Voice Peripheral OTP MCU

HT68RV032/HT68RV033/HT68RV034
HT68RV034L/HT68RV035/HT68RV036

Revision: V2.20 Date: February 12, 2026

www.holtek.com

Features

CPU Features

- Operating Voltage
  - HT68RV032/33/34/35/36
    - f_{SYS}=12MHz: 2.3V~5.5V
  - HT68RV034L
    - f_{SYS}=12MHz: 2.0V~5.5V
- Up to 0.33μs instruction cycle with 12MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 12MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- OTP Program Memory: (2K-16)×16
- RAM Data Memory: 128×8
- Voice Flash Memory: HT68RV032: 2Mbit; HT68RV033: 4Mbit; HT68RV034/HT68RV034L: 8Mbit; HT68RV035: 16Mbit; HT68RV036: 32Mbit
- Watchdog Timer function
- 9 bidirectional I/O lines
- One external interrupt line shared with I/O pin
- One 8-bit programmable Timer/Event Counter with overflow interrupt and prescaler
- Dual Time Base functions for generation of fixed time interrupt signals
- Delta Sigma PWM driver
- Integrated 3.0V LDO
- Serial Interface Module – SIM for SPI or I^2C interface
- Low voltage reset function – LVR
- Support Peripheral IC mode
- Package type: 8-pin SOP

Development Tools

For rapid product development and to simplify device parameter setting, Holtek has provided relevant development tools which users can download from the following link:

https://www.holtek.com/page/tool-detail/dev_plat/voice/voice_workshop

https://www.holtek.com/page/tool-detail/dev_plat/voice/wireless_doorbell_workshop

General Description

The devices are OTP type 8-bit high performance RISC architecture microcontrollers, which are designed for various voice application terminal products such as smart home appliances, consumer electronic products, etc.

For memory features, the memory also includes an area of RAM Data Memory as well as an area of up to 32Mbit Voice Flash Memory.

A single extremely flexible Timer/Event Counter provides timing, event counting and pulse wide measurement functions. In addition, an internal LDO function provides power to the Voice Flash Memory. Communication with the outside world is catered for by including fully integrated SPI and I^2C interface functions, these popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of internal high and low speed oscillators are provided, the two fully integrated system oscillators require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

The devices integrate a Delta Sigma PWM driver, for which the devices have a digital programmable volume control with a wide range. The devices also support the Peripheral IC mode, implementing voice editing together with Holtek Voice Workshop. The inclusion of flexible I/O programming features, Time Base functions, along with many other features ensure that the devices will find excellent use for voice application terminal products.

Selection Table

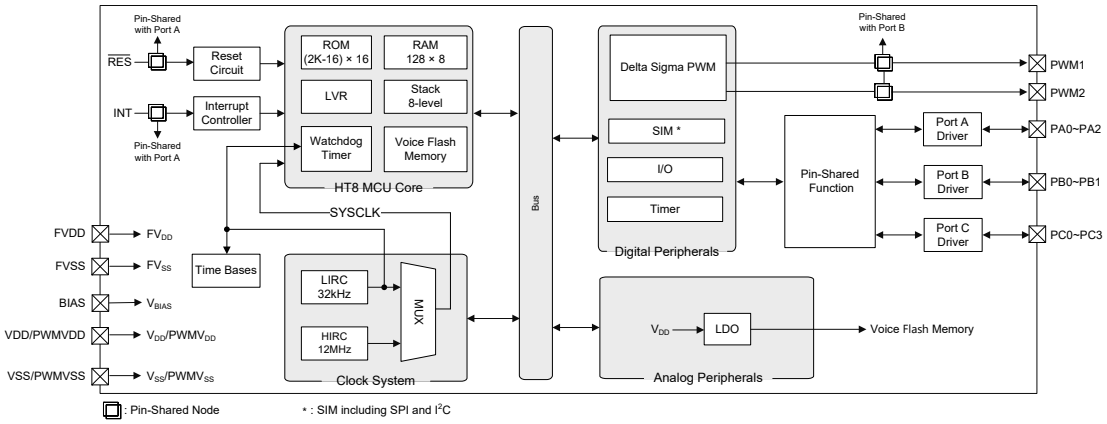
Most features are common to all devices and the main features distinguishing them are operating voltage range, Voice Flash Memory capacity and maximum voice time. The following table summarises the main features of each device.

Part No.	V_{DD}	Program Memory	Data Memory	Voice Flash Memory	Maximum Voice Time
HT68RV032	2.3V~5.5V	(2K-16)×16	128×8	2Mbit	85s
HT68RV033				4Mbit	170s
HT68RV034				8Mbit	340s
HT68RV035				16Mbit	680s
HT68RV036				32Mbit	1360s
HT68RV034L	2.0V~5.5V			8Mbit	340s

Part No.	I/O	External Interrupt	Time Base	Timer	Stack	Package
HT68RV032	5	1	2	8-bit×1	8	8SOP
HT68RV033						
HT68RV034						
HT68RV035						
HT68RV036						
HT68RV034L						

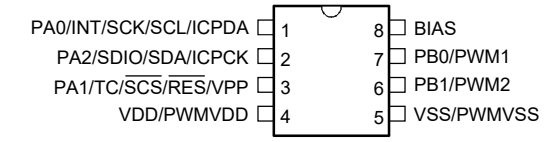
Note: As devices exist in more than one package format, the table reflects the situation for the package with the most pins.

Block Diagram

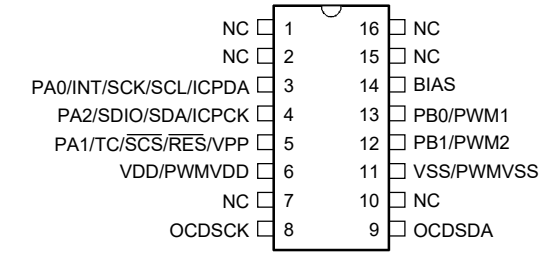


Note: The figure illustrates the Block Diagram of the device with maximum features, the functional differences between the series of devices are provided in the Selection Table.

Pin Assignment



HT68RV032/HT68RV033/HT68RV034/HT68RV034L/HT68RV035/HT68RV036
8 SOP-A



HT68EVV032/HT68EVV033/HT68EVV034/HT68EVV034L/HT68EVV035/HT68EVV036
16 NSOP-A

Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.

2. The OCDSDA and OCDSCK pins are supplied as OCDS dedicated pins and as such only available for the HT68EVV03x (Flash type) device which is the OCDS EV chip for the HT68RV03x device (OTP type).

3. The VPP pin is the High Voltage input for OTP programming and only available for the HT68RV03x device.

4. For the less pin count package type there will be unbounded pins which should be properly configured to avoid unwanted power consumption resulting from floating input conditions. Refer to the "Standby Current Considerations" and "Input/Output Ports" sections.

Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet. As the Pin Description table shows the situation for the package with the most pins, not all pins in the tables will be available on smaller package sizes.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT/SCK/SCL/ICPDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT	INTEG INTC0 PAS0	ST	—	External interrupt input
	SCK	PAS0 IFS	ST	CMOS	SIM SPI serial clock
	SCL	PAS0 IFS	ST	NMOS	SIM I²C clock line
	ICPDA	—	ST	CMOS	ICP data/address

Pin Name	Function	OPT	I/T	O/T	Description
PA1/TC/$\overline{SCS}$/$\overline{RES}$/VPP	PA1	PAPU PAWU PAS0 RSTC	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up. This I/O is pin-shared with VPP and could result in increased current consumption if it is set to output high.
	TC	PAS0 RSTC	ST	—	Timer/Event Counter clock input
	$\overline{SCS}$	PAS0 IFS RSTC	ST	CMOS	SIM SPI slave select
	$\overline{RES}$	RSTC	ST	–	External reset input
	VPP	PAS0 RSTC	PWR	—	High Voltage input for OTP programming, not available for EV chip
PA2/SDIO/SDA/ICPCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	SDIO	PAS0 IFS	ST	CMOS	SIM SPI serial data input (4-wire mode) or SIM SPI serial data input/output (3-wire mode)
	SDA	PAS0 IFS	ST	NMOS	SIM I^2C data line
	ICPCK	—	ST	—	ICP clock input
PB0/PWM1	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PWM1	PBS0	—	CMOS	PWM driver output 1
PB1/PWM2	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PWM2	PBS0	—	CMOS	PWM driver output 2
FVDD	FVDD	—	PWR	—	Voice ROM positive power supply, need to be connected to the BIAS pin
FVSS	FVSS	—	PWR	—	Voice ROM negative power supply, need to be connected to the VSS pin
BIAS	BIAS	—	—	PWR	Positive bias power supply output. A 0.1μF capacitor must be connected as close as possible between this pin and the VSS pin. It cannot be used in any other circuits except for connecting to FVDD
VDD/PWMVDD	VDD	—	PWR	—	Positive power supply
	PWMVDD	—	PWR	—	PWM positive power supply
VSS/PWMVSS	VSS	—	PWR	—	Negative power supply, ground
	PWMVSS	—	PWR	—	PWM negative power supply, ground
OCDSDA	OCDSDA	—	ST	CMOS	OCDS data/address, for EV chip only
OCDSCK	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
NC	NC	—	—	—	No connected

Legend: I/T: Input type; O/T: Output type;
OPT: Optional by register option; PWR: Power;
ST: Schmitt Trigger input; NMOS: NMOS output;
CMOS: CMOS output.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}-0.3V to 6.0V
Input Voltage ..... V_{SS}-0.3V to V_{DD} +0.3V
Storage Temperature..... -60°C to 150°C
Operating Temperature..... -40°C to 85°C
I_{OL} Total (exclude PWM1/PWM2)..... 150mA
I_{OH} Total (exclude PWM1/PWM2)..... -200mA
Total Power Dissipation (exclude PWM1/PWM2)..... 180mW
PWM1/PWM2 I_{OL} Total 500mA
PWM1/PWM2 I_{OH} Total..... -500mA
PWM1/PWM2 Total Power Dissipation (@R_L=8Ω) 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the devices. Functional operation of the devices 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

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency, pin load conditions, temperature and program instruction type, etc., can all exert an influence on the measured values.

Operating Voltage Characteristics

HT68RV032/33/34/35/36

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	f_{SYS}=12MHz	2.3	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32kHz	2.3	—	5.5	V

HT68RV034L

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	f_{SYS}=12MHz	2.0	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32kHz	2.0	—	5.5	V

Operating Current Characteristics

HT68RV032/33/34/35/36

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2.3V	f_{SYS}=32kHz	—	30	50	μA
		3V		—	40	60	
		5V		—	60	80	
	FAST Mode – HIRC	2.3V	f_{SYS}=12MHz	—	1.0	1.4	mA
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	

HT68RV034L

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2.0V	f_{SYS}=32kHz	—	30	50	μA
		3V		—	40	60	
		5V		—	60	80	
	FAST Mode – HIRC	2.0V	f_{SYS}=12MHz	—	1.0	1.4	mA
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	

Note: When using the characteristic table data, the following notes should be taken into consideration:

1. Any digital inputs are setup in a non-floating condition and the I/O pin-shared with VPP is not setup in an output high condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Operating Current values are measured using a continuous NOP instruction program loop.

Standby Current Characteristics

HT68RV032/33/34/35/36

Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max	Max.@ 85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.3V	WDT off	—	0.45	0.80	7.00	μA
		3V		—	0.45	0.90	8.00	μA
		5V		—	0.5	2.0	10.0	μA
		2.3V	WDT on	—	6	12	18	μA
		3V		—	8	16	24	μA
		5V		—	10	20	30	μA
	IDLE0 Mode – LIRC	2.3V	f_{SUB} on	—	6	12	18	μA
		3V		—	8	16	24	μA
		5V		—	10	20	30	μA
	IDLE1 Mode – HIRC	2.3V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	540	750	900	μA
		5V		—	800	1200	1440	μA

HT68RV034L

Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max	Max.@ 85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.0V	WDT off	—	0.45	0.80	7.00	μA
		3V		—	0.45	0.90	8.00	μA
		5V		—	0.5	2.0	10.0	μA
		2.0V	WDT on	—	6	12	18	μA
		3V		—	8	16	24	μA
		5V		—	10	20	30	μA
	IDLE0 Mode – LIRC	2.0V	f_{SUB} on	—	6	12	18	μA
		3V		—	8	16	24	μA
		5V		—	10	20	30	μA
	IDLE1 Mode – HIRC	2.0V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	540	750	900	μA
		5V		—	800	1200	1440	μA

Note: When using the characteristic table data, the following notes should be taken into consideration:

1. Any digital inputs are setup in a non-floating condition and the I/O pin-shared with VPP is not setup in an output high condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

A.C. Characteristics

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency and temperature etc., can all exert an influence on the measured values.

High Speed Internal Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at a user selected HIRC frequency and user selected voltage of either 3V or 5V.

HT68RV032/33/34/35/36

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	12MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	12	+1%	MHz
			-40°C~85°C	-2%	12	+2%	
		2.3V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C~85°C	-3%	12	+3%	

HT68RV034L

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	12MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	12	+1%	MHz
			-40°C~85°C	-2%	12	+2%	
		2.0V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C~85°C	-3%	12	+3%	

Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.3V to 3.6V (HT68RV032/33/34/35/36) / 2.0V to 3.6V (HT68RV034L) and fixed at 5V for application voltage ranges from 3.3V to 5.5V.
3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

Low Speed Internal Oscillator Characteristics – LIRC

HT68RV032/33/34/35/36

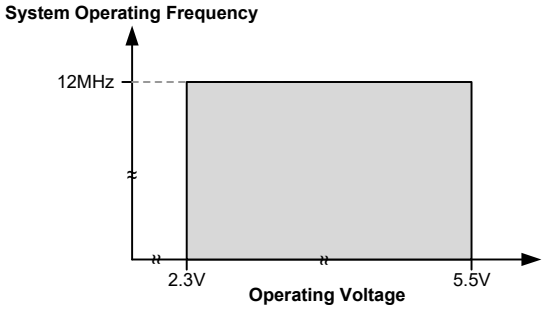
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	2.3V~5.5V	25°C	—	32	—	kHz
			-40°C~85°C	—	32	—	
t_{START}	LIRC Start-up Time	—	-40°C~85°C	—	—	500	μs

HT68RV034L

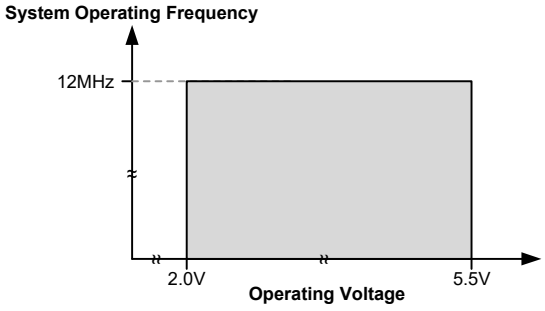
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	2.0V~5.5V	25°C	—	32	—	kHz
			-40°C~85°C	—	32	—	
t_{START}	LIRC Start-up Time	—	-40°C~85°C	—	—	500	μs

Operating Frequency Characteristic Curves

HT68RV032/33/34/35/36



HT68RV034L



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time (Wake-up from Condition where f_{SYS} is Off)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time (Wake-up from Condition where f_{SYS} is On)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	System Speed Switch Time (FAST to SLOW Mode or SLOW to FAST Mode)	—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
t_{RSTD}	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR_{POR}=5V/ms	5	16	18	ms
	System Reset Delay Time (LVRC/RSTC Software Reset)	—	—				
	System Reset Delay Time (Reset Source from WDT Overflow or $\overline{RES}$ Pin Reset)	—	—	5	16	18	
t_{RES}	External Reset Minimum Low Pulse Width	—	—	10	—	—	μs
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

Note: 1. For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.

2. The time units, shown by the symbols t_{HIRC} etc., are the inverse of the corresponding frequency values as provided in the frequency tables. For example, $t_{HIRC}=1/f_{HIRC}$, $t_{SYS}=1/f_{SYS}$ etc.

3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.

4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports Except $\overline{RES}$ and PB0~PB1 Pins	5V	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	
	Input Low Voltage for $\overline{RES}$ Pin	—	$V_{DD}\geq 2.7V$	0	—	$0.4V_{DD}$	V
		—	2.3V≤V_{DD}<2.7V (HT68RV032/33/34/35/36) 2.0V≤V_{DD}<2.7V (HT68RV034L)	0	—	$0.3V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports Except $\overline{RES}$ and PB0~PB1 Pins	5V	—	3.5	—	5.0	V
		—		$0.8V_{DD}$	—	V_{DD}	
	Input High Voltage for $\overline{RES}$ Pin	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	Sink Current for I/O Ports Except PB0~PB1 Pins	3V	$V_{OL}=0.1V_{DD}$	5	10	—	mA
		5V		10	20	—	
I_{OH}	Source Current for I/O Ports Except PB0~PB1 Pins	3V	$V_{OH}=0.9V_{DD}$	-2.5	-5.0	—	
		5V		-5	-10	—	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
R_{PH}	Pull-high Resistance for I/O Ports Except PB0~PB1 Pins (Note)	3V	—	20	60	100	kΩ
		5V		10	30	50	
I_{LEAK}	Input Leakage Current for I/O Ports Except PB0~PB1 Pins	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{TC}	TC Input Pin Minimum Pulse Width	—	—	25	—	—	ns
t_{INT}	Interrupt Pin Minimum Pulse Width	—	—	0.3	—	—	μs

Note: The R_{PH} internal pull-high resistance value is calculated by connecting to ground and enabling the input pin with a pull-high resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R_{PH} value.

Memory Characteristics

Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
OTP Program Memory							
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 1.9V (only available for HT68RV034L)	-3%	1.9	+3%	V
		—	LVR enable, voltage select 2.2V	-3%	2.2	+3%	
		—	LVR enable, voltage select 2.55V	-3%	2.55	+3%	
		—	LVR enable, voltage select 3.15V	-3%	3.15	+3%	
t_{LVR}	Minimum Low Voltage Width to Reset	—	TLVR[1:0]=00B	120	240	480	μs
			TLVR[1:0]=01B	0.5	1.0	2.0	ms
			TLVR[1:0]=10B	1	2	4	ms
			TLVR[1:0]=11B	2	4	8	ms
I_{LVR}	Additional Current for LVR Enable	5V	—	—	—	14	μA

PWM/PB0/PB1 Driver Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for PB0~PB1 Pins	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for PB0~PB1 Pins	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
R_{PH}	Pull-high Resistance for PB0~PB1 Pins (Note)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
SR_{RISE}	Output Rising Edge Slew Rate for PB0~PB1 Pins	3V	PWDC=0000B~1111B C_{LOAD}=20pF	—	210	—	V/μs
		5V		—	350	—	
SR_{FALL}	Output Falling Edge Slew Rate for PB0~PB1 Pins	3V	PWDC=0000B~1111B C_{LOAD}=20pF	—	210	—	V/μs
		5V		—	350	—	
I_{OL}	PWM1/PWM2 Sink Current	3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=0000B	4.2	6.0	—	mA
		5V		10.5	15.0	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=0001B	9.1	13.0	—	mA
		5V		20.3	29.0	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=0010B	13.3	19.0	—	mA
		5V		29.4	42.0	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=0011B	17.5	25.0	—	mA
		5V		38.5	55.0	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=0100B	20.3	29.0	—	mA
		5V		44.1	63.0	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=0101B	26.6	38.0	—	mA
		5V		56.7	81.0	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=0110B	30.1	43.0	—	mA
		5V		64.4	92.0	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=0111B	33.1	48.0	—	mA
		5V		71.4	102.0	—	mA
		3V	VOL=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=1000B	40.6	58.0	—	mA
		5V		84	120	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=1001B	46.9	67.0	—	mA
		5V		95.9	137.0	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=1010B	53.2	76.0	—	mA
		5V		107.8	154.0	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=1011B	58.1	83.0	—	mA
		5V		117.6	168.0	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=1100B	72	90	—	mA
		5V		144	180	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=1101B	87.2	109.0	—	mA
		5V		170.4	213.0	—	mA
	PWM1/PWM2 Sink Current	3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=1110B	116.8	146.0	—	mA
		5V		217.6	272.0	—	mA
		3V	V_{OL}=0.1V_{DD}, DRV_EN=1, PWDC[3:0]=1111B	144	180	—	mA
		5V		256.8	321.0	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH}	PWM1/PWM2 Source Current	3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=0000B	-4.2	-6.0	—	mA
		5V		-11.2	-16.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=0001B	-9.1	-13.0	—	mA
		5V		-21.7	-31.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=0010B	-13.3	-19.0	—	mA
		5V		-31.5	-45.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=0011B	-17.5	-25.0	—	mA
		5V		-40.6	-58.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=0100B	-19.6	-28.0	—	mA
		5V		-46.2	-66.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=0101B	-25.9	-37.0	—	mA
		5V		-59.5	-85.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=0110B	-29.4	-42.0	—	mA
		5V		-67.2	-96.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=0111B	-32.9	-47.0	—	mA
		5V		-73.5	-105.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=1000B	-39.2	-56.0	—	mA
		5V		-86.1	-123.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=1001B	-44.8	-64.0	—	mA
		5V		-97.3	-139.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=1010B	-50.4	-72.0	—	mA
		5V		-109.2	-156.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=1011B	-55.3	-79.0	—	mA
		5V		-117.6	-168.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=1100B	-68	-85	—	mA
		5V		-143.2	-179.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=1101B	-81.6	-102.0	—	mA
		5V		-167.2	-209.0	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=1110B	-106.4	-133.0	—	mA
		5V		-208	-260	—	mA
		3V	V_{OH}=0.9V_{DD}, DRV_EN=1, PWDC[3:0]=1111B	-128	-160	—	mA
		5V		-240	-300	—	mA

LDO Electrical Characteristics

V_{IN}=V_{OUT}+0.3V, C_{LOAD}=0.1μF, Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IN}	Supply Voltage	—	—	3.3	—	5.5	V
V_{OUT}	Output Voltage	—	Ta=25°C, I_{LOAD}=1mA, V_{OUT}=3.0V	-3%	3.0	3%	V
		—	Ta=-40°C~85°C, I_{LOAD}=1mA, V_{OUT}=3.0V	-8%	3.0	8%	V
I_Q	Quiescent Current	5V	No load	—	—	200	μA
I_{OUT}	Output Current	—	ΔV_{OUT}=0.145V	70	—	—	mA
TC	Temperature Coefficient	—	Ta=-40°C~85°C, I_{LOAD}=10mA	—	±1.5	±2.0	mV/°C

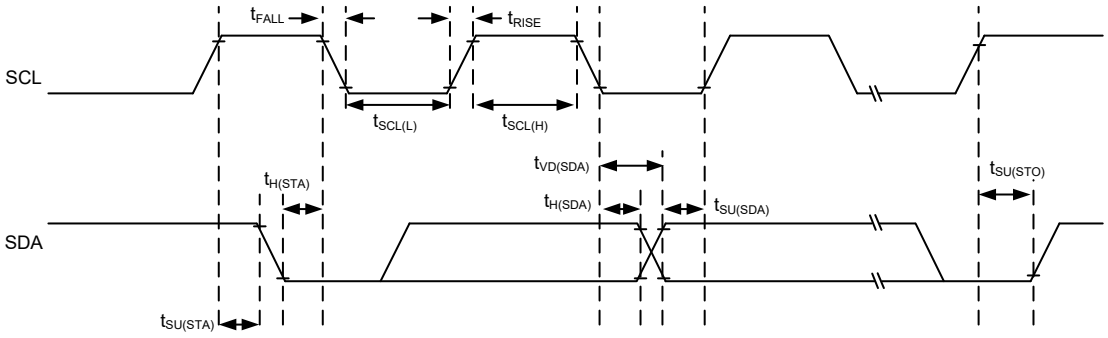
Note: Load regulation is measured at a constant junction temperature, using pulse testing with a low ON time and is guaranteed up to the maximum power dissipation. Power dissipation is determined by the input/output differential voltage and the output current. Guaranteed maximum power dissipation will not be available over the full input/output range. The maximum allowable power dissipation at any ambient temperature is PD = ($T_{J(MAX)}$-Ta)/θ_{JA}.

