHz
		2.2V~3.6V	Ta=-40°C~85°C	-12%	4	+12%	MHz
		3.0V~5.5V	Ta=-40°C~85°C	-12%	4	+12%	MHz
		3.0V~5.5V	Ta=-40°C~85°C	-12%	8	+12%	MHz
		4.5V~5.5V	Ta=-40°C~85°C	-12%	12	+12%	MHz
f_{ERC}	System Clock (ERC)	5V	Ta=25°C, R=120KΩ*	-2%	4	+2%	MHz
		5V	Ta=0~70°C, R=120KΩ*	-5%	4	+5%	MHz
		5V	Ta=-40°C~85°C, R=120KΩ*	-7%	4	+7%	MHz
		2.2V~5.5V	Ta=-40°C~85°C, R=120KΩ*	-11%	4	+11%	MHz
f_{LXT}	System Clock (LXT)	—	—	—	32768	—	Hz
t_{TIMER}	Timer Input Frequency (TCn)	—	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	3V	—	5	10	15	kHz
		5V	—	6.5	13	19.5	kHz
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up time Period	—	—	2	—	128	t_{SYS} t_{SYS} t_{SYS}
t_{INT}	Interrupt Fulse Width	—	—	1	—	—	μs
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1	2	ms
RESTD	Reset Delay Time	—	—	—	100	—	ms

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

2.*For f_{REC}, as the resistor tolerance will influence the frequency a percision 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	3V	t_{AD}=0.5μs	-2	—	2	LSB
		5V					
INL	ADC Integral Non-Linearity	3V	t_{AD}=0.5μs	-4	—	4	LSB
		5V					
I_{ADC}	Additional Power Consumption if A/D Converter is Used	3V	—	—	0.5	0.75	mA
		5V		—	1.0	1.5	mA

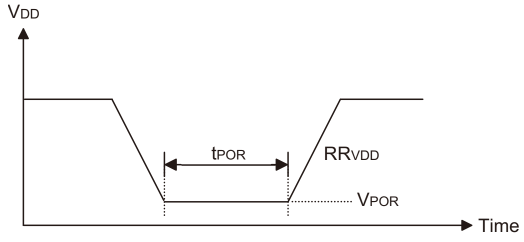
DAC Electrical Characteristics

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DAC}	DAC operating voltage	—	—	2.4	—	—	V
I_Q	DAC quiescent current	5V	Code= 0000H V_{OL}=00H	—	2	3	mA
I_{DAC}	DAC operating current	5V	1 kHz sin wave, full-scale (8K sample rate)	—	3	4.5	mA
$\overline{RES}$	Resolution	—	—	—	—	12	bit
V_O	Output Voltage Level	—	—	0.01	—	0.99	V_{DD}

Power-on Reset Characteristics

Ta=25°C

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



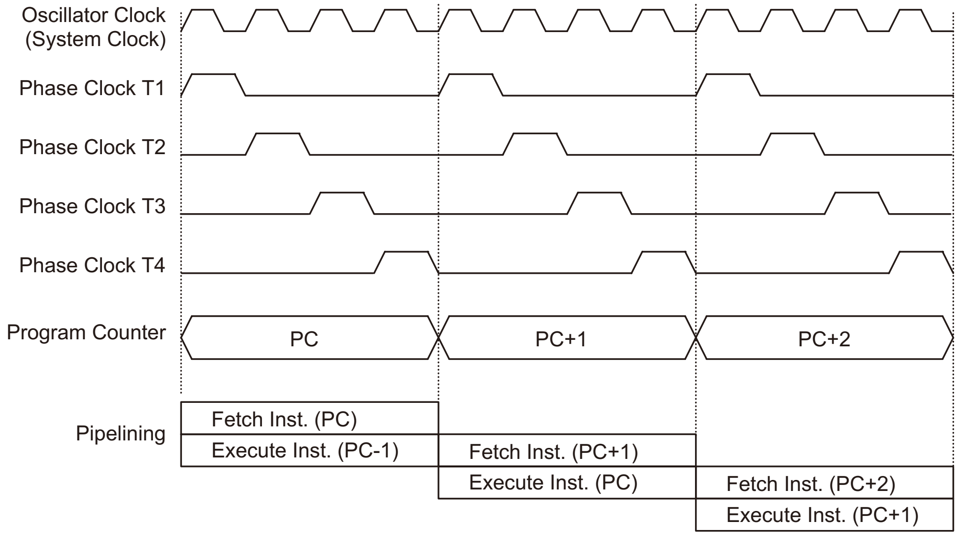
System Architecture

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.

