

## Features

- MPEG I/II-Layer 3 Hardwired Decoder
  - Stand-alone MP3 Decoder
  - 48, 44.1, 32, 24, 22.05, 16 kHz Sampling Frequency
  - Separated Digital Volume Control on Left and Right Channels (Software Control using 31 Steps)
  - Bass, Medium, and Treble Control (31 Steps)
  - Bass Boost Sound Effect
  - Ancillary Data Extraction
  - CRC Error and MPEG Frame Synchronization Indicators
- Programmable Audio Output for Interfacing with Common Audio DAC
  - PCM Format Compatible
  - I<sup>2</sup>S Format Compatible
- 8-bit MCU C51 Core Based ( $F_{MAX} = 20$  MHz)
- 2304 Bytes of Internal RAM
- 64K Bytes of Code Memory
  - AT89C51SND1C: Flash (100K Erase/Write Cycles)
  - AT83C51SND1C: ROM
- 4K Bytes of Boot Flash Memory (AT89C51SND1C)
  - ISP: Download from USB or UART
- USB Rev 1.1 Controller
  - Full Speed Data Transmission
- Built-in PLL
  - MP3 Audio Clocks
  - USB Clock
- MultiMedia Card<sup>®</sup> Interface Compatibility
- Atmel DataFlash<sup>®</sup> SPI Interface Compatibility
- IDE/ATAPI Interface
- 2 Channels 10-bit ADC, 8 kHz (8-true bit)
  - Battery Voltage Monitoring
  - Voice Recording Controlled by Software
- Up to 44 Bits of General-purpose I/Os
  - 4-bit Interrupt Keyboard Port for a 4 x n Matrix
  - SmartMedia<sup>®</sup> Software Interface
- 2 Standard 16-bit Timers/Counters
- Hardware Watchdog Timer
- Standard Full Duplex UART with Baud Rate Generator
- Two Wire Master and Slave Modes Controller
- SPI Master and Slave Modes Controller
- Power Management
  - Power-on Reset
  - Software Programmable MCU Clock
  - Idle Mode, Power-down Mode
- Operating Conditions:
  - 3V,  $\pm 10\%$ , 25 mA Typical Operating at 25°C
  - Temperature Range: -40°C to +85°C
- Packages
  - TQFP80, BGA81, PLCC84 (Development Board)
  - Dice

## Description

The AT8xC51SND1C are fully integrated stand-alone hardwired MPEG I/II-Layer 3 decoder with a C51 microcontroller core handling data flow and MP3-player control.

The AT89C51SND1C includes 64K Bytes of Flash memory and allows In-System Programming through an embedded 4K Bytes of Boot Flash memory.



**Single-Chip  
Flash  
Microcontroller  
with MP3  
Decoder and  
Human Interface**

**AT83C51SND1C  
AT89C51SND1C**

**Preliminary**

Rev. 4109E-8051-06/03



The AT83C51SND1C includes 64K Bytes of ROM memory.

The AT8xC51SND1C include 2304 Bytes of RAM memory.

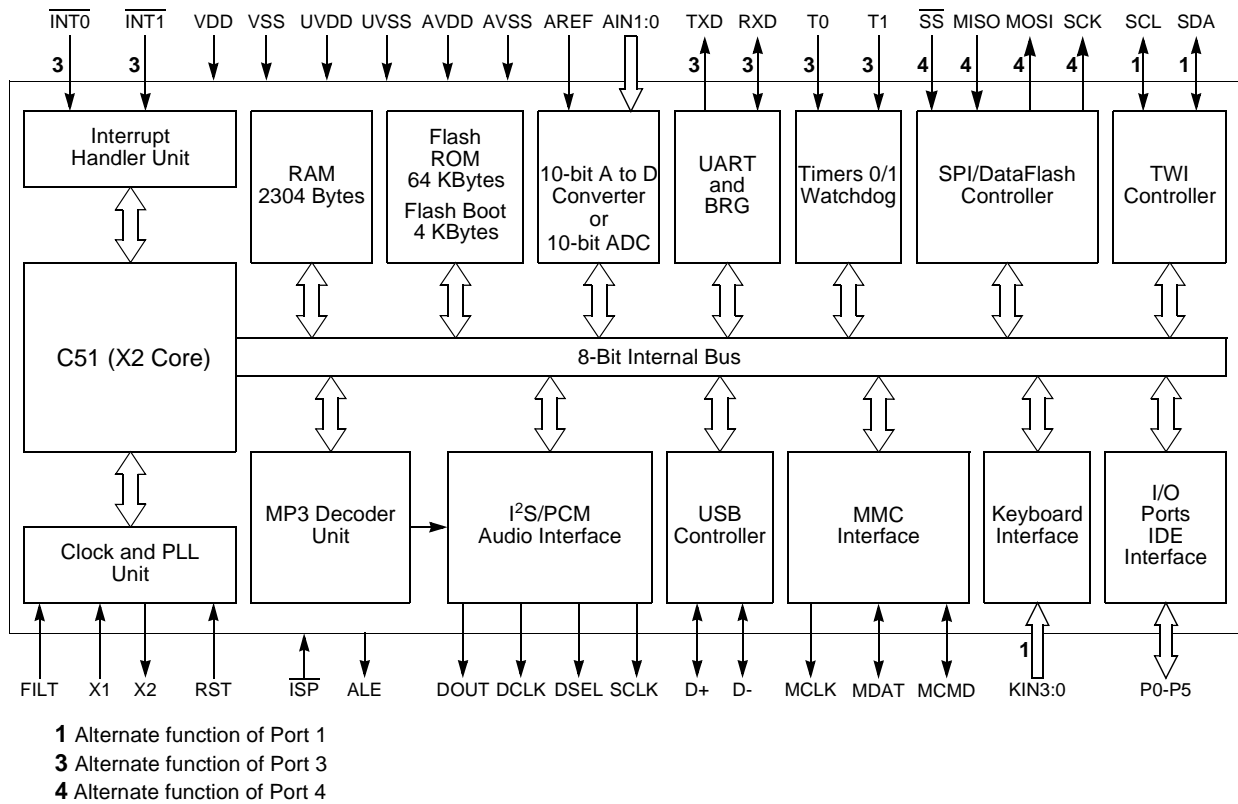
The AT8xC51SND1C provides the necessary features for human interface like timers, keyboard port, serial or parallel interface (USB, TWI, SPI, IDE), ADC input, I<sup>2</sup>S output, and all external memory interface (NAND or NOR Flash, SmartMedia, MultiMedia, DataFlash cards).

## Typical Applications

- MP3-Player
- PDA, Camera, Mobile Phone MP3
- Car Audio/Multimedia MP3
- Home Audio/Multimedia MP3

## Block Diagram

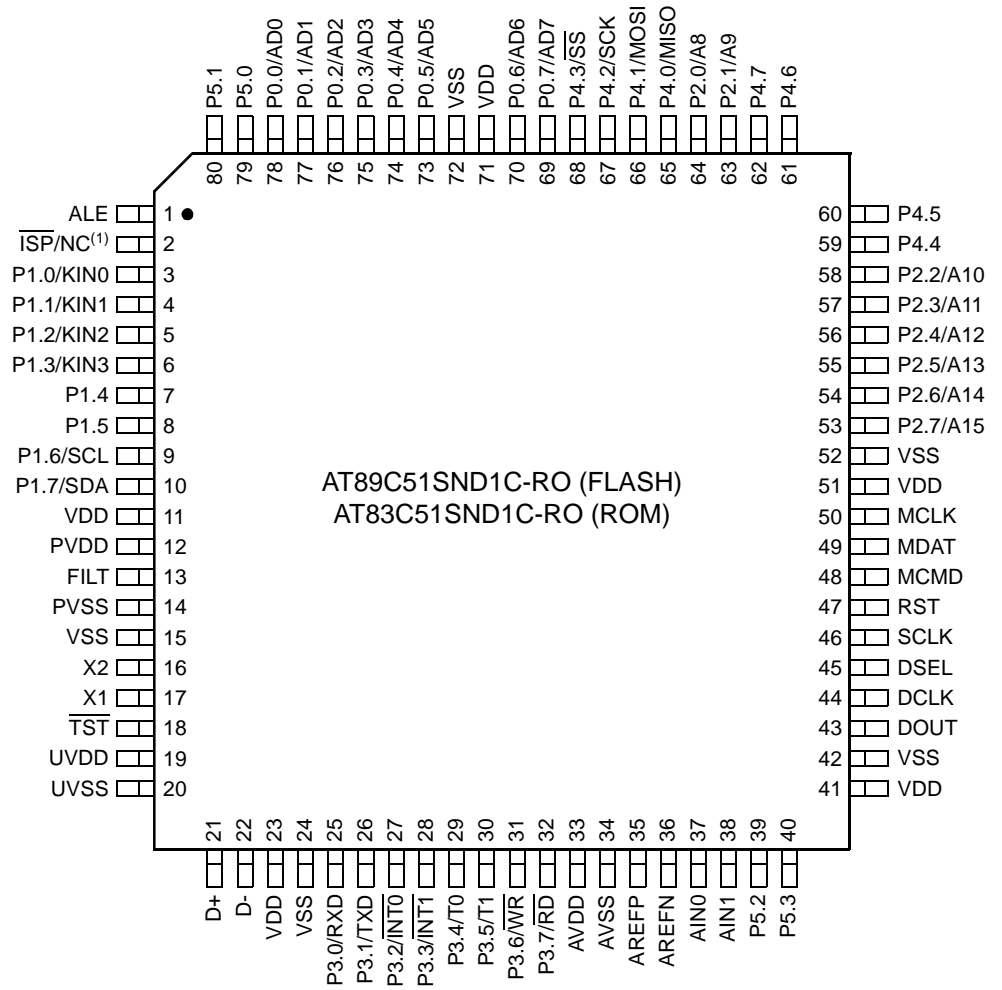
Figure 1. AT8xC51SND1C Block Diagram



## Pin Description

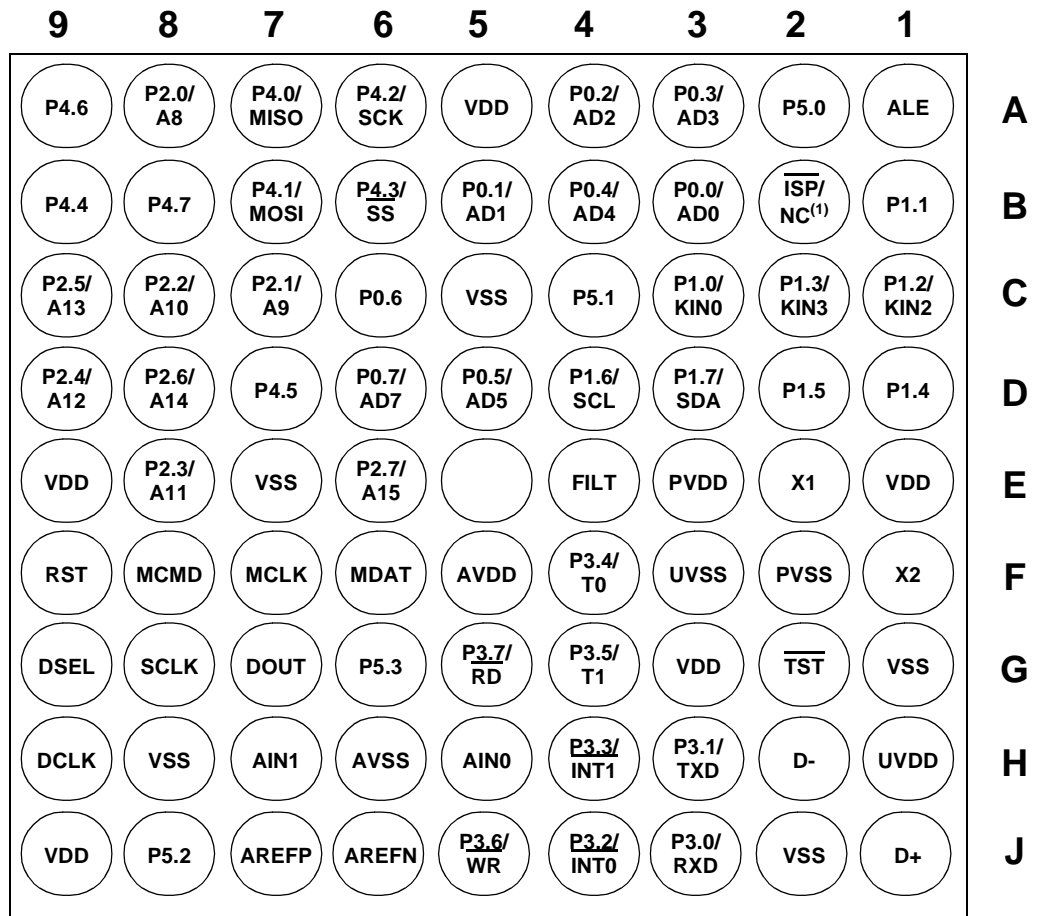
### Pinouts

Figure 2. AT89C51SND1C 80-pin QFP Package



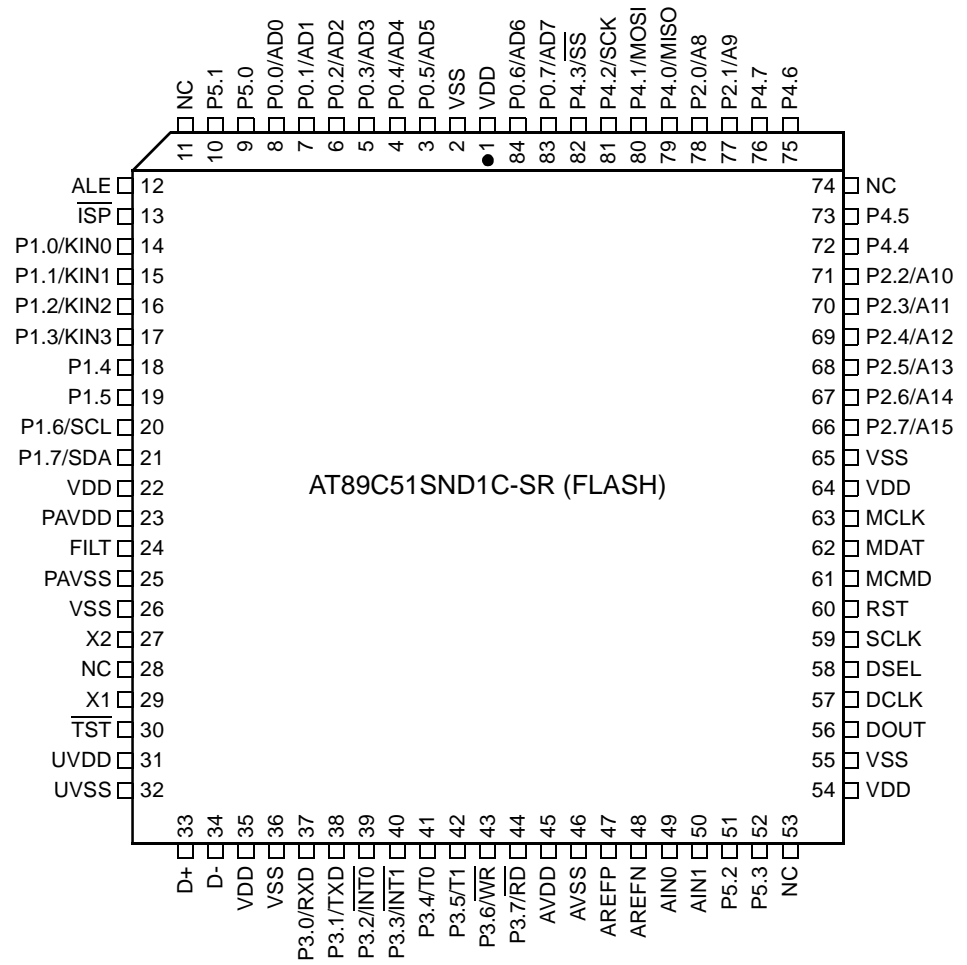
Note: 1.  $\overline{\text{ISP}}$  pin is only available in AT89C51SND1C product.  
Do not connect this pin on AT83C51SND1C product.