I²C Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{I2C}	I²C Standard Mode (100kHz) f_{SYS} Frequency (Note)	—	No clock debounce	2	—	—	MHz
		—	2 system clock debounce	4	—	—	MHz
		—	4 system clock debounce	4	—	—	MHz
	I²C Fast Mode (400kHz) f_{SYS} Frequency (Note)	—	No clock debounce	4	—	—	MHz
		—	2 system clock debounce	8	—	—	MHz
		—	4 system clock debounce	8	—	—	MHz
f_{SCL}	SCL Clock Frequency	3V/5V	Standard mode	—	—	100	kHz
			Fast mode	—	—	400	
$t_{SCL(H)}$	SCL Clock High Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
$t_{SCL(L)}$	SCL Clock Low Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
t_{FALL}	SCL and SDA Fall Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
t_{RISE}	SCL and SDA Rise Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
$t_{SU(SDA)}$	SDA Data Setup Time	3V/5V	Standard mode	0.25	—	—	μs
			Fast mode	0.1	—	—	
$t_{H(SDA)}$	SDA Data Hold Time	3V/5V	—	0.1	—	—	μs
$t_{VD(SDA)}$	SDA Data Valid Time	3V/5V	—	—	—	0.6	μs
$t_{SU(STA)}$	Start Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	
$t_{H(STA)}$	Start Condition Hold Time	3V/5V	Standard mode	4.0	—	—	μs
			Fast mode	0.6	—	—	μs
$t_{SU(STO)}$	Stop Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	

Note: Using the debounce function can make the transmission more stable and reduce the probability of communication failure due to interference.

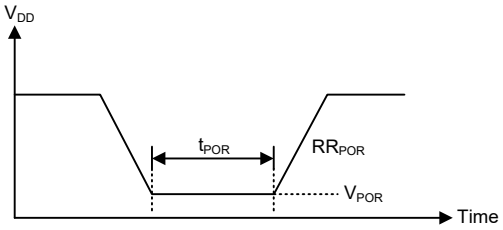


I²C Timing Diagram

Power-on Reset Characteristics

Ta=-40°C~85°C

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



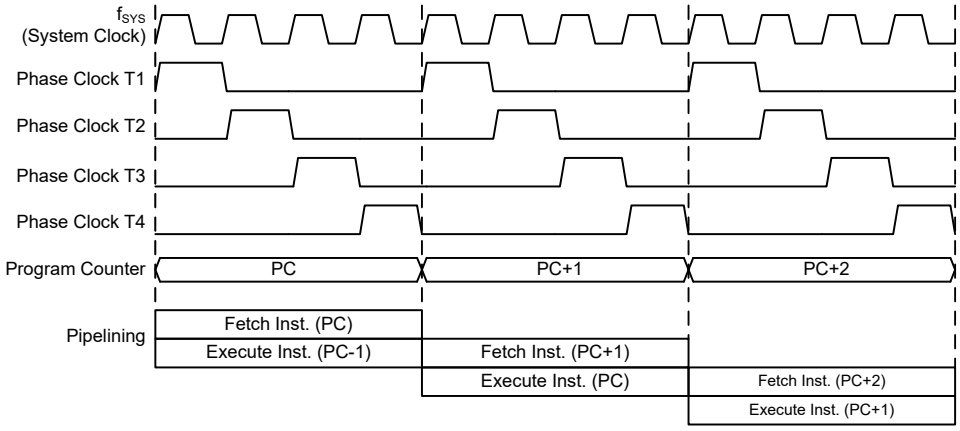
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The range of device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions which need one more cycle. An 8-bit wide ALU is used in practically all instruction set operations, which carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O control system with maximum reliability and flexibility. This makes the device suitable for affordable, high-volume production for controller applications.

Clocking and Pipelining

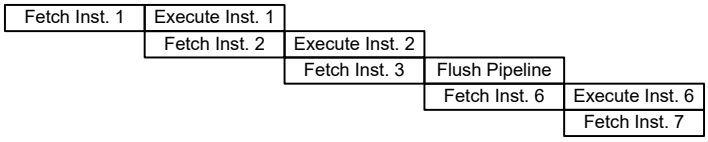
The main system clock, derived from either an HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the 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 machine 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 demands a jump to a non-consecutive Program Memory address. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

Program Counter	
High Byte	Low Byte (PCL)
PC10~PC8	PCL7~PCL0

Program Counter

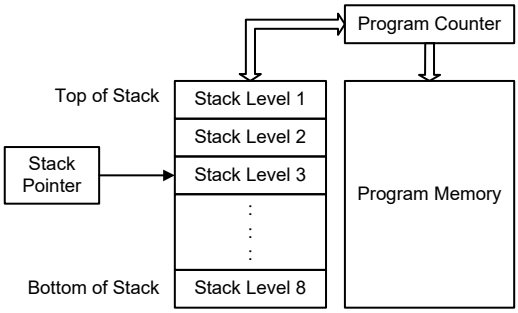
The lower byte of the Program Counter, known as the Program Counter Low Byte register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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. Manipulating the PCL register may cause program branching, so an extra cycle is needed to pre-fetch.

Stack

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

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

If the stack is overflow, the first Program Counter save in the stack will be lost.



Arithmetic and Logic Unit – ALU

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

- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA

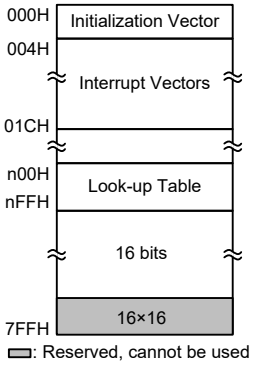
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement:
  INCA, INC, DECA, DEC
- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

OTP Program Memory

The Program Memory is the location where the user code or program is stored. The devices are supplied with One-Time Programmable, OTP memory where users can program their application code into the devices.

Structure

The Program Memory has a capacity of (2K-16)×16 bits. Note that the subtractive 16×16 bits space is reserved and cannot be used.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 a separate table pointer register.



Program Memory Structure

Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the address of the look up data to be retrieved in the table pointer registers, TBLP and TBHP. These registers define the total address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the Program Memory using the “TABRD [m]” or “TABRDL [m]” instruction. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

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

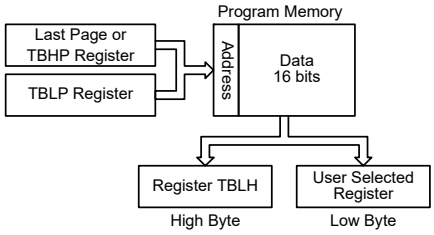


Table Program Example

The following example shows how the table pointer and table data are defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "0700H" which refers to the start address of the last page within the 2K Program Memory of the device. The table pointer low byte register 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 "0706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by TBLP and TBHP if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

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

Table Read Program Example

```
tempreg1 db ? ; temporary register #1
tempreg2 db ? ; temporary register #2
:
:
mov a,06h     ; initialise low table pointer - note that this address is referenced
mov tblp,a    ; to the last page or the page that tbhp pointed
mov a,07h     ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1 ; transfers value in table referenced by table pointer data at program
               ; memory address "0706h" transferred to tempreg1 and TBLH
dec tblp       ; reduce value of table pointer by one
tabrd tempreg2 ; transfers value in table referenced by table pointer data at
               ; program memory address "0705h" transferred to tempreg2 and TBLH
               ; in this example the data "1AH" is transferred to tempreg1
               ; and data "0FH" to register tempreg2 and the data "00H" is
               ; transferred to TBLH
:
:
org 0700h      ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

In Circuit Programming – ICP

The provision of OTP type Program Memory and Flash type Voice ROM provide users with a means of convenient, which can program the applications One-Time into the device and program and modify the voice data.

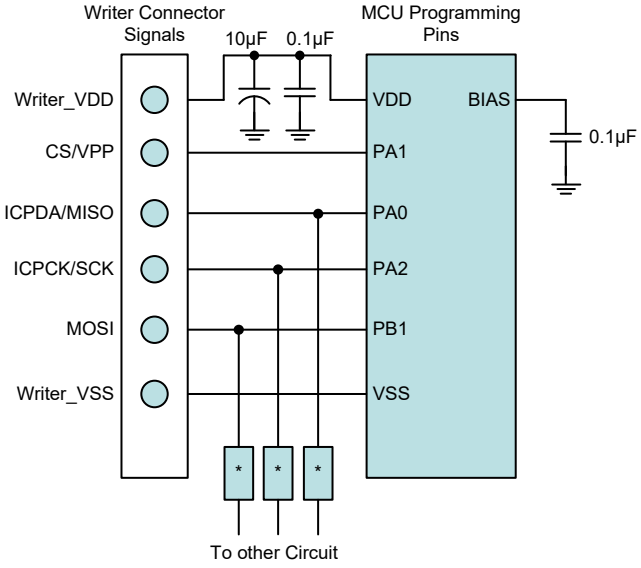
As an additional convenience, Holtek has provided a means of programming the microcontroller in- circuit using a 6-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with an un-programmed microcontroller, and then programming or upgrading the voice data at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest voice data releases without removal and re- insertion of the devices.

The Holtek Flash MCU to Writer Programming Pin correspondence table is as follows:

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPCK/SCK	PA2	Programming Serial Data/Address
ICPDA/MISO	PA0	Programming Data
CS/VPP	PA1	Programming Chip Select / OTP ROM power supply (8.5V)
MOSI	PB1	Programming Clock
VDD	VDD	Power Supply
VSS	VSS	Ground

The Program Memory can be programmed serially in-circuit using this 6-wire interface. Data is downloaded and uploaded serially on two pins and an additional line for the clock and one pin for chip select. Two additional lines are required for the power supply. The technical details regarding the in-circuit programming of the devices are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process, the user must take control of the ICPDA/MISO, ICPCK/SCK, MOSI and CS/VPP pins for data and clock programming purposes to ensure that no other outputs are connected to these four pins.

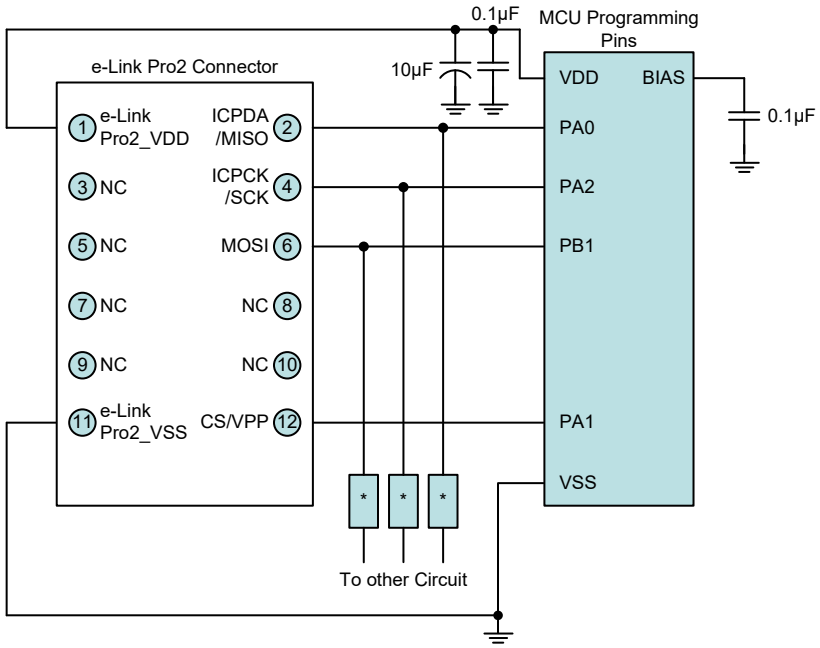


Note: 1. * may be resistor or capacitor. The resistance of * must be greater than 1kΩ or the capacitance of * must be less than 1nF.

2. When the MCU is programmed, the CS/VPP pin will output a high voltage. If there is an external circuit, it may cause damage to the external circuit.

3. The capacitors must be close to the MCU.

The HT68RV03x can be programmed using the Voice Workshop via the e-Link Pro2. The connection method is shown below:



Note: The capacitors must be close to the MCU.

On Chip Debug Support – OCDS

There is an EV chip named HT68EVV03x which is used to emulate the HT68RV03x device. This EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function and package type. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

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.

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

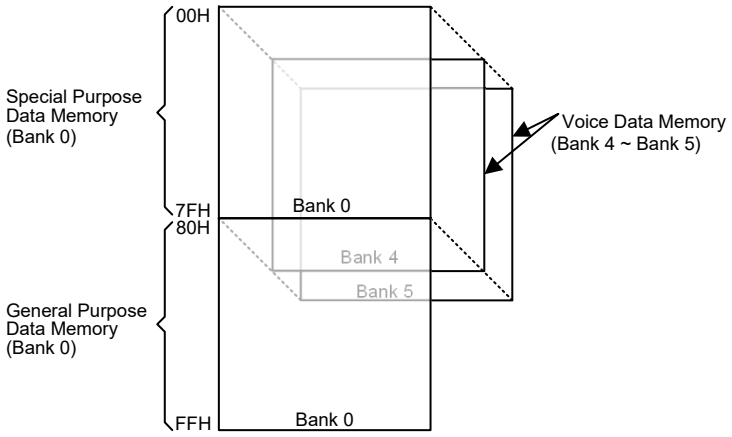
The devices also provide a dedicated for the Voice Data Memory. This area is used to store voice data.

Structure

The Data Memory is subdivided into several banks, all of which are implemented in 8-bit wide RAM. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH. The addresses of the Voice Data Memory area is in the area of 00H~7FH of Bank 4~5. Switching between the different Data Memory banks is achieved by setting the Data Memory Bank Pointer to the correct value. The start address of the Data Memory for the device is the address 00H.

Special Purpose Data Memory	General Purpose Data Memory		Voice Data Memory
Located Bank	Capacity	Address	Address
Bank 0	128×8	Bank 0: 80H~FFH	Bank 4: 00H~7FH Bank 5: 00H~7FH

Data Memory Summary



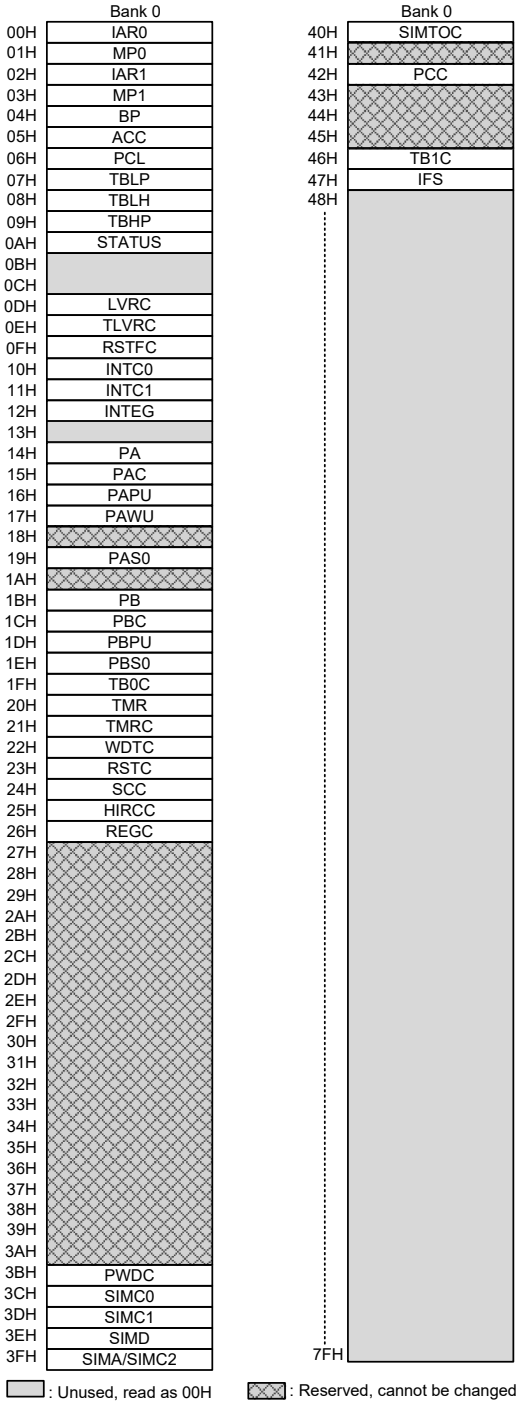
Data Memory Structure

General Purpose Data Memory

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

Special Purpose Data Memory

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



Special Purpose Data Memory Structure

Voice Flash Memory

The devices include a fully integrated 2Mbit~32Mbit Voice Flash Memory to store voice data, which can be edited using Holtek Voice Workshop. After edited in the HT-IDE3000 using Holtek voice library, it can then be programmed using the HOPE3200, e-Writer32 or e-Link Pro2. For more details about how to use the voice libraries, refer to the related documentation.

The Voice Flash Memory is powered by an internal 3.0V LDO whose power supply ranges from 2.3V to 3.6V (HT68RV032/33/34/35/36) / 2.0V to 3.6V (HT68RV034L). When the VDD is greater than 3.6V, the LDO should be set to "on". For more control methods, refer to the Voltage Regulator section.

Device	Capacity
HT68RV032	2Mbit
HT68RV033	4Mbit
HT68RV034/HT68RV034L	8Mbit
HT68RV035	16Mbit
HT68RV036	32Mbit

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

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

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

Indirect Addressing Program