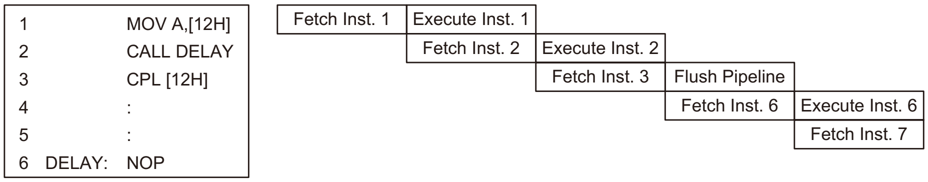
Clocking and Pipelining

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
```

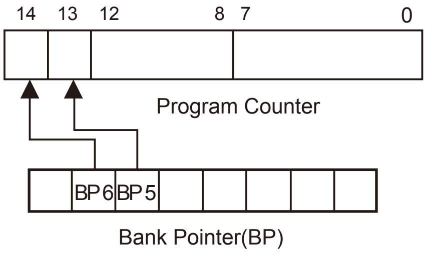
Instruction Fetching

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
HT46R068B	PC13~PC8	PCL7~PCL0
HT46R069B	PC14~PC8	

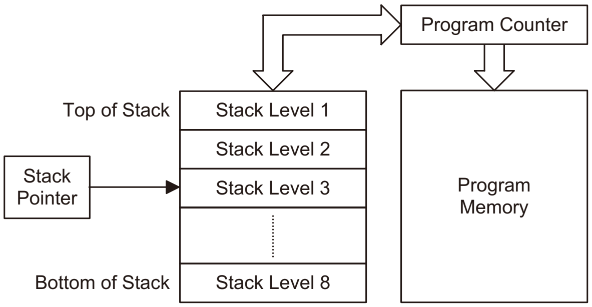


The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available 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
HT46R068B HT46R069B	8

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

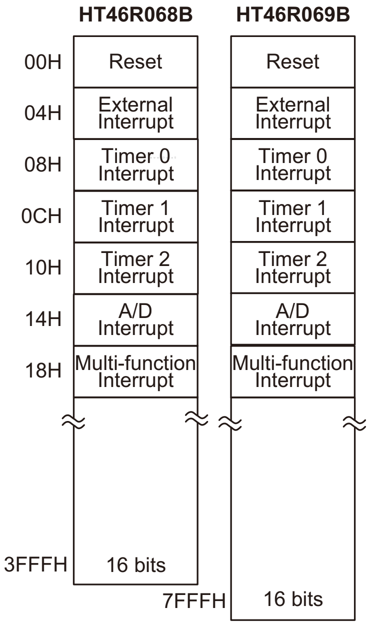
The Program Memory is the location where the user code or program is stored. The device is supplied with One-Time Programmable, OTP, memory where users can program their application code into the device. By using the appropriate programming tools, OTP devices offer users the flexibility to freely develop 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 16Kx16/32Kx16. 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	Banks
HT46R068B	16Kx16	0,1
HT46R069B	32Kx16	0~3

The devices have their Program Memory divided into a number of banks which are selected using the Bank Pointer register. The HT46R068B has its Program Memory divided into two Banks, Bank 0 and Bank 1. The required Bank is selected using Bit 5 of the BP Register. The HT46R069B has its Program Memory divided into four banks, from Bank0 to Bank3. The required Bank is selected using Bit 5 and Bit 6 of the BP Register.



Special Vectors

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

- Reset Vector
  This vector is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.
- External interrupt vector
  This vector is used by the external interrupt. If the external interrupt pin on the device receives an edge transition, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full. The external interrupt active edge transition type, whether high to low, low to high or both is specified in the CTRL1 register.
- Timer/Event 0/1/2 counter interrupt vector
  This internal vector is used by the Timer/Event Counters. If a Timer/Event Counter overflow occurs, the program will jump to its respective location and begin execution if the associated Timer/Event Counter interrupt is enabled and the stack is not full.
- Multi-function interrupt vector
  The Multi-function Interrupt vector is shared by several internal functions: a Time Base overflow, an SPI/I^{2}C or SPIA 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:

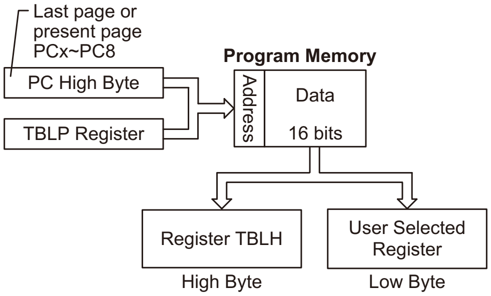


Table Program Example

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 "7F00H" which refers to the start address of the last page within the 32K Program Memory of the 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 "7F06H" 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(s)	Table Location														
	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
TABRDL [m]	1	1	1	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Note: PC14~PC8: Current Program Counter bits
@7~@0: Table Pointer TBLP bits
For the HT46R068B, the Table address location is 14 bits, i.e. from b13~b0
For the HT46R069B, 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,  060h
mov  bp, a         ; select the last bank of prog. memory
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 tempregl
                   ; data at prog. memory address "7F06" 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 "7F06" 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  7F00h         ; 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

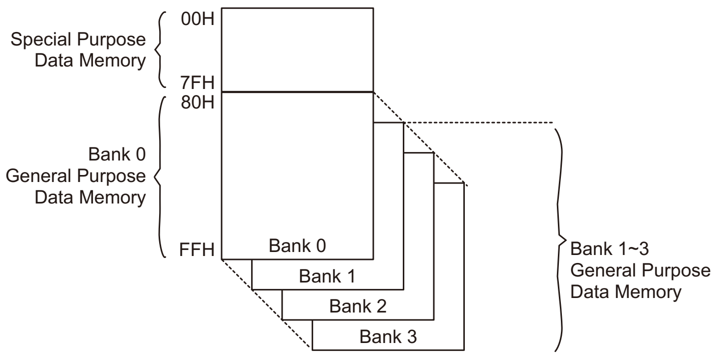
Divided into two sections, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some 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
HT46R068B	512x8	0~3
HT46R069B	1024x8	0~7

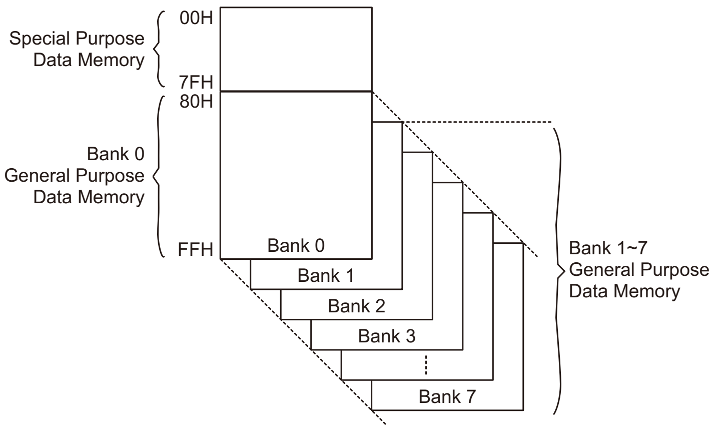
The two sections of Data Memory, the Special Purpose and General Purpose Data Memory are located at consecutive locations. All are implemented in RAM and are 8 bits wide but the length of each memory section is dictated by the type of microcontroller chosen. The start address of the Data Memory for all 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 several banks, which are selected using a Bank Pointer. Only data in Bank 0 can be directly addressed, data in Bank 1~Bank 7 must be indirectly addressed.



HT46R068B



HT46R069B

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 meomory 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 from Bank 0 to Bank 7. 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 from Bank 0 and Bank 7. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, is the address specified by the related Memory Pointer. Note that indirect addressing using MP1 and IAR1 must be used to access any data in Bank 1~Bank 7 . The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.

	HT46R068B	HT46R069B
00H	IAR0	IAR0
01H	MP0	MP0
02H	IAR1	IAR1
03H	MP1	MP1
04H	BP	BP
05H	ACC	ACC
06H	PCL	PCL
07H	TBLP	TBLP
08H	TBLH	TBLH
09H	WDTS	WDTS
0AH	STATUS	STATUS
0BH	INTC0	INTC0
0CH	TMR0	TMR0
0DH	TMR0C	TMR0C
0EH	TMR1	TMR1

	HT46R068B	HT46R069B
0FH	TMR1C	TMR1C
10H	PA	PA
11H	PAC	PAC
12H	PAPU	PAPU
13H	PAWK	PAWK
14H	PB	PB
15H	PBC	PBC
16H	PBPU	PBPU
17H	PC	PC
18H	PCC	PCC
19H	PCPU	PCPU
1AH	CTRL0	CTRL0
1BH	CTRL1	CTRL1
1CH	SCOMC	SCOMC
1DH	PWM1	PWM1
1EH	INTC1	INTC1
1FH	PWM0	PWM0
20H	ADRL	ADRL
21H	ADRH	ADRH
22H	ADCR	ADCR
23H	ACSR	ACSR
24H	MFIC	MFIC
25H	PD	PD
26H	PDC	PDC
27H	PDPU	PDPU
28H	PE	PE
29H	PEC	PEC
2AH	PEPU	PEPU
2BH	PF	PF
2CH	PFC	PFC
2DH	PFPU	PFPU
2EH		
2FH		
30H	PWM2	PWM2
31H	CTRL2	CTRL2
32H		
…	…	…
3AH		
3BH	PG	PG
3CH	PGC	PGC
3DH	PGPU	PGPU
3EH		PH
3FH		PHC
40H		PHPU
41H	TMR2L	TMR2L
42H	TMR2H	TMR2H

	HT46R068B	HT46R069B
43H	TMR2C	TMR2C
44H	PWM3	PWM3
45H		
46H	SIMC0	SIMC0
47H	SIMC1	SIMC1
48H	SIMD	SIMD
49H	SIMA/SIMC2	SIMA/SIMC2
4AH	SPIAC0	SPIAC0
4BH	SPIAC1	SPIAC1
4CH	SPIAD	SPIAD
4DH	ANCSR0	ANCSR0
4EH	ANCSR1	ANCSR1
4FH		
50H	DAL	DAL
51H	DAH	DAH
52H	VOL	VOL
53H		
54H	LVDC	LVDC
…..		
7FH		
Genernal purpose data memory	514 bytes 4 banks (80H~FFH)	1024 bytes 8 banks (80H~FFH)

Indirect Addressing Program Example

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

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

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Bank Pointer – BP

In the HT46R068B and HT46R069B devices, the Data Memory is divided into several Banks, from Bank 0 to Bank 7. A Bank Pointer is used to select the required Data Memory bank. Only data in Bank 0 can be directly addressed as data in Bank 1~Bank 7 must be indirectly addressed using Memory Pointer MP1 and Indirect Addressing Register IAR1. Using Memory Pointer MP0 and Indirect Addressing Register IAR0 will always access data from Bank 0, irrespective of the value of the Bank Pointer. Memory Pointer MP1 and Indirect Addressing Register IAR1 can indirectly address data in either Bank 0 or Bank 1~Bank 7 depending upon the value of the Bank Pointer.

The Data Memory is initialised to Bank 0 after a reset, except for the WDT time-out reset in the Idle/Sleep Mode, in which case, the Data Memory bank remains unaffected. It should be noted that Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within either Bank 0 or Bank 1~Bank 7. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer.

- **HT46R068B**

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

Bit 7~6 unimplemented, read as "0"

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

Bit 4~2 unimplemented, read as "0"

Bit 1,0 **DMBP1, DMBP0**: Data Memory Bank Pointer
00:Bank 0
01:Bank 1
10:Bank 2
11:Bank 3

- **HT46R069B**

Bit	7	6	5	4	3	2	1	0
BP	—	PMBP1	PMBP0	—	—	DMBP2	DMBP1	DMBP0
R/W	—	R/W	R/W	—	—	R/W	R/W	R/W
POR	—	0	0	—	—	0	0	0

Bit 7 unimplemented, read as "0"

Bit 6, 5 **PMBP1, PMBP0**: Program Memory Bank Pointer
00: Bank 0
01: Bank 1
10: Bank 2
11: Bank 3

Bit 4~3 unimplemented, read as "0"

Bit 2~0 **DMBP2, DMBP1, DMBP0**: Data Memory Bank Pointer
000: Bank 0
001: Bank 1
010: Bank 2
011: Bank 3
100: Bank 4
101: Bank 5
110: Bank 6
111: Bank 7

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 occured.

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 not 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.

System Control Registers – CTRL0, CTRL1, CTRL2

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

- **CTRL0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCFG	PFDCS	PWMSEL	PWMC1	PWMC0	PFDC	LXTLP	CLKMOD
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 **PCFG**: I/O configuration
0: (PWM0/TC1)/INT/PFD pin-shared with PA4/PA3/PA1
1: (PWM0/TC1)/INT/PFD pin-shared with PB5/PB4/PB3

Bit 6 **PFDCS**: PFD clock source
0: timer0
1: timer1

Bit 5 **PWMSEL**: PWM type selection
0: 6+2
1: 7+1

Bit 4 **PWMC1**: I/O or PWM1
0: I/O
1: PWM1

Bit 3 **PWMC0**: I/O or PWM0
0: I/O
1: PWM0

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

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

Bit 0 **CLKMOD**: system clock mode selection.
0: High speed system clock
1: LXT system clock, high speed oscillator stopped

Note: If PWM0/1/2/3 output is selected by PWMC0/1/2/3 bit, f_{TP} comes always from f_{SYS}. (f_{TP} is the clock source for timer0, time base and PWM)

- **CTRL1 Register**

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

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

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

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

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

- **CTRL2 Register**

Bit	7	6	5	4	3	2	1	0
Name	DACEN	—	PWMC3	PWMC2	—	—	—	LXTEN
R/W	R/W	—	R/W	R/W	—	—	—	R/W
POR	0	—	0	0	—	—	—	1

Bit 7 **DACEN**: DAC disable/enable control
0: disable
1: enable

Bit 6 unimplemented, read as "0"

Bit 5 **PWMC3**: IO or PWM3 control
0: I/O
1: PWM3 output

Bit4 **PWMC2**: IO or PWM2 control
0: I/O
1: PWM2 output

Bit 3~1 unimplemented, read as "0"

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

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

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.

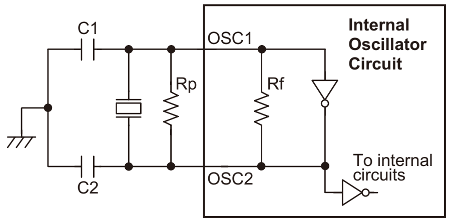
System Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for other functions such as the Watchdog Timer, Timer/Event Counter, Time Base etc. The system oscillator can be provided from a choice of three high speed oscillators, the HXT, ERC or HIRC oscillators, or a single low speed, LXT crystal oscillator. The LIRC oscillator is used only as a Watchdog Timer clock source.

Type	Name	Freq.	Pins	Function
External Crystal	HXT	400kHz~12MHz	OSC1/ OSC2	High Speed System Clock
External RC	ERC	400kHz~12MHz	OSC1	High Speed System Clock
Internal Highb Speed RC	HIRC	4, 8 or 12MHz	—	High Speed System Clock
External Low Speed Crystal	LXT	32768Hz	XT1/ XT2	Low Speed System Clock Clock source for: Watchdog, Time Base, Timer/Event Counters 0/1 Clock/SPI/SPIA
Internal Low Speed RC	LIRC	13kHz	—	Watchdog Timer Clock

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.

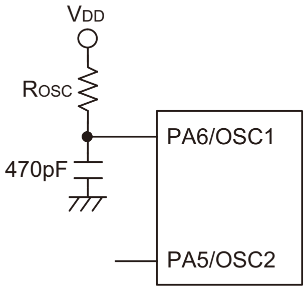
Crystal/Resonator Oscillator — HXT

Crystal Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
12MHz	8pF	10pF
8MHz	8pF	10pF
4MHz	8pF	10pF
1MHz	100pF	100pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

External RC Oscillator – ERC

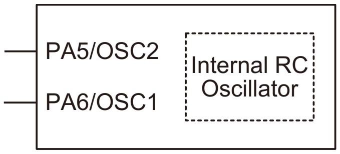
Using the ERC oscillator only requires that a resistor, with a value between 24kΩ and 1.5MΩ, is connected between OSC1 and V_{DD}, and a capacitor is connected between OSC and ground, providing a low cost oscillator configuration. It is only the external resistor that determines the oscillation frequency; 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 120K 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 PA6, leaving pin PA5 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 PA5 and PA6 are free for use as normal I/O pins.



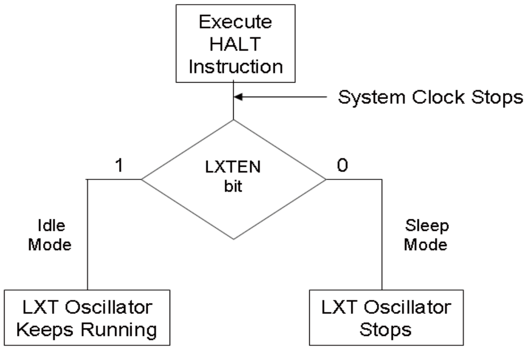
Note: PA5/PA6 used as normal I/Os

Internal RC Oscillator — HIRC

External 32768Hz Crystal Oscillator – LXT

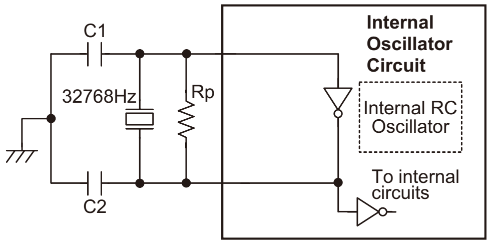
The LXT oscillator is used both as the slow system clock and also as a selectable source clock for some peripheral functions including the Watchdog Timer, Time Base, Timer/Event Counters and SPI functions. It must be first enabled using a configuration option.

To select the LXT oscillator to be the low speed system oscillator, the CLKMOD bit in the CTRL0 register should be set high. When a HALT instruction is executed, the system clock is stopped, but the LXTEN bit in the CTRL2 register determines if the LXT oscillator continues running when the microcontroller powers down. Setting the LXTEN bit high will enable the LXT to keep running after a HALT instruction is executed and enable the LXT oscillator to remain as a possible clock source for the Watchdog Timer, the Time-Base and the Timer/Event Counter 0/1.



LXTEN Bit Function

The LXT oscillator is implemented using a 32768Hz crystal connected to pins XT1/XT2. However, for some crystals and to ensure oscillation and accurate frequency generation, it is normally necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer specification. The external parallel feedback resistor, Rp, may also be required.



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_P=5M~10MΩ is recommended.		

32768Hz Crystal Recommended Capacitor Values

LXT Oscillator Low Power Function

The LXT oscillator can function in one of two modes, the Quick Start Mode and the Low Power Mode. The mode selection is executed using the LXTLP bit in the CTRL0 register.

LXTLP Bit	LXT Mode
0	Quick Start
1	Low-power

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

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

Internal Low Speed Oscillator – LIRC

The LIRC is a fully self-contained free running on-chip RC oscillator with a typical frequency of 13kHz at 5V requiring no external components. When the device enters the Idle/Sleep Mode, the system clock will stop running but the WDT oscillator continues to free-run and to keep the watchdog active. However, to preserve power in certain applications the LIRC can be disabled via a configuration option.

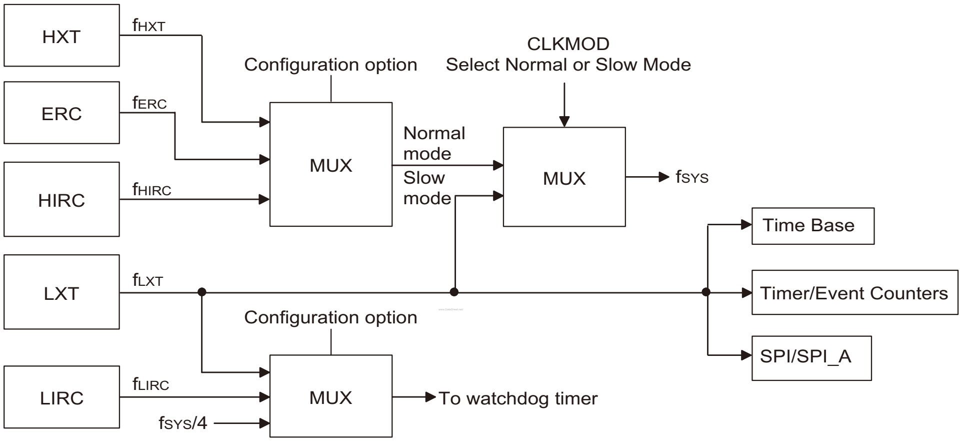
Operating Modes

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

Mode Types and Selection

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

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



System Clock Configurations

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.

- **Operating Mode Control**

HALT Instruction	CLKMOD bit	LXTEN bit	High Speed System Clock XTAL/IRC/ERC	Low Speed System Clock LXT	Operating Mode
Not executed	0	x	Run	On	Normal
	1	x	Stop	On	Slow
Executed	x	1	Stop	On	Idle
	x	0	Stop	Off	Sleep

Mode Switching

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

When a HALT instruction is executed and the LXT oscillator is not running, the system enters the Sleep mode the following conditions exist:

- The system oscillator will stop running and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the LIRC oscillator. The WDT will stop if its clock source originates from the system clock.
- The I/O ports will maintain their present condition.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Standby Current Considerations

As the main reason for entering the Idle/Sleep Mode is to keep the current consumption of the MCU to as low a value as possible, perhaps only in the order of several micro-amps, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised.

Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs.

If the configuration options have enabled the Watchdog Timer internal oscillator LIRC then this will continue to run when in the Idle/Sleep Mode and will thus consume some power. For power sensitive applications it may be therefore preferable to use the system clock source for the Watchdog Timer. The LXT, if configured for use, will also consume a limited amount of power, as it continues to run when the device enters the Idle Mode. To keep the LXT power consumption to a minimum level the LXTLP bit in the CTRL0 register, which controls the low power function, should be set high.

Wake-up

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

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

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

Pins PA0 to PA7 can be setup via the PAWUK register to permit a negative transition on the pin to wake-up the system. When a PA0 to PA7 pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction.

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

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

Wake-up Source	Oscillator Type	
	ERC, IRC	Crystal
External $\overline{\text{RES}}$	t_{RSDT} + t_{SST1}	t_{RSDT} + t_{SST2}
PA Port	t_{SST1}	t_{SST2}
Interrupt		
WDT Overflow		

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

2. t_{RSTD} is power-on delay, typical time=100ms

3. t_{SST1}= 2 or 128 t_{SYS}

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

Wake-up Delay Time

Watchdog Timer

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

Watchdog Timer Operation

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

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

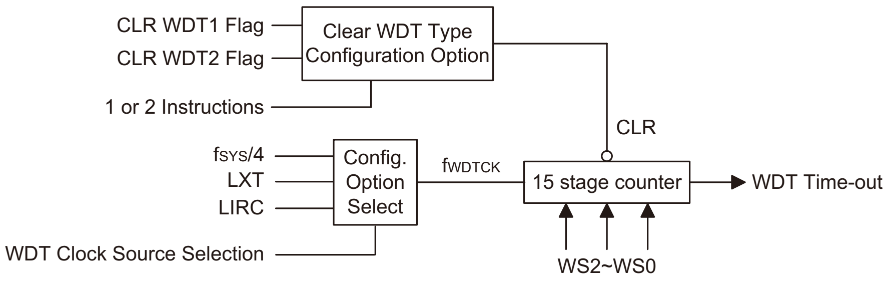
Configuration Option	CTRL1 Register	WDT Function
Disable	Disable	OFF
Disable	Enable	ON
Enable	x	ON

Watchdog Timer On/Off Control

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

The Watchdog Timer clock can emanate from three different sources, selected by configuration option. These are LXT, f_{SYS}/4, or LIRC. It is important to note that when the system enters the Idle/Sleep Mode the instruction clock is stopped, therefore if the configuration options have selected f_{SYS}/4 as the Watchdog Timer clock source, the Watchdog Timer will cease to function. For systems that operate in noisy environments, using the LIRC or the LXT as the clock source is therefore the recommended choice. The division ratio of the prescaler is determined by bits 0, 1 and 2 of the WDTS register, known as WS0, WS1 and WS2. If the Watchdog Timer internal clock source is selected and with the WS0, WS1 and WS2 bits of the WDTS register all set high, the prescaler division ratio will be 1:128, which will give a maximum time-out period.

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



Watchdog Timer

- **WDTS Register**

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

Bit 7~3 unimplemented, read as "0"

Bit 2~0 **WS2, WS1, WS0**: WDT time-out period selection

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

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.

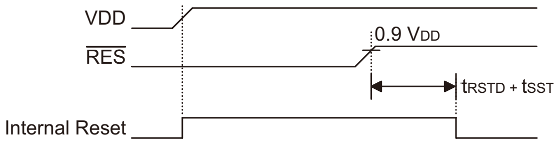
In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully 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 V_{DD} power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{RES}$ pin, whose additional time delay will ensure that the $\overline{RES}$ 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 $\overline{RES}$ 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.

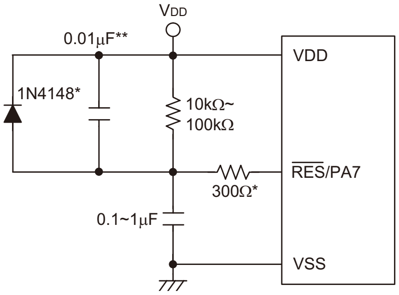


Power-On Reset Timing Chart

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

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

For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



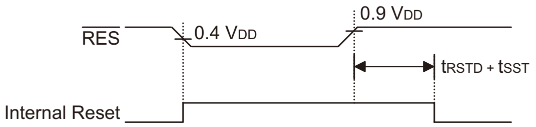
External $\overline{RES}$ Circuit

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