Figure 3. AT8xC51SND1C 81-pin BGA Package



Note: 1.  $\overline{\text{ISP}}$  pin is only available in AT89C51SND1C product.  
Do not connect this pin on AT83C51SND1C product.

**Figure 4.** AT8xC51SND1C 84-pin PLCC Package



## Signals

All the AT8xC51SND1C signals are detailed by functionality in Table 1 to Table 14.

**Table 1.** Ports Signal Description

Signal Name	Type	Description	Alternate Function
P0.7:0	I/O	<b>Port 0</b> P0 is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high impedance inputs. To avoid any parasitic current consumption, floating P0 inputs must be polarized to $V_{DD}$ or $V_{SS}$ .	AD7:0
P1.7:0	I/O	<b>Port 1</b> P1 is an 8-bit bidirectional I/O port with internal pull-ups.	KIN3:0 SCL SDA
P2.7:0	I/O	<b>Port 2</b> P2 is an 8-bit bidirectional I/O port with internal pull-ups.	A15:8
P3.7:0	I/O	<b>Port 3</b> P3 is an 8-bit bidirectional I/O port with internal pull-ups.	RXD TXD $\overline{\text{INT0}}$ $\overline{\text{INT1}}$ T0 T1 $\overline{\text{WR}}$ $\overline{\text{RD}}$
P4.7:0	I/O	<b>Port 4</b> P4 is an 8-bit bidirectional I/O port with internal pull-ups.	MISO MOSI SCK $\overline{\text{SS}}$
P5.3:0	I/O	<b>Port 5</b> P5 is a 4-bit bidirectional I/O port with internal pull-ups.	-

**Table 2.** Clock Signal Description

Signal Name	Type	Description	Alternate Function
X1	I	<b>Input to the on-chip inverting oscillator amplifier</b> To use the internal oscillator, a crystal/resonator circuit is connected to this pin. If an external oscillator is used, its output is connected to this pin. X1 is the clock source for internal timing.	-
X2	O	<b>Output of the on-chip inverting oscillator amplifier</b> To use the internal oscillator, a crystal/resonator circuit is connected to this pin. If an external oscillator is used, leave X2 unconnected.	-
FILT	I	<b>PLL Low Pass Filter input</b> FILT receives the RC network of the PLL low pass filter.	-

**Table 3.** Timer 0 and Timer 1 Signal Description

Signal Name	Type	Description	Alternate Function
INT0	I	<p><b>Timer 0 Gate Input</b> INT0 serves as external run control for timer 0, when selected by GATE0 bit in TCON register.</p> <p><b>External Interrupt 0</b> INT0 input sets IE0 in the TCON register. If bit IT0 in this register is set, bit IE0 is set by a falling edge on INT0#. If bit IT0 is cleared, bit IE0 is set by a low level on INT0#.</p>	P3.2
INT1	I	<p><b>Timer 1 Gate Input</b> INT1 serves as external run control for timer 1, when selected by GATE1 bit in TCON register.</p> <p><b>External Interrupt 1</b> INT1 input sets IE1 in the TCON register. If bit IT1 in this register is set, bit IE1 is set by a falling edge on INT1#. If bit IT1 is cleared, bit IE1 is set by a low level on INT1#.</p>	P3.3
T0	I	<p><b>Timer 0 External Clock Input</b> When timer 0 operates as a counter, a falling edge on the T0 pin increments the count.</p>	P3.4
T1	I	<p><b>Timer 1 External Clock Input</b> When timer 1 operates as a counter, a falling edge on the T1 pin increments the count.</p>	P3.5

**Table 4.** Audio Interface Signal Description

Signal Name	Type	Description	Alternate Function
DCLK	O	<b>DAC Data Bit Clock</b>	-
DOUT	O	<b>DAC Audio Data</b>	-
DSEL	O	<p><b>DAC Channel Select Signal</b> DSEL is the sample rate clock output.</p>	-
SCLK	O	<p><b>DAC System Clock</b> SCLK is the oversampling clock synchronized to the digital audio data (DOUT) and the channel selection signal (DSEL).</p>	-

**Table 5.** USB Controller Signal Description

Signal Name	Type	Description	Alternate Function
D+	I/O	<p><b>USB Positive Data Upstream Port</b> This pin requires an external 1.5 KΩ pull-up to V<sub>DD</sub> for full speed operation.</p>	-
D-	I/O	<b>USB Negative Data Upstream Port</b>	-

**Table 6.** MultiMediaCard Interface Signal Description

Signal Name	Type	Description	Alternate Function
MCLK	O	<b>MMC Clock output</b> Data or command clock transfer.	-
MCMD	I/O	<b>MMC Command line</b> Bidirectional command channel used for card initialization and data transfer commands. To avoid any parasitic current consumption, unused MCMD input must be polarized to $V_{DD}$ or $V_{SS}$ .	-
MDAT	I/O	<b>MMC Data line</b> Bidirectional data channel. To avoid any parasitic current consumption, unused MDAT input must be polarized to $V_{DD}$ or $V_{SS}$ .	-

**Table 7.** UART Signal Description

Signal Name	Type	Description	Alternate Function
RXD	I/O	<b>Receive Serial Data</b> RXD sends and receives data in serial I/O mode 0 and receives data in serial I/O modes 1, 2 and 3.	P3.0
TXD	O	<b>Transmit Serial Data</b> TXD outputs the shift clock in serial I/O mode 0 and transmits data in serial I/O modes 1, 2 and 3.	P3.1

**Table 8.** SPI Controller Signal Description

Signal Name	Type	Description	Alternate Function
MISO	I/O	<b>SPI Master Input Slave Output Data Line</b> When in master mode, MISO receives data from the slave peripheral. When in slave mode, MISO outputs data to the master controller.	P4.0
MOSI	I/O	<b>SPI Master Output Slave Input Data Line</b> When in master mode, MOSI outputs data to the slave peripheral. When in slave mode, MOSI receives data from the master controller.	P4.1
SCK	I/O	<b>SPI Clock Line</b> When in master mode, SCK outputs clock to the slave peripheral. When in slave mode, SCK receives clock from the master controller.	P4.2
$\overline{SS}$	I	<b>SPI Slave Select Line</b> When in controlled slave mode, $\overline{SS}$ enables the slave mode.	P4.3

**Table 9.** TWI Controller Signal Description

Signal Name	Type	Description	Alternate Function
SCL	I/O	<b>TWI Serial Clock</b> When TWI controller is in master mode, SCL outputs the serial clock to the slave peripherals. When TWI controller is in slave mode, SCL receives clock from the master controller.	P1.6
SDA	I/O	<b>TWI Serial Data</b> SDA is the bidirectional Two Wire data line.	P1.7



**Table 10.** A/D Converter Signal Description

Signal Name	Type	Description	Alternate Function
AIN1:0	I	<b>A/D Converter Analog Inputs</b>	-
AREFP	I	<b>Analog Positive Voltage Reference Input</b>	-
AREFN	I	<b>Analog Negative Voltage Reference Input</b> This pin is internally connected to AVSS.	-

**Table 11.** Keypad Interface Signal Description

Signal Name	Type	Description	Alternate Function
KIN3:0	I	<b>Keypad Input Lines</b> Holding one of these pins high or low for 24 oscillator periods triggers a keypad interrupt.	P1.3:0

**Table 12.** External Access Signal Description

Signal Name	Type	Description	Alternate Function
A15:8	I/O	<b>Address Lines</b> Upper address lines for the external bus. Multiplexed higher address and data lines for the IDE interface.	P2.7:0
AD7:0	I/O	<b>Address/Data Lines</b> Multiplexed lower address and data lines for the external memory or the IDE interface.	P0.7:0
ALE	O	<b>Address Latch Enable Output</b> ALE signals the start of an external bus cycle and indicates that valid address information is available on lines A7:0. An external latch is used to demultiplex the address from address/data bus.	-
$\overline{\text{ISP}}$	I/O	<b>ISP Enable Input AT89C51SND1C Only</b> This signal must be held to GND through a pull-down resistor at the falling reset to force execution of the internal bootloader.	-
$\overline{\text{RD}}$	O	<b>Read Signal</b> Read signal asserted during external data memory read operation.	P3.7
$\overline{\text{WR}}$	O	<b>Write Signal</b> Write signal asserted during external data memory write operation.	P3.6

**Table 13. System Signal Description**

Signal Name	Type	Description	Alternate Function
RST	I	<b>Reset Input</b> Holding this pin high for 64 oscillator periods while the oscillator is running resets the device. The Port pins are driven to their reset conditions when a voltage lower than $V_{IL}$ is applied, whether or not the oscillator is running. This pin has an internal pull-down resistor which allows the device to be reset by connecting a capacitor between this pin and $V_{DD}$ . Asserting RST when the chip is in Idle mode or Power-Down mode returns the chip to normal operation.	-
$\overline{TST}$	I	<b>Test Input</b> Test mode entry signal. This pin must be set to $V_{DD}$ .	-

**Table 14. Power Signal Description**

Signal Name	Type	Description	Alternate Function
VDD	PWR	<b>Digital Supply Voltage</b> Connect these pins to +3V supply voltage.	-
VSS	GND	<b>Circuit Ground</b> Connect these pins to ground.	-
AVDD	PWR	<b>Analog Supply Voltage</b> Connect this pin to +3V supply voltage.	-
AVSS	GND	<b>Analog Ground</b> Connect this pin to ground.	-
PVDD	PWR	<b>PLL Supply voltage</b> Connect this pin to +3V supply voltage.	-
PVSS	GND	<b>PLL Circuit Ground</b> Connect this pin to ground.	-
UVDD	PWR	<b>USB Supply Voltage</b> Connect this pin to +3V supply voltage.	-
UVSS	GND	<b>USB Ground</b> Connect this pin to ground.	-

## Internal Pin Structure

**Table 15.** Detailed Internal Pin Structure

Circuit <sup>(1)</sup>	Type	Pins
	Input	$\overline{\text{TST}}$
	Input/Output	RST
	Input/Output	P1 <sup>(2)</sup> P2 <sup>(3)</sup> P3 P4 P53:0
	Input/Output	P0 MCMD MDAT $\overline{\text{ISP}}$
	Output	ALE SCLK DCLK DOUT DSEL MCLK
	Input/Output	D+ D-

- Notes:
- For information on resistors value, input/output levels, and drive capability, refer to the Section "DC Characteristics", page 181.
  - When the Two Wire controller is enabled, P<sub>1</sub>, P<sub>2</sub>, and P<sub>3</sub> transistors are disabled allowing pseudo open-drain structure.
  - In Port 2, P<sub>1</sub> transistor is continuously driven when outputting a high level bit address (A15:8).

## Clock Controller

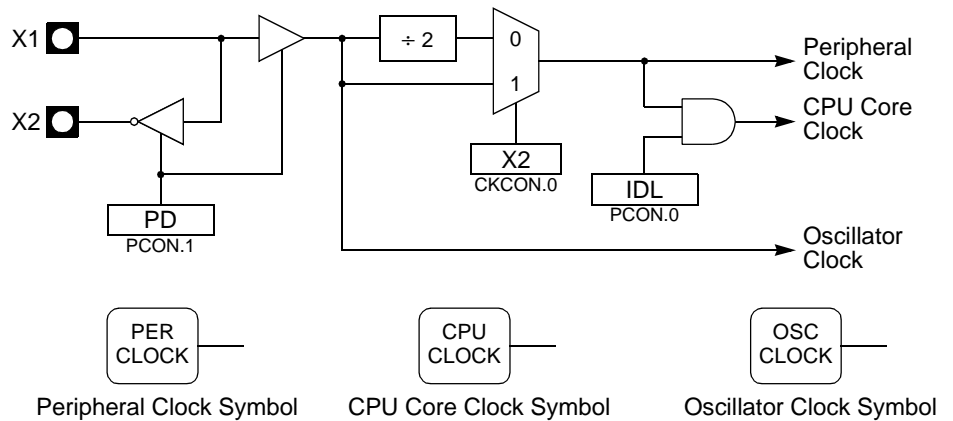
The AT8xC51SND1C clock controller is based on an on-chip oscillator feeding an on-chip Phase Lock Loop (PLL). All internal clocks to the peripherals and CPU core are generated by this controller.

## Oscillator

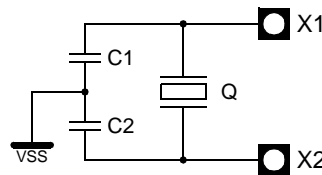
The AT8xC51SND1C X1 and X2 pins are the input and the output of a single-stage on-chip inverter (see Figure 5) that can be configured with off-chip components such as a Pierce oscillator (see Figure 6). Value of capacitors and crystal characteristics are detailed in the section “DC Characteristics”.

The oscillator outputs three different clocks: a clock for the PLL, a clock for the CPU core, and a clock for the peripherals as shown in Figure 5. These clocks are either enabled or disabled, depending on the power reduction mode as detailed in the section “Power Management” on page 46. The peripheral clock is used to generate the Timer 0, Timer 1, MMC, ADC, SPI, and Port sampling clocks.

**Figure 5.** Oscillator Block Diagram and Symbol



**Figure 6.** Crystal Connection

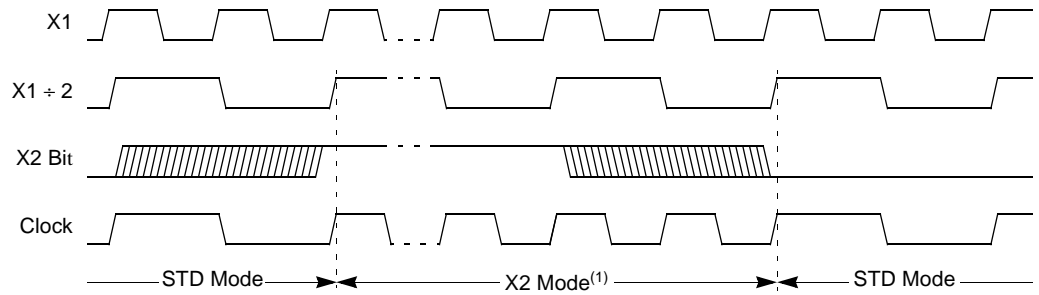


## X2 Feature

Unlike standard C51 products that require 12 oscillator clock periods per machine cycle, the AT8xC51SND1C need only 6 oscillator clock periods per machine cycle. This feature called the “X2 feature” can be enabled using the X2 bit<sup>(1)</sup> in CKCON (see Table 16) and allows the AT8xC51SND1C to operate in 6 or 12 oscillator clock periods per machine cycle. As shown in Figure 5, both CPU and peripheral clocks are affected by this feature. Figure 7 shows the X2 mode switching waveforms. After reset the standard mode is activated. In standard mode the CPU and peripheral clock frequency is the oscillator frequency divided by 2 while in X2 mode, it is the oscillator frequency.