```
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                ; increase memory pointer
     sdz block              ; check if last memory location has been cleared
     jmp loop
continue:
```

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

Bank Pointer – BP

For these devices, the Data Memory is divided into several banks. Selecting the required Data Memory area is achieved using the Bank Pointer. The DMBP2~DMBP0 bits in the BP register is used to select Data Memory Banks. The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in IDLE or SLEEP Mode, in which case, the Data Memory bank remains unaffected. It should be noted that the Special Purpose Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within any bank. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from banks other than Bank 0 must be implemented using Indirect Addressing.

- **BP Register**

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

Bit 7~3 Unimplemented, read as “0”

Bit 2~0 **DMBP2~DMBP0**: Data Memory Bank Selection
000: Bank 0
100: Bank 4
101: Bank 5
Others: Undefined

Accumulator – ACC

The Accumulator is central to the operation of any microcontroller and is closely related with operations carried out by the ALU. The Accumulator is the place where all intermediate results from the ALU are stored. Without the Accumulator it would be necessary to write the result of each calculation or logical operation such as addition, subtraction, shift, etc., to the Data Memory resulting in higher programming and timing overheads. Data transfer operations usually involve

the temporary storage function of the Accumulator; for example, when transferring data between one user-defined register and another, it is necessary to do this by passing the data through the Accumulator as no direct transfer between two registers is permitted.

Program Counter Low Byte Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller. With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition, operations related to the status register may give different results due to the different instruction operations. The TO flag can be affected only by a system power-up, a WDT time-out or by executing the "CLR WDT" or "HALT" instruction. The PDF flag is affected only by executing the "HALT" or "CLR WDT" instruction or during a system power-up. The Z, OV, AC and C flags generally reflect the status of the latest operations.

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

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

• **STATUS Register**

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

"x": unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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

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

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

Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

The "C" flag is also affected by a rotate through carry instruction.

Oscillators

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

Oscillator Overview

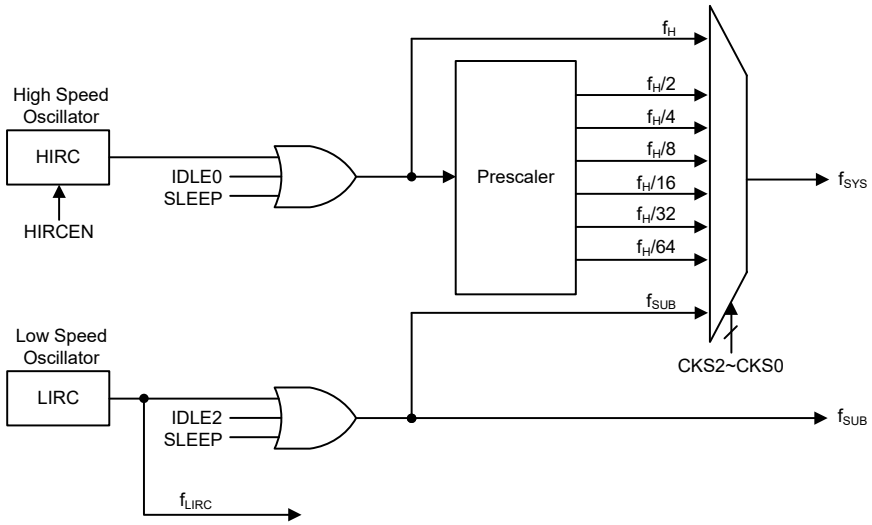
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

Type	Name	Frequency
Internal High Speed RC	HIRC	12MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two oscillator sources, one high speed oscillator and one low speed oscillator. The high speed system clock is sourced from the internal 12MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.



System Clock Configurations

Internal High Speed RC Oscillator – HIRC

The high speed internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has one fixed frequency of 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.

Internal 32kHz Oscillator – LIRC

The internal 32kHz System Oscillator is also a fully integrated RC oscillator with a typical frequency of 32kHz, requiring no external components for its implementation.

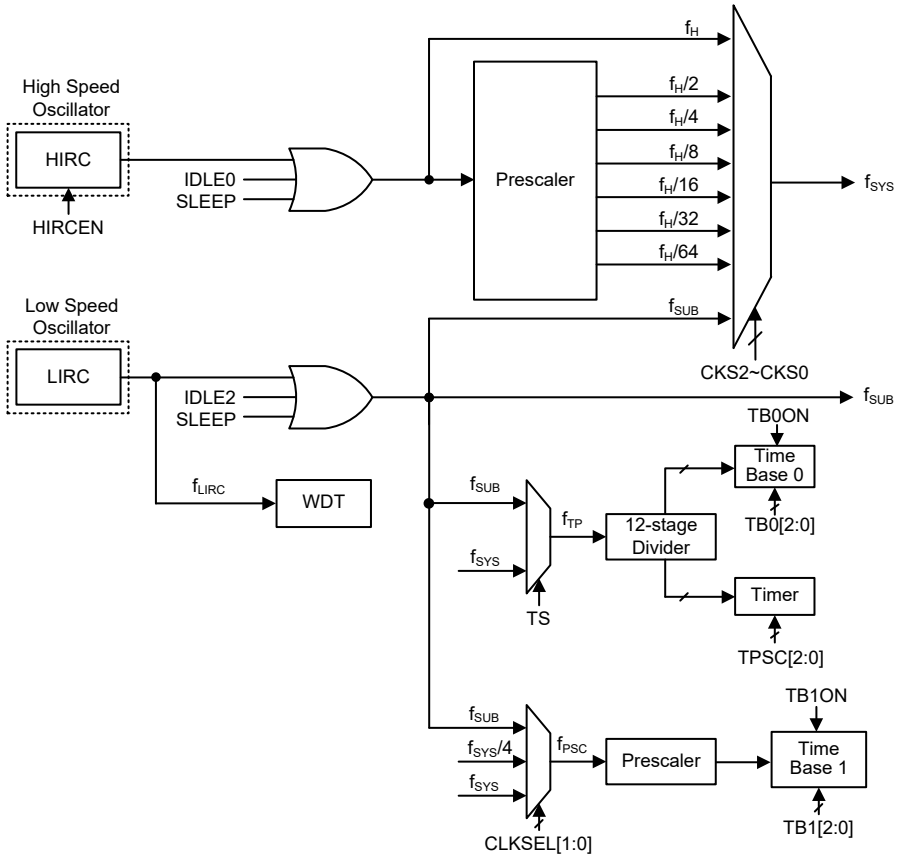
Operating Modes and System Clocks

Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice versa, lower speed clocks reduce current consumption. As Holtek has provided the device with both high and low speed clock sources and the means to switch between them dynamically, the user can optimise the operation of their microcontroller to achieve the best performance/power ratio.

System Clocks

The device has many different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock options using register programming, a clock system can be configured to obtain maximum application performance.

The main system clock, can come from a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.



Device Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillator will stop to conserve the power or continue to oscillate to provide the clock source, f_H~$f_H/64$, for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the FAST Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Modes are used when the microcontroller CPU is switched off to conserve power.

Operation Mode	CPU	Register Setting			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
FAST	On	x	x	000~110	f_H~f_H/64	On	On	On
SLOW	On	x	x	111	f_{SUB}	On/Off (1)	On	On
IDLE0	Off	0	1	000~110	Off	Off	On	On
				111	On			
IDLE1	Off	1	1	xxx	On	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off	On
				111	Off			
SLEEP	Off	0	0	xxx	Off	Off	Off	On/Off (2)

"x": Don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.
2. The f_{LIRC} clock will be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

FAST Mode

This is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source which will come from the high speed oscillator, HIRC. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from the LIRC oscillator.

SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bit both are low. In the SLEEP mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral function will also be stopped. However, the f_{LIRC} clock can continue to operate if the WDT function is enabled.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is low and the FSIDEN bit in the SCC register is high. In the IDLE0 Mode the CPU will be switched off but the low speed oscillator will be turned on to drive some peripheral functions.

IDLE1 Mode

The IDLE1 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is high. In the IDLE1 Mode the CPU will be switched off but both the high and low speed oscillators will be turned on to provide a clock source to keep some peripheral functions operational.

IDLE2 Mode

The IDLE2 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is low. In the IDLE2 Mode the CPU will be switched off but the high speed oscillator will be turned on to provide a clock source to keep some peripheral functions operational.

Control Registers

The SCC and HIRCC registers are used to control the system clock and the HIRC oscillator configurations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN

System Operating Mode Control Register List

- **SCC Register**

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	0	0	0	—	—	—	0	0

Bit 7~5 **CKS2~CKS0**: System clock selection
000: f_H
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: f_{SUB}
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from f_H or f_{SUB}, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4~2 Unimplemented, read as "0"

Bit 1 **FHIDEN**: High frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing a "HALT" instruction.

Bit 0 **FSIDEN**: Low frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing a "HALT" instruction.

Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.
Clock switching delay time = $4\times t_{SYS}+[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$, where $t_{Curr.}$ indicates the current clock period, $t_{Tar.}$ indicates the target clock period and t_{SYS} indicates the current system clock period.

• **HIRCC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection
00: Reserved, cannot be used
01: 12MHz
10: Reserved, cannot be used
11: Reserved, cannot be used
It should be noted that the HIRC[1:0] bits must be fixed at "01".
When the HIRC oscillator is enabled the clock frequency will automatically be changed after the HIRCF flag is set to 1.

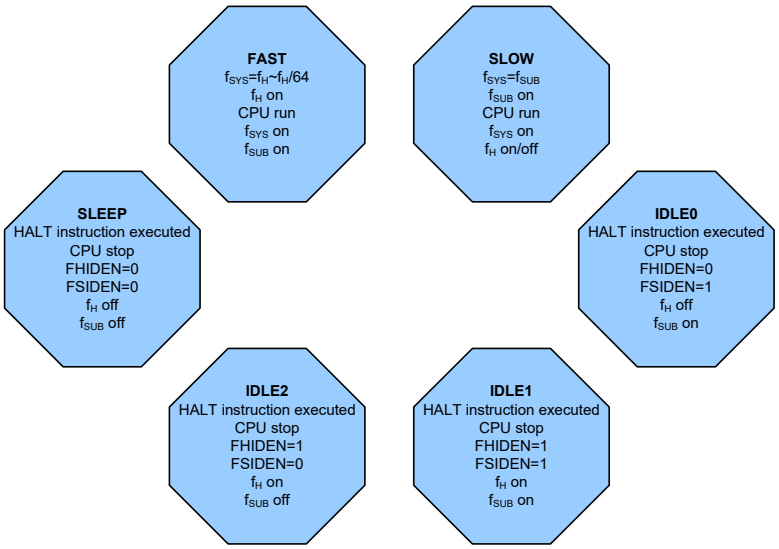
Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

Bit 0 **HIRCEN**: HIRC oscillator enable control
0: Disable
1: Enable

Operating Mode Switching

The device can switch between operating modes dynamically allowing the user to select the best performance/power ratio for the present task in hand. In this way microcontroller operations that do not require high performance can be executed using slower clocks thus requiring less operating current and prolonging battery life in portable applications.

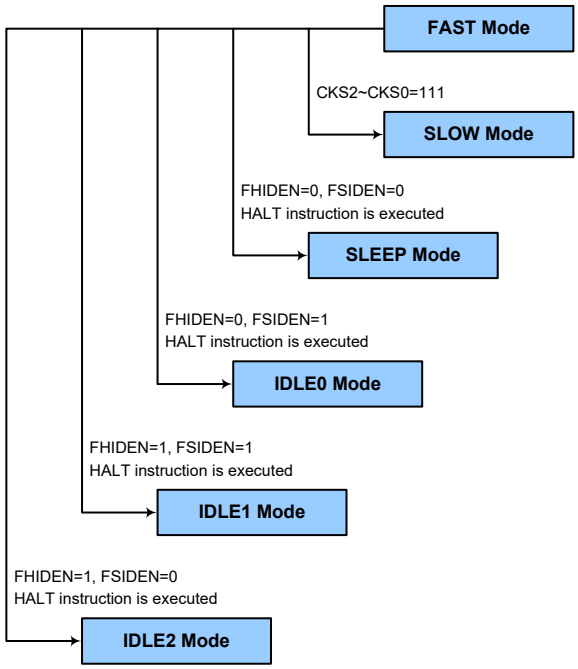
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the FHIDEN and FSIDEN bits in the SCC register.



FAST Mode to SLOW Mode Switching

When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by setting the CKS2~CKS0 bits to "111" in the SCC register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

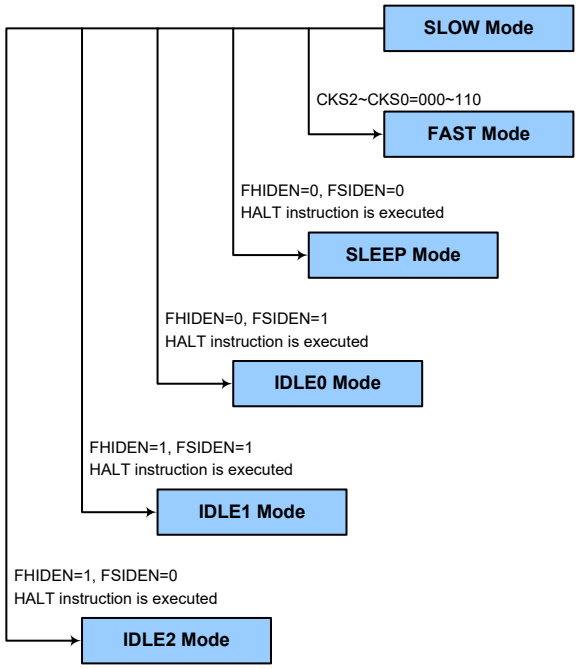
The SLOW Mode system clock is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

In SLOW mode the system clock is derived from f_{SUB}. When system clock is switched back to the FAST mode from f_{SUB}, the CKS2~CKS0 bits should be set to "000"~"110" and then the system clock will respectively be switched to $f_H \sim f_H/64$.

However, if f_H is not used in SLOW mode and thus switched off, it will take some time to re-oscillate and stabilise when switching to the FAST mode from the SLOW Mode. This is monitored using the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the System Start Up Time Characteristics.



Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "0". In this mode all the clocks and functions will be switched off except the WDT function. When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the FHIDEN bit in the SCC register equal to "0" and the FSIDEN bit in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The f_H clock will be stopped and the application program will stop at the "HALT" instruction, but the f_{SUB} clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE1 Mode

There is only one way for the device to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The f_H and f_{SUB} clocks will be on but the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE2 Mode

There is only one way for the device to enter the IDLE2 Mode and that is to execute the "HALT" instruction in the application program with the FHIDEN bit in the SCC register equal to "1" and the FSIDEN bit in the SCC register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

- The f_H clock will be on but the f_{SUB} clock will be off and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Modes, 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. This also applies to the device which has different package types, as there may be unbonded pins. These pins must either be setup as outputs or if setup as inputs must have pull-high resistors connected. In addition, the I/O pin-shared with VPP must not be set to output high, as this could 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. Also note that additional standby current will also be required if the LIRC oscillator has been enabled.

In the IDLE1 and IDLE2 Modes the high speed oscillator is on, if the peripheral function clock source is derived from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

To minimise power consumption the device can enter the SLEEP or any IDLE Mode, where the CPU will be switched off. However, when the device is woken up again, it will take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

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

- An external falling edge on Port A
- An external $\overline{\text{RES}}$ pin reset
- A system interrupt
- A WDT overflow

If the system is woken up by an external $\overline{\text{RES}}$ pin 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 flag. 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, other flags remain in their original status.

Each pin on Port A can be set using the PAWU register to permit a negative transition on the pin to wake-up the system. When a pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{LIRC} which is sourced from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of $[(2^8\text{-}2^0)\sim2^8]\sim[(2^{15}\text{-}2^7)\sim2^{15}]$ to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as Watchdog Timer the enable/disable operation.

• **WDTC Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3 **WDTEN**: WDT function enable control
0: Disable
1: Enable

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $[(2^8-2^0)\sim2^8]/f_{LIRC}$
001: $[(2^9-2^1)\sim2^9]/f_{LIRC}$
010: $[(2^{10}-2^2)\sim2^{10}]/f_{LIRC}$
011: $[(2^{11}-2^3)\sim2^{11}]/f_{LIRC}$
100: $[(2^{12}-2^4)\sim2^{12}]/f_{LIRC}$
101: $[(2^{13}-2^5)\sim2^{13}]/f_{LIRC}$
110: $[(2^{14}-2^6)\sim2^{14}]/f_{LIRC}$
111: $[(2^{15}-2^7)\sim2^{15}]/f_{LIRC}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Refer to the $\overline{RES}$ Pin Reset section.

Bit 2 **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.

Bit 1 **LRF**: LVR control register software reset flag
Refer to the Low Voltage Reset section.

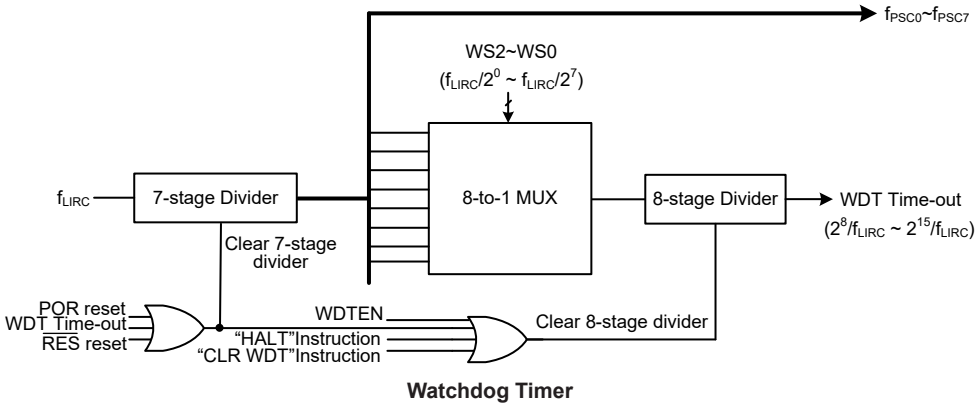
Bit 0 Unimplemented, read as "0"

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there is one bit, WDTEN, in the WDTC register to offer the enable/disable control.

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO and PDF bits 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 using the Watchdog Timer software clear instruction, the second is via a HALT instruction. The third is an external hardware reset, which means a low level on the external reset pin.

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



Watchdog Timer

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power that 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 device is running. One example of this is where after power has been applied and the device 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 registers remain unchanged allowing the device to proceed with normal operation after the reset line is allowed to return high.