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

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

  This type of reset occurs when the microcontroller is already running and the $\overline{\text{RES}}$ 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.

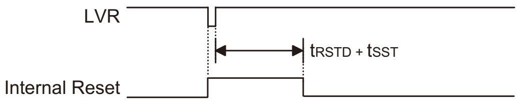


$\overline{\text{RES}}$ Reset Timing Chart

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

- 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.

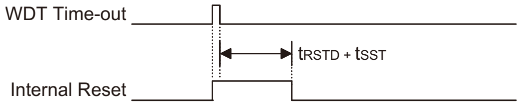


Low Voltage Reset Timing Chart

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

- Watchdog Time-out Reset during Normal Operation

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

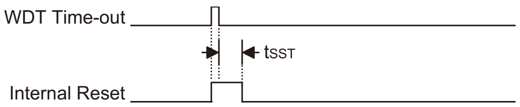


WDT Reset during Normal Operation Timing Chart

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

- 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 128 or 2 clock cycles if the system clock source is provided by ERC or HIRC. The SST is 128 clock cycle for HXT or LXT. It is described in the following table:

System Clock Source	HIRC		ERC	HXT
LXT	**None**	**XT1/XT2**		
Power on	128 HIRC	128 HIRC	128 ERC	128 HXT
Normal Mode wakeup	2 HIRC	2 HIRC	2 ERC	128 HXT
Slow Mode wakeup	X	2 LXT	X	X
Sleep Mode wakeup	2 HIRC	128 LXT	2 ERC	128 HXT

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	$\overline{\text{RES}}$ 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	HT46R068B	HT46R069B	Power-on Reset	$\overline{\text{RES}}$ or LVR Reset (Normal Operation)	$\overline{\text{RES}}$ or LVR Reset (Idle/Sleep)	WDT Time-out (Normal Operation)	WDT Time-out (Idle/Sleep)
PCL	●	●	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
MP0	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	●		––0– ––00	––0– ––00	––0– ––00	––0– ––00	––u– ––uu
BP		●	–00– –000	–00– –000	–00– –000	–00– –000	–uu– –uuu
ACC	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	●	●	–––– –111	–––– –111	–––– –111	–––– –111	–––– –uuu
STATUS	●	●	––00 xxxx	––uu uuuu	––01 uuuu	––1u uuuu	––11 uuuu
INTC0	●	●	–000 0000	–000 0000	–000 0000	–000 0000	–uuu uuuu
INTC1	●	●	–000 –000	–000 –000	–000 –000	–000 –000	–uuu –uuu
MFIC	●	●	–000 –000	–000 –000	–000 –000	–000 –000	–uuu –uuu
TMR0	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	●	●	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu
TMR1	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	●	●	0000 1–––	0000 1–––	0000 1–––	0000 1–––	uuuu u–––
TMR2L	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2H	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2C	●	●	0000 1–––	0000 1–––	0000 1–––	0000 1–––	uuuu u–––
PA	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAWK	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	●	●	–000 0000	–000 0000	–000 0000	–000 0000	–uuu uuuu
PB	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPU	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PE	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEPU	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PF	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFC	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFPU	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PG	●		–––– ––11	–––– ––11	–––– ––11	–––– ––11	–––– ––uu
PGC	●		–––– ––11	–––– ––11	–––– ––11	–––– ––11	–––– ––uu

Register	HT46R068B	HT46R069B	Power-on Reset	$\overline{\text{RES}}$ or LVR Reset (Normal Operation)	$\overline{\text{RES}}$ or LVR Reset (Idle/Sleep)	WDT Time-out (Normal Operation)	WDT Time-out (Idle/Sleep)
PGPU	●		–––– ––00	–––– ––00	–––– ––00	–––– ––00	–––– ––uu
PG		●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PGC		●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PGPU		●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PH		●	––11 1111	––11 1111	––11 1111	––11 1111	––uu uuuu
PHC		●	––11 1111	––11 1111	––11 1111	––11 1111	––uu uuuu
PHPU		●	––00 0000	––00 0000	––00 0000	––00 0000	––uu uuuu
CTRL0	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTRL1	●	●	1000 1010	1000 1010	1000 1010	1000 1010	uuuu uuuu
CTRL2	●	●	0–00 –––1	0–00 –––1	0–00 –––1	0–00 –––1	u–uu –––u
SCOMC	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWM0	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM1	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM2	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM3	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMC0	●	●	1110 000–	1110 000–	1110 000–	1110 000–	uuuu uuu–
SIMC1	●	●	1000 0001	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/ SIMC2	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SPIAC0	●	●	111– ––01	111– ––01	111– ––01	111– ––01	uuuu uuuu
SPIAC1	●	●	–000 0000	–000 0000	–000 0000	–000 0000	uuuu uuuu
SPIAD	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
DAL	●	●	0000 ––––	0000 ––––	0000 ––––	0000 ––––	uuuu ––––
DAH	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
VOL	●	●	0000 ––––	0000 ––––	0000 ––––	0000 ––––	uuuu ––––
LVDC	●	●	––00 ––––	––00 ––––	––00 ––––	––00 ––––	––uu ––––
ADRL	●	●	xxxx ––––	xxxx ––––	xxxx ––––	xxxx ––––	uuuu ––––
ADRH	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	●	●	01–– 0000	01–– 0000	01–– 0000	01–– 0000	uu–– uuuu
ACSR	●	●	11–– 0000	11–– 0000	11–– 0000	11–– 0000	uu–– uuuu
ANCSR0	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ANCSR1	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

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 and PFPU located in the Data Memory. The pull-high resistors are implemented using weak PMOS transistors. Note that pin PA7 does not have a pull-high resistor selection.

Port A Wake-up

If the HALT instruction is executed, the device will enter the Idle/Sleep Mode, where the system clock will stop resulting in power being conserved, a feature that is important for battery and other low-power applications.

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

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

- **HT46R068B**

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

- **HT46R069B**

Register Name	POR	Bit							
		7	6	5	4	3	2	1	0
PAWK	00H	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
PAC	FFH	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	00H	—	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PBC	FFH	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	00H	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PCC	FFH	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	00H	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PDC	FFH	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	00H	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PEC	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	FFH	PGC7	PGC6	PGC5	PGC4	PGC3	PGC2	PGC1	PGC0
PGPU	00H	PGPU7	PGPU6	PGPU5	PGPU4	PGPU3	PGPU2	PGPU1	PGPU0
PHC	3FH	—	—	PHC5	PHC4	PHC3	PHC2	PHC1	PHC0
PHPU	00H	—	—	PHPU5	PHPU4	PHPU3	PHPU2	PHPU1	PHPU0

"—" Unimplemented, read as "0"

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

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

PAPUn/PBPUn/PCPUn/PDPUn/PEPUn/PFPUn/PGPUn/PHPUn: 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, PGC, PHC which controls the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

Pin-shared Functions

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

- External Interrupt Input
  The external interrupt pin, INT, is pin-shared with an I/O pin. To use the pin as an external interrupt input the correct bits in the INTC0 register must be programmed. The pin must also be setup as an input by setting the PAC3 bit in the Port Control Register. A pull-high resistor can also be selected via the appropriate port pull-high resistor register. Note that even if the pin is setup as an external interrupt input the I/O function still remains.
- External Timer/Event Counter Input
  The Timer/Event Counter pins, TC0, TC1 and TC2 are pin-shared with I/O pins. For these shared pins to be used as Timer/Event Counter inputs, the Timer/Event Counter must be configured to be in the Event Counter or Pulse Width Capture Mode. This is achieved by setting the appropriate bits in the Timer/Event Counter Control Register. The pins must also be setup as inputs by setting the appropriate bit in the Port Control Register. Pull-high resistor options can also be selected using the port pull-high resistor registers. Note that even if the pin is setup as an external timer input the I/O function still remains.
- PFD Output
  The PFD function output is pin-shared with an I/O pin. The output function of this pin is chosen using the CTRL0 register. Note that the corresponding bit of the port control register, must setup the pin as an output to enable the PFD output. If the port control register has setup the pin as an input, then the pin will function as a normal logic input with the usual pull-high selection, even if the PFD function has been selected.
- PWM Outputs
  The PWM function whose outputs are pin-shared with I/O pins. The PWM output functions are chosen using the CTRL0 and CTRL2 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 Pinss
  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 up to 16 inputs to the A/D converter. All of these analog inputs are pin-shared with I/O pins. If these pins are to be used as A/D inputs and not as I/O pins then the corresponding PCRn bits in the ANCSR0 and ANCSR1 registers, must be properly setup. There are no configuration options associated with the A/D converter. If chosen as I/O pins, then full pull-high resistor configuration options remain, however if used as A/D inputs then any pull-high resistor configuration options associated with these pins will be automatically disconnected.

Pin Remapping Configuration

The pin remapping function enables the function pins PWM0/TC1, INT and PFD to be located on different port pins. It is important not to confuse the Pin Remapping function with the Pin-shared function, these two functions have no interdependence.

The PCFG bit in the CTRL0 register allows the three function pins PWM0/TC1, INT and PFD to be remapped to different port pins. After power up, this bit will be reset to zero, which will define the default port pins to which these three functions will be mapped. Changing this bit will move the functions to other port pins.

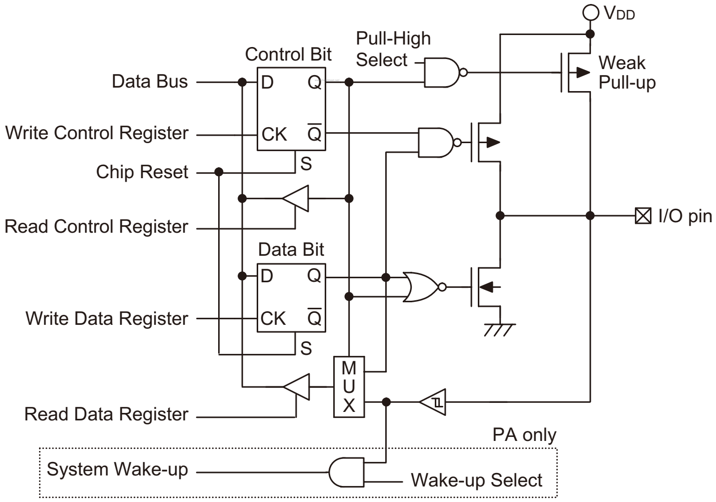
Examination of the pin names on the package diagrams will reveal that some pin function names are repeated, this indicates a function pin that can be remapped to other port pins. If the pin name is bracketed then this indicates its alternative location. Pin names without brackets indicates its default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

PCFG Bit Status		
PCFG Bit	**0**	**1**
Pin Mapping	(PWM0/TC1)/PA4 INT/PA3 PFD/PA1	[(PWM0/TC1)]/PB5 [INT]/PB4 [PFD]/PB3

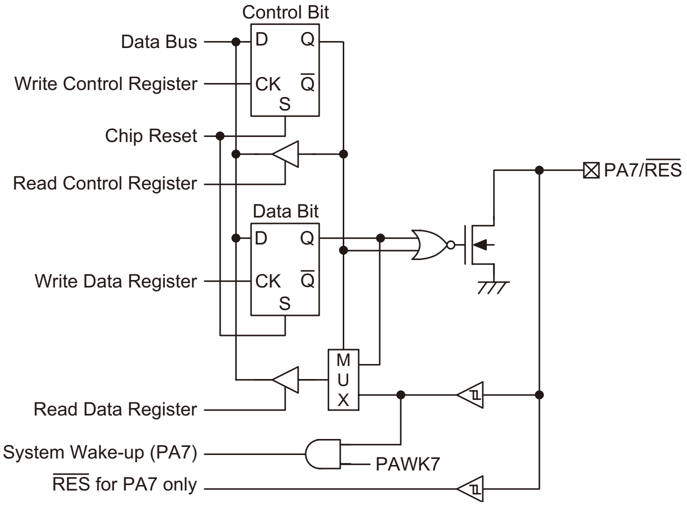
Pin Remapping

I/O Pin Structures

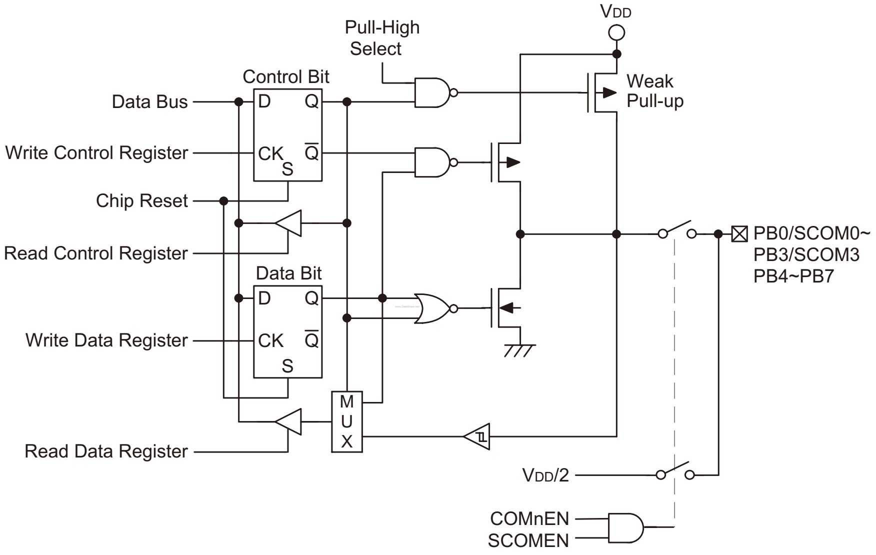
The diagrams illustrate the I/O pin internal structures. As the exact logical construction of the I/O pin may differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins.



Generic Input/Output Ports



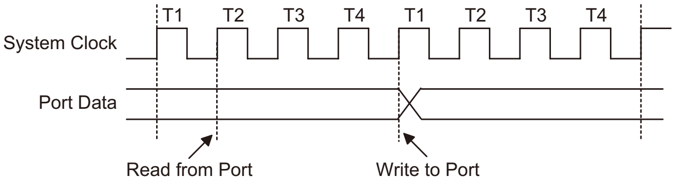
PA7 NMOS Input/Output Port



PB Input/Output Port

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 two 8-bit and one 16-bit timer. As the timers have three different operating modes, they can be configured to operate as a general timer, an external event counter or as a pulse width capture device. The provision of an internal prescaler to the clock circuitry on gives added range to the timers.

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

Configuring the Timer/Event Counter Input Clock Source

The Timer/Event Counter clock source can originate from various sources, an internal clock or an external pin. The internal clock source is used when the timer is in the timer mode or in the pulse width capture mode. For some Timer/Event Counters, this internal clock source is first divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register bits. An external clock source is used when the timer is in the event counting mode, the clock source being provided on an external timer pin TCn. Depending upon the condition of the TnEG bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.

Timer Registers – TMR0, TMR1, TMR2L, TMR2H

The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. These registers are known as TMR0, TMR1, TMR2L and TMR2H. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit Timer/Event Counter or FFFFH for the 16-bit Timer/Event Counters, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting. Note that to achieve a maximum full range count of FFH or FFFFH, the preload register must first be cleared to all zeros. It should be noted that after power-on, the preload registers will be in an unknown condition. Note that if the Timer/Event Counter is in an OFF condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter the next time an overflow occurs.

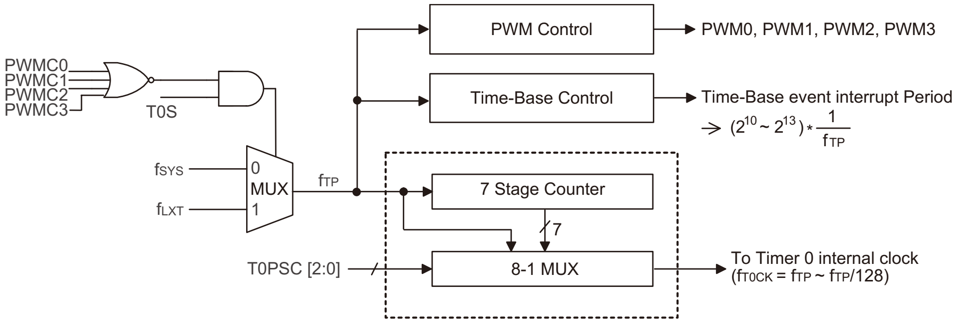
Timer Control Registers – TMR0C, TMR1C, TMR2C

The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in three different modes, the options of which are determined by the contents of their respective control register.

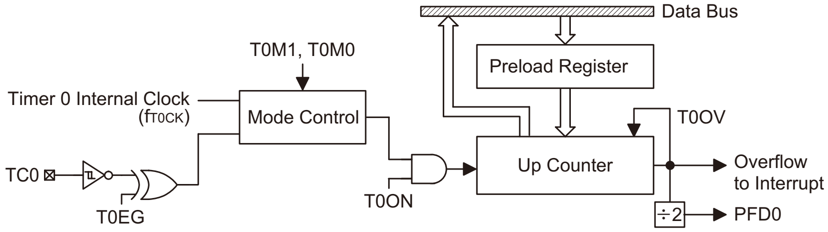
The Timer Control Register is known as TMRnC. It is the Timer Control Register together with its corresponding timer register that control the full operation of the Timer/Event Counter. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data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

To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width capture mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair TnM1/TnM0, must be set to the required logic levels.

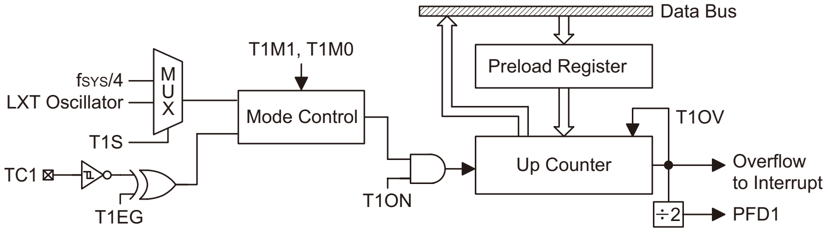
The timer-on bit, which is bit 4 of the Timer Control Register and known as TnON, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. Bits 0~2 of the Timer Control Register determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or pulse width capture mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as TnEG. The TnS bit selects the internal clock source.



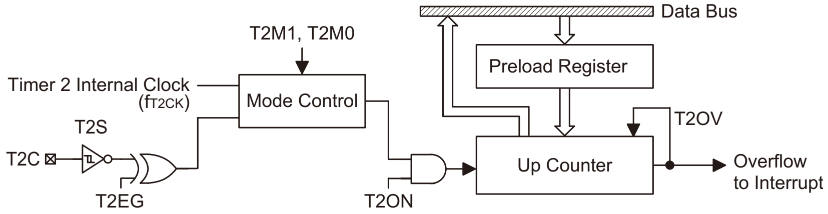
Clock Structure for Timer/PWM/Time Base



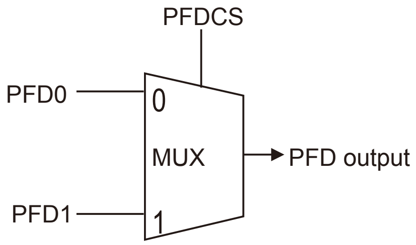
8-bit Timer/Event Counter 0 Structure



8-bit Timer/Event Counter 1 Structure



16-bit Timer/Event Counter 2 Structure



Note: If PWM0/PWM1/PWM2/PWM4 is enabled, then f_{TP} comes from f_{SYS} (ignore T0S)

- **TMR0C Register**

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

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

Bit 5 **T0S**: timer clock source
0: f_{SYS}
1: LXT oscillator
T0S selects the clock source for fTP which is provided for Timer 0~2, the Time-Base and the PWM. If the PWM is enabled, then f_{SYS} will be selected, overriding the T0S selection.

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 raising 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_{TP}
001: $f_{TP}/2$
010:$f_{TP}/4$
011:$f_{TP}/8$
100:$f_{TP}/16$
101:$f_{TP}/32$
110:$f_{TP}/64$
111:$f_{TP}/128$

- **TMR1C Register**

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

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

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

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

Bit 3 **T1EG**: Event counter active edge selection
0: count on raising edge
1: count on falling edge
Pulse Width Capture active edge selection
0: start counting on falling edge, stop on raising 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	T2S	T2ON	T2EG	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	1	—	—	—

Bit 7,6 **T2M1, T2M0**: Timer 1 operation mode selection
00: no mode available
01: event counter mode
10: timer mode
11: pulse width capture mode

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

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

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

Bit 2~0 unimplemented, read as "0"

To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width capture mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair TnM1/TnM0, must be set to the required logic levels.

The timer-on bit, which is bit 4 of the Timer Control Register and known as TnON, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. Bits 0~2 of the Timer Control Register determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or pulse width capture mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as TnEG. The TnS bit selects the internal clock source if used.

Timer Mode

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

Control Register Operating Mode
Select Bits for the Timer Mode

Bit7	Bit6
1	0

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

Event Counter Mode

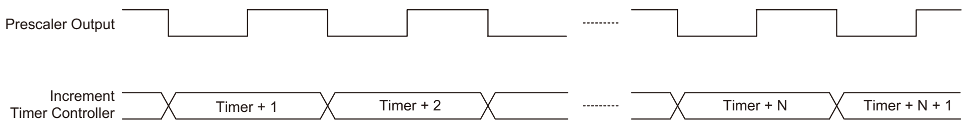
In this mode, a number of externally changing logic events, occurring on the external timer TCn pin, can be recorded by the Timer/Event Counter. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode
Select Bits for the Event Counter Mode

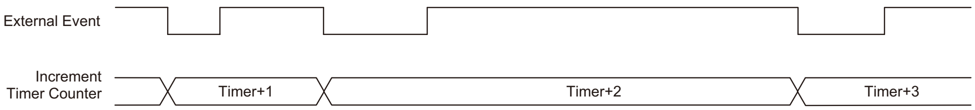
Bit7	Bit6
0	1

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 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.



Timer Mode Timing Chart



Event Counter Mode Timing Chart (TnEG=1)

Pulse Width Capture Mode

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

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

In this mode the internal clock, f_{SYS} , $f_{SYS}/4$ 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 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 T0PSC0~T0PSC2 of the TMR0C register can be used to define a division ratio for the internal clock source of the Timer/Event Counter enabling longer time out periods to be setup.

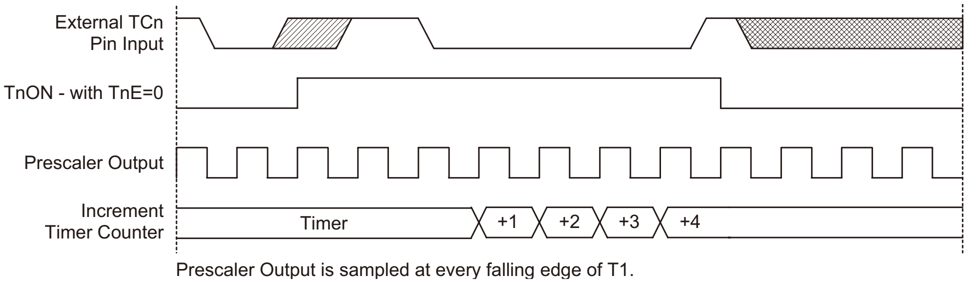
PFD Function

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

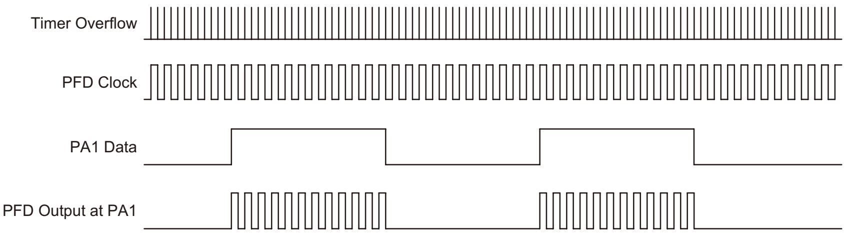
The Timer/Event Counter overflow signal is the clock source for the PFD function, which is controlled by PFDCS bit in CTRL0. For applicable devices the clock source can come from either Timer/Event Counter 0 or Timer/Event Counter 1. The output frequency is controlled by loading the required values into the timer prescaler and timer registers to give the required division ratio. The counter will begin to count-up from this preload register value until full, at which point an overflow signal is generated, causing both the PFD outputs to change state. The counter will then be automatically reloaded with the preload register value and continue counting-up.

If the CTRL0 register has selected the PFD function, then for PFD output to operate, it is essential for the Port A control register PAC, to setup the PFD pins as outputs. PA1 must be set high to activate the PFD. The output data bits can be used as the on/off control bit for the PFD outputs. Note that the PFD outputs will all be low if the output data bit is cleared to zero.

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



Pulse Width Capture Mode Timing Chart (TnE=0)



PFD Function

I/O Interfacing

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

Programming Considerations

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

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

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

Timer Program Example

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

PFD Programming Example