Note: 1. The X2 bit reset value depends on the X2B bit in the Hardware Security Byte (see Table 22 on page 22). Using the AT89C51SND1C (Flash Version) the system can boot either in standard or X2 mode depending on the X2B value. Using AT83C51SND1C (ROM Version) the system always boots in standard mode. X2B bit can be changed to X2 mode later by software.

**Figure 7. Mode Switching Waveforms**



Note: 1. In order to prevent any incorrect operation while operating in X2 mode, user must be aware that all peripherals using clock frequency as time reference (timers, etc.) will have their time reference divided by 2. For example, a free running timer generating an interrupt every 20 ms will then generate an interrupt every 10 ms.

## PLL

### PLL Description

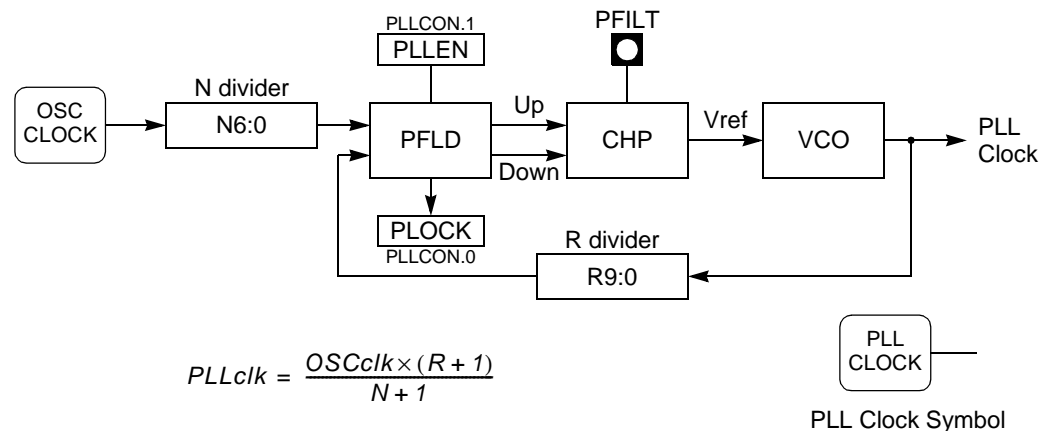
The AT8xC51SND1C PLL is used to generate internal high frequency clock (the PLL Clock) synchronized with an external low-frequency (the Oscillator Clock). The PLL clock provides the MP3 decoder, the audio interface, and the USB interface clocks. Figure 8 shows the internal structure of the PLL.

The PFLD block is the Phase Frequency Comparator and Lock Detector. This block makes the comparison between the reference clock coming from the N divider and the reverse clock coming from the R divider and generates some pulses on the Up or Down signal depending on the edge position of the reverse clock. The PLEN bit in PLLCON register is used to enable the clock generation. When the PLL is locked, the bit PLOCK in PLLCON register (see Table 17) is set.

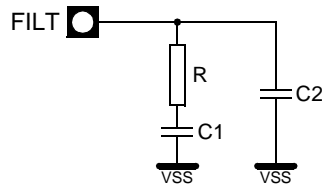
The CHP block is the Charge Pump that generates the voltage reference for the VCO by injecting or extracting charges from the external filter connected on PFILT pin (see Figure 9). Value of the filter components are detailed in the Section “DC Characteristics”.

The VCO block is the Voltage Controlled Oscillator controlled by the voltage  $V_{ref}$  produced by the charge pump. It generates a square wave signal: the PLL clock.

**Figure 8. PLL Block Diagram and Symbol**



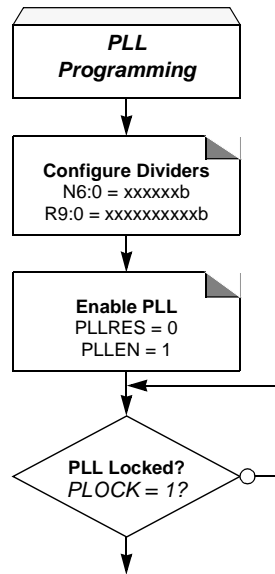
**Figure 9.** PLL Filter Connection



## PLL Programming

The PLL is programmed using the flow shown in Figure 10. As soon as clock generation is enabled, the user must wait until the lock indicator is set to ensure the clock output is stable. The PLL clock frequency will depend on MP3 decoder clock and audio interface clock frequencies.

**Figure 10.** PLL Programming Flow



## Registers

**Table 16.** CKCON Register  
CKCON (S:8Fh) – Clock Control Register

7	6	5	4	3	2	1	0
-	WDX2	-	-	-	T1X2	T0X2	X2
Bit Number	Bit Mnemonic	Description					
7	-	<b>Reserved</b> The value read from this bit is indeterminate. Do not set this bit.					
6	WDX2	<b>Watchdog Clock Control Bit</b> Set to select the oscillator clock divided by 2 as watchdog clock input (X2 independent). Clear to select the peripheral clock as watchdog clock input (X2 dependent).					
5 - 3	-	<b>Reserved</b> The values read from these bits are indeterminate. Do not set these bits.					
2	T1X2	<b>Timer 1 Clock Control Bit</b> Set to select the oscillator clock divided by 2 as timer 1 clock input (X2 independent). Clear to select the peripheral clock as timer 1 clock input (X2 dependent).					
1	T0X2	<b>Timer 0 Clock Control Bit</b> Set to select the oscillator clock divided by 2 as timer 0 clock input (X2 independent). Clear to select the peripheral clock as timer 0 clock input (X2 dependent).					
0	X2	<b>System Clock Control Bit</b> Clear to select 12 clock periods per machine cycle (STD mode, $F_{CPU} = F_{PER} = F_{OSC}/2$ ). Set to select 6 clock periods per machine cycle (X2 mode, $F_{CPU} = F_{PER} = F_{OSC}$ ).					

Reset Value = 0000 000Xb (AT89C51SND1C) or 0000 0000b (AT83C51SND1C)

**Table 17.** PLLCON Register  
PLLCON (S:E9h) – PLL Control Register

7	6	5	4	3	2	1	0
R1	R0	-	-	PLLRES	-	PLLEN	PLOCK
Bit Number	Bit Mnemonic	Description					
7 - 6	R1:0	<b>PLL Least Significant Bits R Divider</b> 2 LSB of the 10-bit R divider.					
5 - 4	-	<b>Reserved</b> The values read from these bits are always 0. Do not set these bits.					
3	PLLRES	<b>PLL Reset Bit</b> Set this bit to reset the PLL. Clear this bit to free the PLL and allow enabling.					
2	-	<b>Reserved</b> The value read from this bit is always 0. Do not set this bit.					
1	PLLEN	<b>PLL Enable Bit</b> Set to enable the PLL. Clear to disable the PLL.					
0	PLOCK	<b>PLL Lock Indicator</b> Set by hardware when PLL is locked. Clear by hardware when PLL is unlocked.					

Reset Value = 0000 1000b

**Table 18.** PLLNDIV Register

PLLNDIV (S:EEh) – PLL N Divider Register

7	6	5	4	3	2	1	0
-	N6	N5	N4	N3	N2	N1	N0
Bit Number	Bit Mnemonic	Description					
7	-	<b>Reserved</b> The value read from this bit is always 0. Do not set this bit.					
6 - 0	N6:0	<b>PLL N Divider</b> 7 - bit N divider.					

Reset Value = 0000 0000b

**Table 19.** PLLRDIV Register

PLLRDIV (S:EFh) – PLL R Divider Register

7	6	5	4	3	2	1	0
R9	R8	R7	R6	R5	R4	R3	R2
Bit Number	Bit Mnemonic	Description					
7 - 0	R9:2	<b>PLL Most Significant Bits R Divider</b> 8 MSB of the 10-bit R divider.					

Reset Value = 0000 0000b



## Program/Code Memory

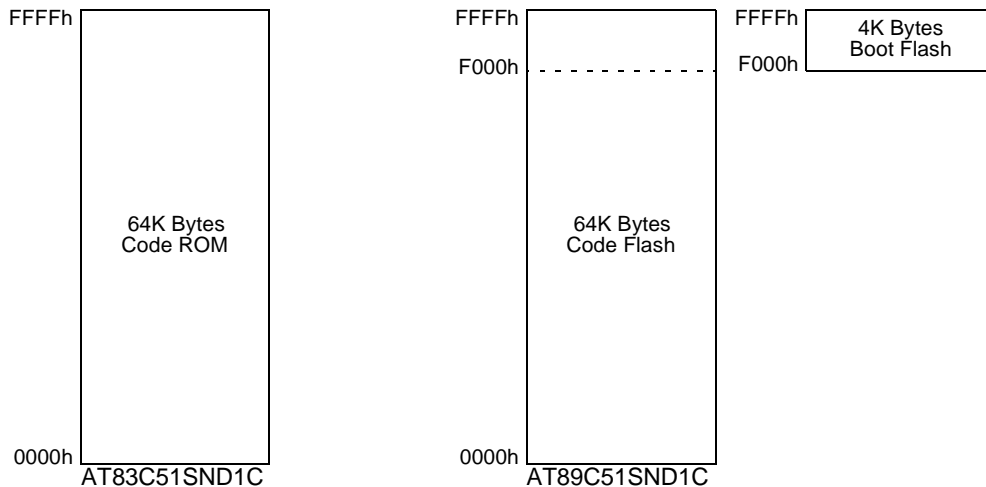
The AT8xC51SND1C implement 64K Bytes of on-chip program/code memory. Figure 11 shows the split of internal and external program/code memory spaces depending on the product.

The AT83C51SND1C product provides the internal program/code memory in ROM memory while the AT89C51SND1C product provides it in Flash memory. These 2 products do not allow external code memory execution.

The Flash memory increases EPROM and ROM functionality by in-circuit electrical erasure and programming. The high voltage needed for programming or erasing Flash cells is generated on-chip using the standard  $V_{DD}$  voltage, made possible by the internal charge pump. Thus, the AT89C51SND1C can be programmed using only one voltage and allows In-application software programming. Hardware programming mode is also available using common programming tools. See the application note 'Programming T89C51x and AT89C51x with Device Programmers'.

The AT89C51SND1C implements an additional 4K Bytes of on-chip boot Flash memory provided in Flash memory. This boot memory is delivered programmed with a standard boot loader software allowing In-System Programming (ISP). It also contains some Application Programming Interface routines named API routines allowing In Application Programming (IAP) by using user's own boot loader.

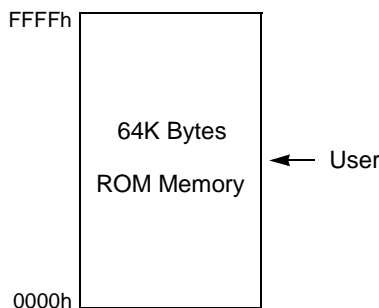
**Figure 11.** Program/Code Memory Organization



## ROM Memory Architecture

As shown in Figure 11 the AT83C51SND1C ROM memory is composed of one space detailed in the following paragraphs.

**Figure 12.** AT83C51SND1C Memory Architecture



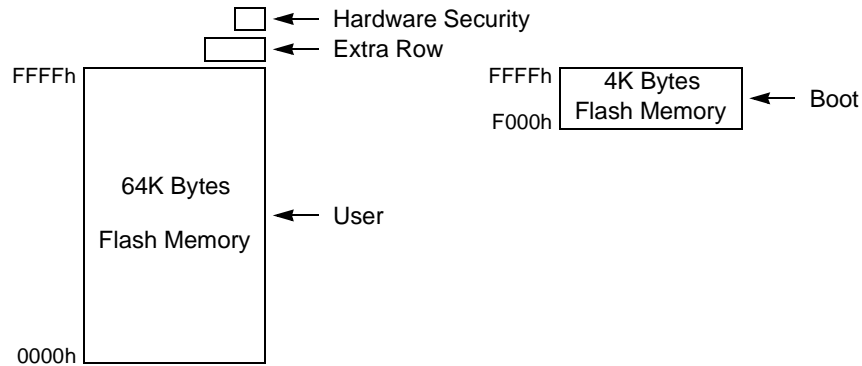
### User Space

This space is composed of a 64K Bytes ROM memory programmed during the manufacturing process. It contains the user's application code.

### Flash Memory Architecture

As shown in Figure 13 the AT89C51SND1C Flash memory is composed of four spaces detailed in the following paragraphs.

Figure 13. AT89C51SND1C Memory Architecture



### User Space

This space is composed of a 64K Bytes Flash memory organized in 512 pages of 128 Bytes. It contains the user's application code.

This space can be read or written by both software and hardware modes.

### Boot Space

This space is composed of a 4K Bytes Flash memory. It contains the boot loader for In-System Programming and the routines for In Application Programming.

This space can only be read or written by hardware mode using a parallel programming tool.

### Hardware Security Space

This space is composed of one Byte: the Hardware Security Byte (HSB see Table 22) divided in 2 separate nibbles. The MSN contains the X2 mode configuration bit and the Boot Loader Jump Bit as detailed in Section "Boot Memory Execution", page 19 and can be written by software while the LSN contains the lock system level to protect the memory content against piracy as detailed in Section "Hardware Security System", page 19 and can only be written by hardware.

### Extra Row Space

This space is composed of 2 Bytes:

- The Software Boot Vector (SBV, see Table 23).  
This Byte is used by the software boot loader to build the boot address.
- The Software Security Byte (SSB, see Table 24).  
This Byte is used to lock the execution of some boot loader commands.

## Hardware Security System

The AT89C51SND1C implements three lock bits LB2:0 in the LSN of HSB (see Table 22) providing three levels of security for user's program as described in Table 22 while the AT83C51SND1C is always set in read disabled mode.

Level 0 is the level of an erased part and does not enable any security feature.

Level 1 locks the hardware programming of both user and boot memories.

Level 2 locks also hardware verifying of both user and boot memories

Level 3 locks also the external execution.

**Table 20.** Lock Bit Features<sup>(1)</sup>

Level	LB2 <sup>(2)</sup>	LB1	LB0	Internal Execution	External Execution	Hardware Verifying	Hardware Programming	Software Programming
0	U	U	U	Enable	Enable	Enable	Enable	Enable
1	U	U	P	Enable	Enable	Enable	Disable	Enable
2	U	P	X	Enable	Enable	Disable	Disable	Enable
3 <sup>(3)</sup>	P	X	X	Enable	Disable	Disable	Disable	Enable

Notes: 1. U means unprogrammed, P means programmed and X means don't care (programmed or unprogrammed).

2. LB2 is not implemented in the AT8xC51SND1C products.

3. AT89C51SND1C products are delivered with third level programmed to ensure that the code programmed by software using ISP or user's boot loader is secured from any hardware piracy.

## Boot Memory Execution

As internal C51 code space is limited to 64K Bytes, some mechanisms are implemented to allow boot memory to be mapped in the code space for execution at addresses from F000h to FFFFh. The boot memory is enabled by setting the ENBOOT bit in AUXR1 (see Figure 21). The three ways to set this bit are detailed in the following sections.