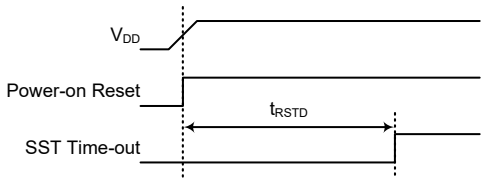
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller.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

Reset Functions

There are several 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.



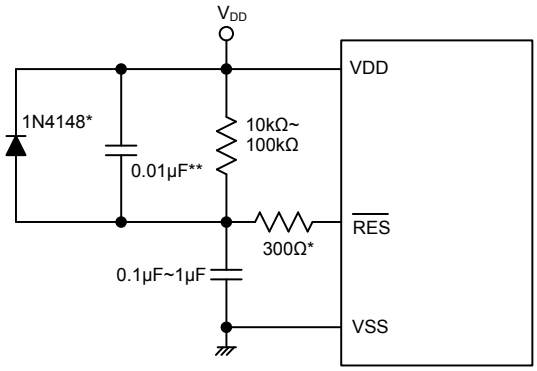
Power-on Reset Timing Chart

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

As the reset pin is shared with I/O pins, the reset function must be selected using the control register, RSTC. 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 $\overline{\text{RES}}$ pin, whose additional time delay will ensure that the $\overline{\text{RES}}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{\text{RES}}$ line reaches a certain voltage value, the reset delay time, t_{RSTD}, is invoked to provide an extea delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Time.

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

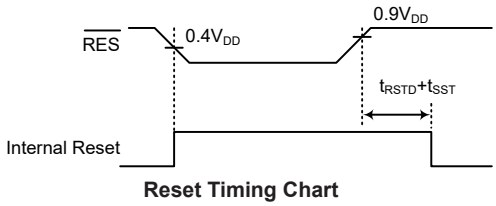
For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



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

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

Pulling the $\overline{\text{RES}}$ Pin low using external hardware will also execute a device reset. In this case, as in the case of other resets, the Program Counter will reset to zero and program execution initiated from this point.



Reset Timing Chart

There is an internal reset control register, RSTC, which is used to provide a reset when the device operates abnormally due to the environmental noise interference. If the content of the RSTC register is set to any value other than 01010101B or 10101010B, it will reset the device after a delay time, t_{SRESET}. After power on the register will have a value of 01010101B.

RSTC7 ~ RSTC0 Bits	Reset Function
01010101B	I/O
10101010B	$\overline{RES}$
Any other value	Reset MCU

Internal Reset Function Control

- **RSTC Register**

Bit	7	6	5	4	3	2	1	0
Name	RSTC7	RSTC6	RSTC5	RSTC4	RSTC3	RSTC2	RSTC1	RSTC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7~0 **RSTC7~RSTC0**: Reset function control
01010101: I/O or other pin-shared functions
10101010: $\overline{RES}$ pin
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET}, and the RSTF bit in the RSTFC register will be set to 1.
All resets will reset this register to POR value except the WDT time out hardware warm reset. Note that when this register is set to 10101010B to select the $\overline{RES}$ pin function, this configuration has higher priority than other related pin-shared controls.

- **RSTFC Register**

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

"x": unknown

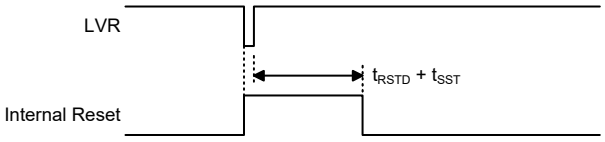
Bit 7~4 Unimplemented, read as "0"
Bit 3 **RSTF**: Reset control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the RSTC control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program
Bit 2 **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.
Bit 1 **LRF**: LVR control register software reset flag
Refer to the Low Voltage Reset section.
Bit 0 Unimplemented, read as "0"

Low Voltage Reset – LVR

The microcontrollers contain a low voltage reset circuit in order to monitor the supply voltage of the devices and provide an MCU reset when the value falls below a certain predefined level. If the supply voltage of the devices drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the devices

internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVR Electrical Characteristics. If the duration of 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 t_{LVR} value can be selected by the TLVR1~TLVR0 bits in the TLVRC register. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by environmental noise, the LVR will reset the devices after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set high. After power on the LVRC register will have the value of 01010101B.

Note that the LVR function will be automatically disabled when the devices enter the SLEEP or IDLE mode.



Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select
01010101: 1.9V (only available for HT68RV034L)
00110011: 2.2V
10011001: 2.55V
10101010: 3.15V
11110000: LVR disable
Other values: Generates a MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by the LVR voltage value above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps for greater than a t_{LVR} time. In this situation the register contents will remain the same after such a reset occurs. Any register value, other than the four defined register values above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time, t_{SRESET}. However in this situation the register contents will be reset to the POR value.

- **TLVRC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TLVR1	TLVR0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	1

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **TLVR1~TLVR0**: Minimum low voltage width to reset time (t_{LVR}) selection
00: (7~8)×t_{LIRC}
01: (31~32)×t_{LIRC}
10: (63~64)×t_{LIRC}
11: (127~128)×t_{LIRC}

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Refer to the $\overline{\text{RES}}$ Pin Reset section.

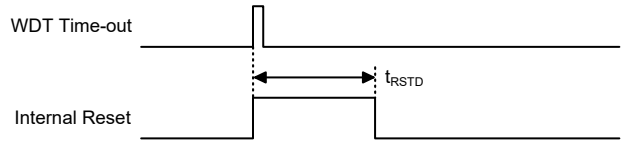
Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set to 1 when a specific low voltage reset condition occurs. This bit can only be cleared to zero by application program.

Bit 1 **LRF**: LVRC register software reset flag
0: Not occurred
1: Occurred
This bit is set high if the LVRC register contains any non-defined LVR voltage register values. This in effect acts like a software reset function. This bit can only be cleared to zero by application program.

Bit 0 Unimplemented, read as "0"

Watchdog Time-out Reset during Normal Operation

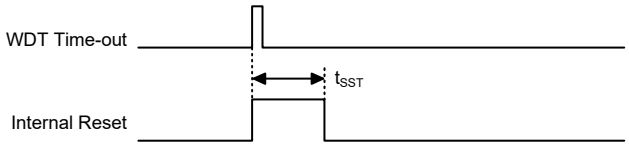
When the Watchdog Time-out Reset during normal operation in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

The Watchdog Time-out Reset during SLEEP or IDLE Mode is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to "0" and the TO and PDF flags will be set to "1". Refer to the System Start Up Time Characteristics for t_{SST} details.



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

TO	PDF	Reset Conditions
0	0	Power-on reset
u	u	$\overline{\text{RES}}$ or LVR reset during FAST or SLOW Mode operation
1	u	WDT time-out reset during FAST or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

"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, Time Bases	Cleared after reset, WDT begins counting
Timer/Event Counter	Timer/Event Counter will be turned off
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	Reset (Power On)	$\overline{\text{RES}}$ Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
IAR1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	---- -000	---- -000	---- -000	---- -uuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---- -xxx	---- -uuu	---- -uuu	---- -uuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
LVRC	0101 0101	0101 0101	0101 0101	uuuu uuuu
TLVRC	---- --01	---- --01	---- --01	---- --uu
RSTFC	---- 0x0-	---- uuu-	---- uuu-	---- uuu-
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTEG	---- --00	---- --00	---- --00	---- --uu
PA	1111 -111	1111 -111	1111 -111	uuuu –uuu
PAC	1111 -111	1111 -111	1111 -111	uuuu –uuu
PAPU	0000 -000	0000 -000	0000 -000	uuuu –uuu
PAWU	0000 -000	0000 -000	0000 -000	uuuu –uuu
PAS0	--00 0000	--00 0000	--00 0000	--uu uuuu

Register	Reset (Power On)	$\overline{\text{RES}}$ Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PB	---- --11	---- --11	---- --11	---- --uu
PBC	---- --11	---- --11	---- --11	---- --uu
PBPU	---- --00	---- --00	---- --00	---- --uu
PBS0	---- 0000	---- 0000	---- 0000	---- uuuu
TB0C	0--- -000	0--- -000	0--- -000	u--- -uuu
TMR	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMRC	0000 1000	0000 1000	0000 1000	uuuu uuuu
WDTC	---- 1111	---- 1111	---- 1111	---- uuuu
RSTC	0101 0101	0101 0101	0101 0101	uuuu uuuu
SCC	000- --00	000- --00	000- --00	uuu- --uu
HIRCC	---- 0001	---- 0001	---- 0001	---- uuuu
REGC	---- --00	---- --00	---- --00	---- --uu
PWDC	0--- 0100	0--- 0100	0--- 0100	u--- uuuu
SIMC0	1110 0000	1110 0000	1110 0000	uuuu uuuu
SIMC1	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMTOC	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCC	---- 1111	---- 1111	---- 1111	---- uuuu
TB1C	0-00 -000	0-00 -000	0-00 -000	u-uu -uuu
IFS	---- 0000	---- 0000	---- 0000	---- uuuu

Note: “u” stands for unchanged
“x” stands for unknown
“-” stands for unimplemented

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. With the input or output designation of every pin fully under user program control, pull-high selections for all ports and wake-up selections on certain pins, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	—	PA2	PA1	PA0
PAC	PAC7*	PAC6*	PAC5*	PAC4*	—	PAC2	PAC1	PAC0
PAPU	PAPU7**	PAPU6**	PAPU5**	PAPU4**	—	PAPU2	PAPU1	PAPU0
PAWU	PAWU7**	PAWU6**	PAWU5**	PAWU4**	—	PAWU2	PAWU1	PAWU0
PB	—	—	—	—	—	—	PB1	PB0
PBC	—	—	—	—	—	—	PBC1	PBC0
PBPU	—	—	—	—	—	—	PBPU1	PBPU0
PCC	—	—	—	—	PCC3*	PCC2*	PCC1*	PCC0*

"—": Unimplemented, read as "0"

I/O Logic Function Register List

Note: The control bit with an asterisk * should be cleared to 0 and the control bit with an asterisk ** should remain the POR value after power-on reset. This can set these unbonded and not internally used lines as outputs to prevent additional power consumption.

Pull-high Resistors

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

Note that the pull-high resistor can be controlled by the relevant pull-high control register only when the pin-shared functional pin is selected as a digital input or NMOS output. Otherwise, the pull-high resistors cannot be enabled.

- **PxPU Register**

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

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

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

I/O Port Wake-up

The HALT instruction forces the microcontroller into the SLEEP or IDLE Mode which preserves power, a feature that is important for battery and other low-power applications. Various methods exist to wake up the microcontroller, one of which is to change the logic condition on one of the I/O pins with wake up ability from high to low. This function is especially suitable for applications that can be woken up via external switches. Each I/O pin with wake-up ability can be selected individually to have this wake-up feature using the PAWU register.

Note that the wake-up function can be controlled by the wake-up control register only when the pin is selected as a general purpose input and the MCU enters the IDLE or SLEEP mode.

• **PAWU Register**

Bit	7	6	5	4	3	2	1	0
Name	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
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

PAWUn: I/O port wake-up function control
0: Disable
1: Enable
The PAWUn bit is used to control the pin wake-up function.

I/O Port Control Registers

Each I/O port has its own control register known as PAC~PBC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. 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.

• **PxC Register**

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

PxCn: I/O Port x Pin type selection
0: Output
1: Input
The PxCn bit is used to control the pin type selection. Here the "x" can be A or B. However, the actual available bits for each I/O Port may be different.

Pin-shared Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For these pins, the desired function of the multi-function I/O pins is selected by a series of registers via the application program control.

Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The device includes Port "x" output function selection register "n", labeled as PxSn, and Input Function Selection register, labeled as IFS, which can select the desired functions of the multi-function pin-shared pins.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as INT, TC, etc., which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bit fields. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be setup as input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	—	—	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PBS0	—	—	—	—	PBS03	PBS02	PBS01	PBS00
IFS	—	—	—	—	—	SCSBPS	SDIOPS	SCKLPS

Pin-shared Function Selection Register List

- **PAS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
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 5~4 **PAS05~PAS04**: PA2 pin-shared function selection
00: PA2
01: PA2
10: PA2
11: SDIO/SDA

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00: PA1/TC/$\overline{\text{RES}}$/VPP
01: PA1/TC/$\overline{\text{RES}}$/VPP
10: PA1/TC/$\overline{\text{RES}}$/VPP
11: $\overline{\text{SCS}}$

Bit 1~0 **PAS01~PAS00**: PA0 pin-shared function selection
00: PA0/INT
01: PA0/INT
10: PA0/INT
11: SCK/SCL

• **PBS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PBS03	PBS02	PBS01	PBS00
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"
Bit 3~2 **PBS03~PBS02**: PB1 pin-shared function selection
00: PB1
01: PB1
10: PB1
11: PWM2
Bit 1~0 **PBS01~PBS00**: PB0 pin-shared function selection
00: PB0
01: PB0
10: PB0
11: PWM1

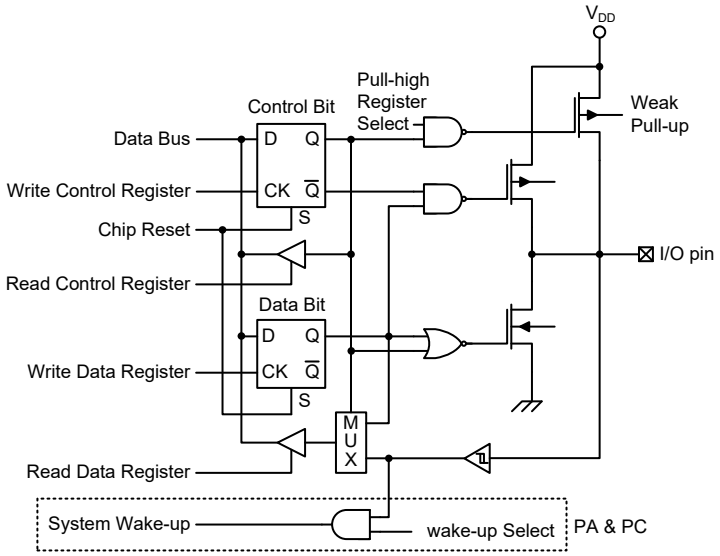
• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	SCSBPS	SDIOPS	SCKLPS
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"
Bit 3 Unimplement, can be read/write
Bit 2 **SCSBPS**: $\overline{\text{SCS}}$ input source pin selection
0: PA1
1: Reserved
Bit 1 **SDIOPS**: SDIO/SDA input source pin selection
0: PA2
1: Reserved
Bit 0 **SCKLPS**: SCK/SCL input source pin selection
0: PA0
1: Reserved

I/O Pin Structures

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



Logic Function Input/Output Structure

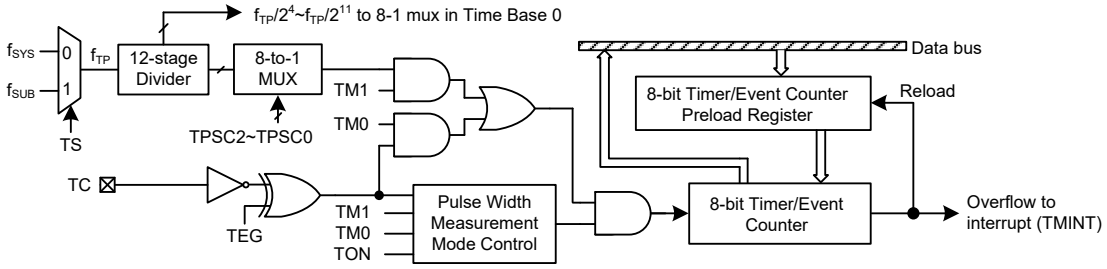
Programming Considerations

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

Port A has the additional capability of providing wake-up functions. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the Port A pins. Single or multiple pins on Port A can be setup to have this function.

Timer/Event Counter