```
org  04h       ; external interrupt vector
org  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

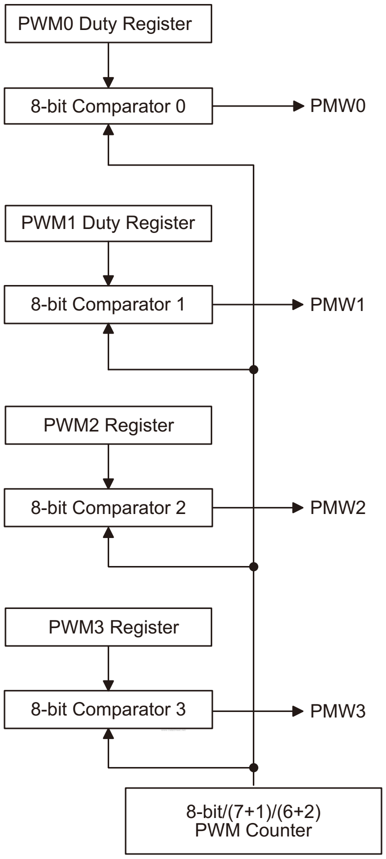
The device includes a Time Base function which is used to generate a regular time interval signal.

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

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

Pulse Width Modulator

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



PWM Block Diagram

Device	Channels	Mode	Pins	Registers
HT46R068B HT46R069B	4	6+2 7+1	PA4 PC3 PC2 PD1	PWM0 PWM1 PWM2 PWM3

PWM Operation

A single register, known as PWMn and located in the Data Memory is assigned to each Pulse Width Modulator channel. It is here that the 8-bit value, which represents the overall duty cycle of one modulation cycle of the output waveform, should be placed. To increase the PWM modulation frequency, each modulation cycle is subdivided into two or four individual modulation subsections, known as the 7+1 mode or 6+2 mode respectively. The required mode and the on/off control for each PWM channel is selected using the CTRL0 and CTRL2 registers. Note that when using the PWM, it is only necessary to write the required value into the PWMn register and select the required mode setup and on/off control using the CTRL0 and CTRL2 registers, the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware. The PWM clock source is the system clock f_{SYS}. This method of dividing the original modulation cycle into a further 2 or 4 sub-cycles enable the generation of higher PWM frequencies which allow a wider range of applications to be served. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood. As the PWM clock is the system clock, f_{SYS}, and as the PWM value is 8-bits wide, the overall PWM cycle frequency is $f_{SYS}/256$. However, when in the 7+1 mode of operation the PWM modulation frequency will be $f_{SYS}/128$, while the PWM modulation frequency for the 6+2 mode of operation will be $f_{SYS}/64$.

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

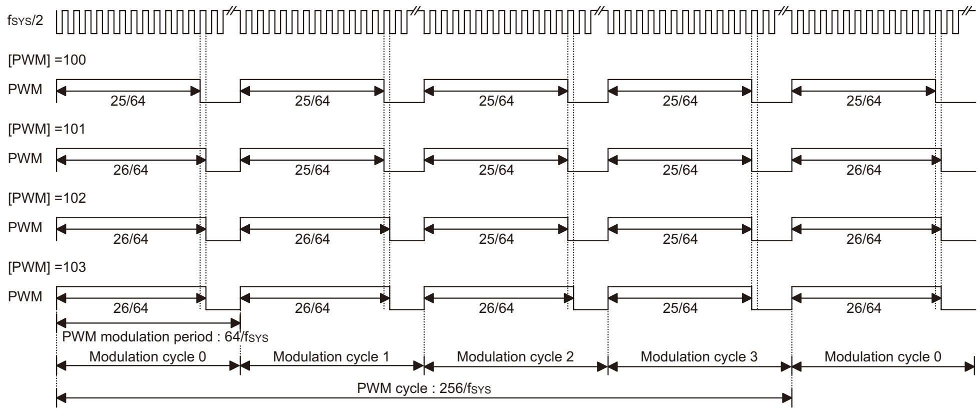
6+2 PWM Mode

Each full PWM cycle, as it is controlled by an 8-bit PWM register, has 256 clock periods. However, in the 6+2 PWM mode, each PWM cycle is subdivided into four individual sub-cycles known as modulation cycle 0 ~ modulation cycle 3, denoted as i in the table. Each one of these four sub-cycles contains 64 clock cycles. In this mode, a modulation frequency increase of four is achieved. The 8-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit2~bit7 is denoted here as the DC value. The second group which consists of bit0~bit1 is known as the AC value. In the 6+2 PWM mode, the duty cycle value of each of the four modulation sub-cycles is shown in the following table.

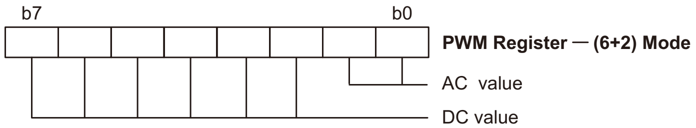
Parameter	AC (0~3)	DC (Duty Cycle)
Modulation cycle i (i=0~3)	i < AC	$\frac{DC+1}{64}$
	i≥AC	$\frac{DC}{64}$

6+2 Mode Modulation Cycle Values

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



6+2 PWM Mode



PWM Register for 6+2 Mode

7+1 PWM Mode

Each full PWM cycle, as it is controlled by an 8-bit PWM register, has 256 clock periods. However, in the 7+1 PWM mode, each PWM cycle is subdivided into two individual sub-cycles known as modulation cycle 0 ~ modulation cycle 1, denoted as i in the table. Each one of these two sub-cycles contains 128 clock cycles. In this mode, a modulation frequency increase of two is achieved. The 8-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit1~bit7 is denoted here as the DC value. The second group which consists of bit0 is known as the AC value. In the 7+1 PWM mode, the duty cycle value of each of the two modulation sub-cycles is shown in the following table.

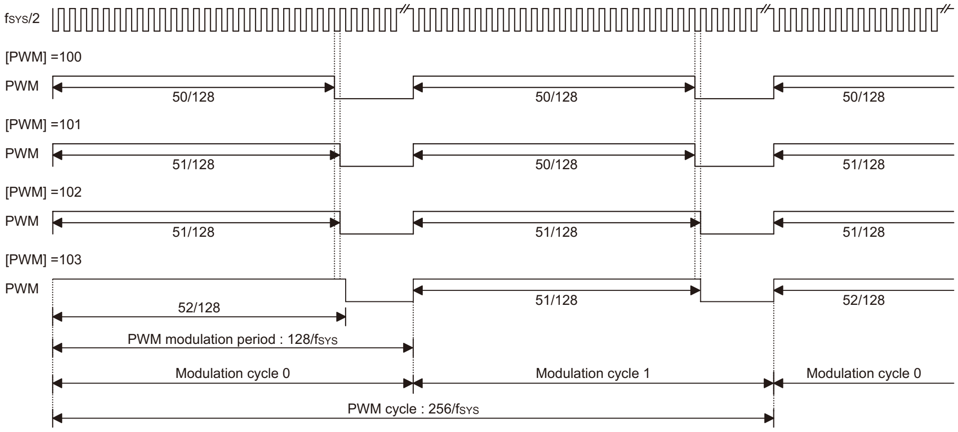
Parameter	AC (0~1)	DC (Duty Cycle)
Modulation cycle i (i=0~1)	i < AC	$\frac{DC+1}{128}$
	i≥AC	$\frac{DC}{128}$

7+1 Mode Modulation Cycle Values

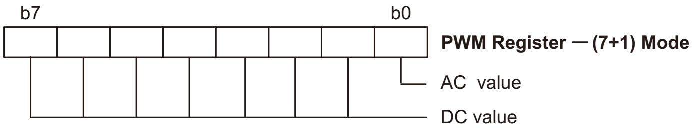
The following diagram illustrates the waveforms associated with the 7+1 mode PWM operation. It is important to note how the single PWM cycle is subdivided into 2 individual modulation cycles, numbered 0 and 1 and how the AC value is related to the PWM value.

PWM Output Control

The PWM outputs are pin-shared with the I/O pins PA4, PC2 and PC3. To operate as a PWM output and not as an I/O pin, the correct bits must be set in the CTRL0 and CTRL2 register. A zero value must also be written to the corresponding bit in the I/O port control register PAC.4, PCC.2 and PCC.3 to ensure that the corresponding PWM output pin is setup as an output. After these two initial steps have been carried out, and of course after the required PWM value has been written into the PWMn register, writing a high value to the corresponding bit in the output data register PA.4, PC.2 and PC.3 will enable the PWM data to appear on the pin. Writing a zero value will disable the PWM output function and force the output low. In this way, the Port data output registers can be used as an on/off control for the PWM function. Note that if the CTRL0 and CTRL2 registers have selected the PWM function, but a high value has been written to its corresponding bit in the PAC or PCC control register to configure the pin as an input, then the pin can still function as a normal input line, with pull-high resistor options.



7+1 PWM Mode



PWM Register for 7+1 Mode

PWM Programming Example

The following sample program shows how the PWM0 output is setup and controlled.

```
mov a,64h          ; setup PWM value of decimal 100
mov pwm0,a
set ctrl0.5        ; select the 7+1 PWM mode
set ctrl0.3        ; select pin PA4 to have a PWM function
clr pac.4          ; setup pin PA4 as an output
set pa.4           ; enable the PWM output
: :
clr pa.4           ; disable the PWM output_ pin
                   ; PA4 forced low
```

Analog to Digital Converter

The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a , they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the , 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 4/8-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.

Part No.	Input Channels	Conversion Bits	Input Pins
HT46R068B	16	12	PA0~PA3 PC0~PC1 PC6~PC7 PE0~PE7
HT46R069B			

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

A/D Converter Data Registers – ADRL, ADRH

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

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

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

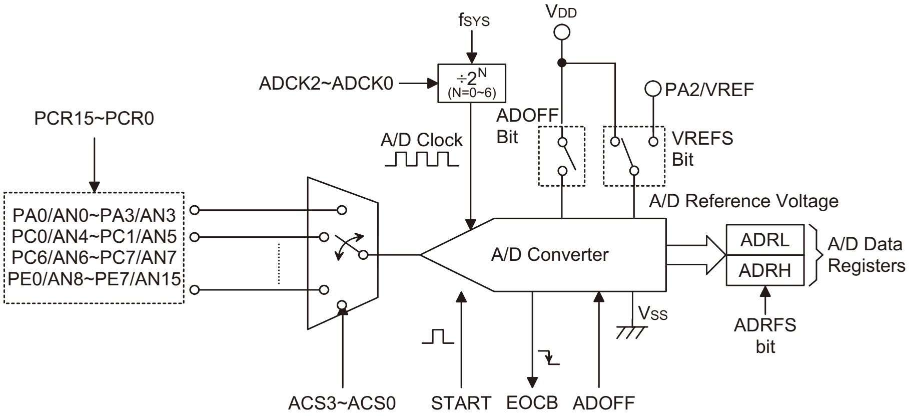
A/D Data Registers

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

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

The ACS3~ACS0 bits in the ADCR register define the channel number. As the device contains only one actual analog to digital converter circuit, each of the individual 8 analog inputs must be routed to the converter. It is the function of the ACS3~ACS0 bits in the ADCR register to determine which analog channel is actually connected to the internal A/D converter.

The two control registers, ANCSR1,ANCSR0, determine which pins on PA0~PA3, PC0, PC1, PC6, PC7, PE0~PE7 are used as analog inputs for the A/D converter and which pins are to be used as normal I/O pins. If the 16-bit address on PCR15~PCR0 has a value of "FFH", then all 16 pins, namely AN0~AN15 will all be set as analog inputs. To reduce the power consumption in normal run, the ADC module can be turned off by setting ADONB=1 in ACSR. Once the ADC module is turned off, the ADC module and analog channel have no power consumption no matter what voltage level is on analog input. If the I/O lines is selected as an I/O function and the analog input pin voltage is not equal to V_{DD} or V_{SS}, there user may have to take care IDD/ISTB current consumed by the pin-shared logic input function no matter the ADC module is on or off. If the ADC module is turned off, then all the ADC pin-shared I/O pins will be setup as normal I/Os.



A/D Converter Structure

- **ADRH, ADRL Register**

Bit	ADRH								ADRL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
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	—	—	ACS3	ACS2	ACS1	ACS0
R/W	R/W	R	—	—	R/W	R/W	R/W	R/W
POR	0	1	0	0	0	0	0	0

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

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

Bit 5, Bit 4 unimplemented, read as "0"

Bit 3~0 **ACS3, ACS2, ACS1, ACS0**: Select A/D channel
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000: AN8
1001: AN9
1010: AN10
1011: AN11
1100: AN12
1101: AN13
1110: AN14
1111: AN15

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

- **ACSR Register**

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

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

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

Bit 5 unimplemented, read as "0"

Bit 4 **VREFS**: Select ADC reference voltage
0: V_{DD}
1: VERF pin
This bit is used to select the reference voltage for the A/D converter. If the bit is high, then the A/D converter reference voltage is supplied on teh external VREF pin. If the pin is low, then the internal reference is used which is taken from the pwoer supply pin V_{DD}.

Bit 3 unimplemented, read as "0"

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

- **ANCSR0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCR7	PCR6	PCR5	PCR4	PCR3	PCR2	PCR1	PCR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PCR7**: Define PC7 is A/D input or not
0: Not A/D input
1: A/D input, AN7

Bit 6 **PCR6**: Define PC6 is A/D input or not
0: Not A/D input
1: A/D input, AN6

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

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

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

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

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

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

- **ANCSR1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCR15	PCR14	PCR13	PCR12	PCR11	PCR10	PCR9	PCR8
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 **PCR15**: Define PE7 is A/D input or not
0: Not A/D input
1: A/D input, AN15

Bit 6 **PCR14**: Define PE6 is A/D input or not
0: Not A/D input
1: A/D input, AN14

Bit 5 **PCR13**: Define PE5 is A/D input or not
0: Not A/D input
1: A/D input, AN13

Bit 4 **PCR12**: Define PE4 is A/D input or not
0: Not A/D input
1: A/D input, AN12

Bit 3 **PCR11**: Define PE3 is A/D input or not
0: Not A/D input
1: A/D input, AN11

Bit 2 **PCR10**: Define PE2 is A/D input or not
0: Not A/D input
1: A/D input, AN10

Bit 1 **PCR9**: Define PE1 is A/D input or not
0: Not A/D input
1: A/D input, AN9

Bit 0 **PCR8**: Define PE0 is A/D input or not
0: Not A/D input
1: A/D input, AN8

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

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

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

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

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

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

A/D Clock Period Examples

A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins on Port A, Port C and Port E. Bits PCR15~PCR0 in the ANCSR0 and ANCSR1 registers, determine whether the input pins are setup as normal input/output pins or whether they are setup as analog inputs. In this way, pins can be changed under program control to change their function from normal I/O operation to analog inputs and vice versa. Pull-high resistors, which are setup through register programming, apply to the input pins only when they are used as normal I/O pins, if setup as A/D inputs the pull-high resistors will be automatically disconnected. Note that it is not necessary to first setup the A/D pin as an input in the PAC, PCC and PEC port control registers to enable the A/D input as when the PCR15~PCR0 bits enable an A/D input, the status of the port control register will be overridden.

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCS2, 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 PCR15~PCR0 bits in the ANCSR0, ANCSR1 registers.
- Step 3
  Enable the A/D by clearing the ADONB in the ACSR register to zero.

- Step 4
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS3~ACS0 bits which are also contained in the register.
- Step 5
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, the INTC0 interrupt control register must be set to "1", the A/D converter interrupt bit, ADE, must also be set to "1".
- Step 6
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR register from "0" to "1" and then to "0" again. Note that this bit should have been originally set to "0".
- Step 7
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value. As an alternative method if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

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

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

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

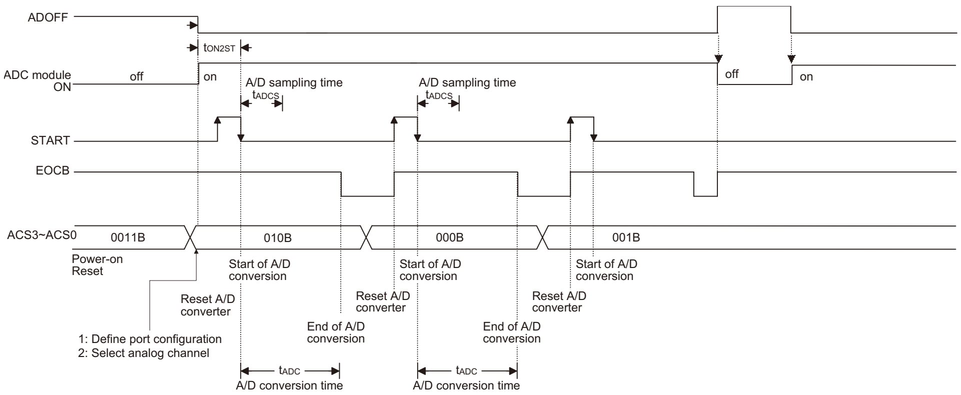
Programming Considerations

When programming, special attention must be given to the PCR[15:0] bits in the register. If these bits are all cleared to zero no external pins will be selected for use as A/D input pins allowing the pins to be used as normal I/O pins. When this happens the internal A/D circuitry will be power down. Setting the ADONB bit high has the ability to power down the internal A/D circuitry, which may be an important consideration in power sensitive applications.

A/D Transfer Function

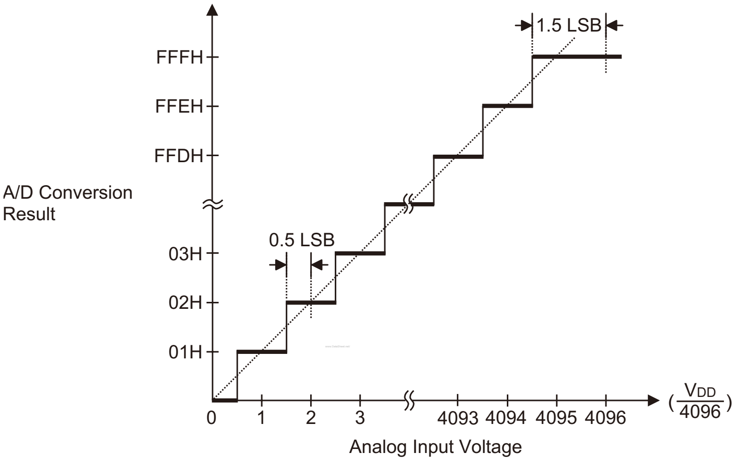
As the device contain a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the V_{DD} voltage, this gives a single bit analog input value of $V_{DD}/4096$. The diagram show the ideal transfer function between the analog input value and the digitised output value for the A/D converter.