### Software Boot Mapping

The software way to set ENBOOT consists in writing to AUXR1 from the user's software. This enables boot loader or API routines execution.

### Hardware Condition Boot Mapping

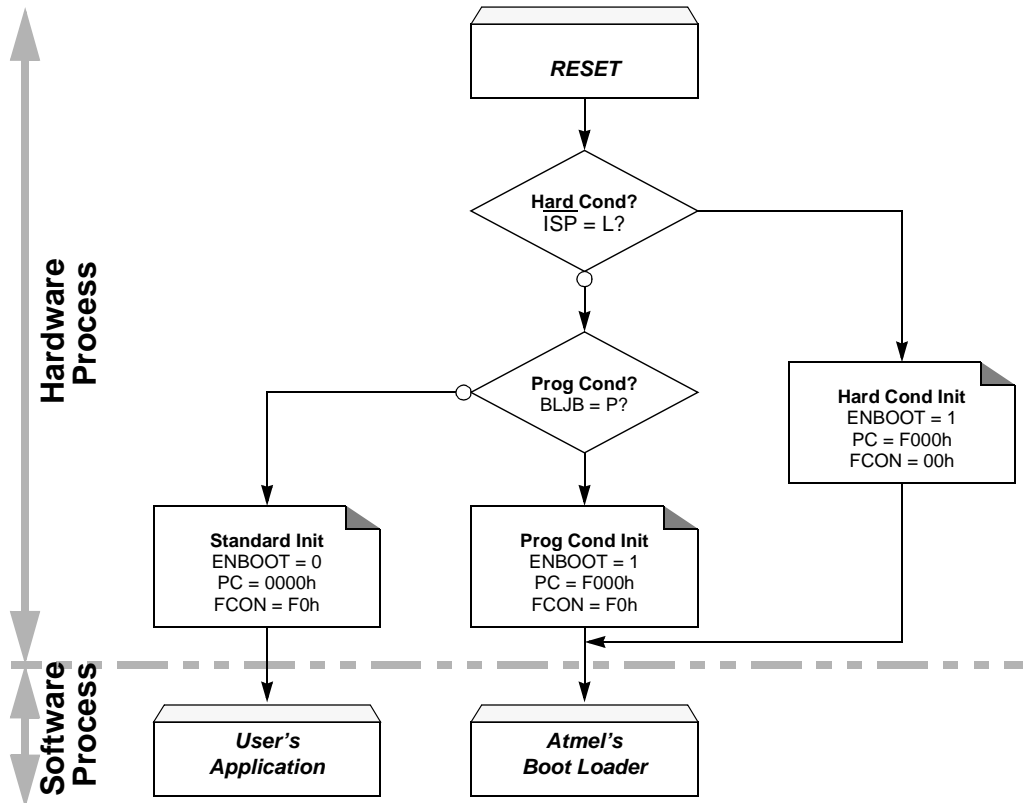
The hardware condition is based on the  $\overline{\text{ISP}}$  pin. When driving this pin to low level, the chip reset sets ENBOOT and forces the reset vector to F000h instead of 0000h in order to execute the boot loader software.

As shown in Figure 14 the hardware condition always allows in-system recovery when user's memory has been corrupted.

### Programmed Condition Boot Mapping

The programmed condition is based on the Boot Loader Jump Bit (BLJB) in HSB. As shown in Figure 14 when this bit is programmed (by hardware or software programming mode), the chip reset set ENBOOT and forces the reset vector to F000h instead of 0000h, in order to execute the boot loader software.

Figure 14. Hardware Boot Process Algorithm



The software process (boot loader) is detailed in the “Boot Loader Datasheet” Document.

## Preventing Flash Corruption

See Section “Reset Recommendation to Prevent Flash Corruption”, page 47.

## Registers

**Table 21.** AUXR1 Register  
AUXR1 (S:A2h) – Auxiliary Register 1

7	6	5	4	3	2	1	0
-	-	ENBOOT	-	GF3	0	-	DPS
Bit Number	Bit Mnemonic	Description					
7 - 6	-	<b>Reserved</b> The value read from these bits are indeterminate. Do not set these bits.					
5	ENBOOT <sup>1</sup>	<b>Enable Boot Flash</b> Set this bit to map the boot Flash in the code space between at addresses F000h to FFFFh. Clear this bit to disable boot Flash.					
4	-	<b>Reserved</b> The value read from this bit is indeterminate. Do not set this bit.					
3	GF3	<b>General Flag</b> This bit is a general-purpose user flag.					
2	0	<b>Always Zero</b> This bit is stuck to logic 0 to allow INC AUXR1 instruction without affecting GF3 flag.					
1	-	<b>Reserved for Data Pointer Extension.</b>					
0	DPS	<b>Data Pointer Select Bit</b> Set to select second data pointer: DPTR1. Clear to select first data pointer: DPTR0.					

Reset Value = XXXX 00X0b

Note: 1. ENBOOT bit is only available in AT89C51SND1C product.

## Hardware Bytes

**Table 22.** HSB Byte – Hardware Security Byte

7	6	5	4	3	2	1	0
X2B	BLJB	-	-	-	LB2	LB1	LB0
Bit Number	Bit Mnemonic	Description					
7	X2B <sup>(1)</sup>	<b>X2 Bit</b> Program this bit to start in X2 mode. Unprogram (erase) this bit to start in standard mode.					
6	BLJB <sup>(2)</sup>	<b>Boot Loader Jump Bit</b> Program this bit to execute the boot loader at address F000h on next reset. Unprogram (erase) this bit to execute user's application at address 0000h on next reset.					
5 - 4	-	<b>Reserved</b> The value read from these bits is always unprogrammed. Do not program these bits.					
3	-	<b>Reserved</b> The value read from this bit is always unprogrammed. Do not program this bit.					
2 - 0	LB2:0	<b>Hardware Lock Bits</b> Refer to for bits description.					

Reset Value = XXUU UXXX, UUUU UUUU after an hardware full chip erase.

- Note:
1. X2B initializes the X2 bit in CKCON during the reset phase.
  2. In order to ensure boot loader activation at first power-up, AT89C51SND1C products are delivered with BLJB programmed.
  3. Bits 0 to 3 (LSN) can only be programmed by hardware mode.

**Table 23.** SBV Byte – Software Boot Vector

7	6	5	4	3	2	1	0
ADD15	ADD14	ADD13	ADD12	ADD11	ADD10	ADD9	ADD8
Bit Number	Bit Mnemonic	Description					
7 - 0	ADD15:8	<b>MSB of the user's boot loader 16-bit address location</b> Refer to the boot loader datasheet for usage information (boot loader dependent)					

Reset Value = XXXX XXXX, UUUU UUUU after an hardware full chip erase.

**Table 24.** SSB Byte – Software Security Byte

7	6	5	4	3	2	1	0
SSB7	SSB6	SSB5	SSB4	SSB3	SSB2	SSB1	SSB0
Bit Number	Bit Mnemonic	Description					
7 - 0	SSB7:0	<b>Software Security Byte Data</b> Refer to the boot loader datasheet for usage information (boot loader dependent)					

Reset Value = XXXX XXXX, UUUU UUUU after an hardware full chip erase.

## Data Memory

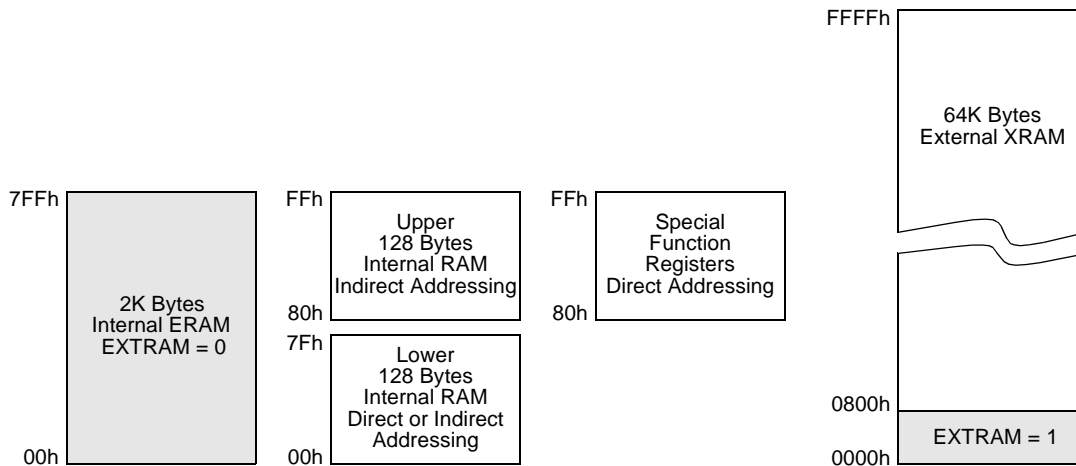
The AT8xC51SND1C provides data memory access in 2 different spaces:

1. The internal space mapped in three separate segments:
  - The lower 128 Bytes RAM segment
  - The upper 128 Bytes RAM segment
  - The expanded 2048 Bytes RAM segment
2. The external space.

A fourth internal segment is available but dedicated to Special Function Registers, SFRs, (addresses 80h to FFh) accessible by direct addressing mode. For information on this segment, refer to the Section “Special Function Registers”, page 30.

Figure 15 shows the internal and external data memory spaces organization.

**Figure 15.** Internal and External Data Memory Organization



## Internal Space

### Lower 128 Bytes RAM

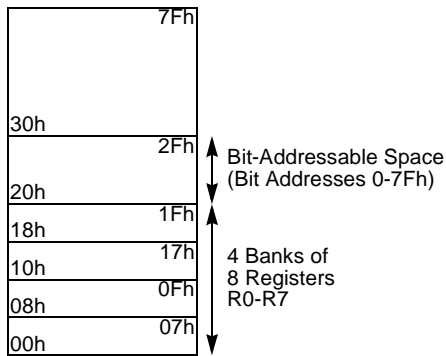
The lower 128 Bytes of RAM (see Figure 16) are accessible from address 00h to 7Fh using direct or indirect addressing modes. The lowest 32 Bytes are grouped into 4 banks of 8 registers (R0 to R7). 2 bits RS0 and RS1 in PSW register (see Table 28) select which bank is in use according to Table 25. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing, and can be used for context switching in interrupt service routines.

**Table 25.** Register Bank Selection

RS1	RS0	Description
0	0	Register bank 0 from 00h to 07h
0	1	Register bank 1 from 08h to 0Fh
1	0	Register bank 2 from 10h to 17h
1	1	Register bank 3 from 18h to 1Fh

The next 16 Bytes above the register banks form a block of bit-addressable memory space. The C51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00h to 7Fh.

**Figure 16.** Lower 128 Bytes Internal RAM Organization



**Upper 128 Bytes RAM**

The upper 128 Bytes of RAM are accessible from address 80h to FFh using only indirect addressing mode.

**Expanded RAM**

The on-chip 2K Bytes of expanded RAM (ERAM) are accessible from address 0000h to 07FFh using indirect addressing mode through MOVX instructions. In this address range, EXTRAM bit in AUXR register (see Table 29) is used to select the ERAM (default) or the XRAM. As shown in Figure 15 when EXTRAM = 0, the ERAM is selected and when EXTRAM = 1, the XRAM is selected (see Section “External Space”).

The ERAM memory can be resized using XRS1:0 bits in AUXR register to dynamically increase external access to the XRAM space. Table 26 details the selected ERAM size and address range.

**Table 26.** ERAM Size Selection

XRS1	XRS0	ERAM Size	Address
0	0	256 Bytes	0 to 00FFh
0	1	512 Bytes	0 to 01FFh
1	0	1K Byte	0 to 03FFh
1	1	2K Bytes	0 to 07FFh

Note: Lower 128 Bytes RAM, Upper 128 Bytes RAM, and expanded RAM are made of volatile memory cells. This means that the RAM content is indeterminate after power-up and must then be initialized properly.



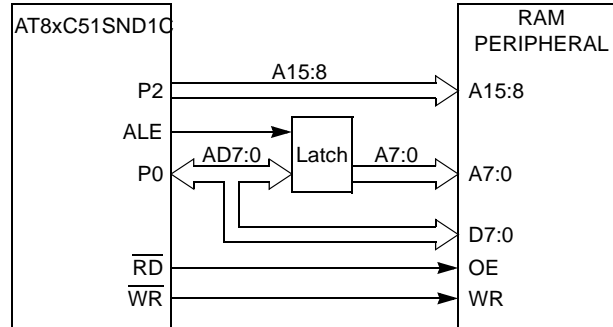
## External Space

### Memory Interface

The external memory interface comprises the external bus (port 0 and port 2) as well as the bus control signals ( $\overline{RD}$ ,  $\overline{WR}$ , and ALE).

Figure 17 shows the structure of the external address bus. P0 carries address A7:0 while P2 carries address A15:8. Data D7:0 is multiplexed with A7:0 on P0. Table 27 describes the external memory interface signals.

**Figure 17.** External Data Memory Interface Structure



**Table 27.** External Data Memory Interface Signals

Signal Name	Type	Description	Alternate Function
A15:8	O	<b>Address Lines</b> Upper address lines for the external bus.	P2.7:0
AD7:0	I/O	<b>Address/Data Lines</b> Multiplexed lower address lines and data for the external memory.	P0.7:0
ALE	O	<b>Address Latch Enable</b> ALE signals indicates that valid address information are available on lines AD7:0.	-
$\overline{RD}$	O	<b>Read</b> Read signal output to external data memory.	P3.7
$\overline{WR}$	O	<b>Write</b> Write signal output to external memory.	P3.6

### Page Access Mode

The AT8xC51SND1C implement a feature called Page Access that disables the output of DPH on P2 when executing MOVX @DPTR instruction. Page Access is enable by setting the DPHDIS bit in AUXR register.

Page Access is useful when application uses both ERAM and 256 Bytes of XRAM. In this case, software modifies intensively EXTRAM bit to select access to ERAM or XRAM and must save it if used in interrupt service routine. Page Access allows external access above 00FFh address without generating DPH on P2. Thus ERAM is accessed using MOVX @Ri or MOVX @DPTR with DPTR < 0100h, and XRAM is accessed using MOVX @DPTR with DPTR ≥ 0100h while keeping P2 for general I/O usage.

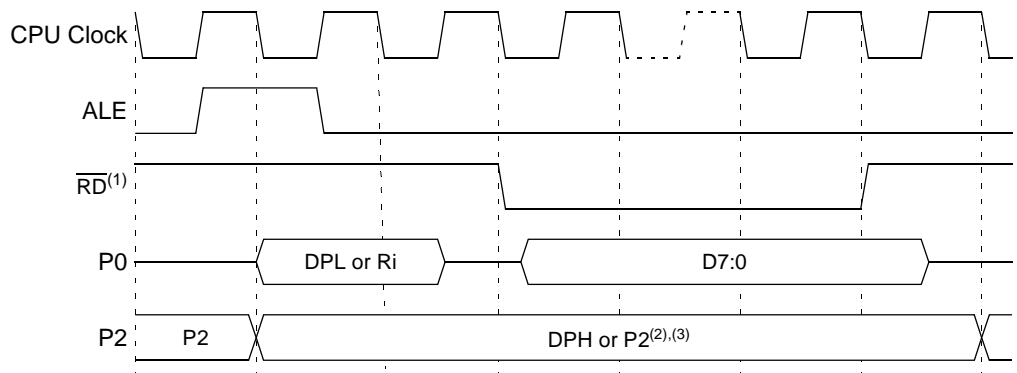
## External Bus Cycles

This section describes the bus cycles the AT8xC51SND1C executes to read (see Figure 18), and write data (see Figure 19) in the external data memory. External memory cycle takes 6 CPU clock periods. This is equivalent to 12 oscillator clock period in standard mode or 6 oscillator clock periods in X2 mode. For further information on X2 mode, refer to the Section “X2 Feature”, page 12.