The provision of the Timer/Event Counter forms an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contains an 8-bit Timer/Event Counter, which contains an 8-bit programmable count-up counter and the clock may come from an external or internal clock source. As the timer has three different operating modes, it can be configured to operate as a general timer, an external event counter or a pulse width measurement device.



8-bit Timer/Event Counter

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 and Pulse Width Measurement Mode. For the Timer/Event Counter, this internal clock source can be configured by the TS bit in the TMRC Timer Control Register to be derived from the f_{SYS} or f_{SUB} clock, the division ratio of which is selected by the TPSC2~TPSC0 bits in the TMRC Timer/Event Control Register.

An external clock source is used when the Timer/Event Counter is in the Event Counter Mode, the clock source is provided on the external TC pin. Depending upon the condition of the TEG bit, each high to low or low to high transition on the external timer pin will increase the counter by one.

Timer/Event Counter Registers

There are two registers related to the Timer/Event Counter. The first is the TMR register that contains the actual value of the timer and into which an initial value can be preloaded. Writing to the TMR register will transfer the specified data to the Timer/Event Counter. Reading the TMR register will read the contents of the Timer/Event Counter. The second is the TMRC control register, which is used to define the operating mode, select the internal clock source, control the counting enable or disable and select the active edge.

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMR	D7	D6	D5	D4	D3	D2	D1	D0
TMRC	TM1	TM0	TS	TON	TEG	TPSC2	TPSC1	TPSC0

Timer/Event Counter Register List

Timer Register – TMR

The timer register TMR is the place where the actual timer value is stored. The value in the timer register 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, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be loaded with the preload register value and continue counting.

Note that to achieve a maximum full range count of FFH, the preload register must first be cleared. 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 until an overflow occurs.

• **TMR Register**

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

Bit 7~0 **D7~D0**: Timer preload register byte

Timer Control Register – TMRC

The flexible features of the Holtek microcontroller Timer/Event Counter are implemented by operating in three different modes, the options of which are determined by the contents of control register bits.

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

To select which of the three modes the timer is to operate in, namely the Timer Mode, the Event Counter Mode or the Pulse Width Measurement Mode, the TM1~TM0 bits in the Timer Control Register must be set to the required logic levels. The timer-on bit TON provides the basic on/off control of the respective timer. Setting the bit to high allows the counter to run. Clearing the bit stops the counter. When the internal clock source is used, it can be sourced from the f_{SYS} or f_{SUB} clock selected by setting the TS bit. Bits TPSC2~TPSC0 determine the division ratio of the selected clock source. The internal clock selection will have no effect if an external clock source is used. If the timer is in the Event Counter or Pulse Width Measurement Mode, the active transition edge type is selected by the logic level of the TEG bit in the Timer Control Register.

• **TMRC Register**

Bit	7	6	5	4	3	2	1	0
Name	TM1	TM0	TS	TON	TEG	TPSC2	TPSC1	TPSC0
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 **TM1~TM0**: Timer/Event Counter operating mode selection
00: Unused
01: Event Counter Mode
10: Timer Mode
11: Pulse Width Measurement Mode

Bit 5 **TS**: Timer f_{TP} clock source selection
0: f_{SYS}
1: f_{SUB}

Bit 4 **TON**: Timer/Event Counter counting enable
0: Disable
1: Enable

Bit 3 **TEG**: Timer/Event Counter active edge selection
Event Counter Mode
0: Count on rising edge
1: Count on falling edge
Pulse Width Measurement Mode
0: Start counting on falling edge, stop on rising edge
1: Start counting on rising edge, stop on falling edge

Bit 2~0 **TPSC2~TPSC0**: Timer internal clock selection
000: f_{TP}
001: $f_{TP}/2$
010: $f_{TP}/4$
011: $f_{TP}/8$
100: $f_{TP}/16$
101: $f_{TP}/32$
110: $f_{TP}/64$
111: $f_{TP}/128$

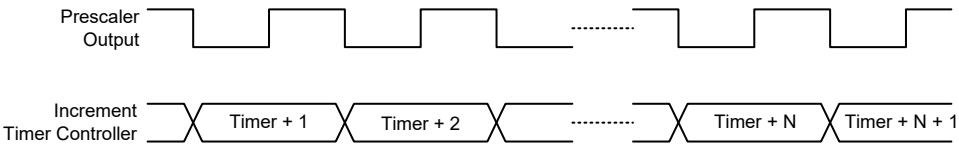
Timer/Event Counter Operating Modes

The Timer/Event Counter can operate in one of three operating modes, Timer Mode, Event Counter Mode or Pulse Width Measurement Mode. The operating mode is selected using the TM1 and TM0 bits in the TMRC register.

Timer Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "10" respectively. 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.

When operating in this mode the internal clock f_{TP} is used as the timer clock, which can be selected to be devirved from f_{SYS} or f_{SUB} by setting the TS bit in the TMRC register. The division of the f_{TP} clock is selected by the TPSC2~TPSC0 bits in the same register. The timer-on bit TON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increases by one. When the timer reaches its maximum 8-bit, FFH Hex, value and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. It should be noted that in the Timer mode, even if the device is in the IDLE/SLEEP mode, if the selected internal clock is still activated and a timer overflow occurs, it will generate a timer interrupt and corresponding wake-up source.



Timer Mode Timing Chart

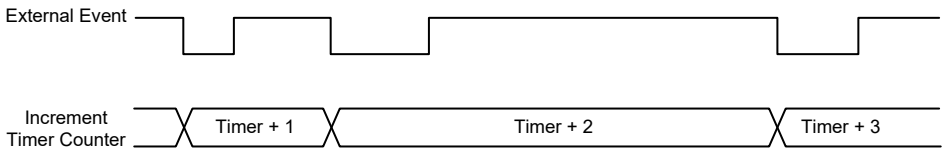
Event Counter Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "01" respectively. In this mode, a number of externally changing logic events, occurring on the external timer TC pin, can be recorded by the Timer/Event Counter.

When operating in this mode, the external timer pin, TC, is used as the Timer/Event Counter clock source. After the other bits in the Timer Control Register have been properly configured, the enable bit TON, can be set high to enable the Timer/Event Counter. If the Active Edge Selection bit, TEG, is low, the Timer/Event Counter will increase each time the TC pin receives a low to high transition. If the TEG bit is high, the counter will increase each time the TC 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.

As the exernal pin TC is pin-shared with PA1, the corresponding bit in the I/O port control register must be set high to set the pin as the input. It should be noted that in the Event Counter mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (TEG=1)

Pulse Width Measurement Mode

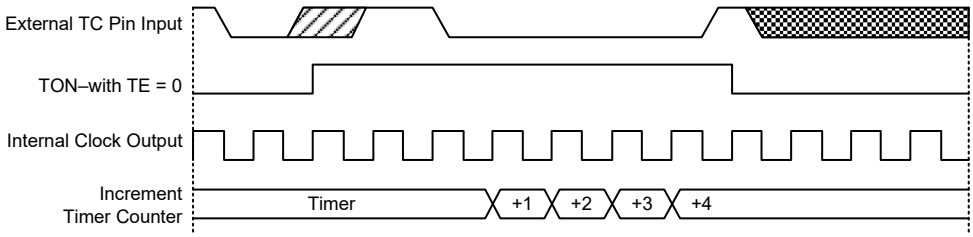
To select this mode, bits TM1 and TM0 in the TMRC register should be set to "11" respectively. In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin.

When operating in this mode the internal clock f_{TP} is used as the timer clock, which can be selected to be devirved from f_{SYS} or f_{SUB} by setting the TS bit in the TMRC register. The division of the f_{TP} clock is selected by the TPSC2~TPSC0 bits in the same register. After the other bits in the Timer Control Register have been properly configured, the enable bit TON, 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 TC pin.

If the active Edge Selection bit TEG is low, once a high to low transition has been received on the TC pin, the Timer/Event Counter will start counting based on the selected internal clock source until the TC 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 Selection bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the TC pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will then be automatically reset to zero. It is important to note that in the pulse width Measurement Mode, the enable bit is automatically reset to zero when the external control signal on the TC pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under application 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 TC 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 measurement until the enable bit is set high again by the application 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.

It should be noted that in the Pulse Width Measurement mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC pin if the selected internal clock source is still activated and the external signal continues to change state. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Pulse Width Measurement Mode Timing Chart (TEG=0)

Prescaler

When the prescaler clock is used as the Timer/Event Counter clock source, it should be noted that the prescaler is the same as the Time Base 0. For more detailed information concerning the prescaler, refer to the "Time Base Interrupts" section.

Programming Considerations

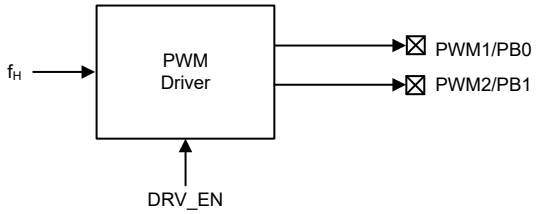
When running in the Timer Mode, if the internal system clock is used as the timer clock source, the timer can be synchronised with the overall operation of the microcontroller. In this mode when the timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector. For the Pulse Width Measurment Mode, the internal 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 errors in measured values requiring programmers to take this into account during programming. The same applies if the timer is configured to operate in the Event Counter Mode, which again is an external event and not synchronised with the internal 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, it should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialised before using them for the first time. The associated timer interrupt enable bit in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The active edge selection, timer operating mode selection and clock source control bits in timer control register must also be correctly issued to ensure the timer is properly configured for the required applications. It is also important to ensure that a desired initial value is first loaded into the timer register before the timer is switched on. 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 to generate an interrupt signal. If the Timer/Event Counter interrupt is enabled this will in turn allow program branch to its interrupt vector. However irrespective of whether the interrupt is enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in the IDLE/SLEEP mode. This situation may occur if the Timer/Event Counter internal clock source is still activated or if the external signal continues to change state. In such cases, the Timer/Event Counter will continue to count and if an overflow occurs the device will be woken up. 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.

Delta Sigma PWM Driver

The devices contain a fully integrated Delta Sigma PWM driver, which completes with volume control function. The voice control can be adjusted using the PWDC3~PWDC0 bits in the PWDC register.



Delta Sigma PWM Driver Block Diagram

PWM Driver Register

The control register exist to control the PWM1/PWM2 output I/O drive current to change the volume.

• PWDC Register

Bit	7	6	5	4	3	2	1	0
Name	DRV_EN	—	—	—	PWDC3	PWDC2	PWDC1	PWDC0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	1	0	0

Bit 7 **DRV_EN**: PWM output driver control
0: Disable
1: Enable

Bit 6~4 Unimplemented, read as "0"

Bit 3~0 **PWDC3~PWDC0**: PWM1/PWM 2 source/sink current control
0000: Source/Sink current = Level 0
0001: Source/Sink current = Level 1
0010: Source/Sink current = Level 2
0011: Source/Sink current = Level 3
0100: Source/Sink current = Level 4
0101: Source/Sink current = Level 5
0110: Source/Sink current = Level 6
0111: Source/Sink current = Level 7
1000: Source/Sink current = Level 8
1001: Source/Sink current = Level 9
1010: Source/Sink current = Level 10
1011: Source/Sink current = Level 11
1100: Source/Sink current = Level 12
1101: Source/Sink current = Level 13
1110: Source/Sink current = Level 14
1111: Source/Sink current = Level 15

Note: Refer to the PWM/PB0/PB1 Driver Characteristics section to obtain the exact value.

Serial Interface Module – SIM

The devices contain a Serial Interface Module, which includes both the three/four-line SPI interface or two-line I^2C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I^2C based hardware such as sensors, etc. The SIM interface pins are pin-shared with other I/O pins and therefore the SIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As both interface types share the same pins and registers, the choice of whether the SPI or I^2C type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O pins are selected using pull-high control registers when the SIM function is enabled and the corresponding pins are used as SIM input pins.

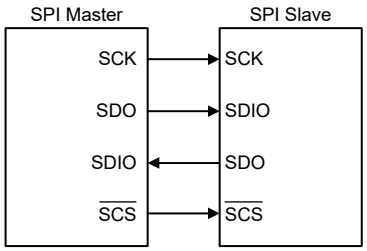
SPI Interface

The SPI interface is often used to communicate with external peripheral devices such as sensors etc. Originally developed by Motorola, the three/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.

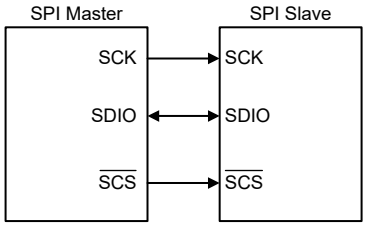
The communication is full-duplex or half-duplex and operates as a slave/master type, where the device can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, the device provided only one pin. If the master needs to control multiple 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 or half-duplex synchronous serial data link. For the four-line SPI interface with pin names SDIO, SDO, SCK and $\overline{SCS}$, the SDIO and SDO pins are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. For the three-line SPI interface with pin names SDIO, SCK and $\overline{SCS}$, the SDIO pin is the Serial Data Input/Output line, the SCK pin is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I^2C function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. After the desired SPI configuration has been set it can be 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. The SPI master/slave connection is either 3-wire mode or 4-wire mode selected by the SIMWIRE bit in the SIMC0 register.



4-wire Mode SPI Master/Slave Connection – SIMWIRE=0

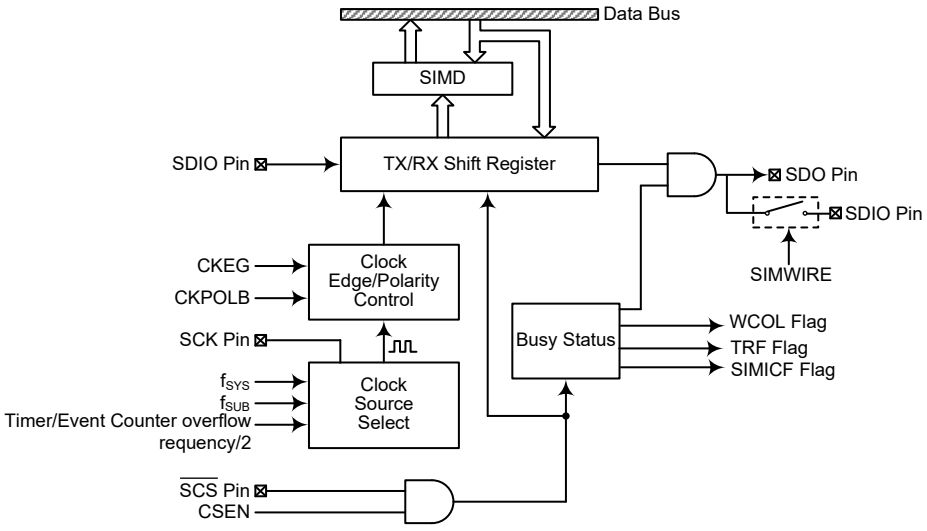


3-wire Mode SPI Master/Slave Connection – SIMWIRE=1

The SPI function in the devices offer the following features:

- Full duplex/Half-duplex (3-wire mode) synchronous data transfer
- Both Master and Slave modes
- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge