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



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

A/D Conversion Timing



Ideal A/D Transfer Function

A/D Programming Example

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

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

```
    clr EADI              ; disable ADC interrupt
    mov a,00000001B
    mov ACSR,a            ; select fSYS/8 as A/D clock and ADONB=0
    mov a,0Fh             ; setup ANCSR0 and ANCSR1 to configure pins AN0~AN3
    mov ANCSR0,a
    mov a, 00h
    mov ANCSR1,a          ;
    mov a,00000000B       ;
    mov ADCR,a            ; select AN0 to be connected to the A/D converter
    :
Start_conversion:
    clr START
    set START             ; reset A/D
    clr START             ; start A/D
Polling_EOC:
    sz  EOCB              ; poll the ADCR register EOCB bit to detect end
                          ; of A/D conversion
    jmp polling_EOC       ; continue polling
    mov a,ADRL            ; read low byte conversion result value
    mov adrl_buffer,a     ; save result to user defined register
    mov a,ADRH            ; read high byte conversion result value
    mov adrh_buffer,a     ; save result to user defined register
    :
    jmp start_conversion  ; start next A/D conversion
```

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

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

```
     clr EADI            ; disable ADC interrupt
     mov a,00000001B
     mov ACSR,a          ; select fSYS/8 as A/D clock and ADONB=0
     mov a,0Fh           ; setup ANCSR0 and ANCSR1 to configure pins AN0~AN3
     mov ANCSR0,a
     mov a, 00h
     mov ANCSR1,a        ;
     mov    a,00000000B;
     mov ADCR, a         ; select AN0 to be connected to the A/D converter
     :
Start_conversion:
     clr START
     set START           ; reset A/D
     clr START           ; start A/D
     clr ADF             ; clear ADC interrupt request flag
     set EADI            ; enable ADC interrupt
     set EMI             ; enable global interrupt
     :
     :
; ADC interrupt service routine
ADC_:
     mov acc_stack,a     ; save ACC to user defined memory
     mov a,STATUS
     mov status_stack,a  ; save STATUS to user defined memory
     :
     :
     mov a,ADRL          ; read low byte conversion result value
     mov adrl_buffer,a   ; save result to user defined register
     mov a,ADRH          ; read high byte conversion result value
     mov adrh_buffer,a   ; save result to user defined register
     :
     :
EXIT_ISR:
     mov a,status_stack
     mov STATUS,a        ; restore STATUS from user defined memory
     mov a,acc_stack     ; restore ACC from user defined memory
     clr ADF             ; clear ADC interrupt flag
     reti
```

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

Interrupts

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

The device contains a single external interrupt and multiple internal interrupts. The external interrupt is controlled by the action of the external interrupt pin, while the internal interrupts are generated by the various functions such as Timer/Event Counters, and Time Base.

Interrupt Register

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

- **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	T1F	T0F	EIF	ET1I	ET0I	EEI	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 unimplemented, read as “0”

Bit 6 **T1F**: Timer/Event Counter 1 interrupt request flag
0: inactive
1: active

Bit 5 **T0F**: Timer/Event Counter 0 interrupt request flag
0: inactive
1: active

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

Bit 3 **ET1I**: Timer/Event Counter 1 interrupt enable
0: disable
1: enable

Bit 2 **ET0I**: Timer/Event Counter 0 interrupt enable
0: disable
1: enable

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

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

- **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	MFF	ADF	T2F	—	EMFI	EADI	ET2I
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
0: inactive
1: active

Bit 5 **ADF**: A/D converter interrupt request flag
0: inactive
1: active

Bit 4 **T2F**: Timer/Event Counter 2 interrupt request flag
0: inactive
1: active

Bit 3 unimplemented, read as "0"

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

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

Bit 0 **ET2I**: Timer/Event Counter 2 interrupt enable
0: disable
1: enable

- **MFIC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	SIF	SIMF	TBF	—	ESII	ESIM	ETBI
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 **SIF**: SPIA interrupt request flag
0: inactive
1: active

Bit 5 **SIMF**: SIM interrupt request flag
0: inactive
1: active

Bit 4 **TBF**: Time Base interrupt request flag
0: inactive
1: active

Bit 3 unimplemented, read as "0"

Bit 2 **ESII**: SPIA interrupt enable
0: disable
1: enable

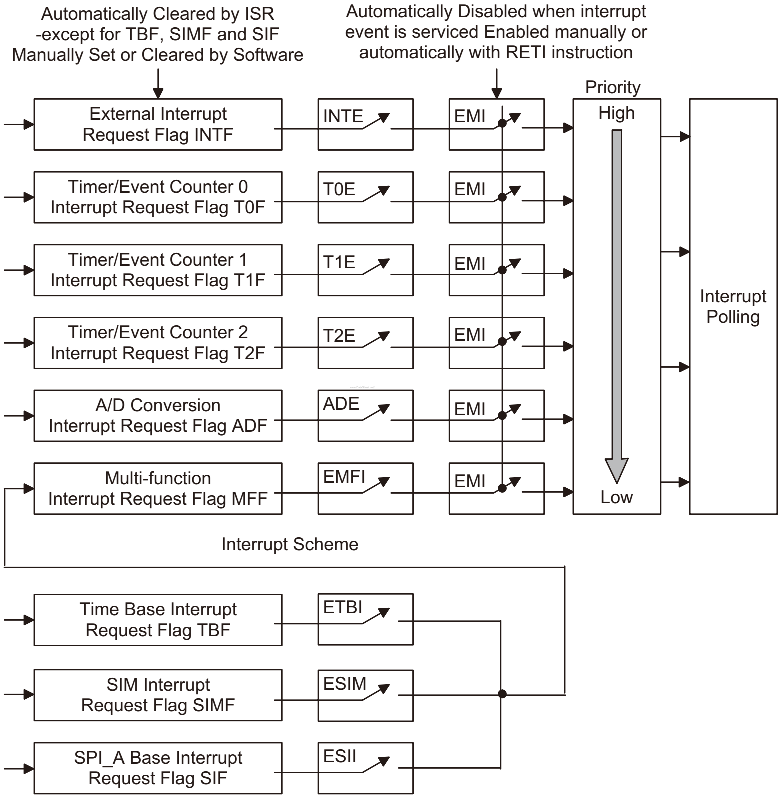
Bit 1 **ESIM**: SIM interrupt enable
0: disable
1: enable

Bit 0 **ETBI**: Time Base interrupt enable
0: disable
1: enable

Interrupt Operation

A Timer/Event Counter overflow, an active edge on the external interrupt pin, a serial data byte transmitted or received completion, or a Time Base event will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP statement which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI instruction, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

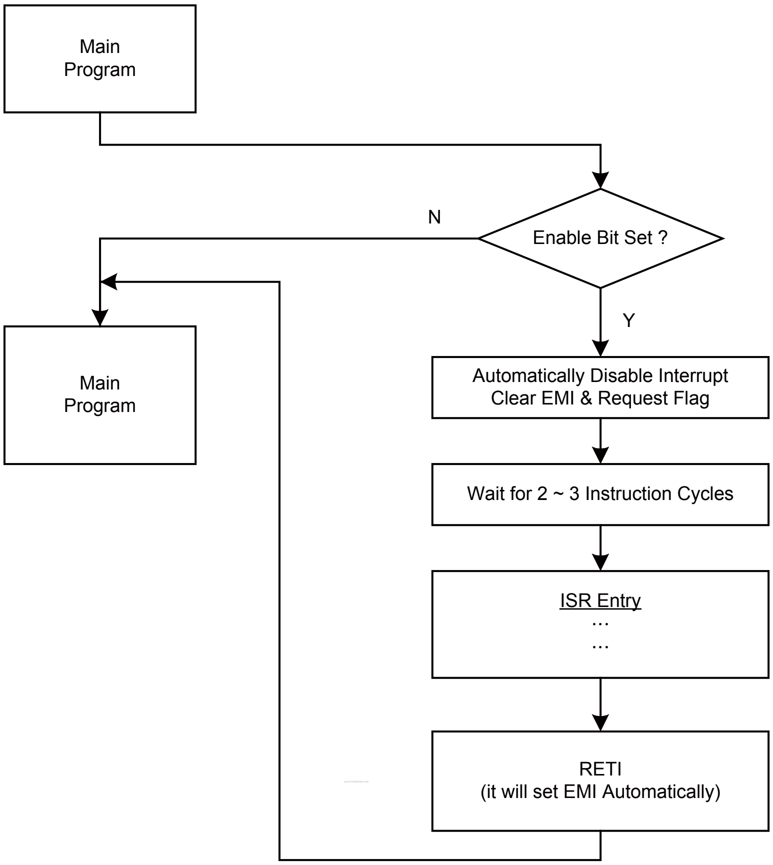
The various interrupt enable bits, together with their associated request flags, are shown in the following diagram with their order of priority.


Automatically Cleared by ISR
-except for TBF, SIMF and SIF
Manually Set or Cleared by Software
Automatically Disabled when interrupt
event is serviced Enabled manually or
automatically with RETI instruction
Priority
High
Low
External Interrupt Request Flag INTF
INTE
EMI
Timer/Event Counter 0 Interrupt Request Flag T0F
T0E
EMI
Timer/Event Counter 1 Interrupt Request Flag T1F
T1E
EMI
Timer/Event Counter 2 Interrupt Request Flag T2F
T2E
EMI
A/D Conversion Interrupt Request Flag ADF
ADE
EMI
Multi-function Interrupt Request Flag MFF
EMFI
EMI
Interrupt Polling
Time Base Interrupt Request Flag TBF
ETBI
SIM Interrupt Request Flag SIMF
ESIM
SPI_A Base Interrupt Request Flag SIF
ESII

Interrupt Scheme

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.


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

Interrupt Flow

Interrupt Priority

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

Interrupt Source	Priority	Vector
External Interrupt	1	04H
Timer/Event Counter 0 Overflow	2	08H
Timer/Event Counter 1 Overflow	3	0CH
Timer/Event Counter 2 Overflow	4	10H
A/D Interrupt	5	14H
Multi-function Interrupt (Time Base, SIM, SPIA)	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.

External Interrupt

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

INTEG1	INTEG0	Edge Trigger Type
0	0	External interrupt disable
0	1	Rising edge Trigger
1	0	Falling edge Trigger
1	1	Both edge Trigger

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

Timer/Event Counter Interrupt

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

Multi-function Interrupt

Unlike the other independent interrupts, the Multi-function Interrupt has no independent source, but rather is formed from other existing interrupt sources, namely the Time-base Interrupt, SIM Interrupt and SPIA Interrupt. A Multi-function Interrupt request will take place when the Multi-function Interrupt request flag, MFF is set. The Multi-function Interrupt flag will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function Interrupt is enabled and the stack is not full, and either one of the interrupts contained within each of Multi-function Interrupt occurs, a subroutine call to the Multi-function Interrupt vector will take place. When the interrupt is serviced, the Multi-Function Interrupt 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 flags from the original source of the Multi-function Interrupt, namely the Time-base Interrupt, SIM Interrupt and SPIA Interrupt will not be automatically reset and must be manually reset by the application program. After a Multi-function has been generated, the application program can determine which interrupt source has occurred by interrogating the interrupt request flags, SIF, SIMF or TBF within the MFIC register.

Programming Considerations

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

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

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

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

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

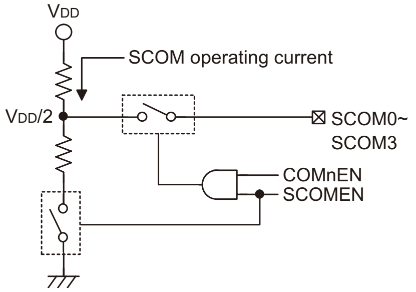
To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program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

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 the PB0~ PB3 pins. 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.



SCOM Circuit

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	—	ISEL1	ISEL0	SCOMEN	COM3EN	COM2EN	COM1EN	COM0EN
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 Reserved Bit
1: Unpredictable operation - bit must NOT be set high
0: Correct level - bit must be reset to zero for correct operation

Bit 6,5 **ISEL1, ISEL0**: SCOM operating current selection (V_{DD}=5V)
00: 25μA
01: 50μA
10: 100μA
11: 200μA

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: I/O
1: SCOM3

Bit 2 **COM2EN**: PB2 or SCOM2 selection
0: I/O
1: SCOM2

Bit 1 **COM1EN**: PB1 or SCOM1 selection
0: I/O
1: SCOM1

Bit 0 **COM0EN**: PB0 or SCOM0 selection
0: I/O
1: SCOM0

Serial Interface Module – SIM

These devices contain a Serial Interface Module, which includes both the four line SPI interface or the two line I^{2}C 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^{2}C 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^{2}C type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O are selected using pull-high control registers, and also if the SIM function is enabled.

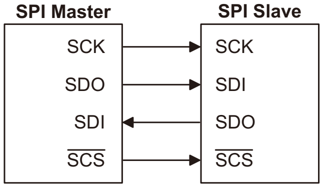
SPI Interface

This SPI interface function which is part of the Serial Interface Module, should not be confused with the other independent SPI function, known as SPIA, which is described in another section of this datasheet.

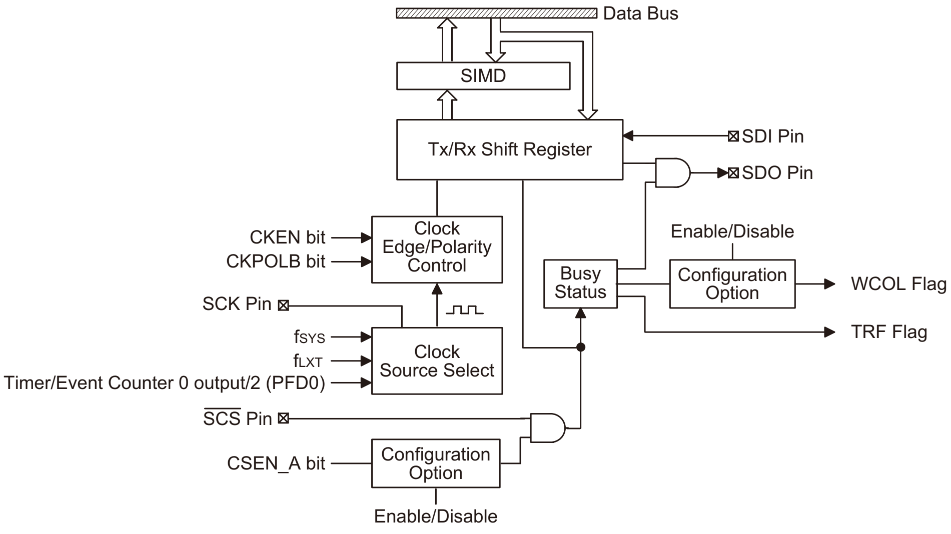
The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the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

The communication is full duplex and operates as a slave/master type, where the device can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, but this device provided only one $\overline{SCS}$ pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pin to select the slave devices.

- 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^{2}C function pins, the SPI interface must first be enabled by selecting the SIM enable configuration option and setting the correct bits in the SIMC0 and SIMC2 registers. After the SPI configuration option has been configured it can also be additionally disabled or enabled using the SIMEN bit in the SIMC0 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 utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to "1" to enable $\overline{SCS}$ pin function, set CSEN bit to "0" the $\overline{SCS}$ pin will be floating state.



SPI Master/Slave Connection



SPI Block Diagram

The SPI function in this device offers the following features:

- Full duplex synchronous data transferBoth 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

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 and SIMEN.

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 SIMC0 register will have no effect. Another two SPI configuration options determine if the CSEN and WCOL bits are to be used.

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two registers SIMC0 and SIMC2. Note that the SIMC1 register is only used by the I^{2}C interface.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF

SIM Registers List

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

- **SIMD Regisater**

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

"x" unknown

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I^{2}C function. The SIMC1 register is not used by the SPI function, only by the I^{2}C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Although not connected with the SPI function, the SIMC0 register is also used to control the Peripheral Clock Prescaler. Register SIMC2 is used for other control functions such as LSB/MSB selection, write collision flag etc.

- **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
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{LXT}
100: SPI master mode; SPI clock is Timer/Event Counter 0 output/2 (PFD0)
101: SPI slave mode
110: I^{2}C slave mode
111: Unused mode

These bits setup the overall operating mode of the SIM function. As well as selecting if the I^{2}C 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 0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **PCKEN**: Peripheral Clock Pin Control
Described elsewhere.

Bit 3~2 **PCKP1, PCKP0**: Select PCK output pin frequency
Described elsewhere.

Bit 1 **SIMEN**: SIM Control
0: Disable
1: Enable

The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will be in a floating condition and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I^{2}C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I^{2}C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I^{2}C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 unimplemented, read as "0"

- **SIMC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 Undefined bit
These bit can be read or written by user software program.

Bit 5 **CKPOLB**: Determines the base condition of the clock line
0: the SCK line will be high when the clock is inactive
1: the SCK line will be low when the clock is inactive

The CKPOLB 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 CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4 **CKEG**: Determines SPI SCK active clock edge type
CKPOLB=0
0: SCK is high base level and data capture at SCK rising edge
1: SCK is high base level and data capture at SCK falling edge

CKPOLB=1
0: SCK is low base level and data capture at SCK falling edge
1: SCK is low base level and data capture at SCK rising edge

The CKEG and CKPOLB 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 CKPOLB 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 CKPOLB 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 CKPOLB bit.

Bit 3 **MLS**: SPI Data shift order
0: LSB
1: MSB

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.

Bit 2 **CSEN**: SPI $\overline{\text{SCS}}$ pin Control
0: Disable
1: Enable

The CSEN bit is used as an enable/disable for the $\overline{\text{SCS}}$ pin. If this bit is low, then the $\overline{\text{SCS}}$ pin will be disabled and placed into a floating condition. If the bit is high the $\overline{\text{SCS}}$ pin will be enabled and used as a select pin. Note that using the CSEN bit can be disabled or enabled via configuration option.

Bit 1 **WCOL**: SPI Write Collision flag
0: No collision
1: Collision

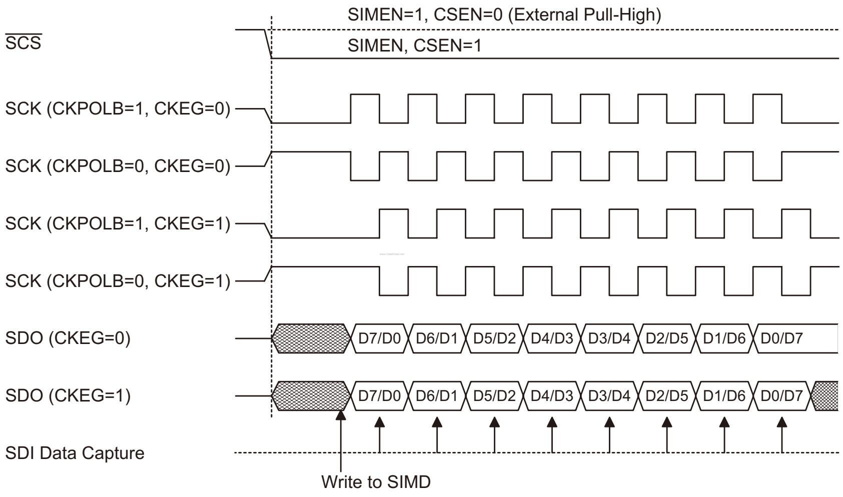
The WCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program. Note that using the WCOL bit can be disabled or enabled via configuration option.