Slow peripherals can be accessed by stretching the read and write cycles. This is done using the M0 bit in AUXR register. Setting this bit changes the width of the  $\overline{RD}$  and  $\overline{WR}$  signals from 3 to 15 CPU clock periods.

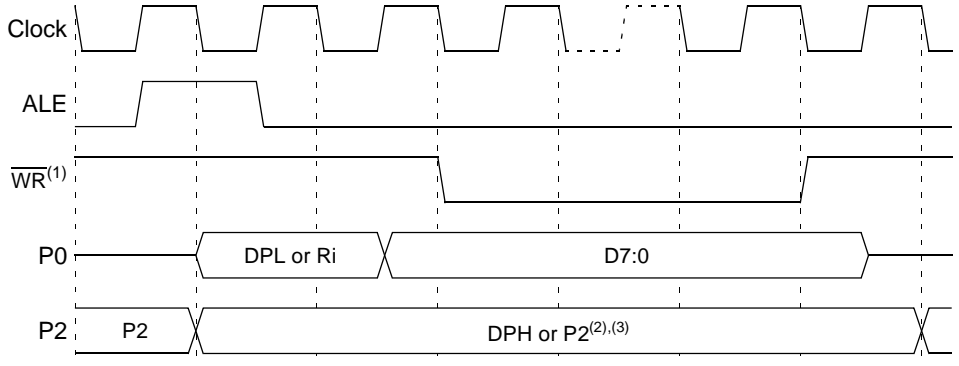
For simplicity, Figure 18 and Figure 19 depict the bus cycle waveforms in idealized form and do not provide precise timing information. For bus cycle timing parameters refer to the Section “AC Characteristics”.

**Figure 18.** External Data Read Waveforms



- Notes:
1.  $\overline{RD}$  signal may be stretched using M0 bit in AUXR register.
  2. When executing MOVX @Ri instruction, P2 outputs SFR content.
  3. When executing MOVX @DPTR instruction, if DPHDIS is set (Page Access Mode), P2 outputs SFR content instead of DPH.

**Figure 19.** External Data Write Waveforms



- Notes:
1.  $\overline{WR}$  signal may be stretched using M0 bit in AUXR register.
  2. When executing MOVX @Ri instruction, P2 outputs SFR content.
  3. When executing MOVX @DPTR instruction, if DPHDIS is set (Page Access Mode), P2 outputs SFR content instead of DPH.

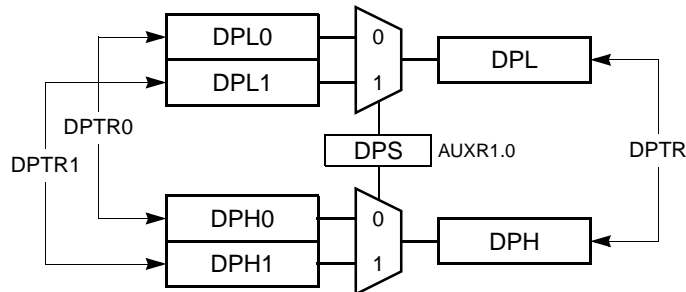
## Dual Data Pointer

### Description

The AT8xC51SND1C implement a second data pointer for speeding up code execution and reducing code size in case of intensive usage of external memory accesses.

DPTR0 and DPTR1 are seen by the CPU as DPTR and are accessed using the SFR addresses 83h and 84h that are the DPH and DPL addresses. The DPS bit in AUXR1 register (see Table 21) is used to select whether DPTR is the data pointer 0 or the data pointer 1 (see Figure 20).

**Figure 20.** Dual Data Pointer Implementation



### Application

Software can take advantage of the additional data pointers to both increase speed and reduce code size, for example, block operations (copy, compare, search ...) are well served by using one data pointer as a “source” pointer and the other one as a “destination” pointer.

Below is an example of block move implementation using the 2 pointers and coded in assembler. The latest C compiler also takes advantage of this feature by providing enhanced algorithm libraries.

The INC instruction is a short (2 Bytes) and fast (6 CPU clocks) way to manipulate the DPS bit in the AUXR1 register. However, note that the INC instruction does not directly force the DPS bit to a particular state, but simply toggles it. In simple routines, such as the block move example, only the fact that DPS is toggled in the proper sequence matters, not its actual value. In other words, the block move routine works the same whether DPS is '0' or '1' on entry.

```
; ASCII block move using dual data pointers
; Modifies DPTR0, DPTR1, A and PSW
; Ends when encountering NULL character
; Note: DPS exits opposite of entry state unless an extra INC AUXR1 is added
```

```
AUXR1    EQU    0A2h

move:    mov    DPTR,#SOURCE ; address of SOURCE
         inc    AUXR1       ; switch data pointers
         mov    DPTR,#DEST  ; address of DEST
mv_loop: inc    AUXR1       ; switch data pointers
         movx   A,@DPTR     ; get a Byte from SOURCE
         inc    DPTR        ; increment SOURCE address
         inc    AUXR1       ; switch data pointers
         movx  @DPTR,A      ; write the Byte to DEST
         inc    DPTR        ; increment DEST address
         jnz   mv_loop      ; check for NULL terminator
end_move:
```

## Registers

**Table 28.** PSW Register  
*PSW (S:8Eh) – Program Status Word Register*

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	F1	P
Bit Number	Bit Mnemonic	Description					
7	CY	<b>Carry Flag</b> Carry out from bit 1 of ALU operands.					
6	AC	<b>Auxiliary Carry Flag</b> Carry out from bit 1 of addition operands.					
5	F0	<b>User Definable Flag 0</b>					
4 - 3	RS1:0	<b>Register Bank Select Bits</b> Refer to Table 25 for bits description.					
2	OV	<b>Overflow Flag</b> Overflow set by arithmetic operations.					
1	F1	<b>User Definable Flag 1</b>					
0	P	<b>Parity Bit</b> Set when ACC contains an odd number of 1's. Cleared when ACC contains an even number of 1's.					

Reset Value = 0000 0000b

**Table 29.** AUXR Register  
*AUXR (S:8Eh)* – Auxiliary Control Register

7	6	5	4	3	2	1	0
-	EXT16	M0	DPHDIS	XRS1	XRS0	EXTRAM	AO
Bit Number	Bit Mnemonic	Description					
7	-	<b>Reserved</b> The value read from this bit is indeterminate. Do not set this bit.					
6	EXT16	<b>External 16-bit Access Enable Bit</b> Set to enable 16-bit access mode during MOVX instructions. Clear to disable 16-bit access mode and enable standard 8-bit access mode during MOVX instructions.					
5	M0	<b>External Memory Access Stretch Bit</b> Set to stretch RD or WR signals duration to 15 CPU clock periods. Clear not to stretch RD or WR signals and set duration to 3 CPU clock periods.					
4	DPHDIS	<b>DPH Disable Bit</b> Set to disable DPH output on P2 when executing MOVX @DPTR instruction. Clear to enable DPH output on P2 when executing MOVX @DPTR instruction.					
3 - 2	XRS1:0	<b>Expanded RAM Size Bits</b> Refer to Table 26 for ERAM size description.					
1	EXTRAM	<b>External RAM Enable Bit</b> Set to select the external XRAM when executing MOVX @Ri or MOVX @DPTR instructions. Clear to select the internal expanded RAM when executing MOVX @Ri or MOVX @DPTR instructions.					
0	AO	<b>ALE Output Enable Bit</b> Set to output the ALE signal only during MOVX instructions. Clear to output the ALE signal at a constant rate of $F_{CPU}/3$ .					

Reset Value = X000 1101b

## Special Function Registers

The Special Function Registers (SFRs) of the AT8xC51SND1C derivatives fall into the categories detailed in Table 30 to Table 46. The relative addresses of these SFRs are provided together with their reset values in Table 47. In this table, the bit-addressable registers are identified by Note 1.

**Table 30.** C51 Core SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
ACC	E0h	Accumulator								
B	F0h	B Register								
PSW	D0h	Program Status Word	CY	AC	F0	RS1	RS0	OV	F1	P
SP	81h	Stack Pointer								
DPL	82h	Data Pointer Low Byte								
DPH	83h	Data Pointer High Byte								

**Table 31.** System Management SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
PCON	87h	Power Control	SMOD1	SMOD0	-	-	GF1	GF0	PD	IDL
AUXR	8Eh	Auxiliary Register 0	-	EXT16	M0	DPHDIS	XRS1	XRS0	EXTRAM	AO
AUXR1	A2h	Auxiliary Register 1	-	-	ENBOOT <sup>(1)</sup>	-	GF3	0	-	DPS
NVERS	FBh	Version Number	NV7	NV6	NV5	NV4	NV3	NV2	NV1	NV0

Note: 1. ENBOOT bit is only available in AT89C51SND1C product.

**Table 32.** PLL and System Clock SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
CKCON	8Fh	Clock Control	-	-	-	-	-	-	-	X2
PLLCON	E9h	PLL Control	R1	R0	-	-	PLLRES	-	PLLEN	PLOCK
PLLNDIV	EEh	PLL N Divider	-	N6	N5	N4	N3	N2	N1	N0
PLLRDIV	EFh	PLL R Divider	R9	R8	R7	R6	R5	R4	R3	R2

**Table 33.** Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
IEN0	A8h	Interrupt Enable Control 0	EA	EAUD	EMP3	ES	ET1	EX1	ET0	EX0
IEN1	B1h	Interrupt Enable Control 1	-	EUSB	-	EKB	EADC	ESPI	EI2C	EMMC
IPH0	B7h	Interrupt Priority Control High 0	-	IPHAUD	IPHMP3	IPHS	IPHT1	IPHX1	IPHT0	IPHX0
IPL0	B8h	Interrupt Priority Control Low 0	-	IPLAUD	IPLMP3	IPLS	IPLT1	IPLX1	IPLT0	IPLX0
IPH1	B3h	Interrupt Priority Control High 1	-	IPHUSB	-	IPHKB	IPHADC	IPHSPI	IPHI2C	IPHMMC
IPL1	B2h	Interrupt Priority Control Low 1	-	IPLUSB	-	IPLKB	IPLADC	IPLSPI	IPLI2C	IPLMMC

**Table 34.** Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
P0	80h	8-bit Port 0								
P1	90h	8-bit Port 1								
P2	A0h	8-bit Port 2								
P3	B0h	8-bit Port 3								
P4	C0h	8-bit Port 4								
P5	D8h	4-bit Port 5	-	-	-	-				

**Table 35.** Flash Memory SFR

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
FCON <sup>(1)</sup>	D1h	Flash Control	FPL3	FPL2	FPL1	FPL0	FPS	FMOD1	FMOD0	FBUSY

Note: 1. FCON register is only available in AT89C51SND1C product.

**Table 36.** Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
TCON	88h	Timer/Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
TMOD	89h	Timer/Counter 0 and 1 Modes	GATE1	C/T1#	M11	M01	GATE0	C/T0#	M10	M00
TL0	8Ah	Timer/Counter 0 Low Byte								
TH0	8Ch	Timer/Counter 0 High Byte								
TL1	8Bh	Timer/Counter 1 Low Byte								
TH1	8Dh	Timer/Counter 1 High Byte								
WDTRST	A6h	Watchdog Timer Reset								
WDTPRG	A7h	Watchdog Timer Program	-	-	-	-	-	WTO2	WTO1	WTO0

**Table 37. MP3 Decoder SFRs**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
MP3CON	AAh	MP3 Control	MPEN	MPBBST	CRCEN	MSKANC	MSKREQ	MSKLAY	MSKSYN	MSKCRC
MP3STA	C8h	MP3 Status	MPANC	MPREQ	ERRLAY	ERRSYN	ERRCRC	MPFS1	MPFS0	MPVER
MP3STA1	AFh	MP3 Status 1	-	-	-	MPFREQ	MPBREQ	-	-	-
MP3DAT	ACH	MP3 Data	MPD7	MPD6	MPD5	MPD4	MPD3	MPD2	MPD1	MPD0
MP3ANC	ADh	MP3 Ancillary Data	AND7	AND6	AND5	AND4	AND3	AND2	AND1	AND0
MP3VOL	9Eh	MP3 Audio Volume Control Left	-	-	-	VOL4	VOL3	VOL2	VOL1	VOL0
MP3VOR	9Fh	MP3 Audio Volume Control Right	-	-	-	VOR4	VOR3	VOR2	VOR1	VOR0
MP3BAS	B4h	MP3 Audio Bass Control	-	-	-	BAS4	BAS3	BAS2	BAS1	BAS0
MP3MED	B5h	MP3 Audio Medium Control	-	-	-	MED4	MED3	MED2	MED1	MED0
MP3TRE	B6h	MP3 Audio Treble Control	-	-	-	TRE4	TRE3	TRE2	TRE1	TRE0
MP3CLK	EBh	MP3 Clock Divider	-	-	-	MPCD4	MPCD3	MPCD2	MPCD1	MPCD0

**Table 38. Audio Interface SFRs**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
AUDCON0	9Ah	Audio Control 0	JUST4	JUST3	JUST2	JUST1	JUST0	POL	DSIZ	HLR
AUDCON1	9Bh	Audio Control 1	SRC	DRQEN	MSREQ	MUDRN	-	DUP1	DUP0	AUDEN
AUDSTA	9Ch	Audio Status	SREQ	UDRN	AUBUSY	-	-	-	-	-
AUDDAT	9Dh	Audio Data	AUD7	AUD6	AUD5	AUD4	AUD3	AUD2	AUD1	AUD0
AUDCLK	ECh	Audio Clock Divider	-	-	-	AUCD4	AUCD3	AUCD2	AUCD1	AUCD0



**Table 39. USB Controller SFRs**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
USBCON	BCh	USB Global Control	USBE	SUSPCLK	SDRMWUP	-	UPRSM	RMWUPE	CONFIG	FADDEN
USBADDR	C6h	USB Address	FEN	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0
USBINT	BDh	USB Global Interrupt	-	-	WUPCPU	EORINT	SOFINT	-	-	SPINT
USBIEN	BEh	USB Global Interrupt Enable	-	-	EWUPCPU	EEORINT	ESOFINT	-	-	ESPINT
UEPNUM	C7h	USB Endpoint Number	-	-	-	-	-	-	EPNUM1	EPNUM0
UEPCONX	D4h	USB Endpoint X Control	EPEN	NAKIEN	NAKOUT	NAKIN	DTGL	EPDIR	EPTYPE1	EPTYPE0
UEPSTAX	CEh	USB Endpoint X Status	DIR	RXOUTB1	STALLRQ	TXRDY	STLCRC	RXSETUP	RXOUTB0	TXCMP
UEPRST	D5h	USB Endpoint Reset	-	-	-	-	-	EP2RST	EP1RST	EP0RST
UEPINT	F8h	USB Endpoint Interrupt	-	-	-	-	-	EP2INT	EP1INT	EP0INT
UEPIEN	C2h	USB Endpoint Interrupt Enable	-	-	-	-	-	EP2INTE	EP1INTE	EP0INTE
UEPDATX	CFh	USB Endpoint X FIFO Data	FDAT7	FDAT6	FDAT5	FDAT4	FDAT3	FDAT2	FDAT1	FDAT0
UBYCTX	E2h	USB Endpoint X Byte Counter	-	BYCT6	BYCT5	BYCT4	BYCT3	BYCT2	BYCT1	BYCT0
UFNUML	BAh	USB Frame Number Low	FNUM7	FNUM6	FNUM5	FNUM4	FNUM3	FNUM2	FNUM1	FNUM0
UFNUMH	BBh	USB Frame Number High	-	-	CRCOK	CRCERR	-	FNUM10	FNUM9	FNUM8
USBCLK	EAh	USB Clock Divider	-	-	-	-	-	-	USBCD1	USBCD0