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.



SPI Block Diagram

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. The SIMC1 register is only used by the I²C interface.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	SIMFLT	SIMTRS	SIMWIRE	SIMEN	SIMICF
SIMC2	D7	DATA_IND	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
SIMD	D7	D6	D5	D4	D3	D2	D1	D0

SPI Register List

SPI Data Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C 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

Bit 7~0 **D7~D0**: SIM data register bit 7 ~ bit 0

SPI Control Registers

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 I²C function. The SIMC1 register is not used by the SPI function, only by the I²C function. Register SIMC0 is used to control the enable/disable function, to select SPI master/slave connection mode and to set the data transmission clock frequency. 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	SIMFLT	SIMTRS	SIMWIRE	SIMEN	SIMICF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SIM2~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_{SUB}
100: SPI master mode; SPI clock is Timer/Event Counter overflow frequency/2
101: SPI slave mode
110: I²C slave mode
111: Non SIM function

These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C 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 and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **SIMFLT**: SDIO pin state selection for the 3-wire mode only
0: The SDIO pin will remain the output state after transmitting the data
1: The SDIO pin will change to the input state after transmitting the data

This bit should be configured before the SIMD register is written/read or when the TRF bit is high, the hardware does not provide mistake-proof function.

When SIMFLT=1 and the SDIO pin is set to transmitter function, this pin will change to output state after writing the data to the SIMD register; the pin will change to input state by clearing the SIMTRS bit after the SPI data transfer is completely finished. This case does not affect the SIMFLT bit setting.

Bit 3 **SIMTRS**: SDIO pin function selection for the 3-wire mode only
0: Receiver function
1: Transmitter function

Bit 2 **SIMWIRE**: 3-wire Mode Enable bit
0: 4-wire mode, the SDIO pin is used as SPI receiver function only
1: 3-wire mode, the SDIO pin can be used as SPI receiver or transmitter function controlled by the SIMTRS bit

Bit 1 **SIMEN**: SIM Enable 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 SDIO, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I^2C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. 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^2C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I^2C 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^2C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SIM SPI slave mode Incomplete Transfer Flag
0: SIM SPI slave mode incomplete condition is not occurred
1: SIM SPI slave mode incomplete condition is occurred

This bit is only available when the SIM is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SIMEN and CSEN bits both being set to 1 but the $\overline{SCS}$ line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set to 1 together with the TRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the TRF bit will not be set to 1 if the SIMICF bit is set to 1 by software application program.

- **SIMC2 Register**

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

Bit 7 **D7**: Undefined bit

This bit can be read or written by the application program.

Bit 6 **DATA_IND**: Transmit/receive SCK status indication

When the SCK counter is 0 (it has not started to receive data), this bit is 0, otherwise the bit is 1.

Note: This bit is used for the slave mode. When the SCK count error is caused by glitch, which will affect the data reception, this bit can be checked by the firmware to determine whether the data is correct.

Bit 5 **CKPOLB**: SPI clock line base condition selection
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**: SPI SCK clock active edge type selection

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 first
- 1: MSB first

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.

Bit 1 **WCOL**: SPI write collision flag
- 0: No collision
- 1: Collision

The WCOL flag is used to detect whether a data collision has occurred or not. 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. This bit can be cleared by the application program.

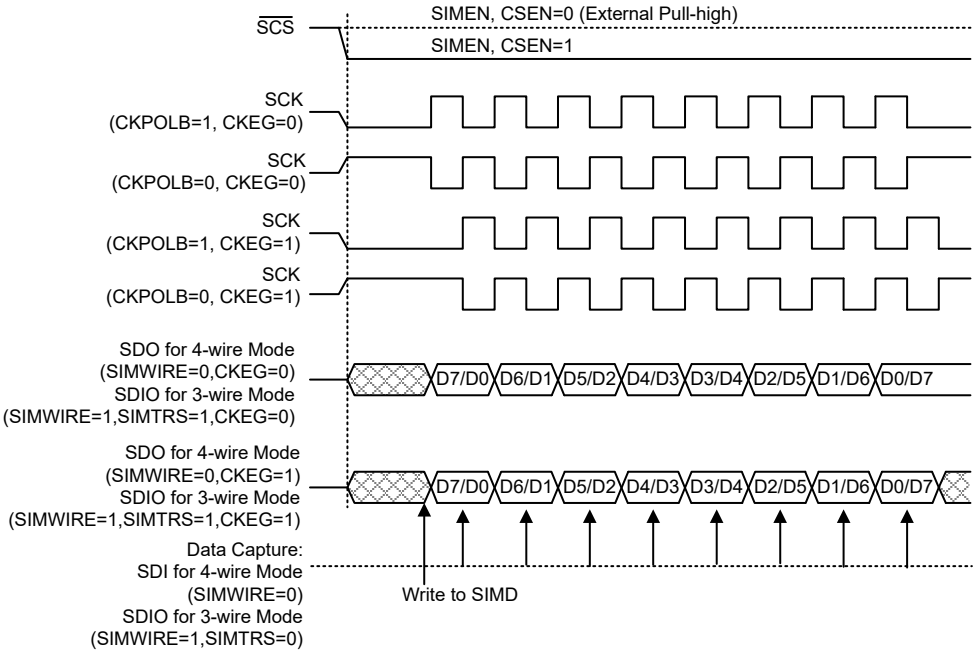
Bit 0 **TRF**: SPI Transmit/Receive complete flag
- 0: SPI data is being transferred
- 1: SPI data transfer is completed

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

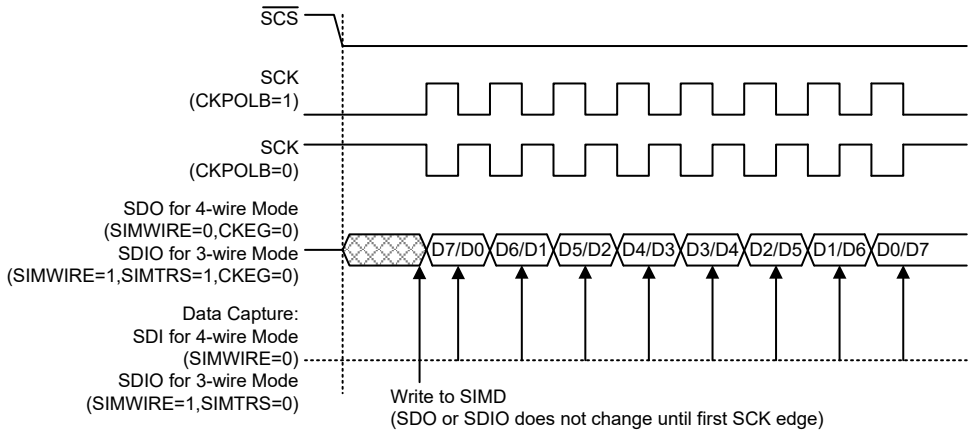
SPI Communication

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 for 4-wire mode and transmission or reception will begin selected by the SIMTRS bit in the SIMC0 register for 3-wire mode. 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 from the output pin and any data on the SDIO pin will be shifted into the SIMD register when in the receiver mode. The master should output an $\overline{\text{SCS}}$ signal to enable the slave devices before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the SCK signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagram shows the relationship between the slave data and SCK signal for various configurations of the CKPOLB and CKEG bits.

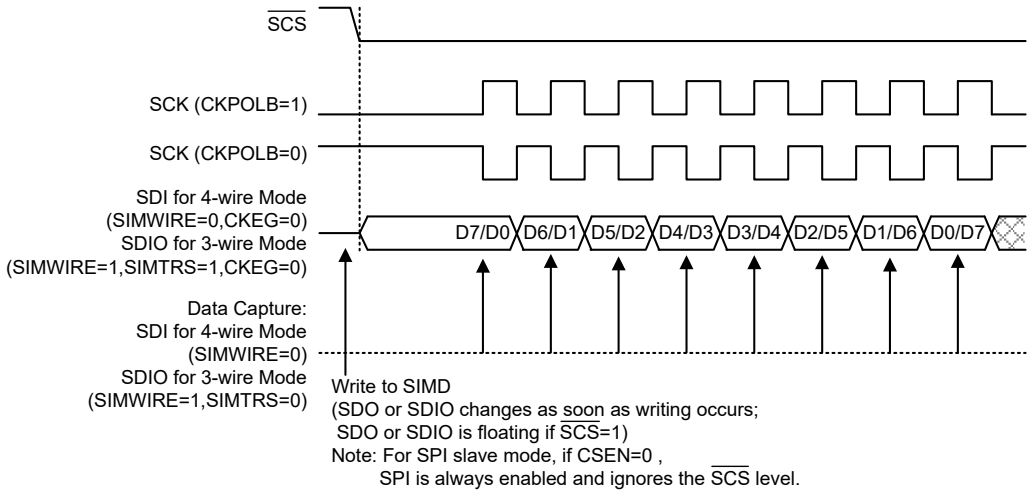
The SPI Master mode will continue to function if the SPI clock is running.



SPI Master Mode Timing

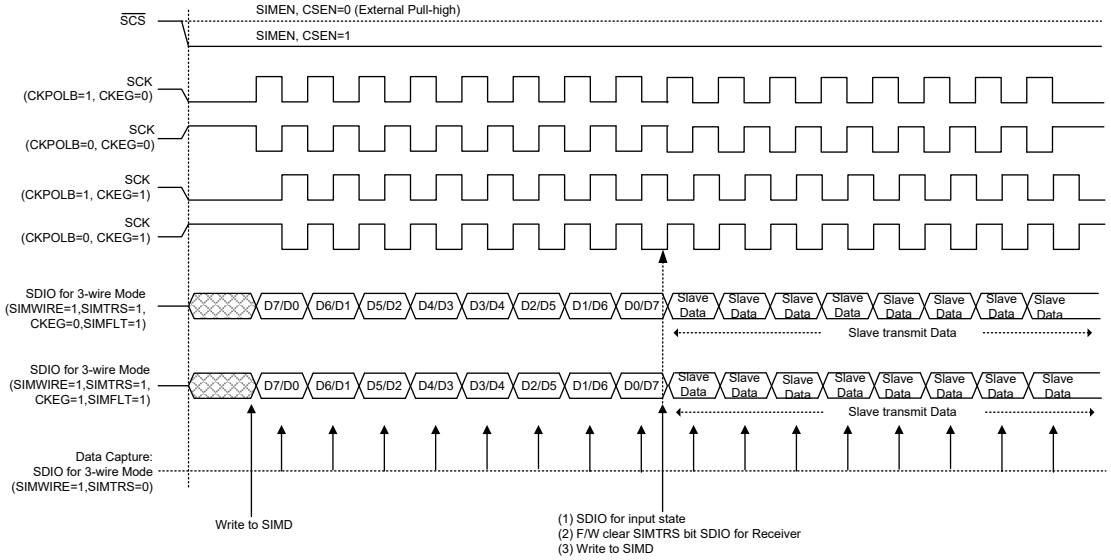


SPI Slave Mode Timing – CKEG=0

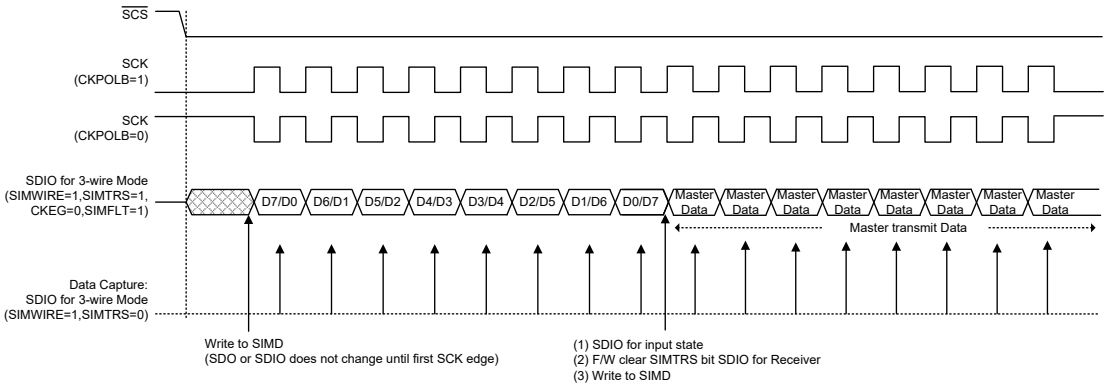


SPI Slave Mode Timing – CKEG=1

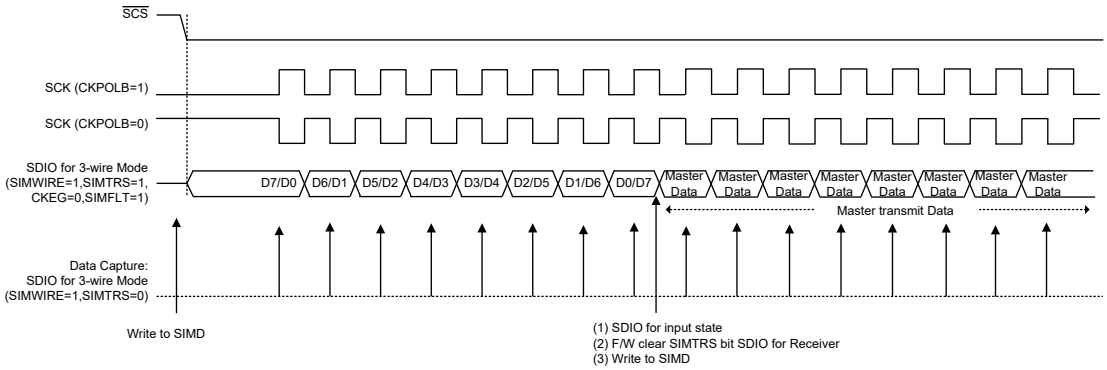
For the SPI 3-wire Mode, when the SIMFLT bit =1, the SDIO transmit status timing is shown below:



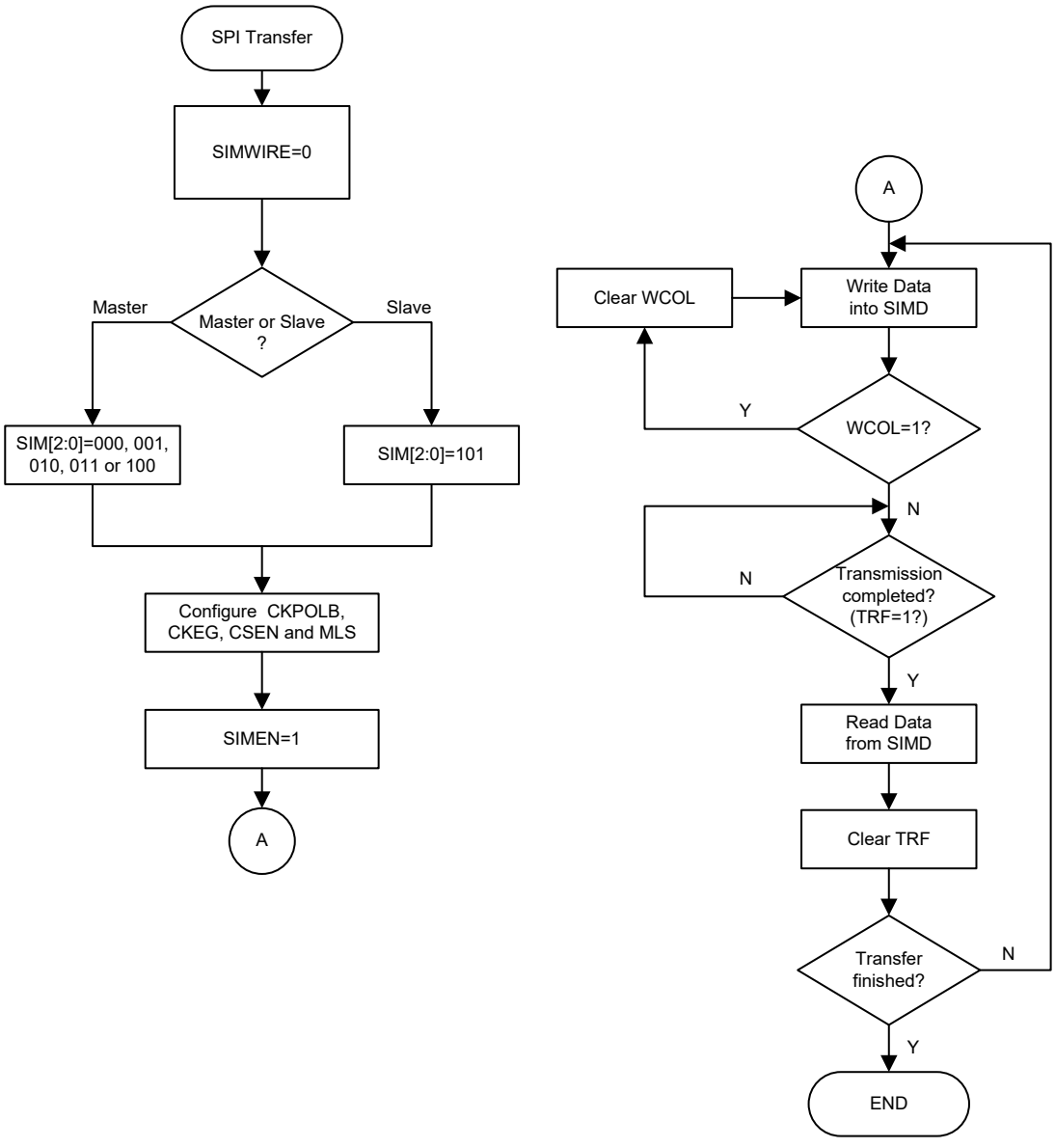
SPI Master Mode Timing



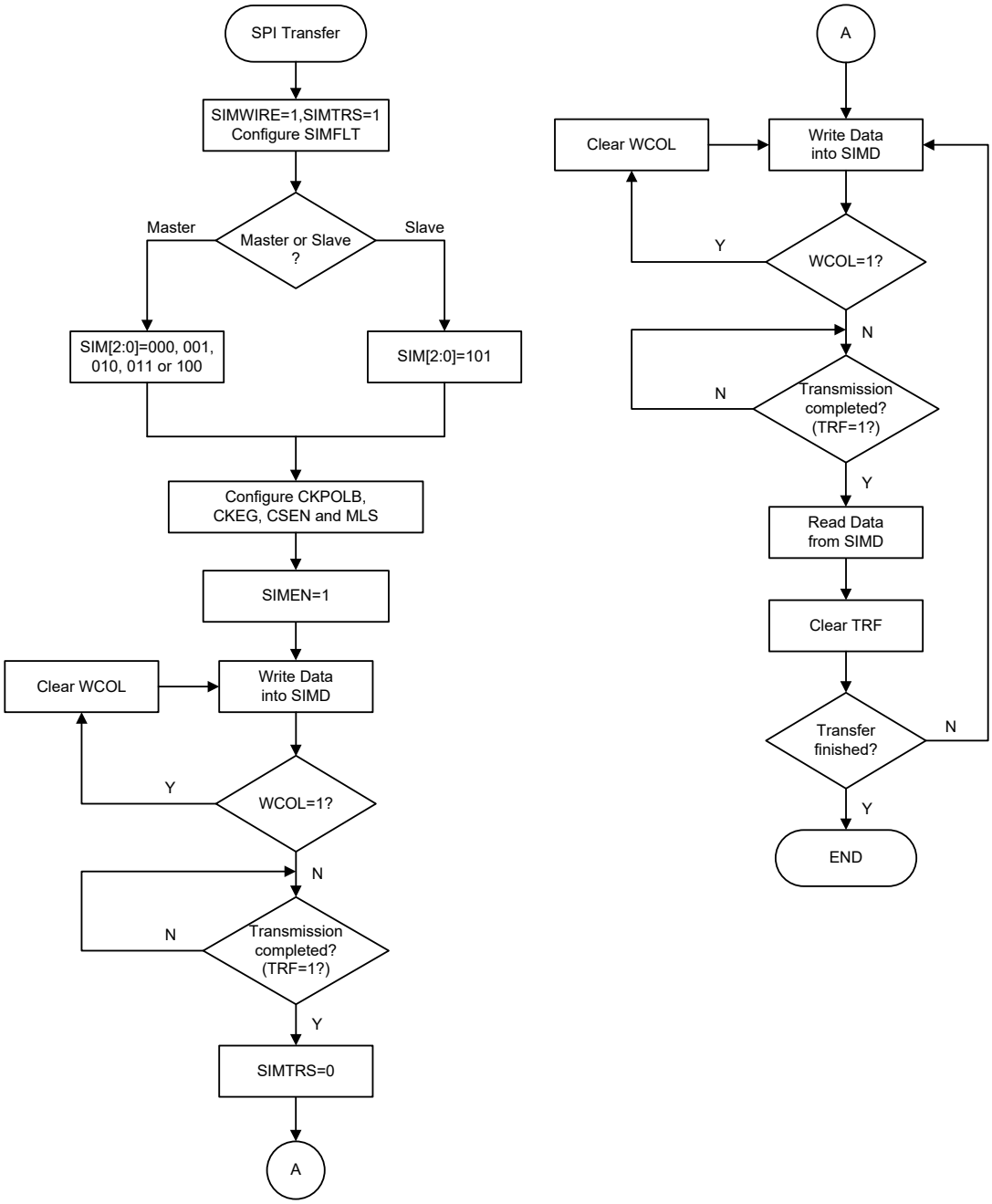
SPI Slave Mode Timing – CKEG=0



SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flow Chart – 4-wire Mode



SPI Transfer Control Flowchart – 3-wire Mode

SPI Bus Enable/Disable

To enable the SPI bus, set CSEN=1 and $\overline{SCS}$=0, then wait for data to be written into the SIMD (TXRX buffer) register. For the Master Mode, after data has been written to the SIMD (TXRX buffer) register, then transmission or reception will start automatically for 4-wire mode and transmission or reception will start selected by the SIMTRS bit in the SIMC0 register for 3-wire mode. When all the data has been transferred, the TRF bit should be set. For the Slave Mode, when clock pulses are received on SCK, data in the TXRX buffer will be shifted out from the output pin or data on SDI will be shifted in when in the receiver mode.

When the SPI bus is disabled, the SCK, SDIO, SDO and $\overline{SCS}$ can become I/O pins or other pin-shared functions using the corresponding pin-shared control bits.

SPI Operation Steps

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

The CSEN bit in the SIMC2 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{SCS}$ line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the $\overline{SCS}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the input line in a floating condition and the output line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and the $\overline{SCS}$, SDIO, SDO and SCK will all become I/O pins or the other functions using the corresponding pin-shared control bits. 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 SIMD 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

  Setup the SPIMWIRE bit in the SIMC0 control register to select either the 3-wire connection mode or 4-wire connection mode. Setup the SIMFLT and SIMTRS bits if the SIMWIRE is high. Select the SPI Master mode and clock source using the SIM2~SIM0 bits in the SIMC0 control register.

- Step 2

  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave devices.

- Step 3

  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.

- Step 4

  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then use the SCK and SDO or SDIO lines to output the data. After this, go to step 5.

  For read operations: the data transferred in on the SDIO line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.

- Step 5

  Check the WCOL 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 TRF bit or wait for an SIM SPI serial bus interrupt.

- Step 7

  Clear the SIMTRS bit if the SIMWIRE bit is high. Read data from the SIMD register.

- Step 8

  Clear TRF.

- Step 9

  Go to step 4.

Slave Mode

- Step 1

  Setup the SIMWIRE bit in the SIMC0 control register to select either the 3-wire connection mode or 4-wire connection mode. Setup the SIMFLT and SIMTRS bits if the SIMWIRE is high. Select the SPI Slave mode using the SIM2~SIM0 bits in the SIMC0 control register

- Step 2

  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.

- Step 3

  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.

- Step 4

  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and $\overline{SCS}$ signal. After this, go to step 5.

  For read operations: the data transferred in on the SDIO line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.

- Step 5

  Check the WCOL 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 TRF bit or wait for an SIM SPI serial bus interrupt.

- Step 7

  Clear the SIMTRS bit if the SIMWIRE bit is high. Read data from the SIMD register.

- Step 8

  Clear TRF.

- Step 9

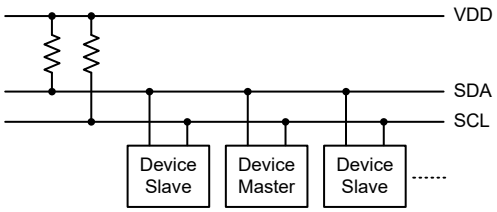
  Go to step 4.

Error Detection

The WCOL bit in the SIMC2 register is provided to indicate errors during data transfer. The bit is set by the SPI serial Interface but must be cleared by the application program. This bit indicates that a data collision has occurred which happens if a write to the SIMD register takes place during a data transfer operation and will prevent the write operation from continuing.

I²C Interface

The I²C 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.

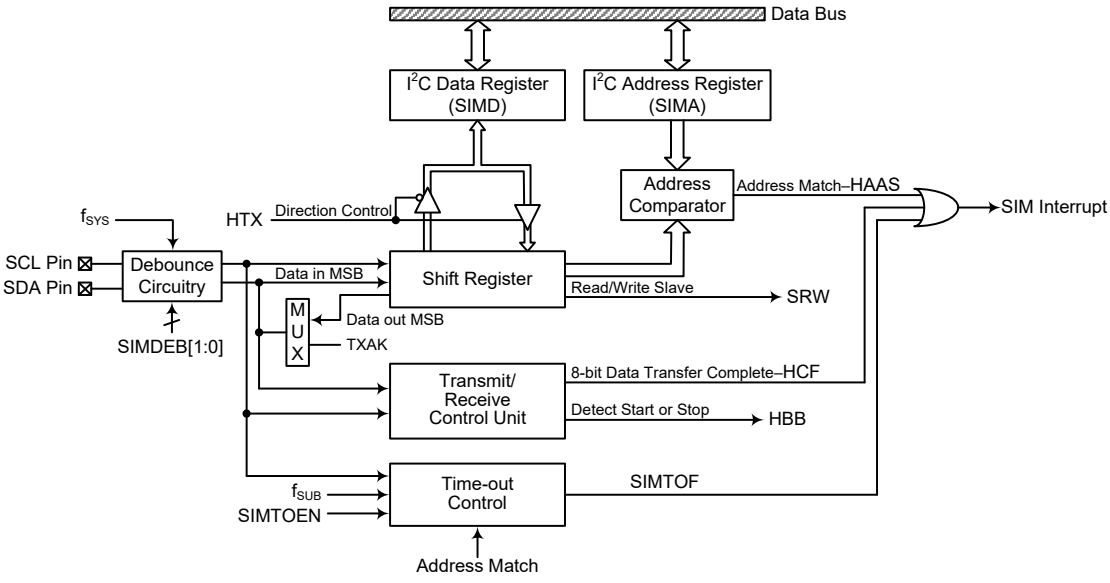


I²C Master/Slave Bus Connection

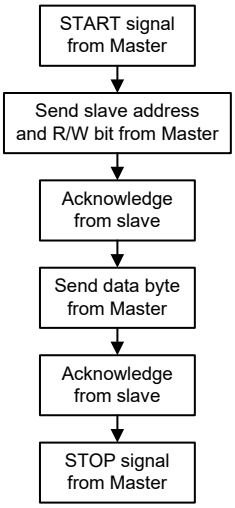
I²C Interface Operation

The I²C 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 I²C bus is identified by a unique address which will be transmitted and received on the I²C bus.

When two devices communicate with each other on the bidirectional I²C 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 the device, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode. The pull-high control function pin-shared with SCL/SDA pin is still applicable even if I²C device is activated and the related internal pull-high function could be controlled by its corresponding pull-high control register.



I²C Interface Block Diagram



I²C Interface Operation

The SIMDEB1 and SIMDEB0 bits determine the debounce time of the I²C interface. This uses the system 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. To achieve the required I²C data transfer speed, there exists a relationship between the system clock, f_{SYS}, and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

I²C Debounce Time Selection	I²C Standard Mode (100kHz)	I²C Fast Mode (400kHz)
No Debounce	f_{SYS} > 2MHz	f_{SYS} > 4MHz
2 system clock debounce	f_{SYS} > 4MHz	f_{SYS} > 8MHz
4 system clock debounce	f_{SYS} > 4MHz	f_{SYS} > 8MHz

I²C Minimum f_{SYS} Frequency Requirement

I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMTOC, one address register SIMA and one data register, SIMD.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMA	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C Register List

I²C Data Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C functions. Before the device writes data to the I²C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I²C bus, the device can read it from the SIMD register. Any transmission or reception of data from the I²C 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

Bit 7~0 **D7~D0**: SIM data register bit 7 ~ bit 0

I²C Address Register

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 implemented.

When a master device, which is connected to the I²C 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.

• **SIMA Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~1 **SIMA6~SIMA0**: I²C slave address
SIMA6~SIMA0 is the I²C slave address bit 6~bit 0.

Bit 0 **D0**: Reserved bit, can be read or written

I²C Control Registers

There are three control registers for the I²C interface, SIMC0, SIMC1 and SIMTOC. The register SIMC0 is used to control the enable/disable function and to select the I²C slave mode and debounce time. The SIMC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, SIMTOC, is used to control the I²C time-out function and is described in the corresponding section.

• **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
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~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_{SUB}
100: SPI master mode; SPI clock is Timer/Event Counter overflow frequency/2
101: SPI slave mode
110: I²C slave mode
111: Non SIM function

These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C 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 and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
1x: 4 system clock debounce
These bits are used to select the I²C debounce time when the SIM is configured as the I²C interface function by setting the SIM2~SIM0 bits to "110".

Bit 1 **SIMEN**: SIM Enable 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 SDIO, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function 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 I²C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C 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 I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SIM SPI Incomplete Flag
The SIMICF bit is only used in the SPI mode and the detailed definition is described in the SPI section.