Bit 0 **TRF**: SPI Transmit/Receive Complete flag
0: Data is being transferred
1: SPI data transmission is completed
The TRF bit is the Transmit/Receive Complete flag and is set "1" automatically when an SPI data transmission is completed, but must set to "0" by the application program. It can be used to generate an interrupt.

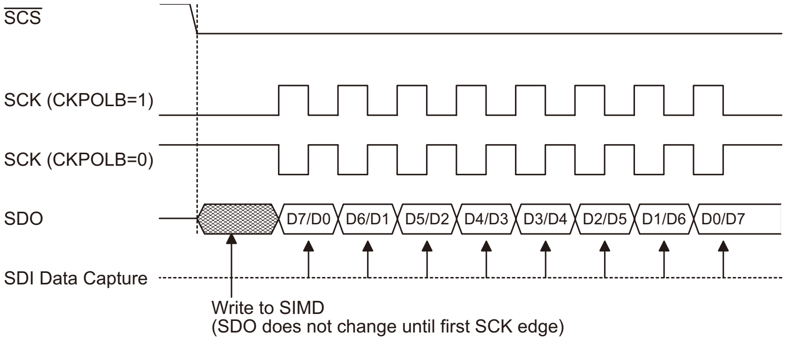
SPI Communication

After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMD 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 SIMD register will be transmitted and any data on the SDI pin will be shifted into the SIMD register. The master should output an $\overline{SCS}$ signal to enable the slave device before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the $\overline{SCS}$ signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagram shows the relationship between the slave data and $\overline{SCS}$ signal for various configurations of the CKPOLB and CKEG bits.

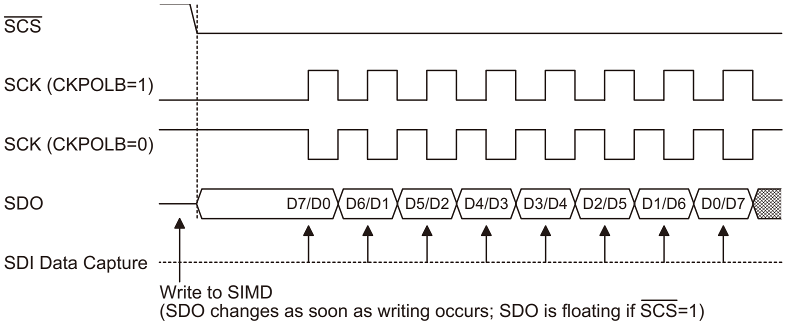
The SPI will continue to function even in the IDLE Mode.



SPI Master Mode Timing

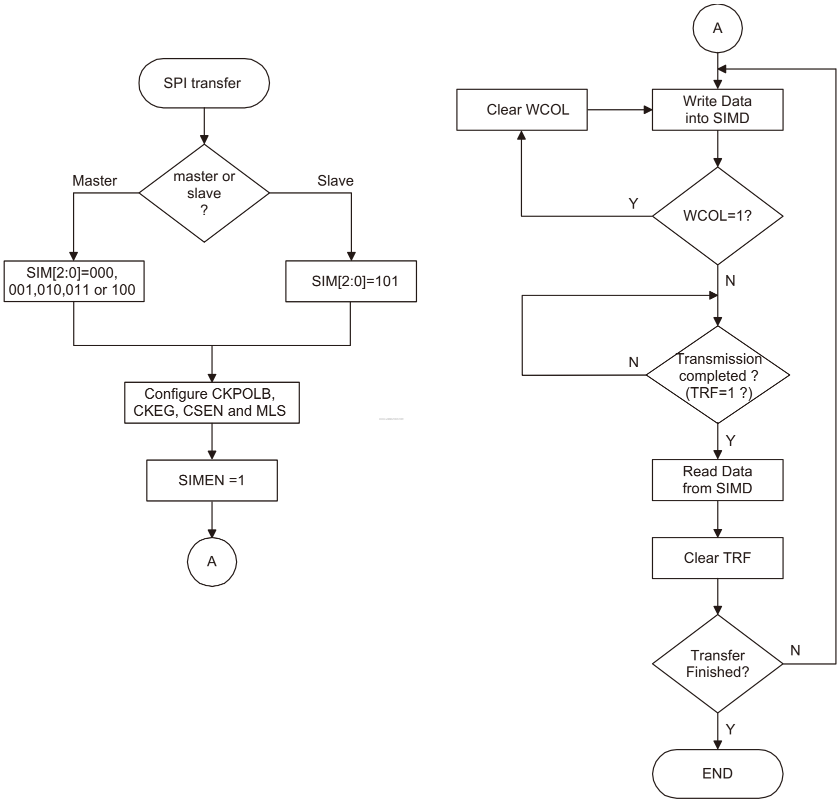

SCS
SCK (CKPOLB=1)
SCK (CKPOLB=0)
SDO
D7/D0 D6/D1 D5/D2 D4/D3 D3/D4 D2/D5 D1/D6 D0/D7
SDI Data Capture
Write to SIMD
(SDO does not change until first SCK edge)

SPI Slave Mode Timing - CKEG=0



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

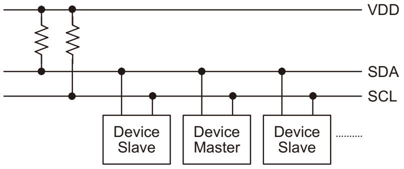
SPI Slave Mode Timing - CKEG=1



SPI Transfer Control Flowchart

I^{2}C Interface

The I^{2}C 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.

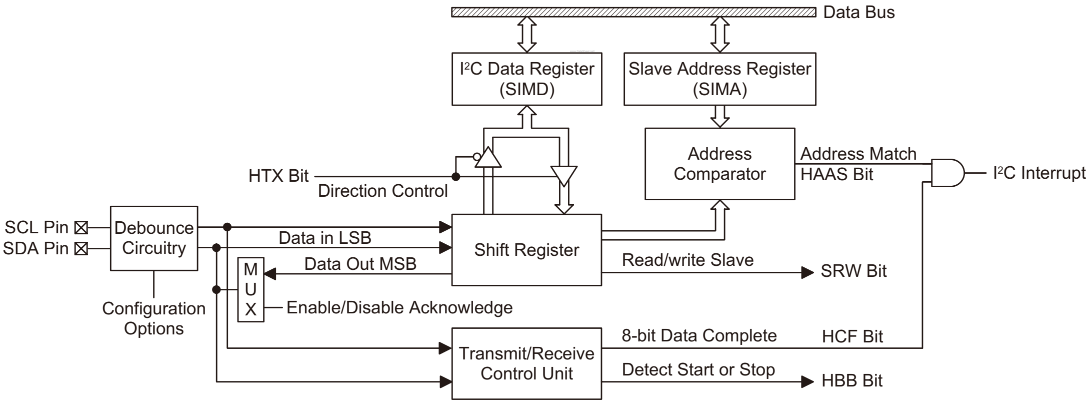


I^{2}C Mater Slave Bus Connection

- I^{2}C Interface Operation

  The I^{2}C 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^{2}C bus is identified by a unique address which will be transmitted and received on the I^{2}C bus.

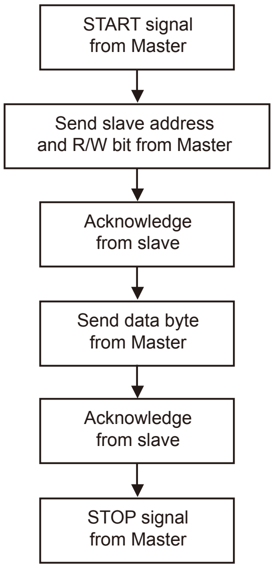
  When two devices communicate with each other on the bidirectional I^{2}C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For these devices, which only operates in slave mode, there are two methods of transferring data on the I^{2}C bus, the slave transmit mode and the slave receive mode.



There are several configuration options associated with the I^{2}C 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 SIMC0 register will have no effect. A configuration option exists to allow a clock other than the system clock to drive the I^{2}C interface. Another configuration option determines the debounce time of the I^{2}C 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 2 or 4 system clocks.

I^{2}C Debounce Time selection	I^{2}C Standard Mode (100kHz)	I^{2}C Fast Mode (400kHz)
No debounce	f_{SYS} > 2MHz	f_{SYS} > 5MHz
2 system clock debounce	f_{SYS} > 4MHz	f_{SYS} > 10MHz
4 system clock debounce	f_{SYS} > 8MHz	f_{SYS} > 20MHz

I^{2}C Minimum f_{SYS} Frequency



I^{2}C Registers

There are three control registers associated with the I^{2}C bus, SIMC0, SIMC1 and SIMA and one data register, SIMD. The SIMD register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I^{2}C bus. Before the microcontroller writes data to the I^{2}C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I^{2}C bus, the microcontroller can read it from the SIMD register. Any transmission or reception of data from the I^{2}C bus must be made via the SIMD register.

Note that the SIMA register also has the name SIMC2 which is used by the SPI function. Bit SIMEN and bits SIM2~SIM0 in register SIMC0 are used by the I^{2}C interface.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	D0

I^{2}C Registers List

- **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
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{LXT}
100: SPI master mode; SPI clock is Timer/Event Counter 0 output/2 (PFD0)
101: SPI slave mode
110: I^{2}C slave mode
111: Unused mode

These bits setup the overall operating mode of the SIM function. As well as selecting if the I^{2}C 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 Timer/Event Counter 0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **PCKEN**: Peripheral Clock Pin control
Described elsewhere.

Bit 3~2 **PCKP1, PCKP0**: Select PCK output pin frequency
Described elsewhere.

Bit 1 **SIMEN**: SIM Control
0: Disable
1: Enable

The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will be in a floating condition and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I^{2}C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I^{2}C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I^{2}C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 unimplemented, read as "0"

- **SIMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **HCF**: I^{2}C Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer
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.

Bit 6 **HAAS**: I^{2}C Bus address match flag
0: Not address match
1: Address match
The HAAS 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.

Bit 5 **HBB**: I^{2}C Bus busy flag
0: I^{2}C Bus is not busy
1: I^{2}C Bus is busy
The HBB flag is the I^{2}C busy flag. This flag will be "1" when the I^{2}C bus is busy which will occur when a START signal is detected. The flag will be set to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX**: Select I^{2}C slave device is transmitter or receiver
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK**: I^{2}C Bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag
The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8-bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set TXAK bit to "0" before further data is received.

Bit 2 **SRW**: I^{2}C Slave Read/Write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode
The SRW flag is the I^{2}C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I^{2}C bus. When the transmitted address and slave address is match, that is when the HAAS flag is set high, the slave device will check the SRW flag to determine whether it should be in transmit mode or receive mode. If the SRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

Bit 1 **IAMWU**: I^{2}C Address Match Wake-up Control
0: Disable
1: Enable – must be cleared byu the application program after wake-up

The I^{2}C module can run without using the internal clock, and generate an interrupt if the SIM interrupt is enabled, which can be used in the sleep mode, idle mode, normal mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I^{2}C address match wake up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

Note: if RNIC=1 and MCU is powered down, the slave-receiver can remain operational but the slave-transmitter will not operate as it needs the system clock.

Bit 0 **RXAK**: I^{2}C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave do not receive acknowledge flag

The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I^{2}C Bus.

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

- **SIMD Register**

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

"x" unknown

- **SIMA Register**

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	D0
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

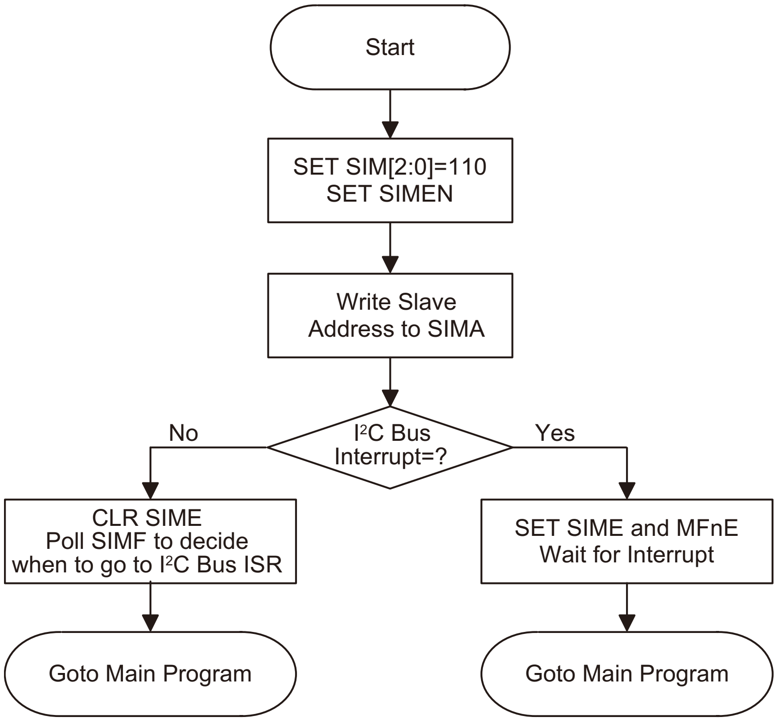
Bit 7~1 **IICA6~ IICA0**: I^{2}C slave address
IICA6~ IICA0 is the I^{2}C slave address bit 6~ bit 0.
The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~ 1 of the SIMA register define the device slave address. Bit 0 is not defined.
When a master device, which is connected to the I^{2}C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

Bit 0 Undefined bit
This bit can be read or written by user software program.

I^{2}C Bus Communication

Communication on the I^{2}C 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^{2}C 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 slave device matches that of the transmitted address, the HAAS bit in the SIMC1 register will be set and an I^{2}C interrupt will be generated. After entering the interrupt service routine, the 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 slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I^{2}C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
  Set the SIM2~SIM0 and SIMEN bits in the SIMC0 register to "1" to enable the I^{2}C bus.
- Step 2
  Write the slave address of the device to the I^{2}C bus address register SIMA.
- Step 3
  Set the SIME and SIM Muti-Function interrupt enable bit of the interrupt control register to enable the SIM interrupt and Multi-function interrupt.



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

The START signal can only be generated by the master device connected to the I²C bus and not by the 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 SIMC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The 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 SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

I^{2}C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the slave device wishes to read data from the I^{2}C bus or write data to the I^{2}C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is "1" then this indicates that the master device wishes to read data from the I^{2}C bus, therefore the slave device must be setup to send data to the I^{2}C bus as a transmitter. If the SRW flag is "0" then this indicates that the master wishes to send data to the I^{2}C bus, therefore the slave device must be setup to read data from the I^{2}C bus as a receiver.

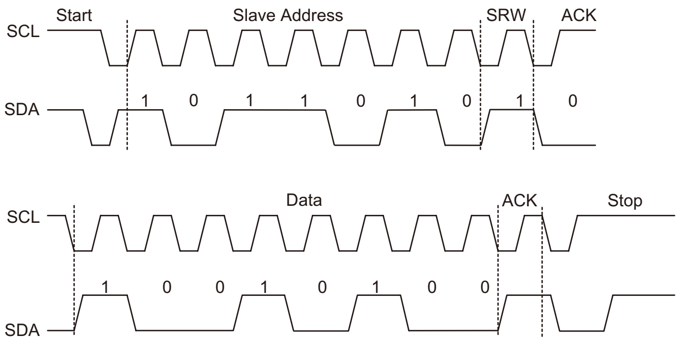
I^{2}C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I^{2}C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The 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 flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be set to "0".

I^{2}C Bus Data and Acknowledge Signal

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 slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I^{2}C Bus. The corresponding data will be stored in the SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the RXAK bit in the SIMC1 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.

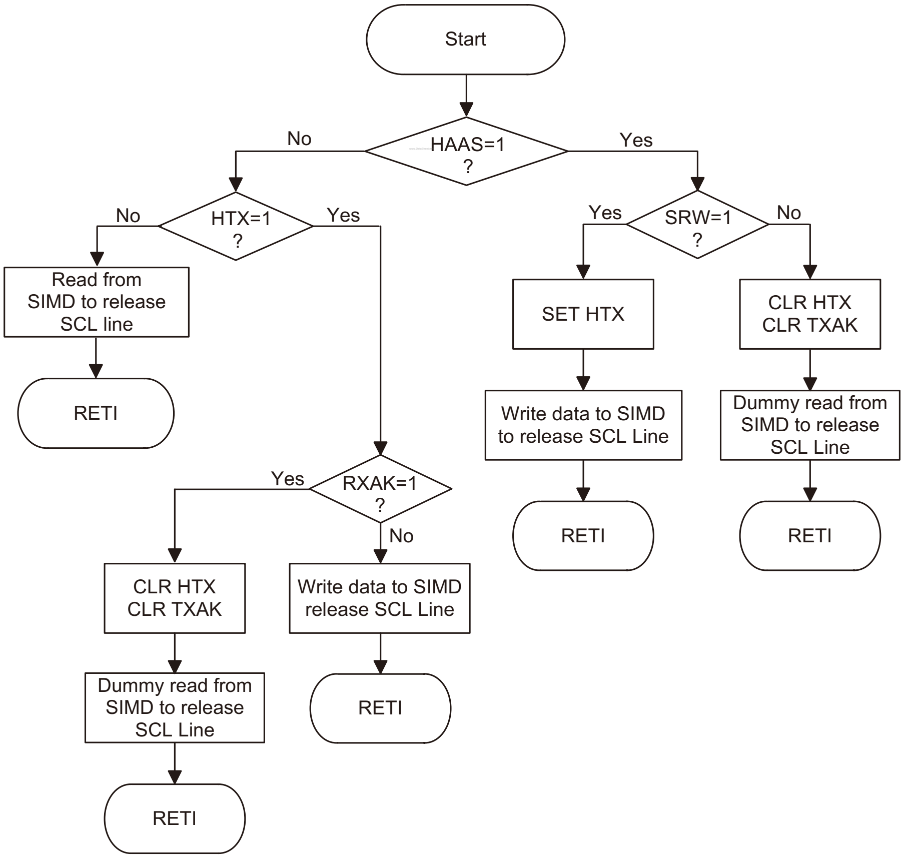


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

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

I^{2}C Communication Timing Diagram

Note: *When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implemented a dummy read from the SIMD register to release the SCL line.



I^{2}C Bus ISR Flow Chart

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 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 output/2 or a divided ratio of the internal fSYS 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.