**Table 40. MMC Controller SFRs**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
MMCON0	E4h	MMC Control 0	DRPTR	DTPTR	CRPTR	CTPTR	MBLOCK	DFMT	RFMT	CRCDIS
MMCON1	E5h	MMC Control 1	BLEN3	BLEN2	BLEN1	BLEN0	DATDIR	DATEN	RESPEN	CMDEN
MMCON2	E6h	MMC Control 2	MMCEN	DCR	CCR	-	-	DATD1	DATD0	FLOWC
MMSTA	DEh	MMC Control and Status	-	-	CBUSY	CRC16S	DATFS	CRC7S	RESPFS	CFLCK
MMINT	E7h	MMC Interrupt	MCBI	EORI	EOCI	EOF1	F2F1	F1F1	F2EI	F1EI
MMMSK	DFh	MMC Interrupt Mask	MCBM	EORM	EOCM	EOFM	F2FM	F1FM	F2EM	F1EM
MMCMD	DDh	MMC Command	MC7	MC6	MC5	MC4	MC3	MC2	MC1	MC0
MMDAT	DCh	MMC Data	MD7	MD6	MD5	MD4	MD3	MD2	MD1	MD0
MMCLK	EDh	MMC Clock Divider	MMCD7	MMCD6	MMCD5	MMCD4	MMCD3	MMCD2	MMCD1	MMCD0

**Table 41. IDE Interface SFR**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
DAT16H	F9h	High Order Data Byte	D15	D14	D13	D12	D11	D10	D9	D8

**Table 42. Serial I/O Port SFRs**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
SCON	98h	Serial Control	FE/SM0	SM1	SM2	REN	TB8	RB8	TI	RI
SBUF	99h	Serial Data Buffer								
SADEN	B9h	Slave Address Mask								
SADDR	A9h	Slave Address								
BDRCON	92h	Baud Rate Control				BRR	TBCK	RBCK	SPD	SRC
BRL	91h	Baud Rate Reload								

**Table 43. SPI Controller SFRs**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
SPCON	C3h	SPI Control	SPR2	SPEN	SSDIS	MSTR	CPOL	CPHA	SPR1	SPR0
SPSTA	C4h	SPI Status	SPIF	WCOL	-	MODF	-	-	-	-
SPDAT	C5h	SPI Data	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0

**Table 44. Two Wire Controller SFRs**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
SSCON	93h	Synchronous Serial Control	SSCR2	SSPE	SSSTA	SSSTO	SSI	SSAA	SSCR1	SSCR0
SSSTA	94h	Synchronous Serial Status	SSC4	SSC3	SSC2	SSC1	SSC0	0	0	0
SSDAT	95h	Synchronous Serial Data	SSD7	SSD6	SSD5	SSD4	SSD3	SSD2	SSD1	SSD0
SSADR	96h	Synchronous Serial Address	SSA7	SSA6	SSA5	SSA4	SSA3	SSA2	SSA1	SSGC

**Table 45. Keyboard Interface SFRs**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
KBCON	A3h	Keyboard Control	KINL3	KINL2	KINL1	KINL0	KINM3	KINM2	KINM1	KINM0
KBSTA	A4h	Keyboard Status	KPDE	-	-	-	KINF3	KINF2	KINF1	KINF0

**Table 46. A/D Controller SFRs**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
ADCON	F3h	ADC Control	-	ADIDL	ADEN	ADEOC	ADSST	-	-	ADCS
ADCLK	F2h	ADC Clock Divider	-	-	-	ADCD4	ADCD3	ADCD2	ADCD1	ADCD0
ADDL	F4h	ADC Data Low Byte	-	-	-	-	-	-	ADAT1	ADAT0
ADDH	F5h	ADC Data High Byte	ADAT9	ADAT8	ADAT7	ADAT6	ADAT5	ADAT4	ADAT3	ADAT2

**Table 47. SFR Addresses and Reset Values**

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h	UEPINT 0000 0000	DAT16H XXXX XXXX		NVERS XXXX XXXX <sup>(2)</sup>					FFh
F0h	B <sup>(1)</sup> 0000 0000		ADCLK 0000 0000	ADCON 0000 0000	ADDL 0000 0000	ADDH 0000 0000			F7h
E8h		PLLCON 0000 1000	USBCLK 0000 0000	MP3CLK 0000 0000	AUDCLK 0000 0000	MMCLK 0000 0000	PLLNDIV 0000 0000	PLLRDIV 0000 0000	EFh
E0h	ACC <sup>(1)</sup> 0000 0000		UBYCTLX 0000 0000		MMCON0 0000 0000	MMCON1 0000 0000	MMCON2 0000 0000	MMINT 0000 0011	E7h
D8h	P5 <sup>(1)</sup> XXXX 1111				MMDAT 1111 1111	MMCMD 1111 1111	MMSTA 0000 0000	MMMSK 1111 1111	DFh
D0h	PSW <sup>(1)</sup> 0000 0000	FCON <sup>(3)</sup> 1111 0000 <sup>(4)</sup>			UEPCONX 1000 0000	UEPRST 0000 0000			D7h
C8h	MP3STA <sup>(1)</sup> 0000 0001						UEPSTAX 0000 0000	UEPDATX XXXX XXXX	CFh
C0h	P4 <sup>(1)</sup> 1111 1111		UEPIEN 0000 0000	SPCON 0001 0100	SPSTA 0000 0000	SPDAT XXXX XXXX	USBADDR 0000 0000	UEPNUM 0000 0000	C7h
B8h	IPL0 <sup>(1)</sup> X000 0000	SADEN 0000 0000	UFNUML 0000 0000	UFNUMH 0000 0000	USBCON 0000 0000	USBINT 0000 0000	USBIEN 0001 0000		BFh
B0h	P3 <sup>(1)</sup> 1111 1111	IEN1 0000 0000	IPL1 0000 0000	IPH1 0000 0000	MP3BAS 0000 0000	MP3MED 0000 0000	MP3TRE 0000 0000	IPH0 X000 0000	B7h
A8h	IEN0 <sup>(1)</sup> 0000 0000	SADDR 0000 0000	MP3CON 0011 1111		MP3DAT 0000 0000	MP3ANC 0000 0000		MP3STA1 0100 0001	AFh
A0h	P2 <sup>(1)</sup> 1111 1111		AUXR1 XXXX 00X0	KBCON 0000 1111	KBSTA 0000 0000		WDRST XXX XXXX	WDTPRG XXXX X000	A7h
98h	SCON 0000 0000	SBUF XXXX XXXX	AUDCON0 0000 1000	AUDCON1 1011 0010	AUDSTA 1100 0000	AUDDAT 1111 1111	MP3VOL 0000 0000	MP3VOR 0000 0000	9Fh
90h	P1 <sup>(1)</sup> 1111 1111	BRL 0000 0000	BDRCON XXX0 0000	SSCON 0000 0000	SSSTA 1111 1000	SSDAT 1111 1111	SSADR 1111 1110		97h
88h	TCON <sup>(1)</sup> 0000 0000	TMOD 0000 0000	TL0 0000 0000	TL1 0000 0000	TH0 0000 0000	TH1 0000 0000	AUXR X000 1101	CKCON 0000 000X <sup>(5)</sup>	8Fh
80h	P0 <sup>(1)</sup> 1111 1111	SP 0000 0111	DPL 0000 0000	DPH 0000 0000				PCON XXXX 0000	87h
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

Reserved

- Notes:
- SFR registers with least significant nibble address equal to 0 or 8 are bit-addressable.
  - NVERS reset value depends on the silicon version: 1000 0100 for AT89C51SND1C product and 0000 0001 for AT83C51SND1C product.
  - FCON register is only available in AT89C51SND1C product.
  - FCON reset value is 00h in case of reset with hardware condition.
  - CKCON reset value depends on the X2B bit (programmed or unprogrammed) in the Hardware Byte.

## Interrupt System

The AT8xC51SND1C, like other control-oriented computer architectures, employ a program interrupt method. This operation branches to a subroutine and performs some service in response to the interrupt. When the subroutine completes, execution resumes at the point where the interrupt occurred. Interrupts may occur as a result of internal AT8xC51SND1C activity (e.g., timer overflow) or at the initiation of electrical signals external to the microcontroller (e.g., keyboard). In all cases, interrupt operation is programmed by the system designer, who determines priority of interrupt service relative to normal code execution and other interrupt service routines. All of the interrupt sources are enabled or disabled by the system designer and may be manipulated dynamically.

A typical interrupt event chain occurs as follows:

- An internal or external device initiates an interrupt-request signal. The AT8xC51SND1C, latches this event into a flag buffer.
- The priority of the flag is compared to the priority of other interrupts by the interrupt handler. A high priority causes the handler to set an interrupt flag.
- This signals the instruction execution unit to execute a context switch. This context switch breaks the current flow of instruction sequences. The execution unit completes the current instruction prior to a save of the program counter (PC) and reloads the PC with the start address of a software service routine.
- The software service routine executes assigned tasks and as a final activity performs a RETI (return from interrupt) instruction. This instruction signals completion of the interrupt, resets the interrupt-in-progress priority and reloads the program counter. Program operation then continues from the original point of interruption.

**Table 48.** Interrupt System Signals

Signal Name	Type	Description	Alternate Function
$\overline{\text{INT0}}$	I	<b>External Interrupt 0</b> See section "External Interrupts", page 39.	P3.2
$\overline{\text{INT1}}$	I	<b>External Interrupt 1</b> See section "External Interrupts", page 39.	P3.3
KIN3:0	I	<b>Keyboard Interrupt Inputs</b> See section "Keyboard Interface", page 179.	P1.3:0

Six interrupt registers are used to control the interrupt system. 2 8-bit registers are used to enable separately the interrupt sources: IEN0 and IEN1 registers (see Table 51 and Table 52).

Four 8-bit registers are used to establish the priority level of the thirteen sources: IPH0, IPL0, IPH1 and IPL1 registers (see Table 53 to Table 56).

## Interrupt System Priorities

Each of the thirteen interrupt sources on the AT8xC51SND1C can be individually programmed to one of four priority levels. This is accomplished by one bit in the Interrupt Priority High registers (IPH0 and IPH1) and one bit in the Interrupt Priority Low registers (IPL0 and IPL1). This provides each interrupt source four possible priority levels according to Table 49.

**Table 49.** Priority Levels

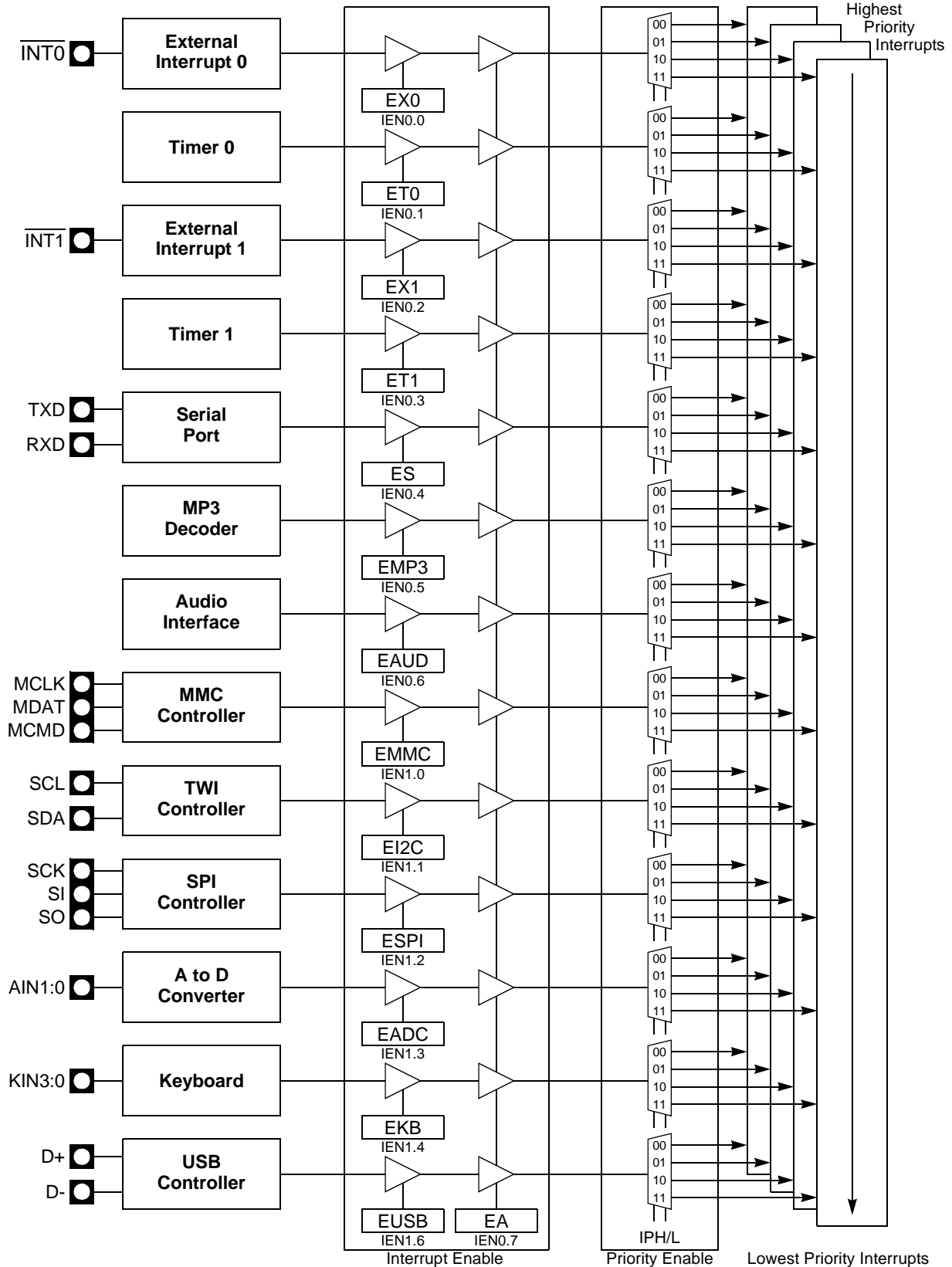
IPHxx	IPLxx	Priority Level
0	0	0 Lowest
0	1	1
1	0	2
1	1	3 Highest

A low-priority interrupt is always interrupted by a higher priority interrupt but not by another interrupt of lower or equal priority. Higher priority interrupts are serviced before lower priority interrupts. The response to simultaneous occurrence of equal priority interrupts is determined by an internal hardware polling sequence detailed in Table 50. Thus, within each priority level there is a second priority structure determined by the polling sequence. The interrupt control system is shown in Figure 21.