- **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**: I²C 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 completion 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. Below is an example of the flow of a two-byte I²C data transfer. First, I²C slave device receives a start signal from I²C master and then HCF bit is automatically cleared to zero. Second, I²C slave device finishes receiving the 1st data byte and then HCF bit is automatically set high. Third, user read the 1st data byte from SIMD register by the application program and then HCF bit is automatically cleared to zero. Fourth, I²C slave device finishes receiving the 2nd data byte and then HCF bit is automatically set to one and so on. Finally, I²C slave device receives a stop signal from I²C master and then HCF bit is automatically set high.

Bit 6 **HAAS**: I²C Bus data transfer completion 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**: I²C Bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy
The HBB flag is the I²C busy flag. This flag will be "1" when the I²C bus is busy which will occur when a START signal is detected. The flag will be cleared to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX**: I²C slave device transmitter/receiver selection
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK**: I²C bus transmit acknowledge flag
0: Slave sends acknowledge flag
1: Slave does not send acknowledge flag
The TXAK flag is the transmit acknowledge flag. After the slave device has received 8 bits of data, this flag will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set the TXAK bit to "0" before further data is received.

Bit 2 **SRW**: I²C 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 I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C 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**: I²C Address Match Wake-Up control
0: Disable
1: Enable – must be cleared by the application program after wake-up
This bit should be set to 1 to enable the I²C address match wake-up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake-up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

Bit 0 **RXAK**: I²C bus receive acknowledge flag
0: Slave receives acknowledge flag
1: Slave does 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 is 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 I²C Bus.

I²C Bus Communication

Communication on the I²C 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 I²C 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 SIM interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from either an address match or the completion of an 8-bit data transfer or the I²C bus time-out occurrence. 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 I²C bus, the microcontroller must initialise the bus; the following are steps to achieve this:

- Step 1

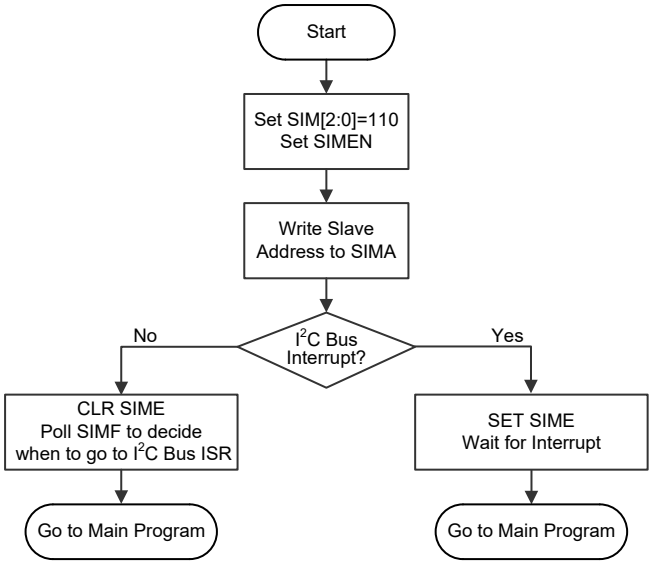
  Set the SIM2~SIM0 bits to "110" and SIMEN bit to "1" in the SIMC0 register to enable the I²C bus.

- Step 2

  Write the slave address of the device to the I²C bus address register SIMA.

- Step 3

  Set the SIME interrupt enable bit of the interrupt control register to enable the SIM 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.

I²C 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 SIM 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 SIM I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits should be examined to see whether the interrupt source has come from either a matching slave address, the completion of a data byte transfer or the I²C bus time-out occurrence. 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.

I²C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the master device wishes to read data from the I²C bus or write data to the I²C 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 I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the SRW flag is "0" then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

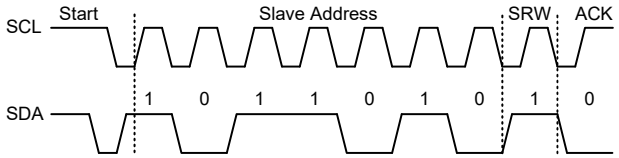
I²C Bus Slave Address Acknowledge Signal

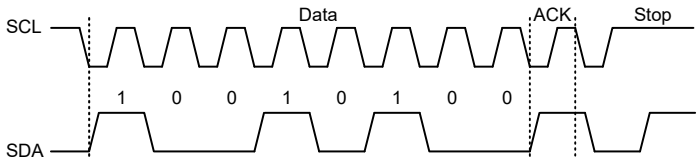
After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal 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 cleared to "0".

I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bit 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 I²C 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.



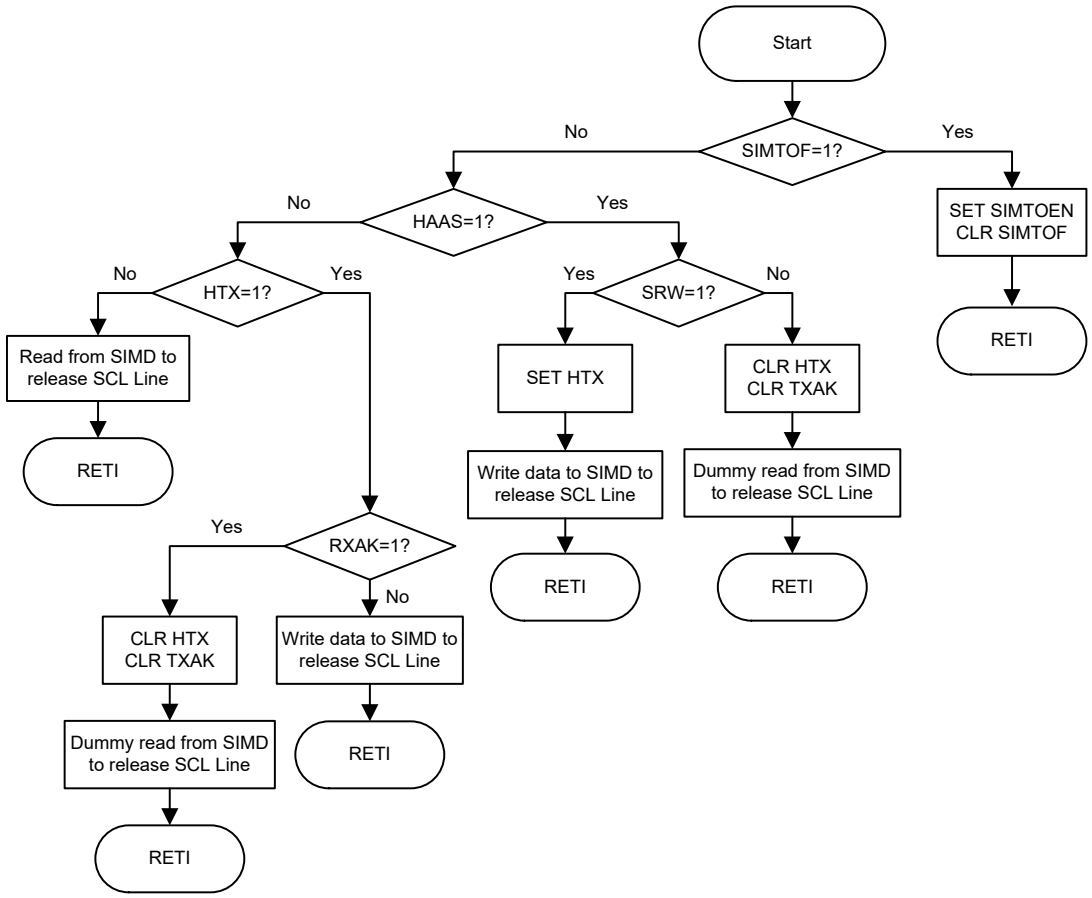


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	·····	S	SA	SR	M	D	A	D	A	·····	P

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 implement a dummy read from the SIMD register to release the SCL line.

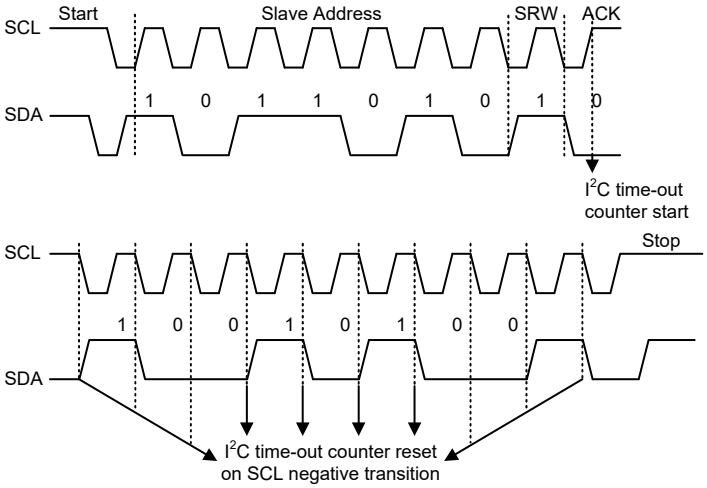
I²C Communication Timing Diagram



I²C Bus ISR Flow Chart

I²C Time-out Control

In order to reduce the I²C lockup problem due to reception of erroneous clock sources, a time-out function is provided. If the clock source connected to the I²C bus is not received for a while, then the I²C circuitry and registers will be reset after a certain time-out period. The time-out counter starts to count on an I²C bus "START" & "address match" condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out period specified by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.



I²C Time-out

When an I²C time-out counter overflow occurs, the counter will stop and the SIMTOEN bit will be cleared to zero and the SIMTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the SIM interrupt vector. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

Registers	After I²C Time-out
SIMD, SIMA, SIMC0	No change
SIMC1	Reset to POR condition

I²C Registers after Time-out

The SIMTOF flag can be cleared by the application program. There are 64 time-out period selections which can be selected using the SIMTOS5~SIMTOS0 bits in the SIMTOC register. The time-out duration is calculated by the formula: ((1~64) × (32/f_{SUB})). This gives a time-out period which ranges from about 1ms to 64ms.

- **SIMTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
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

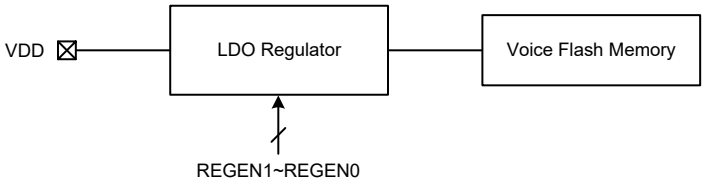
Bit 7 **SIMTOEN**: SIM I²C Time-out control
0: Disable
1: Enable

Bit 6 **SIMTOF**: SIM I²C Time-out flag
0: No time-out occurred
1: Time-out occurred

Bit 5~0 **SIMTOS5~SIMTOS0**: SIM I²C Time-out period selection
I²C Time-out clock source is f_{SUB}/32.
I²C Time-out period is equal to (SIMTOS[5:0]+1) × (32/f_{SUB})

Voltage Regulator – LDO

The devices include a voltage regulator, LDO, which provides power to Voice Flash Memory. The REGC register controls the regulator module in two modes. In the disable mode, the LDO will be switched off and the LDO output will be in a pull-low state. In the bypass mode, the LDO will also be switched off but the VDD pin voltage will be directly passed to provide power the Voice Flash Memory.



LDO Regulator Block Diagram

• **REGC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	REGEN1	REGEN0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

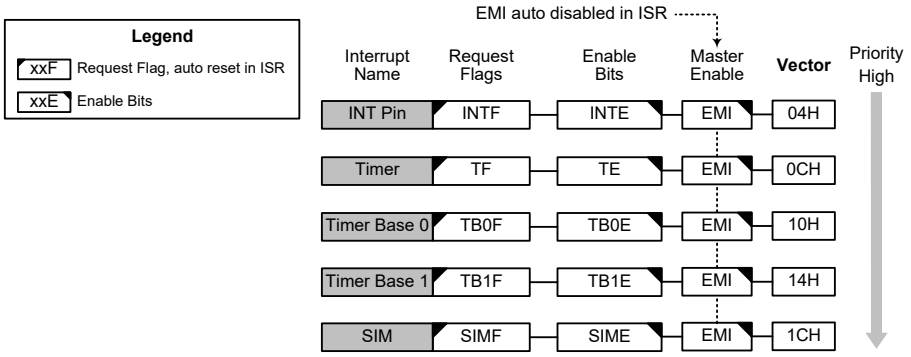
Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **REGEN1~REGEN0**: Regulator on/off control
00: Regulator off, disable mode, the LDO output is pulled low
01: Regulator on, the Voice Flash Memory power is equal to 3.0V
10: Regulator on, the Voice Flash Memory power is equal to 3.0V
11: Regulator off, bypass mode, the Voice Flash Memory power is equal to V_{DD}

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter or Time Base requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The devices contain an external interrupt and several internal interrupt functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions including the Timer/Event Counter, SIM and Time Bases, etc.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. All the interrupt sources have their own individual vector.



Interrupt Structure

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory. The registers fall into two categories. The first is the INTC0~INTC1 registers which setup the primary interrupts, the second is the INTEG register which sets the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INT pin	INTE	INTF	—
Timer/Event Counter	TE	TF	—
Time Base	TBnE	TBnF	n=0~1
SIM Interface	SIME	SIMF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	TF	D5	INTF	TE	D2	INTE	EMI
INTC1	SIMF	D6	TB1F	TB0F	SIME	D2	TB1E	TB0E

Interrupt Register List

- **INTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INTS1	INTS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **INTS1~INTS0**: Interrupt edge control for INT pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TF	D5	INTF	TE	D2	INTE	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 **TF**: Timer/Event Counter interrupt request flag
0: No request
1: Interrupt request

Bit 5 **D5**: Reserved, cannot be changed

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

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

Bit 2 **D2**: Reserved, cannot be changed

Bit 1 **INTE**: INT interrupt control
0: Disable
1: Enable

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

• **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMF	D6	TB1F	TB0F	SIME	D2	TB1E	TB0E
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 **SIMF**: SIM interrupt request flag
0: No request
1: Interrupt request

Bit 6 **D6**: Reserved, cannot be changed

Bit 5 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **SIME**: SIM interrupt control
0: Disable
1: Enable

Bit 2 **D2**: Reserved, cannot be changed

Bit 1 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

Bit 0 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

Interrupt Operation

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

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a "JMP" which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a "RETI", which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

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

External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, is set, which will occur when a transition, whose type is chosen by the edge selection bits, appears on the external interrupt pin. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the external interrupt enable bit, INTE, must first be set. Additionally, the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if its external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that the pull-high resistor selection on the external interrupt pin will remain valid even if the pin is used as an external interrupt input.

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

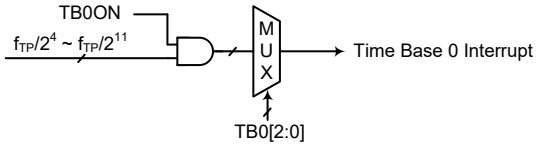
Timer/Event Counter Interrupt

An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TF, is set, which occurs when the Timer/Event Counter overflows. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Timer/Event Counter Interrupt enable bit, TE, must first be set. When the interrupt is enabled, the stack is not full and the Timer/Event Counter overflows, a subroutine call to its interrupt vector will take place. When the interrupt is serviced, the Timer/Event Counter Interrupt flag, TF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

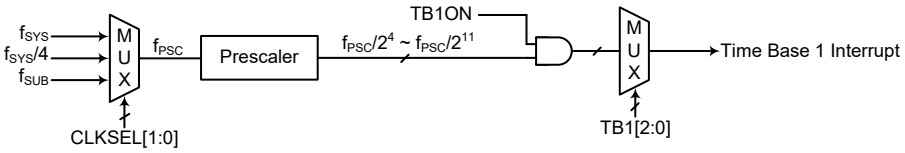
Time Base Interrupts

The function of the Time Base Interrupts are to provide regular time signals in the form of an internal interrupt. It is controlled by the overflow signals from the timer function. When this happens its interrupt request flag TBnF will be set. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and Time Base enable bit, TBnE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to its vector location will take place. When the interrupt is serviced, the interrupt request flag TBnF will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupts are to provide an interrupt signal at fixed time periods. For the Time Base 0, its clock source, f_{TP}, originates from the internal clock source f_{SYS} or f_{SUB}, selected by setting the TS bit and then passes through a divider, which has been described in the Timer/Event Counter chapter. The divided clock source for the Timer Base function is selected by programming the appropriate bits in the TB0C register to obtain longer interrupt periods whose value ranges. For the Time Base 1, its clock source, f_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB1C register to obtain longer interrupt periods whose value ranges. The divided clock source for the Timer Base function is selected by programming the CLKSEL1~CLKSEL0 bits in the TB1C register to obtain longer interrupt periods whose value ranges.



Time Base 0 Interrupt



Time Base 1 Interrupt

- **TB0C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB0ON**: Time Base control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Time Base 0 time-out period selection
000: $2^4/f_{TP}$
001: $2^5/f_{TP}$
010: $2^6/f_{TP}$
011: $2^7/f_{TP}$
100: $2^8/f_{TP}$
101: $2^9/f_{TP}$
110: $2^{10}/f_{TP}$
111: $2^{11}/f_{TP}$

• **TB1C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	—	CLKSEL1	CLKSEL0	—	TB12	TB11	TB10
R/W	R/W	—	R/W	R/W	—	R/W	R/W	R/W
POR	0	—	0	0	—	0	0	0

Bit 7 **TB1ON**: Time Base 1 control
0: Disable
1: Enable

Bit 6 Unimplemented, read as "0"

Bit 5~4 **CLKSEL1~CLKSEL0**: Prescaler clock source selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

Bit 3 Unimplemented, read as "0"

Bit 2~0 **TB12~TB10**: Time Base 1 time-out period selection
000: $2^4/f_{PSC}$
001: $2^5/f_{PSC}$
010: $2^6/f_{PSC}$
011: $2^7/f_{PSC}$
100: $2^8/f_{PSC}$
101: $2^9/f_{PSC}$
110: $2^{10}/f_{PSC}$
111: $2^{11}/f_{PSC}$

Serial Interface Module Interrupt

The Serial Interface Module Interrupt is also known as the SIM interrupt. A SIM Interrupt request will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface, an I^2C slave address match or I^2C bus time-out occurs. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and SIM Interrupt enable bit, SIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the corresponding SIM Interrupt vector, will take place. When the interrupt is serviced, the SIMF flag will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

Interrupt Wake-up Function

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

Programming Considerations

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

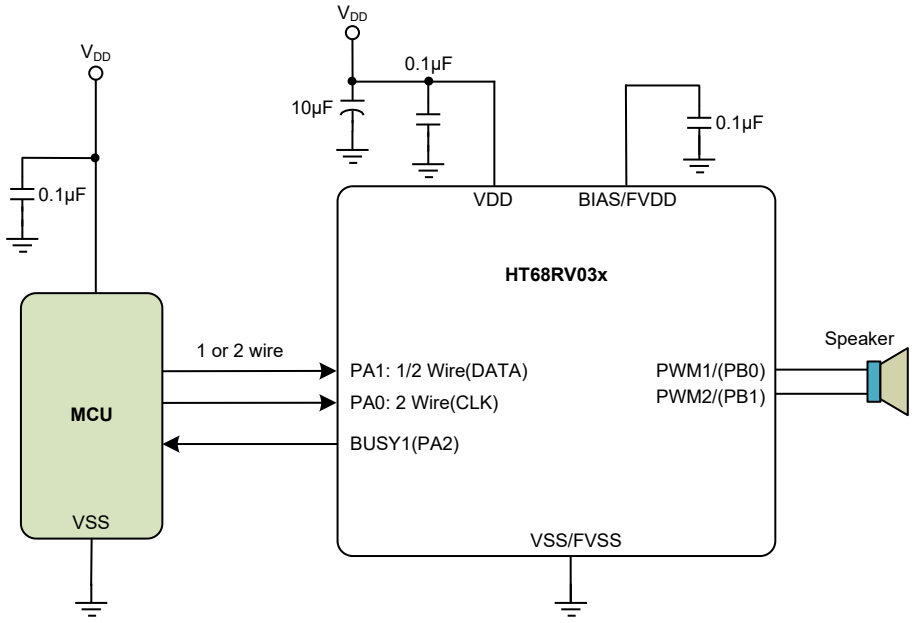
It is recommended that programs do not use the "CALL" instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged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