- **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 elsewhere

Bit 4 **PCKEN**: Peripheral Clock 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: Timer/Event Counter 0 output /2 (PFD0)

Bit 1 **SIMEN**: SIM control
Described elsewhere

Bit 0 unimplemented, read as "0"

Serial Interface – SPIA

The devices contain an independent SPI function. It is important not to confuse this independent SPI function with the additional one contained withing in the combined SIM function, which is described in another section of this datasheet. This independent SPI function will carry the name SPIA to distinguish it from the other one in the SIM.

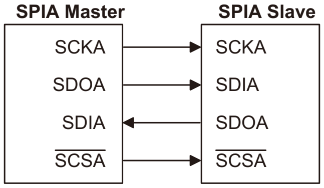
This SPIA interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices, etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying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

The communication is full duplex and operates as a slave/master type, where the device can be either master or slave. Although the SPIA interface specification can control multiple slave devices from a single master, this device is provided only one $\overline{\text{SCSA}}$ pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pins to select the slave devices.

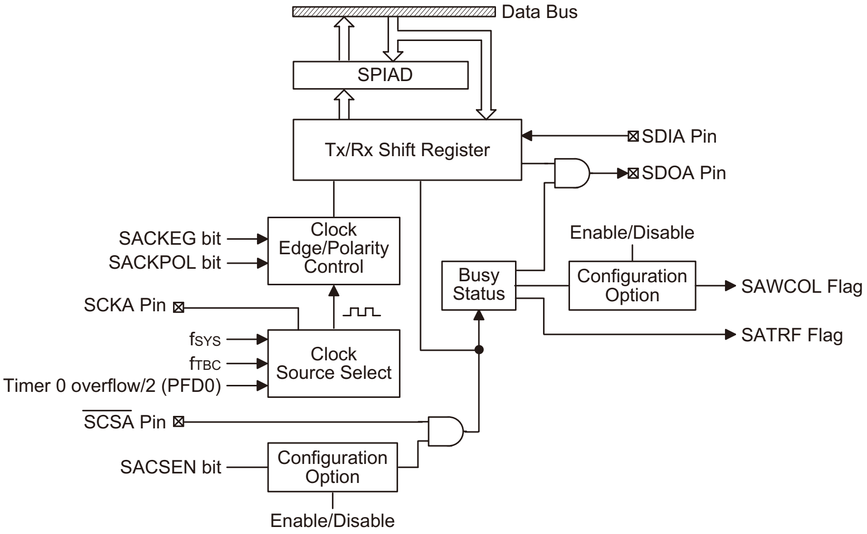
SPIA Interface Operation

The SPIA interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDIA, SDOA, SCKA and $\overline{\text{SCSA}}$. Pins SDIA and SDOA are the Serial Data Input and Serial Data Output lines, SCKA is the Serial Clock line and $\overline{\text{SCSA}}$ is the Slave Select line. As the SPIA interface pins are pin-shared with other functions, the SPIA interface must first be selected by the correct bits in the SPIAC0 and SPIAC1 registers. After the SPIA configuration option has been selected, it can also be additionally disabled or enabled using the SPIAEN bit in the SPIAC0 register. Communication between devices connected to the SPI1 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 $\overline{\text{SASA}}$ pin only one slave device can be utilised.

The $\overline{\text{SCSA}}$ pin is controlled by the application program, set the the SACSEN bit to "1" to enable the $\overline{\text{SCSA}}$ pin function and clear the SACSEN bit to "0" to place the $\overline{\text{SCSA}}$ pin into a floating state.



SPIA Master/Slave Connection



SPIA Block Diagram

The SPIA Serial Interface function includes the following features:

- Full-duplex synchronous data transfer
- Both Master and Slave mode
- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge
- SAWCOL and SACSEN bits enabled or disable select

The status of the SPIA 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 SACSEN and SPIAEN.

There are several configuration options associated with the SPIA interface. One of these is to enable the SPIA function which selects the SPIA pins rather than normal I/O pins. Note that if the configuration option does not select the SPIA function then the SPIAEN bit in the SPIAC0 register will have no effect. Two configuration options, which are used to control the CSEN and WCOL bit functions, are also used to determine if the SACSEN and SAWCOL bits are to be used.

SPIA registers

There are three registers which control the overall operation of the SPIA interface. These are the SPIAD data registers and two control registers SPIAC0 and SPIAC1.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SPIAC0	SASPI2	SASPI1	SASPI0	—	—	—	SPIAEN	—
SPIAC1	—	—	SACKPOL	SACKEG	SAMLS	SACSEN	SAWCOL	SATRF
SPIAD	D7	D6	D5	D4	D3	D2	D1	D0

SPIA Registers List

The SPIAD register is used to store the data being transmitted and received. Before the device writes data to the SPIA bus, the actual data to be transmitted must be placed in the SPIAD register. After the data is received from the SPIA bus, the device can read it from the SPIAD register. Any transmission or reception of data from the SPIA bus must be made via the SPIAD registers.

- **SPIAD 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	×	×	×	×	×	×	×	×

"×" unknown

There are also two control registers for the SPIA interface, SPIAC0 and SPIAC1. Register SPIAC0 is used to control the enable/disable function and to set the data transmission clock frequency. Register SPIAC1 is used for other control functions such as LSB/MSB selection, write collision flag, etc.

- **SPIAC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SASPI2	SASPI1	SASPI0	—	—	—	SPIAEN	—
R/W	R/W	R/W	R/W	—	—	—	R/W	—
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SASPI2~SASPI0:** SPIA Master/Slave Clock Select
000: SPIA master, $f_{SYS}/4$
001: SPIA master, $f_{SYS}/16$
010: SPIA master, $f_{SYS}/64$
011: SPIA master, f_{LXT}
100: SPIA master, Timer 0 overflow/2 (PFD0)
101: SPIA slave
110: Reserved
111: Reserved

Bit 4~2 Unimplemented, read as "0"

Bit 1 **SPIAEN:** SPIA enable or disable
0: Disable
1: Enable

The bit is the overall on/off control for the SPIA interface. When the SPIAEN bit is cleared to zero to disable the SPIA interface, the SDIA, SDOA, SCKA and $\overline{\text{SCSA}}$ lines will lose their SPI function and the SPIA operating current will be reduced to a minimum value. When the bit is high, the SPIA interface is enabled.

Bit 0 Unimplemented, read as "0"

- **SPIAC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SACKPOL	SACKEG	SAMLS	SACSEN	SAWCOL	SATRF
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 **SACKPOL:** Determines the base condition of the clock line
0: SCKA line will be high when the clock is inactive
1: SCKA line will be low when the clock is inactive

The SACKPOL bit determines the base condition of the clock line, if the bit is high, then the SCKA line will be low when the clock is inactive. When the SACKPOL bit is low, then the SCKA line will be high when the clock is inactive.

Bit 4 **SACKEG:** Determines the SPIA SCKA active clock edge type
SACKPOL=0:
0: SCKA has high base level with data capture on SCKA rising edge
1: SCKA has high base level with data capture on SCKA falling edge

SACKPOL=1:
0: SCKA has low base level with data capture on SCKA falling edge
1: SCKA has low base level with data capture on SCKA rising edge

The SACKEG and SACKPOL bits are used to setup the way that the clock signal outputs and inputs data on the SPIA bus. These two bits must be configured before a data transfer is executed otherwise an erroneous clock edge may be generated. The SACKPOL bit determines the base condition of the clock line, if the bit is high, then the SCKA line will be low when the clock is inactive. When the SACKPOL bit is low, then the SCKA line will be high when the clock is inactive. The SACKEG bit determines active clock edge type which depends upon the condition of the SACKPOL bit.

Bit 3 **SAMLS:** data shift order
0: the LSB of data is transmitted first
1: the MSB of data is transmitted first
This is the data shift select bit and is used to select how the data is transferred, either MSB orLSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2 **SACSEN:** SPIA select signal $\overline{\text{SCSA}}$ enable Control
0: Disable
1: Enable

The SACSEN bit is used as an enable/disable for the $\overline{\text{SCSA}}$ pin. If this bit is low, then the $\overline{\text{SCSA}}$ pin will be disabled and placed into other functions. If the bit is high the $\overline{\text{SCSA}}$ pin will be enabled and used as a select pin. Note that the SACSEN function can be enabled or disable via a configuration option.

Bit 1 **SAWCOL**: SPIA Write Collision flag
0: Collision free
1: Collision detected

The SAWCOL flag 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 SPIAD 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 the SAWCOL function can be enabled or disable via a configuration option.

Bit 0 **SATRF1:** SPIA Transmit/Receive Complete flag
0: Data is being transferred
1: SPIA data transmission is completed

The SATRF bit is the Transmit/Receive Complete flag and is set to“1” automatically when an SPIA data transmission is completed, but must cleared to “0” by the application program. It can be used to generate an interrupt.

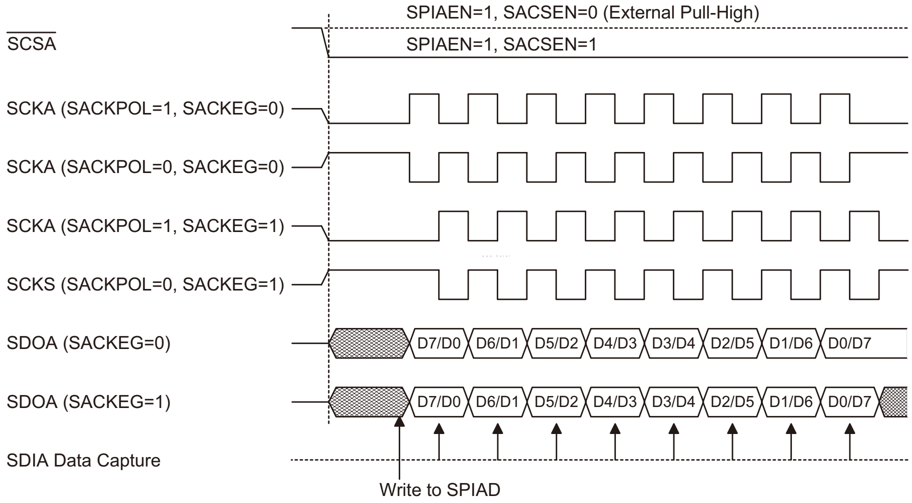
SPIA Communication

After the SPIA interface is enabled by setting the SPIAEN bit high, then in the Master Mode, when data is written to the SPIAD register, transmission/reception will begin simultaneously. When the data transfer is complete, the SATRF 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 SPIAD register will be transmitted and any data on the SDIA pin will be shifted into the SPIAD registers

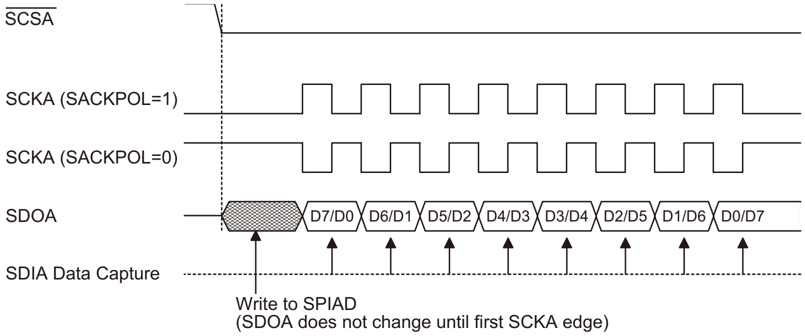
The master should output a $\overline{\text{SCSA}}$ signal to enable the slave device before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the $\overline{\text{SCSA}}$ signal depending upon the configurations of the SACKPOL bit and SACKEG bit. The accompanying timing diagram shows the relationship between the slave data and $\overline{\text{SCSA}}$ signal for various configurations of the SACKPOL and SACKEG bits.

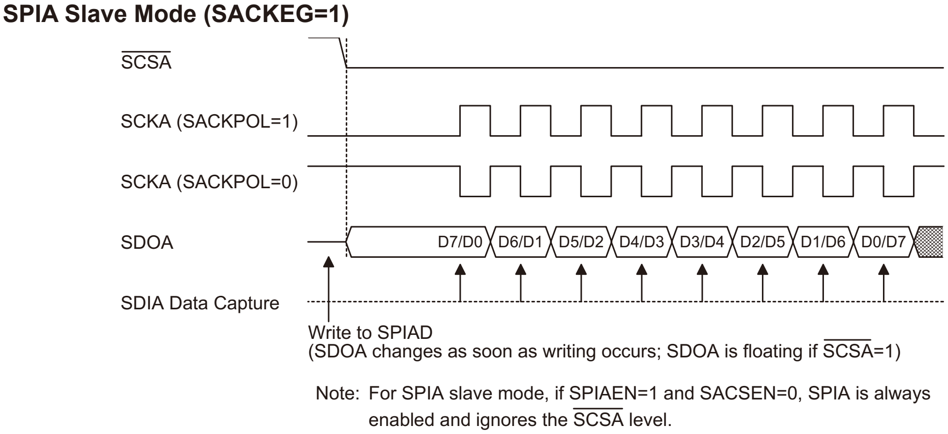
The SPIA will continue to function even in the IDLE Mode.

SPIA Master Mode

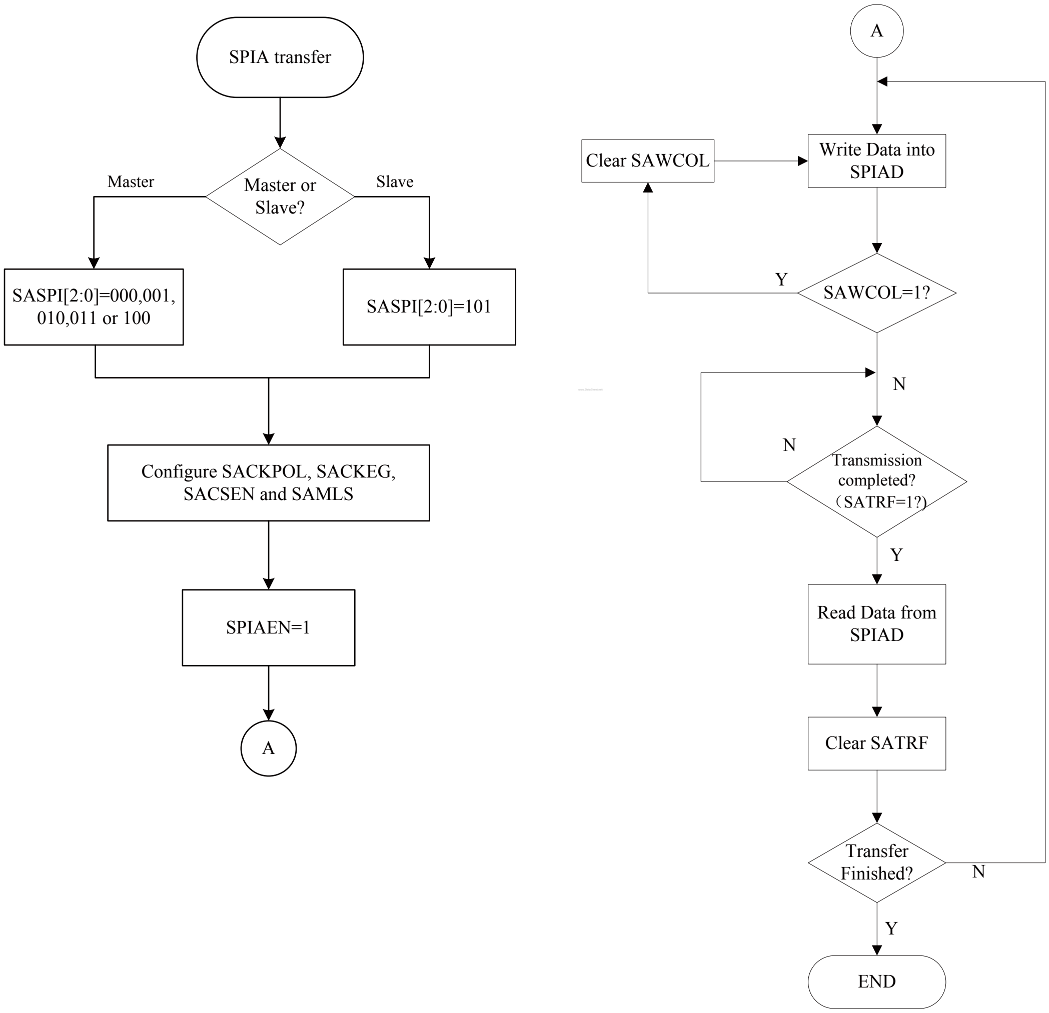


SPIA Slave Mode (SACKEG=0)




SPIA Slave Mode (SACKEG=1)
SCSA
SCKA (SACKPOL=1)
SCKA (SACKPOL=0)
SDOA
D7/D0
D6/D1
D5/D2
D4/D3
D3/D4
D2/D5
D1/D6
D0/D7
SDIA Data Capture
Write to SPIAD
(SDOA changes as soon as writing occurs; SDOA is floating if SCSA=1)
Note: For SPIA slave mode, if SPIAEN=1 and SACSEN=0, SPIA is always enabled and ignores the SCSA level.

SPIA Master/Slave ModeTiming Diagram



SPIA Transfer Control Flowchart

SPIA Bus Enable/Disable

To enable the SPIA bus, set SACSEN = 1 and SCSA=0, then wait for data to be written into the SPIAD (TXRX buffer) register. For the Master Mode, after data has been written to the SPIAD (TXRX buffer) register, then transmission or reception will start automatically. When all the data has been transferred the SATRF bit should be set. For the Slave Mode, when clock pulses are received on SCKA, data in the TXRX buffer will be shifted out or data on SDIA will be shifted in.

To Disable the SPIA bus SCKA, SDIA, SDOA, $\overline{\text{SCSA}}$ will become I/O pins or the other functions.

SPIA Operation

All communication is carried out using the 4-line interface for either Master or Slave Mode.