**Table 50.** Priority within Same Level

Interrupt Name	Priority Number	Interrupt Address Vectors	Interrupt Request Flag Cleared by Hardware (H) or by Software (S)
$\overline{\text{INT0}}$	1 (Highest Priority)	C:0003h	H if edge, S if level
Timer 0	2	C:000Bh	H
$\overline{\text{INT1}}$	3	C:0013h	H if edge, S if level
Timer 1	4	C:001Bh	H
Serial Port	5	C:0023h	S
MP3 Decoder	6	C:002Bh	S
Audio Interface	7	C:0033h	S
MMC Interface	8	C:003Bh	S
Two Wire Controller	9	C:0043h	S
SPI Controller	10	C:004Bh	S
A to D Converter	11	C:0053h	S
Keyboard	12	C:005Bh	S
Reserved	13	C:0063h	-
USB	14	C:006Bh	S
Reserved	15 (Lowest Priority)	C:0073h	-

Figure 21. Interrupt Control System



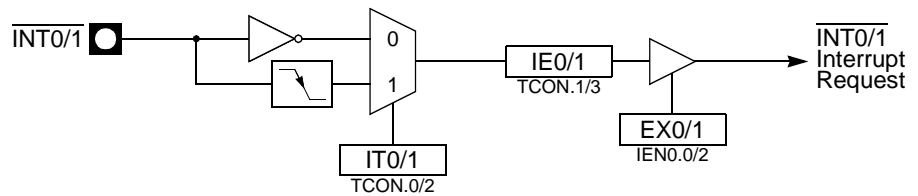
## External Interrupts

### INT1:0 Inputs

External interrupts  $\overline{INT0}$  and  $\overline{INT1}$  ( $\overline{INTn}$ ,  $n = 0$  or  $1$ ) pins may each be programmed to be level-triggered or edge-triggered, dependent upon bits  $IT0$  and  $IT1$  ( $ITn$ ,  $n = 0$  or  $1$ ) in  $TCON$  register as shown in Figure 22. If  $ITn = 0$ ,  $\overline{INTn}$  is triggered by a low level at the pin. If  $ITn = 1$ ,  $\overline{INTn}$  is negative-edge triggered. External interrupts are enabled with bits  $EX0$  and  $EX1$  ( $EXn$ ,  $n = 0$  or  $1$ ) in  $IEN0$ . Events on  $\overline{INTn}$  set the interrupt request flag  $IE_n$  in  $TCON$  register. If the interrupt is edge-triggered, the request flag is cleared by hardware when vectoring to the interrupt service routine. If the interrupt is level-triggered, the interrupt service routine must clear the request flag and the interrupt must be deasserted before the end of the interrupt service routine.

$\overline{INT0}$  and  $\overline{INT1}$  inputs provide both the capability to exit from Power-down mode on low level signals as detailed in section "Exiting Power-down Mode", page 48.

Figure 22.  $\overline{INT1:0}$  Input Circuitry



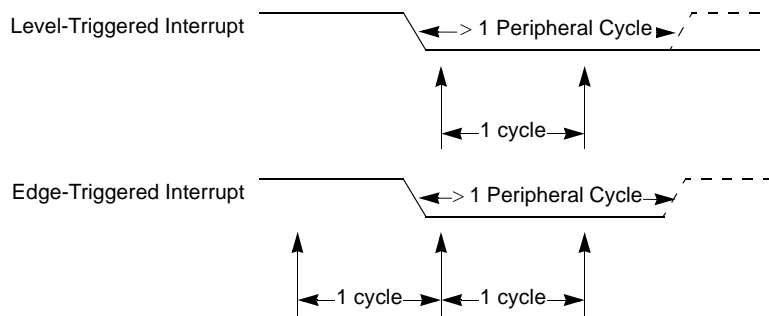
### KIN3:0 Inputs

External interrupts  $KIN0$  to  $KIN3$  provide the capability to connect a matrix keyboard. For detailed information on these inputs, refer to section "Keyboard Interface", page 179.

### Input Sampling

External interrupt pins ( $\overline{INT1:0}$  and  $KIN3:0$ ) are sampled once per peripheral cycle (6 peripheral clock periods) (see Figure 23). A level-triggered interrupt pin held low or high for more than 6 peripheral clock periods (12 oscillator in standard mode or 6 oscillator clock periods in X2 mode) guarantees detection. Edge-triggered external interrupts must hold the request pin low for at least 6 peripheral clock periods.

Figure 23. Minimum Pulse Timings



## Registers

**Table 51.** IEN0 Register  
*IEN0 (S:A8h) – Interrupt Enable Register 0*

7	6	5	4	3	2	1	0
EA	EAUD	EMP3	ES	ET1	EX1	ET0	EX0
Bit Number	Bit Mnemonic	Description					
7	EA	<b>Enable All Interrupt Bit</b> Set to enable all interrupts. Clear to disable all interrupts. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its interrupt enable bit.					
6	EAUD	<b>Audio Interface Interrupt Enable Bit</b> Set to enable audio interface interrupt. Clear to disable audio interface interrupt.					
5	EMP3	<b>MP3 Decoder Interrupt Enable Bit</b> Set to enable MP3 decoder interrupt. Clear to disable MP3 decoder interrupt.					
4	ES	<b>Serial Port Interrupt Enable Bit</b> Set to enable serial port interrupt. Clear to disable serial port interrupt.					
3	ET1	<b>Timer 1 Overflow Interrupt Enable Bit</b> Set to enable timer 1 overflow interrupt. Clear to disable timer 1 overflow interrupt.					
2	EX1	<b>External Interrupt 1 Enable bit</b> Set to enable external interrupt 1. Clear to disable external interrupt 1.					
1	ET0	<b>Timer 0 Overflow Interrupt Enable Bit</b> Set to enable timer 0 overflow interrupt. Clear to disable timer 0 overflow interrupt.					
0	EX0	<b>External Interrupt 0 Enable Bit</b> Set to enable external interrupt 0. Clear to disable external interrupt 0.					

Reset Value = 0000 0000b



**Table 52.** IEN1 Register  
IEN1 (S:B1h) – Interrupt Enable Register 1

7	6	5	4	3	2	1	0
-	EUSB	-	EKB	EADC	ESPI	EI2C	EMMC
Bit Number	Bit Mnemonic	Description					
7	-	<b>Reserved</b> The value read from this bit is always 0. Do not set this bit.					
6	EUSB	<b>USB Interface Interrupt Enable Bit</b> Set this bit to enable <b>USB</b> interrupts. Clear this bit to disable <b>USB</b> interrupts.					
5	-	<b>Reserved</b> The value read from this bit is always 0. Do not set this bit.					
4	EKB	<b>Keyboard Interface Interrupt Enable Bit</b> Set to enable Keyboard interrupt. Clear to disable Keyboard interrupt.					
3	EADC	<b>A to D Converter Interrupt Enable Bit</b> Set to enable ADC interrupt. Clear to disable ADC interrupt.					
2	ESPI	<b>SPI Controller Interrupt Enable Bit</b> Set to enable SPI interrupt. Clear to disable SPI interrupt.					
1	EI2C	<b>Two Wire Controller Interrupt Enable Bit</b> Set to enable Two Wire interrupt. Clear to disable Two Wire interrupt.					
0	EMMC	<b>MMC Interface Interrupt Enable Bit</b> Set to enable MMC interrupt. Clear to disable MMC interrupt.					

Reset Value = 0000 0000b

**Table 53.** IPH0 Register  
IPH0 (S:B7h) – Interrupt Priority High Register 0

7	6	5	4	3	2	1	0
-	IPHAUD	IPHMP3	IPHS	IPHT1	IPHX1	IPHT0	IPHX0
Bit Number	Bit Mnemonic	Description					
7	-	<b>Reserved</b> The value read from this bit is indeterminate. Do not set this bit.					
6	IPHAUD	<b>Audio Interface Interrupt Priority Level MSB</b> Refer to Table 49 for priority level description.					
5	IPHMP3	<b>MP3 Decoder Interrupt Priority Level MSB</b> Refer to Table 49 for priority level description.					
4	IPHS	<b>Serial Port Interrupt Priority Level MSB</b> Refer to Table 49 for priority level description.					
3	IPHT1	<b>Timer 1 Interrupt Priority Level MSB</b> Refer to Table 49 for priority level description.					
2	IPHX1	<b>External Interrupt 1 Priority Level MSB</b> Refer to Table 49 for priority level description.					
1	IPHT0	<b>Timer 0 Interrupt Priority Level MSB</b> Refer to Table 49 for priority level description.					
0	IPHX0	<b>External Interrupt 0 Priority Level MSB</b> Refer to Table 49 for priority level description.					

Reset Value = X000 0000b

**Table 54.** IPH1 Register  
IPH1 (S:B3h) – Interrupt Priority High Register 1

7	6	5	4	3	2	1	0
-	IPHUSB	-	IPHKB	IPHADC	IPHSPI	IPHI2C	IPHMMC
Bit Number	Bit Mnemonic	Description					
7	-	<b>Reserved</b> The value read from this bit is always 0. Do not set this bit.					
6	IPHUSB	<b>USB Interrupt Priority Level MSB</b> Refer to Table 49 for priority level description.					
5	-	<b>Reserved</b> The value read from this bit is always 0. Do not set this bit.					
4	IPHKB	<b>Keyboard Interrupt Priority Level MSB</b> Refer to Table 49 for priority level description.					
3	IPHADC	<b>A to D Converter Interrupt Priority Level MSB</b> Refer to Table 49 for priority level description.					
2	IPHSPI	<b>SPI Interrupt Priority Level MSB</b> Refer to Table 49 for priority level description.					
1	IPHI2C	<b>Two Wire Controller Interrupt Priority Level MSB</b> Refer to Table 49 for priority level description.					
0	IPHMMC	<b>MMC Interrupt Priority Level MSB</b> Refer to Table 49 for priority level description.					

Reset Value = 0000 0000b

**Table 55. IPL0 Register**  
 IPL0 (S:B8h) - Interrupt Priority Low Register 0

7	6	5	4	3	2	1	0
-	IPLAUD	IPLMP3	IPLS	IPLT1	IPLX1	IPLT0	IPLX0
Bit Number	Bit Mnemonic	Description					
7	-	<b>Reserved</b> The value read from this bit is indeterminate. Do not set this bit.					
6	IPLAUD	<b>Audio Interface Interrupt Priority Level LSB</b> Refer to Table 49 for priority level description.					
5	IPLMP3	<b>MP3 Decoder Interrupt Priority Level LSB</b> Refer to Table 49 for priority level description.					
4	IPLS	<b>Serial Port Interrupt Priority Level LSB</b> Refer to Table 49 for priority level description.					
3	IPLT1	<b>Timer 1 Interrupt Priority Level LSB</b> Refer to Table 49 for priority level description.					
2	IPLX1	<b>External Interrupt 1 Priority Level LSB</b> Refer to Table 49 for priority level description.					
1	IPLT0	<b>Timer 0 Interrupt Priority Level LSB</b> Refer to Table 49 for priority level description.					
0	IPLX0	<b>External Interrupt 0 Priority Level LSB</b> Refer to Table 49 for priority level description.					

Reset Value = X000 0000b

**Table 56. IPL1 Register**  
**IPL1 (S:B2h) – Interrupt Priority Low Register 1**

7	6	5	4	3	2	1	0
-	IPLUSB	-	IPLKB	IPLADC	IPLSPI	IPLI2C	IPLMMC
Bit Number	Bit Mnemonic	Description					
7	-	<b>Reserved</b> The value read from this bit is always 0. Do not set this bit.					
6	IPLUSB	<b>USB Interrupt Priority Level LSB</b> Refer to Table 49 for priority level description.					
5	-	<b>Reserved</b> The value read from this bit is always 0. Do not set this bit.					
4	IPLKB	<b>Keyboard Interrupt Priority Level LSB</b> Refer to Table 49 for priority level description.					
3	IPLADC	<b>A to D Converter Interrupt Priority Level LSB</b> Refer to Table 49 for priority level description.					
2	IPLSPI	<b>SPI Interrupt Priority Level LSB</b> Refer to Table 49 for priority level description.					
1	IPLI2C	<b>Two Wire Controller Interrupt Priority Level LSB</b> Refer to Table 49 for priority level description.					
0	IPLMMC	<b>MMC Interrupt Priority Level LSB</b> Refer to Table 49 for priority level description.					

Reset Value = 0000 0000b

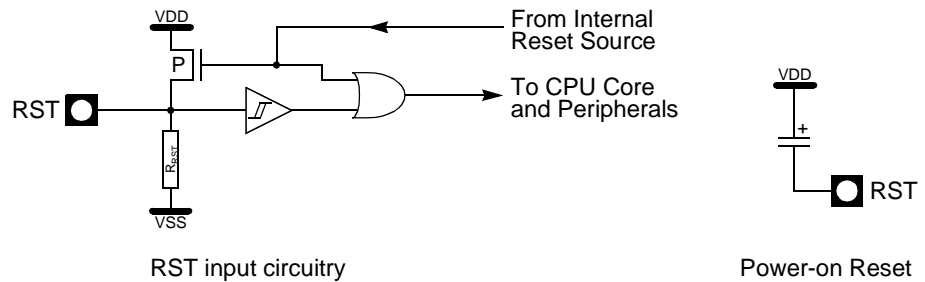
## Power Management

2 power reduction modes are implemented in the AT8xC51SND1C: the Idle mode and the Power-down mode. These modes are detailed in the following sections. In addition to these power reduction modes, the clocks of the core and peripherals can be dynamically divided by 2 using the X2 mode detailed in section “X2 Feature”, page 12.

## Reset

In order to start-up (cold reset) or to restart (warm reset) properly the microcontroller, an high level has to be applied on the RST pin. A bad level leads to a wrong initialization of the internal registers like SFRs, Program Counter... and to unpredictable behavior of the microcontroller. A proper device reset initializes the AT8xC51SND1C and vectors the CPU to address 0000h. RST input has a pull-down resistor allowing power-on reset by simply connecting an external capacitor to  $V_{DD}$  as shown in Figure 24. A warm reset can be applied either directly on the RST pin or indirectly by an internal reset source such as the watchdog timer. Resistor value and input characteristics are discussed in the Section “DC Characteristics” of the AT8xC51SND1C datasheet. The status of the Port pins during reset is detailed in Table 57.

**Figure 24.** Reset Circuitry and Power-On Reset



**Table 57.** Pin Conditions in Special Operating Modes

Mode	Port 0	Port 1	Port 2	Port 3	Port 4	Port 5	MMC	Audio
Reset	Floating	High	High	High	High	High	Floating	<sup>1</sup>
Idle	Data	Data	Data	Data	Data	Data	Data	Data
Power-down	Data	Data	Data	Data	Data	Data	Data	Data

Note: 1. Refer to section “Audio Output Interface”, page 73.