Application Circuits



Note: The capacitor must be close to HT68RV03x.

Peripheral IC Mode

The devices provide the Peripheral mode for editing voice using Holtek Voice Workshop. They also have five interface control modes, implementing both voice data and control interface program programming, as described below.

Features for Peripheral IC Mode

- Operating voltage (V_{DD}): HT68RV032/33/34/35/36: 2.3V~5.5V; HT68RV034L: 2.0V~5.5V
- Standby current about 9.9μA @ V_{DD}=3.3V, 14.5μA @ V_{DD}=5V
- Low Voltage Reset function – LVR
- Five controller mode: One-wire, Two-wire, SPI, I^2C and Direct mode
- PWM Voice output is greater than 0.65W, 1kHz sine wave/THD=10% @ R_L=8Ω; V_{DD}=5V

  Note: When setting the volume for direct drive speaker, the PWM output current and speaker wattage should be taken into consideration, to avoid damage to the speaker
- PWM Voice for Delta Sigma PWM
- Voice Coding modes: ADPCM4, ADPCM5, uPCM8, PCM12 Support Busy State output function for One-wire/Two-wire mode/SPI/I^2C/direct mode
- Busy state output function
- Voice sampling rate supports up to 48kHz mono wav input
- Voice and Sentence arrangement
- Maximum voice time for ADPCM4, HT68RV032: 85s (6kHz); HT68RV033: 170s (6kHz); HT68RV034/HT68RV034L: 340s (6kHz); HT68RV035: 680s (6kHz); HT68RV036: 1360s (6kHz)
- Holtek Voice Workshop tool to aid development
- Package type: 8-pin SOP

Pin Assignment for Peripheral IC Mode

HT68RV032/HT68RV033/HT68RV034
HT68RV034L/HT68RV035/HT68RV036
8 SOP-A

Note: The BUSY1 pin is used in One-wire mode and Two-wire mode; The BUSY2 pin is used in I^2C mode.

Pin Descriptions for Peripheral IC Mode

Pin Name	Function	Type	Playing State	Standby State	Description
CLK/KEY1/ SCL/SCK/PA0	CLK	I	Input pull-high	Input pull-high	Two-wire interface CLK signal
	KEY1	I	Input pull-high	Input pull-high	KEY1 input for Direct mode – active low
	SCL	I	Input pull-high	Input pull-high	I^2C interface SCL signal
	SCK	I	Input pull-high	Input pull-high	SPI interface SCK signal
	PA0	I/O	—	—	General purpose I/O
BUSY1/KEY2/ SDA/SDIO/ PA2	BUSY1	I/O	Output mode	Input pull-high	Output low when voice playing – One-wire or Two-wire
	KEY2	I	Input pull-high	Input pull-high	KEY2 input for Direct mode – active low
	SDA	I	Input pull-high	Input pull-high	I^2C interface SDA signal
	SDIO	I/O	Input pull-high	Input pull-high	SPI interface SDIO signal
	PA2	I/O	—	—	General purpose I/O
DATA/KEY3/ $\overline{SCS}$/BUSY2/ PA1	DATA	I	Input pull-high	Input pull-high	One-wire or Two-wire interface DATA signal
	KEY3	I	Input pull-high	Input pull-high	KEY3 input for Direct mode – active low
	$\overline{SCS}$	I	Input pull-high	Input pull-high	SPI interface $\overline{SCS}$ signal
	BUSY2	I/O	Output mode	Input pull-high	Output low when voice playing – I^2C mode
	PA1	I/O	—	—	General purpose I/O
PWM1/PB0	PWM1	O	Output mode	Output high	Voice PWM output 1
	PB0	I/O	—	—	General purpose I/O
PWM2/PB1	PWM2	O	Output mode	Output high	Voice PWM output 2
	PB1	I/O	—	—	General purpose I/O
BIAS	BIAS	PWR	—	Pull-low	Positive bias voltage
VDD	VDD	PWR	—	—	Positive power supply
VSS	VSS	PWR	—	—	Negative power supply, ground

Legend: I: Input; O: Output; PWR: Power

Control Modes

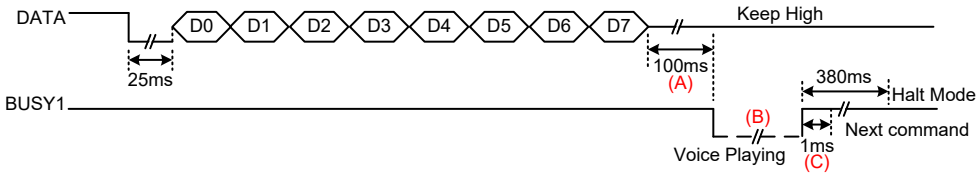
The devices have various control modes, which are the key controlled Direct mode and serial interface controlled One-wire, Two-wire, SPI and I^2C mode. The control mode can be set by users according to their requirements when arranging the voices in Holtek Voice Workshop. The Direct mode provides six keys.

For the detailed operations, refer to AN0684: HT68RV03x Voice Application and Library Description.

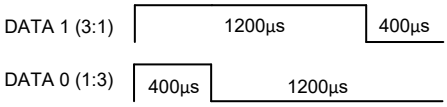
Serial Interface Control Modes

The devices support five serial interface control modes, the One-wire mode, Two-wire mode 1, Two-wire mode 2, SPI and I²C mode. Their control timing and commands are described below.

One-wire Mode



After DATA is pulled low for a time range of 20ms~30ms (25ms recommended), 8-bit of data will be sent, LSB first and MSB last. The value of each bit is expressed in terms of the ratio of high voltage to low voltage.



When the high to low ratio is 3:1, this indicates a value of 1. It is recommended to use the values 1200µs:400µs. When the high to low ratio is 1:3, this indicates a value of 0. It is recommended to use the values 400µs:1200µs.

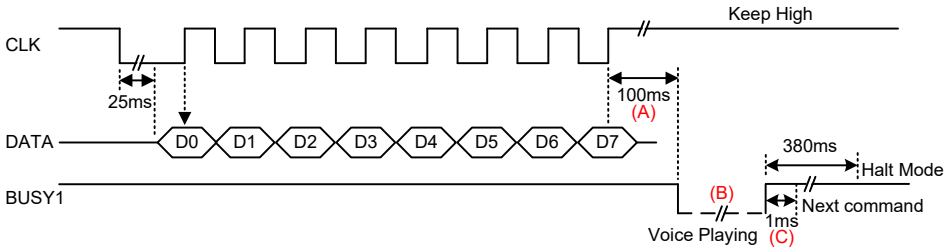
After command end the DATA line needs to be kept at a high level. Maximum 100ms from DATA end to BUSY1 pulled low. Maximum 380ms from Voice end into Halt mode. At least 1ms from BUSY1 end to the next command.

Value range: 80µs:240µs ~ 600µs:1800µs. Note that it is recommended to use 3:1 and 1:3 voltage ratios to ensure communication stability.

Note: In area (A) and (C), no commands can be issued.
In area (B), commands can be issued and the commands are complete.

Two-wire Mode

- Two-wire Mode 1

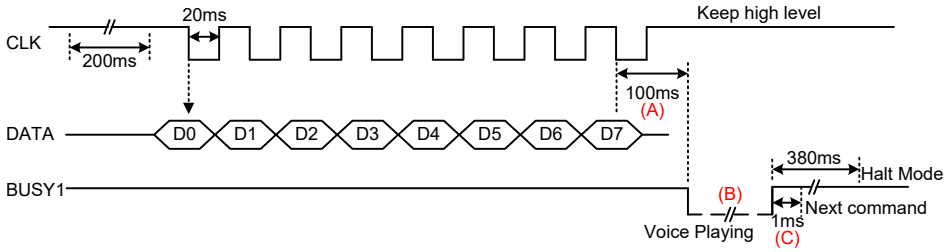


After CLK is pulled low for 20ms~30ms (25ms is recommended), the device will fetch the DATA on the CLK rising edge and 8 bits of data will be sent, LSB first and MSB last. The CLK period range is 200µs~5ms, a value of 600µs is recommended. The low pulse range is 100µs~2.5ms.

After command end the CLK line needs to be kept at a high level. Maximum 100ms from CLK end to BUSY1 pulled low. Maximum 380ms from Voice end into Halt mode. At least 1ms from BUSY1 end to the next command.

Note: In area (A) and (C), no commands can be issued.
In area (B), commands can be issued and the commands are complete.

- Two-wire Mode 2

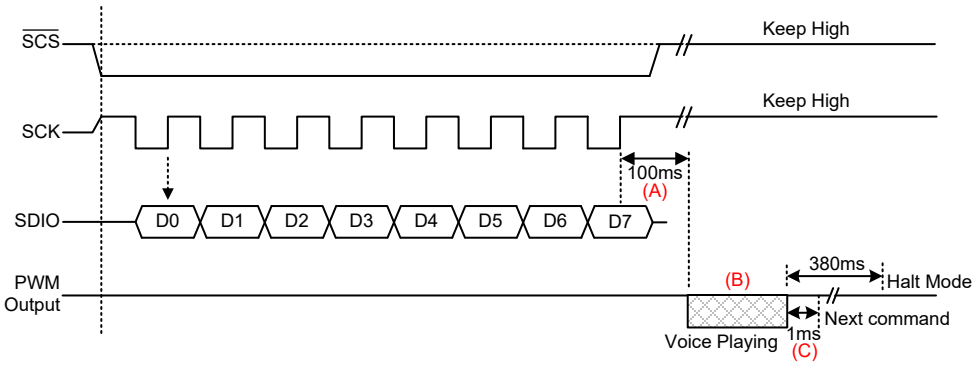


After CLK is pulled high for 200ms~400ms (200ms is recommended), CLK will fetch the DATA on the falling edge and 8 bits of data will be sent, LSB first and MSB last. The first low level of the CLK should be kept for more than 380ms, and the other CLK period range is 8ms~30ms, a value of 20ms is recommended.

After command end the CLK line needs to be kept at a high level. Maximum 100ms from CLK end to BUSY1 pulled low. Maximum 20ms from Voice end into Halt mode. At least 1ms from BUSY1 end to the next command.

Note: In area (A) and (C), no commands can be issued.
In area (B), commands can be issued and the commands are complete.

SPI Mode

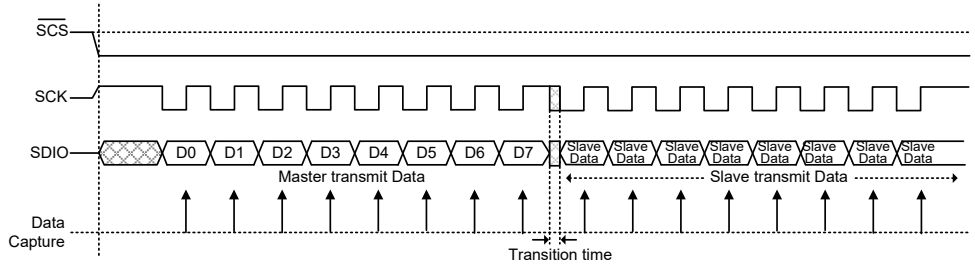


After $\overline{SCS}$ is pulled low, $\overline{SCS}$ will fetch the SDIO on the rising edge and 8 bits of data will be sent, LSB first and MSB last. After command end the SCK and $\overline{SCS}$ lines need to be kept at a high level. Maximum 100ms from SCK end to PWM output. Maximum 380ms from Voice end into Halt mode.

At least 1ms from PWM output return high level to the next command.

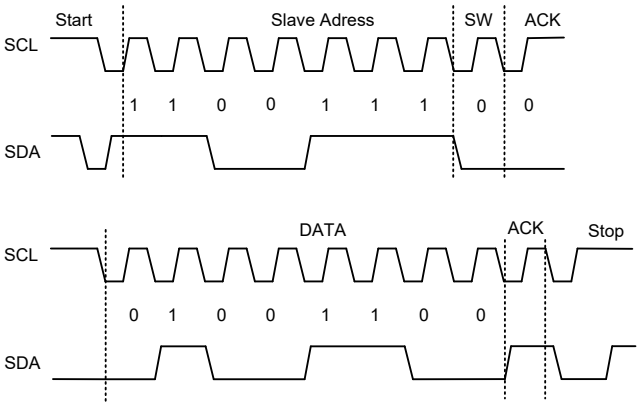
Note: In area (A) and (C), no commands can be issued.
In area (B), commands can be issued and the commands are complete.

The SPI communication rate is up to 10MHz.



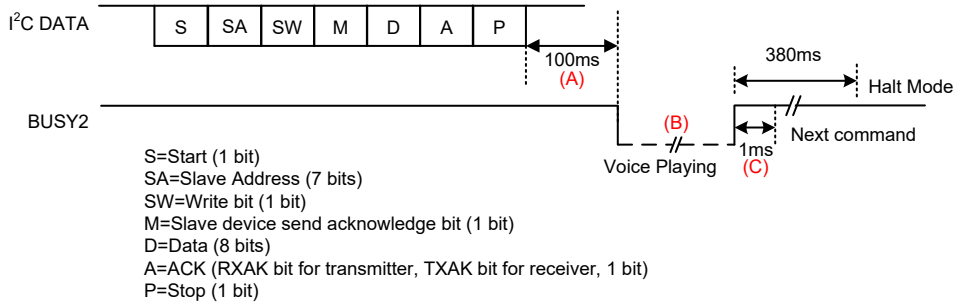
After sending the SPI specific command (F3H), the devices will respond whether it is in a Busy status (50H) or Idle status (56H) after a transition time (100μs).

I²C Mode



When the SCL is high, SDA changes from high level to low level as the Start. After SCL is low, SCL will fetch the SDA on the rising edge. Send the Slave address(1100111b) and the Write bit(0), MSB first and LSB last. Wait for the Slave to respond ACK, send DATA(8bit)(Command Code) and confirm ACK again, the SCL line needs to be kept at a high level. The SDA changes from low to high to stop the transmission process.

The transmit rate is 400kHz in I²C mode.



After command end the SCL/SDA lines need to be kept at a high level. Maximum 100ms from SCL end to BUSY2 pulled low. Maximum 380ms from Voice end into Halt mode. At least 1ms from BUSY2 return high level to the next command.

Note: In area (A) and (C), no commands can be issued.
In area (B), commands can be issued and the commands are complete.

Control Commands

Command Code (Hexadecimal)	Function	Description
00H~7FH	Play voice	Select the desired voice to play, 00H is voice 0 01H is voice 1 And so on, from 0 to 127, there are 128 voices.
80H~DFH	Play sentence	Select the desired sentence to play, 80H is sentence 0 81H is sentence 1 And so on, from 0 to 95, there are 96 sentences.
E0H	Operating voltage selection	When the operating voltage is 2.3V~3.3V (HT68RV032/33/34/35/36) / 2.0V~3.3V (HT68RV034L), this command is required to adjust the MCU voltage. If this command is not used, the default operating voltage will be 3.4V~5.5V. When applied to situations where the operating voltage varies, this command can be used to make the system operate under optimum conditions.
E1H~EFH	Volume selection	E1 is the minimum volume and EF is the maximum volume. There are 15 levels of volume adjustment.
F0H	Suspend mode	The “HALT” instruction will stop the program execution and turn off the system clock.
F1H	Pause voice/sentence	Pause playing the current voice and sentence.
F2H	Play after pause	Continue playing the paused voice and sentence.
F3H	SPI mode specific command	SPI mode check busy state command.
F4H	Loop playback the current voice/sentence	Loop playback for the current voice and sentence.
F5H~F7H	—	Reserved, cannot be used
F8H	Stop playing the current voice/sentence	Stop playing the current voice and sentence.
F9H	—	Reserved, cannot be used
FAH	Voice sector selection	Select low sector
FBH	Voice sector selection	Select high sector
FCH	Voice block selection	Select block1
FDH	Voice block selection	Select block2
FEH	Voice block selection	Select block3
FFH	Voice block selection	Select block4

- Control Command Supplementary Description

  The Voice playing command fixed as ranges from 00H to 7FH, which can be used together with the Select Voice Block command (FCH/FDH/FEH/FFH) and Select Voice Sector command (FAH/FBH) to select the Voice section to play.

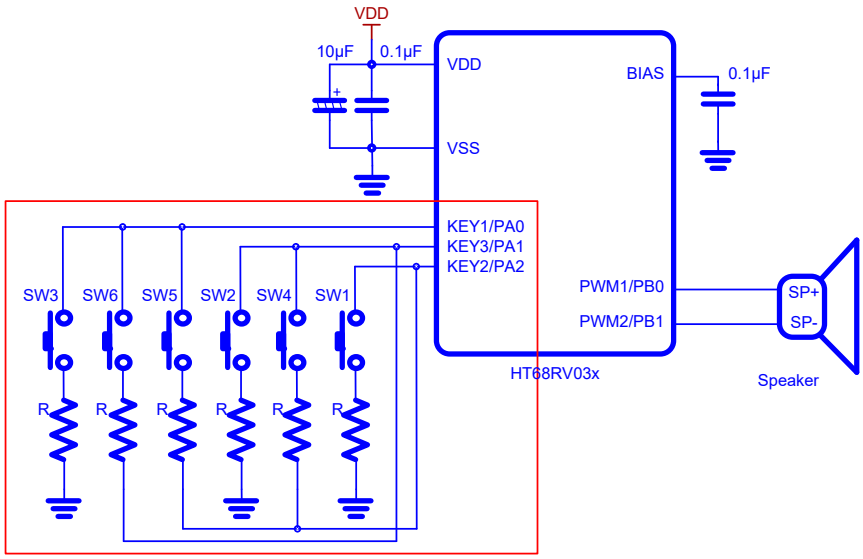
CMD(00H~7FH) (Represents the number of the voice)	Select Voice Block(CMD)	Select Voice Sector(CMD)
0~127	Block1(FCH)(default)	Low(FAH)(default)
128~255	Block1(FCH)	High(FBH)
256~383	Block2(FDH)	Low(FAH)
384~511	Block2(FDH)	High(FBH)
512~639	Block3(FEH)	Low(FAH)
640~767	Block3(FEH)	High(FBH)
768~895	Block4(FFH)	Low(FAH)
896~1023	Block4(FFH)	High(FBH)

Direct Mode

The devices support a key control mode, which can be set according to requirements when arranging voices in the Holtek Voice Workshop. This mode supports six keys, SW1~SW6, which are active low. The key functions are as follows:

Key Pin	Function	Description
SW1	Play/Next	Press again to play the next voice without requiring the current voice to finish.
SW2	Play/Next	Press again to play the next sentence without requiring the current voice to finish.
SW3	Stop	Stop playing the current voice
SW4	Reset	Reset to the first voice.
SW5	Volume Up	Increment volume
SW6	Volume Down	Decrement volume

The key connection is shown in the red box below.

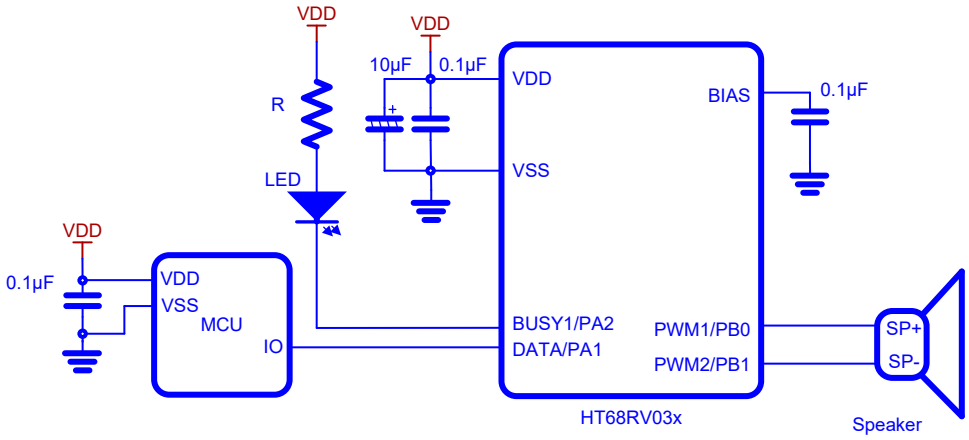


Note: The capacitor must be close to HT68RV03x.

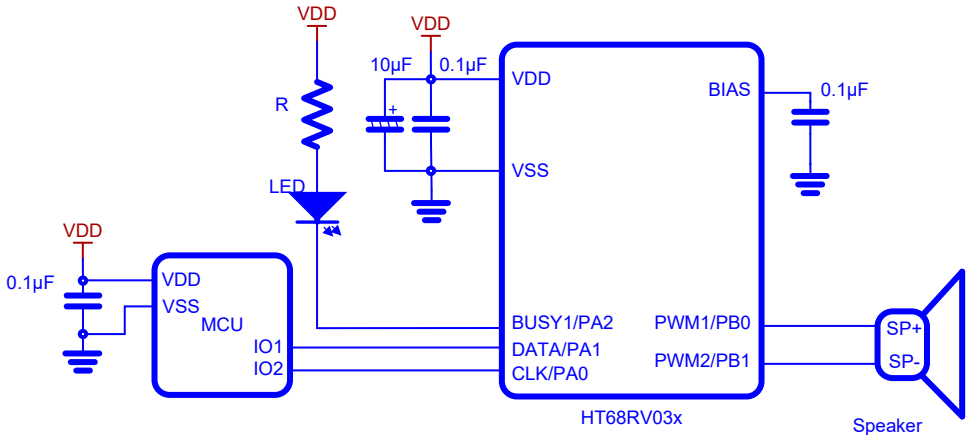
The direct mode volume levels and control mode command code corresponding to the PWDC[3:0] values are shown in the following table.

Direct Mode Volume Level	Control Mode Command Code	PWDC[3:0]
1	E1H	Mute (Ignore the PWDC[3:0]value)
2	E2H	0010B
3	E3H	0011B
4	E4H	0100B
5	E5H	0101B
6	E6H	0110B
7	E7H	0111B
8	E8H	1000B
9	E9H	1001B
10	EAH	1010B
11	EBH	1011B
12	ECH	1100B
13	EDH	1101B
14	EEH	1110B
15	EFH	1111B

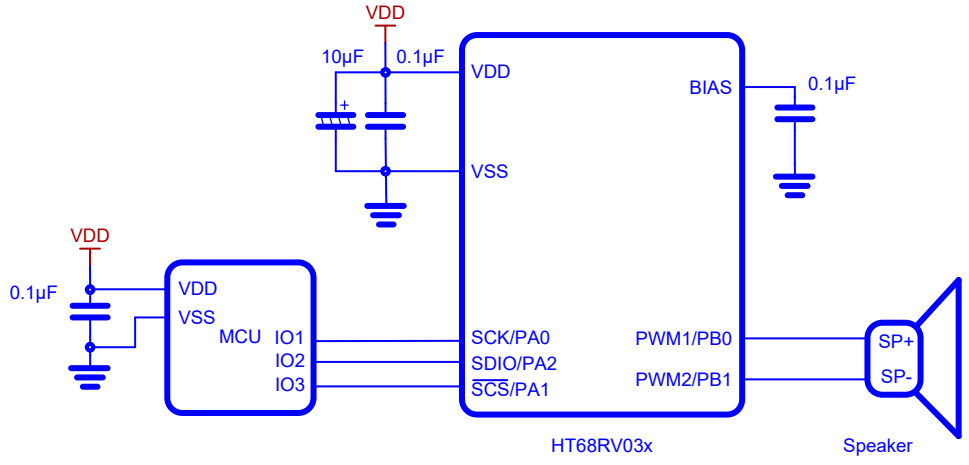
Application Circuits



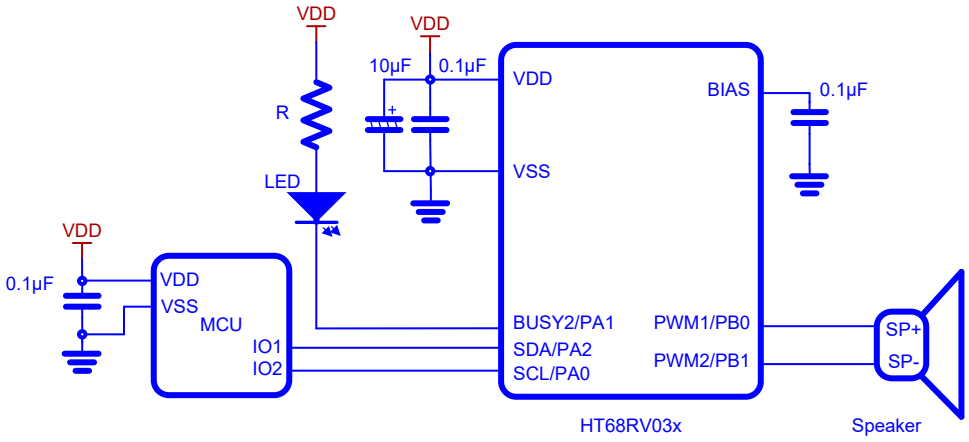
One-wire Mode Application Circuit



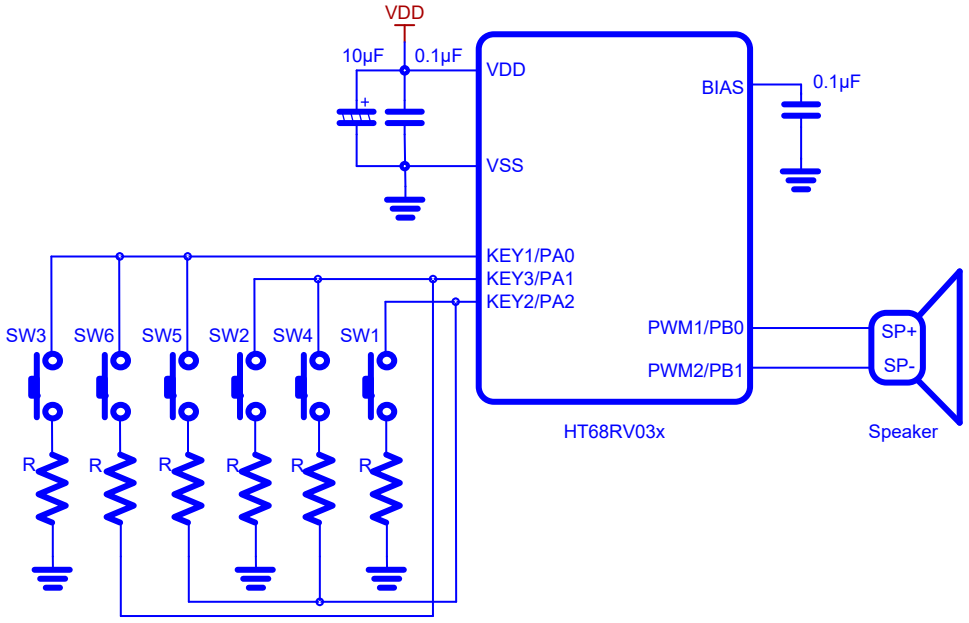
Two-wire Mode Application Circuit



SPI Mode Application Circuit



I²C Mode Application Circuit



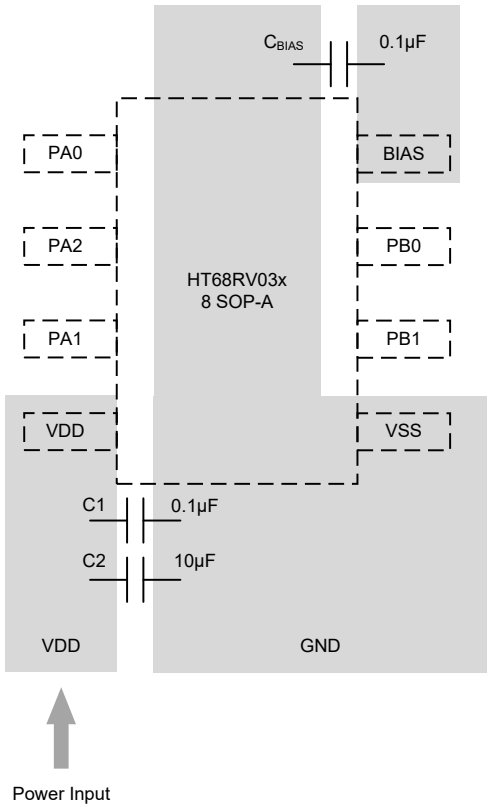
Direct Mode Application Circuit

Note: The capacitor must be close to HT68RV03x.

Layout Recommendations

The recommended Layout methods for the HT68RV03x are as follows:

- The 8-pin SOP Layout is recommended as shown below:



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

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

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch Operation			
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 Operation			
TABRD [m]	Read table (specific page or 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
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.

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

CPL [m] Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − $\overline{C}$

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

SET [m] Set Data Memory

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

Operation $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation $[m].i \leftarrow 1$

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

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

Affected flag(s) None

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

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

Operation ACC ← [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 The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

TABRD [m] Read table (specific page or current page) to TBLH and Data Memory

Description The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no 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) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

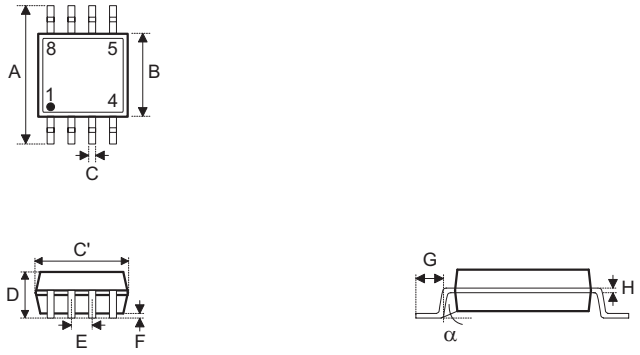
Package Information

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

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

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

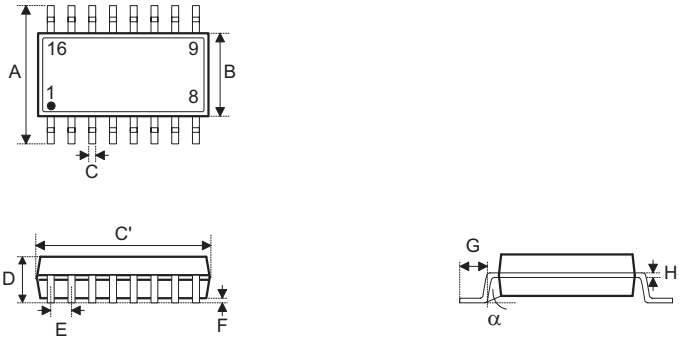
8-pin SOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.012	—	0.020
C′		0.193 BSC	
D	—	—	0.069
E		0.050 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		6.00 BSC	
B		3.90 BSC	
C	0.31	—	0.51
C′		4.90 BSC	
D	—	—	1.75
E		1.27 BSC	
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

16-pin NSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.012	—	0.020
C'		0.390 BSC	
D	—	—	0.069
E		0.050 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		6.00 BSC	
B		3.90 BSC	
C	0.31	—	0.51
C'		9.90 BSC	
D	—	—	1.75
E		1.27 BSC	
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°