The SACSEN bit in the SPIAC1 register controls the overall function of the SPIA interface. Setting this bit high will enable the SPIA interface by allowing the $\overline{\text{SCSA}}$ line to be active, which can then be used to control the SPIA interface. If the SACSEN bit is low, the SPIA interface will be disabled and the $\overline{\text{SCSA}}$ line will be an I/O pin or other functions and can therefore not be used for control of the SPIA interface. If the SACSEN bit and the SPIAEN bit in the SPIAC0 register are set high, this will place the SDIA line in a floating condition and the SDOA line high. If in Master Mode the SCKA line will be either high or low depending upon the clock polarity selection bit SACKPOLB in the SPIAC1 register. If in Slave Mode the SCKA line will be in a floating condition. If SPIAEN is low then the bus will be disabled and $\overline{\text{SCSA}}$, SDIA, SDOA and SCKA will all become I/O pins or other functions. 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 into the SPIAD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode:

Master Mode

- Step 1
  Select the clock source and Master mode using the SASPI2~SASPI0 bits in the SPIAC0 control register
- Step 2
  Setup the SACSEN bit and setup the SAMLS bit to choose if the data is MSB or LSB first, this must be same as the Slave device.
- Step 3
  Setup the SPIAEN bit in the SPIAC0 control register to enable the SPIA interface.
- Step 4
  For write operations: write the data to the SPIAD register, which will actually place the data into the TXRX buffer. Then use the SCKA and $\overline{\text{SCSA}}$ lines to output the data. After this go to step 5.
  For read operations: the data transferred in on the SDIA line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPIAD register.
- Step 5
  Check the SAWCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the SATRF bit or wait for a SPIA serial bus interrupt.

- Step 7
  Read data from the SPIAD register.
- Step 8
  Clear SATRF.
- Step 9
  Go to step 4.

Slave Mode

- Step 1
  Select the SPI Slave mode using the SASPI2~SASPI0 bits in the SPIAC0 control register
- Step 2
  Setup the SACSEN bit and setup the SAMLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master device.
- Step 3
  Setup the SPIAEN bit in the SPIAC0 control register to enable the SPIA interface.
- Step 4
  For write operations: write the data to the SPIAD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCKA and $\overline{\text{SCSA}}$ signal. After this, go to step 5.
  For read operations: the data transferred in on the SDIA line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPIAD register.
- Step 5
  Check the SAWCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the SATRF bit or wait for a SPIA serial bus interrupt.
- Step 7
  Read data from the SPIAD register.
- Step 8
  Clear SATRF.
- Step 9
  Go to step 4.

Error Detection

The SAWCOL bit in the SPIAC register is provided to indicate errors during data transfer. The bit is set by the SPIA serial Interface but must be cleared by the application program. This bit indicates a data collision has occurred which happens if a write to the SPIAD register takes place during a data transfer operation and will prevent the write operation from continuing. The SAWCOL and SACSEN functions can be disabled or enabled by configuration options.

Low Voltage Detector – LVD

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

LVD Register

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

A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

- **LVDC Register**

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

Bit 7~6 unimplemented, read as "0"

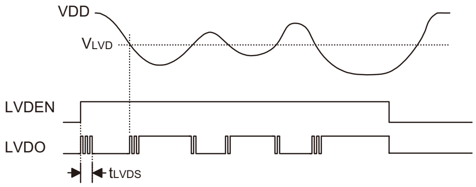
Bit 5 **LVDO**: LVD Output Flag
0: No Low Voltage Detect
1: Low Voltage Detect

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

Bit 3~0: unimplemented, read as "0"

LVD Operation

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



LVD Operation

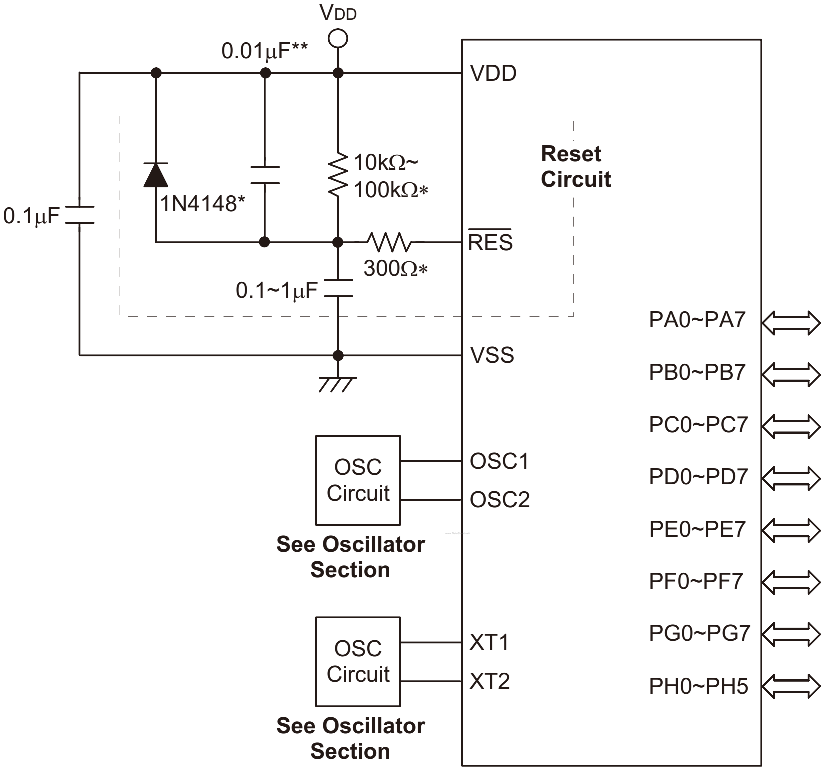
Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the 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 can not 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 Speed System Oscillator Selection - f_H: 1. HXT 2. ERC 3. HIRC
2	Low Speed System Oscillator Selection - f_L: 1. LXT 2. LIRC
3	WDT Clock Selection - f_S: 1. LXT 2. LIRC 3. $f_{SYS}/4$
4	HIRC Frequency Selection: 1. 4MHz 2. 8MHz 3. 12MHz
Reset Pin Options	
5	PA7/$\overline{\text{RES}}$ Pin Options: 1. $\overline{\text{RES}}$ pin 2. I/O pin
Watchdog Options	
6	Watchdog Timer Function: 1. Enable 2. Disable
7	CLRWDT Instructions Selection: 1. 1 instructions 2. 2 instructions
LVR Options	
8	LVR Function: 1. Enable 2. Disable
9	LVR/LVD Voltage Selection: 1. 2.1V/2.2V 2. 3.15V/3.3V 3. 4.2V/4.4V
SIM/SPIA Options	
10	SIM Function: 1. Enable 2. Disable
11	SPI/SPIA - WCOL/SAWCOL bits: 1. Enable 2. Disable
12	SPI/SPIA - CSEN/SACSEN bits: 1. Enable 2. Disable
13	I²C Debounce Time Selection: 1. No debounce 2. 2 system clock debounce 3. 4 system clock debounce

No.	Options
14	SPIA Function: 1. Enable 2. Disable
DAC Options	
15	DAC Function: 1. DAC 2. I/O

Application Circuit



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

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

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

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

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

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

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

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC ″AND″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″AND″ [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

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

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

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

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

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

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

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

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

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

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

Operation: ACC ← ACC − [m] − C

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

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

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

Operation: [m] ← ACC − [m] − C

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s): None

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

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

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

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation: [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation: ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation: [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s): None

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

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

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

Affected flag(s): None

SZ [m] Skip if Data Memory is 0

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

Operation: Skip if [m] = 0

Affected flag(s): None

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

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

Operation: ACC ← [m]
Skip if [m] = 0

Affected flag(s): None

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

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

Operation: Skip if [m].i = 0

Affected flag(s): None

TABRD [m] Read table to TBLH and Data Memory

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

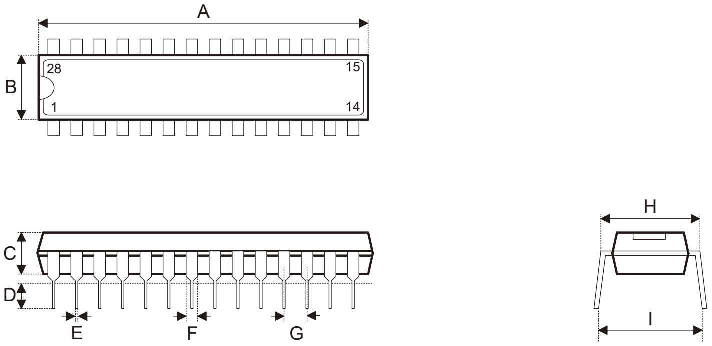
Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Package Information

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

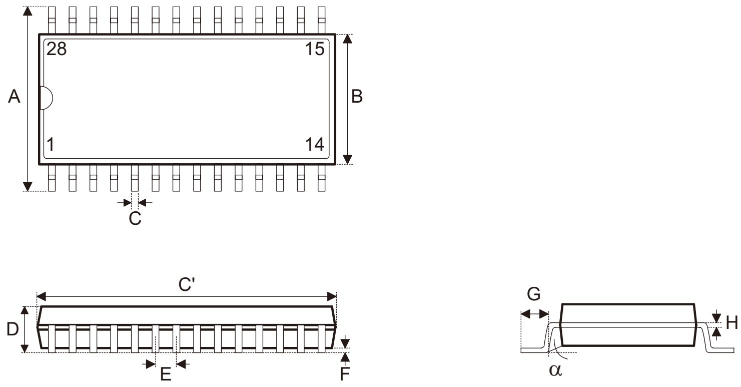
28-pin SKDIP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	1.375	—	1.395
B	0.278	—	0.298
C	0.125	—	0.135
D	0.125	—	0.145
E	0.016	—	0.020
F	0.050	—	0.070
G	—	0.100	—
H	0.295	—	0.315
I	—	0.375	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	34.93	—	35.43
B	7.06	—	7.57
C	3.18	—	3.43
D	3.18	—	3.68
E	0.41	—	0.51
F	1.27	—	1.78
G	—	2.54	—
H	7.49	—	8.00
I	—	9.53	—

28-pin SOP (300mil) Outline Dimensions

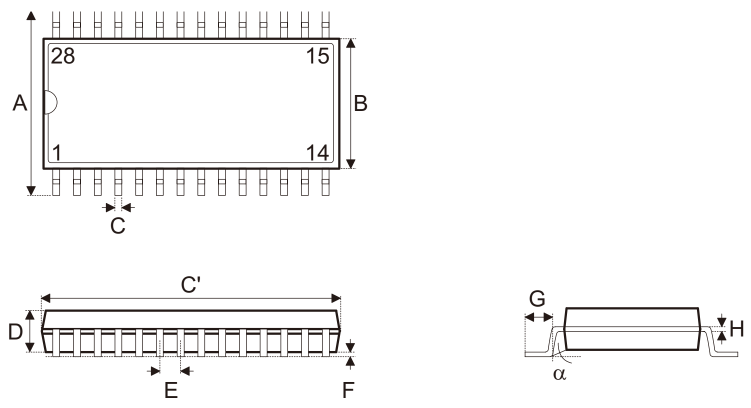


- **MS-013**

Symbol	Dimensions in inch		
	Min.	**Nom.**	**Max.**
A	0.393	—	0.419
B	0.256	—	0.300
C	0.012	—	0.020
C’	0.697	—	0.713
D	—	—	0.104
E	—	0.050	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	**Nom.**	**Max.**
A	9.98	—	10.64
B	6.50	—	7.62
C	0.30	—	0.51
C’	17.70	—	18.11
D	—	—	2.64
E	—	1.27	—
F	0.10	—	0.30
G	0.41	—	1.27
H	0.20	—	0.33
α	0°	—	8°

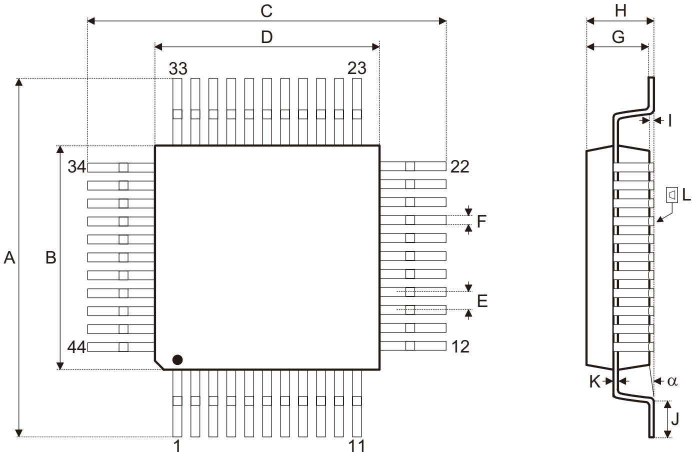
28-pin SSOP (150mil) Outline Dimensions



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°

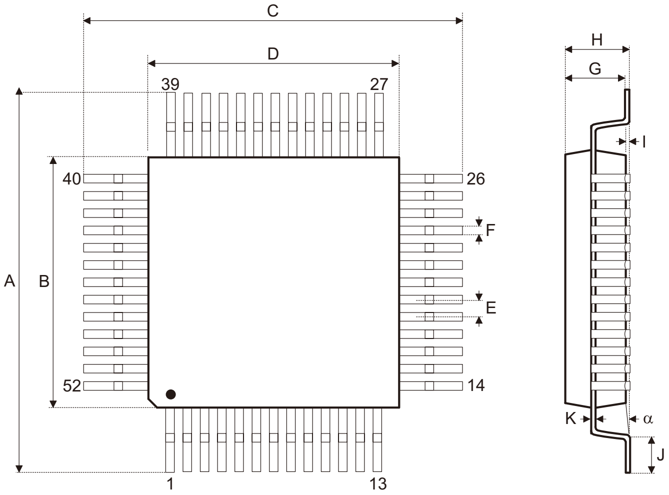
44-pin QFP (10mmx10mm) Outline Dimensions



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

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

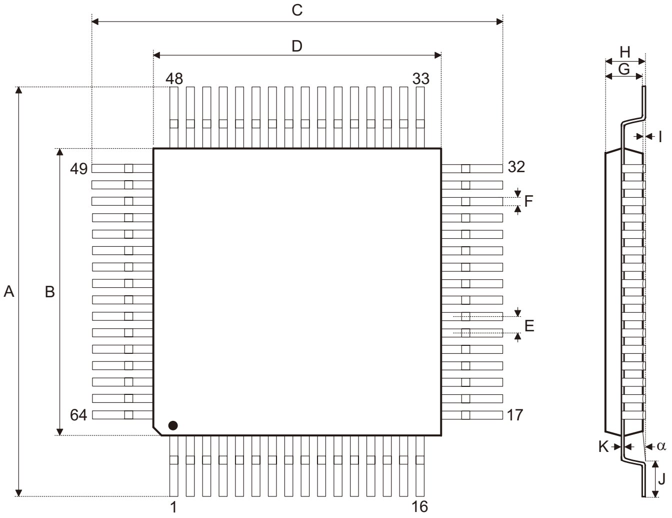
52-pin QFP (14mmx14mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.681	—	0.689
B	0.547	—	0.555
C	0.681	—	0.689
D	0.547	—	0.555
E	—	0.039	—
F	—	0.016	—
G	0.098	—	0.122
H	—	—	0.134
I	—	0.004	—
J	0.029	—	0.041
K	0.004	—	0.008
α	0°	—	7°

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°

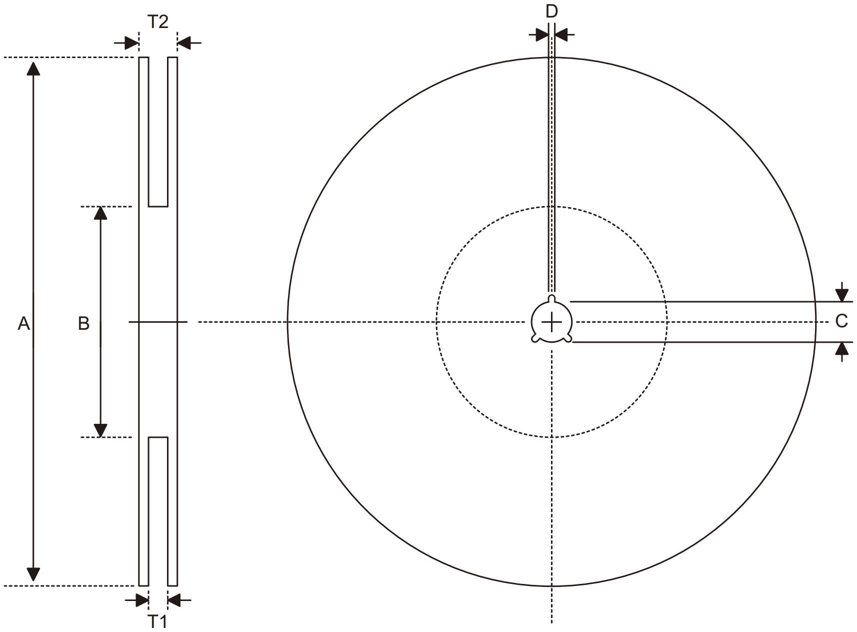
64-pin LQFP (7mmx7mm) Outline Dimensions



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

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	8.90	—	9.10
B	6.90	—	7.10
C	8.90	—	9.10
D	6.90	—	7.10
E	—	0.40	—
F	0.13	—	0.23
G	1.35	—	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	—	0.75
K	0.09	—	0.20
α	0°	—	7°

Reel Dimensions



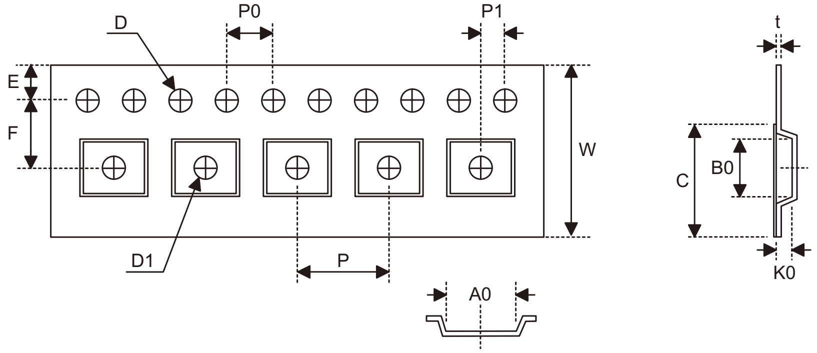
- **SOP 28W (300mil)**

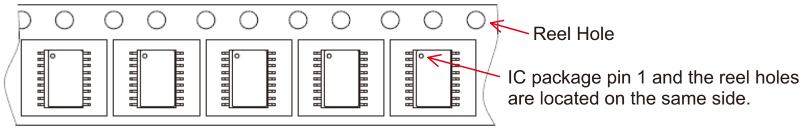
Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 $^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	24.8 $^{+0.3/-0.2}$
T2	Reel Thickness	30.2±0.2

- **SSOP 28S (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

Carrier Tape Dimensions




Reel Hole
IC package pin 1 and the reel holes are located on the same side.

- **SOP 28W (300mil)**

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±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.85±0.10
B0	Cavity Width	18.34±0.10
K0	Cavity Depth	2.97±0.10
t	Carrier Tape Thickness	0.35±0.01
C	Cover Tape Width	21.3±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

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

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