## Cold Reset

2 conditions are required before enabling a CPU start-up:

- $V_{DD}$  must reach the specified  $V_{DD}$  range
- The level on X1 input pin must be outside the specification ( $V_{IH}$ ,  $V_{IL}$ )

If one of these 2 conditions are not met, the microcontroller does not start correctly and can execute an instruction fetch from anywhere in the program space. An active level applied on the RST pin must be maintained till both of the above conditions are met. A reset is active when the level  $V_{IH1}$  is reached and when the pulse width covers the period of time where  $V_{DD}$  and the oscillator are not stabilized. 2 parameters have to be taken into account to determine the reset pulse width:

- $V_{DD}$  rise time,
- Oscillator startup time.

To determine the capacitor value to implement, the highest value of these 2 parameters has to be chosen. Table 58 gives some capacitor values examples for a minimum  $R_{RST}$  of 50 K $\Omega$  and different oscillator startup and  $V_{DD}$  rise times.

**Table 58.** Minimum Reset Capacitor Value for a 50 kΩ Pull-down Resistor<sup>(1)</sup>

Oscillator Start-Up Time	VDD Rise Time		
	1 ms	10 ms	100 ms
5 ms	820 nF	1.2 μF	12 μF
20 ms	2.7 μF	3.9 μF	12 μF

Note: 1. These values assume  $V_{DD}$  starts from 0V to the nominal value. If the time between 2 on/off sequences is too fast, the power-supply de-coupling capacitors may not be fully discharged, leading to a bad reset sequence.

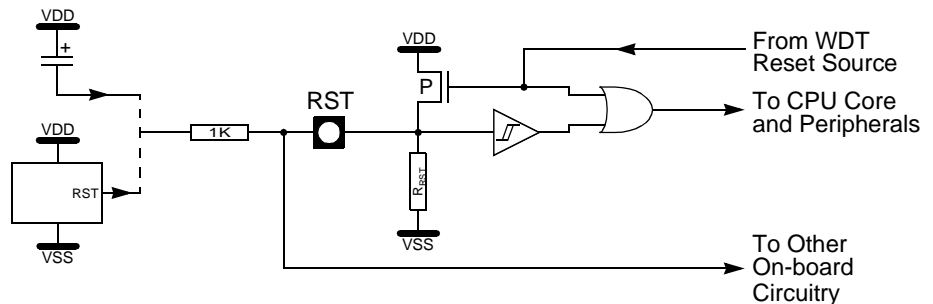
### Warm Reset

To achieve a valid reset, the reset signal must be maintained for at least 2 machine cycles (24 oscillator clock periods) while the oscillator is running. The number of clock periods is mode independent (X2 or X1).

### Watchdog Reset

As detailed in section “Watchdog Timer”, page 59, the WDT generates a 96-clock period pulse on the RST pin. In order to properly propagate this pulse to the rest of the application in case of external capacitor or power-supply supervisor circuit, a 1 kΩ resistor must be added as shown in Figure 25.

**Figure 25.** Reset Circuitry for WDT Reset-out Usage



### Reset Recommendation to Prevent Flash Corruption

An example of bad initialization situation may occur in an instance where the bit ENBOOT in AUXR1 register is initialized from the hardware bit BLJB upon reset. Since this bit allows mapping of the bootloader in the code area, a reset failure can be critical.

If one wants the ENBOOT cleared in order to unmap the boot from the code area (yet due to a bad reset) the bit ENBOOT in SFRs may be set. If the value of Program Counter is accidentally in the range of the boot memory addresses then a Flash access (write or erase) may corrupt the Flash on-chip memory.

It is recommended to use an external reset circuitry featuring power supply monitoring to prevent system malfunction during periods of insufficient power supply voltage (power supply failure, power supply switched off).

### Idle Mode

Idle mode is a power reduction mode that reduces the power consumption. In this mode, program execution halts. Idle mode freezes the clock to the CPU at known states while the peripherals continue to be clocked (refer to section “Oscillator”, page 12). The CPU status before entering Idle mode is preserved, i.e., the program counter and program status word register retain their data for the duration of Idle mode. The contents of the SFRs and RAM are also retained. The status of the Port pins during Idle mode is detailed in Table 57.

## Entering Idle Mode

To enter Idle mode, the user must set the IDL bit in PCON register (see Table 59). The AT8xC51SND1C enters Idle mode upon execution of the instruction that sets IDL bit. The instruction that sets IDL bit is the last instruction executed.

Note: If IDL bit and PD bit are set simultaneously, the AT8xC51SND1C enter Power-down mode. Then it does not go in Idle mode when exiting Power-down mode.

## Exiting Idle Mode

There are 2 ways to exit Idle mode:

1. Generate an enabled interrupt.
  - Hardware clears IDL bit in PCON register which restores the clock to the CPU. Execution resumes with the interrupt service routine. Upon completion of the interrupt service routine, program execution resumes with the instruction immediately following the instruction that activated Idle mode. The general-purpose flags (GF1 and GF0 in PCON register) may be used to indicate whether an interrupt occurred during normal operation or during Idle mode. When Idle mode is exited by an interrupt, the interrupt service routine may examine GF1 and GF0.
2. Generate a reset.
  - A logic high on the RST pin clears IDL bit in PCON register directly and asynchronously. This restores the clock to the CPU. Program execution momentarily resumes with the instruction immediately following the instruction that activated the Idle mode and may continue for a number of clock cycles before the internal reset algorithm takes control. Reset initializes the AT8xC51SND1C and vectors the CPU to address C:0000h.

Note: During the time that execution resumes, the internal RAM cannot be accessed; however, it is possible for the Port pins to be accessed. To avoid unexpected outputs at the Port pins, the instruction immediately following the instruction that activated Idle mode should not write to a Port pin or to the external RAM.

## Power-down Mode

The Power-down mode places the AT8xC51SND1C in a very low power state. Power-down mode stops the oscillator and freezes all clocks at known states (refer to the Section "Oscillator", page 12). The CPU status prior to entering Power-down mode is preserved, i.e., the program counter, program status word register retain their data for the duration of Power-down mode. In addition, the SFRs and RAM contents are preserved. The status of the Port pins during Power-down mode is detailed in Table 57.

Note:  $V_{DD}$  may be reduced to as low as  $V_{RET}$  during Power-down mode to further reduce power dissipation. Notice, however, that  $V_{DD}$  is not reduced until Power-down mode is invoked.

## Entering Power-down Mode

To enter Power-down mode, set PD bit in PCON register. The AT8xC51SND1C enters the Power-down mode upon execution of the instruction that sets PD bit. The instruction that sets PD bit is the last instruction executed.

## Exiting Power-down Mode

If  $V_{DD}$  was reduced during the Power-down mode, do not exit Power-down mode until  $V_{DD}$  is restored to the normal operating level.

There are 2 ways to exit the Power-down mode:

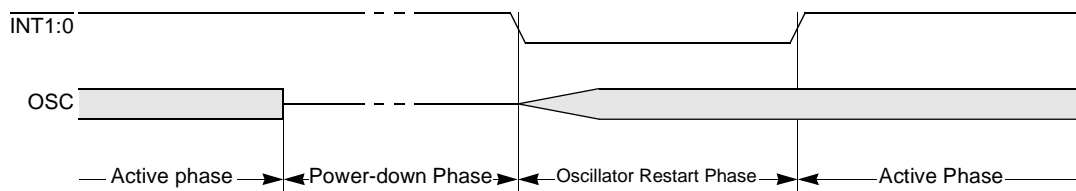
1. Generate an enabled external interrupt.
  - The AT8xC51SND1C provides capability to exit from Power-down using  $\overline{INT0}$ ,  $\overline{INT1}$ , and KIN3:0 inputs. In addition, using KIN input provides high or low level exit capability (see section "Keyboard Interface", page 179). Hardware clears PD bit in PCON register which starts the oscillator and restores the clocks to the CPU and peripherals. Using  $\overline{INTn}$  input, execution



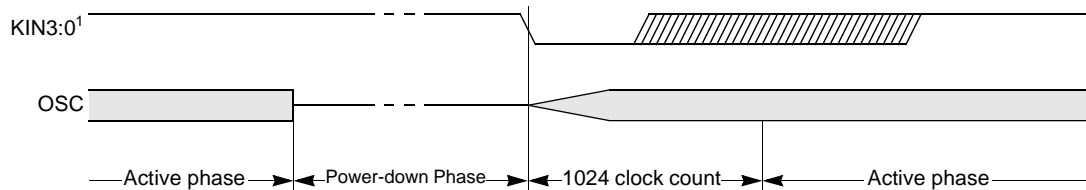
resumes when the input is released (see Figure 26) while using KINx input, execution resumes after counting 1024 clock ensuring the oscillator is restarted properly (see Figure 27). This behavior is necessary for decoding the key while it is still pressed. In both cases, execution resumes with the interrupt service routine. Upon completion of the interrupt service routine, program execution resumes with the instruction immediately following the instruction that activated Power-down mode.

- Note:
1. The external interrupt used to exit Power-down mode must be configured as level sensitive (INT0 and INT1) and must be assigned the highest priority. In addition, the duration of the interrupt must be long enough to allow the oscillator to stabilize. The execution will only resume when the interrupt is deasserted.
  2. Exit from power-down by external interrupt does not affect the SFRs nor the internal RAM content.

**Figure 26.** Power-down Exit Waveform Using  $\overline{\text{INT1:0}}$



**Figure 27.** Power-down Exit Waveform Using KIN3:0



Note: 1. KIN3:0 can be high or low-level triggered.

2. Generate a reset.
  - A logic high on the RST pin clears PD bit in PCON register directly and asynchronously. This starts the oscillator and restores the clock to the CPU and peripherals. Program execution momentarily resumes with the instruction immediately following the instruction that activated Power-down mode and may continue for a number of clock cycles before the internal reset algorithm takes control. Reset initializes the AT8xC51SND1C and vectors the CPU to address 0000h.

- Notes:
1. During the time that execution resumes, the internal RAM cannot be accessed; however, it is possible for the Port pins to be accessed. To avoid unexpected outputs at the Port pins, the instruction immediately following the instruction that activated the Power-down mode should not write to a Port pin or to the external RAM.
  2. Exit from power-down by reset redefines all the SFRs, but does not affect the internal RAM content.

## Registers

**Table 59.** PCON Register

PCON (S:87h) – Power Configuration Register

	7	6	5	4	3	2	1	0
	-	-	-	-	GF1	GF0	PD	IDL

Bit Number	Bit Mnemonic	Description
7 - 4	-	<b>Reserved</b> The value read from these bits is indeterminate. Do not set these bits.
3	GF1	<b>General-purpose flag 1</b> One use is to indicate whether an interrupt occurred during normal operation or during Idle mode.
2	GF0	<b>General-purpose flag 0</b> One use is to indicate whether an interrupt occurred during normal operation or during Idle mode.
1	PD	<b>Power-down Mode bit</b> Cleared by hardware when an interrupt or reset occurs. Set to activate the Power-down mode. If IDL and PD are both set, PD takes precedence.
0	IDL	<b>Idle Mode bit</b> Cleared by hardware when an interrupt or reset occurs. Set to activate the Idle mode. If IDL and PD are both set, PD takes precedence.

Reset Value = XXXX 0000b

## Timers/Counters

The AT8xC51SND1C implement 2 general-purpose, 16-bit Timers/Counters. They are identified as Timer 0 and Timer 1, and can be independently configured to operate in a variety of modes as a Timer or as an event Counter. When operating as a Timer, the Timer/Counter runs for a programmed length of time, then issues an interrupt request. When operating as a Counter, the Timer/Counter counts negative transitions on an external pin. After a preset number of counts, the Counter issues an interrupt request.

The various operating modes of each Timer/Counter are described in the following sections.

### Timer/Counter Operations

For instance, a basic operation is Timer registers THx and TLx ( $x = 0, 1$ ) connected in cascade to form a 16-bit Timer. Setting the run control bit (TRx) in TCON register (see Table 60) turns the Timer on by allowing the selected input to increment TLx. When TLx overflows it increments THx; when THx overflows it sets the Timer overflow flag (TFx) in TCON register. Setting the TRx does not clear the THx and TLx Timer registers. Timer registers can be accessed to obtain the current count or to enter preset values. They can be read at any time but TRx bit must be cleared to preset their values, otherwise, the behavior of the Timer/Counter is unpredictable.

The C/Tx# control bit selects Timer operation or Counter operation by selecting the divided-down peripheral clock or external pin Tx as the source for the counted signal. TRx bit must be cleared when changing the mode of operation, otherwise the behavior of the Timer/Counter is unpredictable.

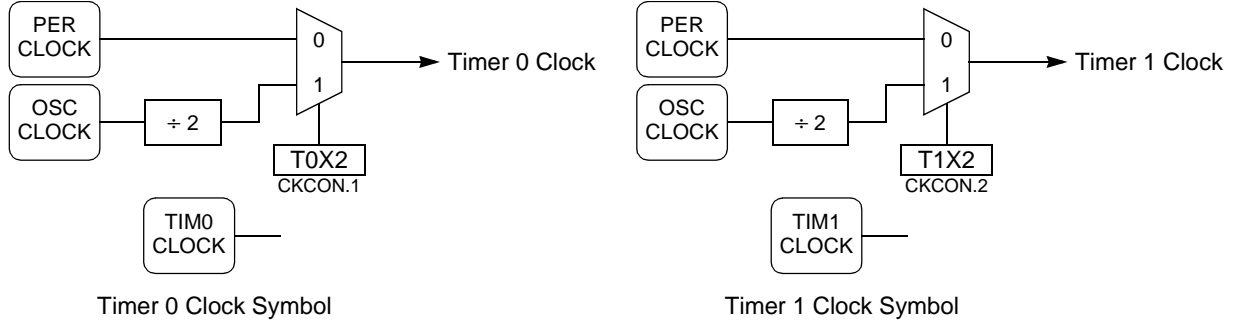
For Timer operation ( $C/Tx\# = 0$ ), the Timer register counts the divided-down peripheral clock. The Timer register is incremented once every peripheral cycle (6 peripheral clock periods). The Timer clock rate is  $F_{PER}/6$ , i.e.,  $F_{OSC}/12$  in standard mode or  $F_{OSC}/6$  in X2 mode.

For Counter operation ( $C/Tx\# = 1$ ), the Timer register counts the negative transitions on the Tx external input pin. The external input is sampled every peripheral cycles. When the sample is high in one cycle and low in the next one, the Counter is incremented. Since it takes 2 cycles (12 peripheral clock periods) to recognize a negative transition, the maximum count rate is  $F_{PER}/12$ , i.e.,  $F_{OSC}/24$  in standard mode or  $F_{OSC}/12$  in X2 mode. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full peripheral cycle.

### Timer Clock Controller

As shown in Figure 28, the Timer 0 (FT0) and Timer 1 (FT1) clocks are derived from either the peripheral clock ( $F_{PER}$ ) or the oscillator clock ( $F_{OSC}$ ) depending on the T0X2 and T1X2 bits in CKCON register. These clocks are issued from the Clock Controller block as detailed in Section "Clock Controller", page 12. When T0X2 or T1X2 bit is set, the Timer 0 or Timer 1 clock frequency is fixed and equal to the oscillator clock frequency divided by 2. When cleared, the Timer clock frequency is equal to the oscillator clock frequency divided by 2 in standard mode or to the oscillator clock frequency in X2 mode.

**Figure 28.** Timer 0 and Timer 1 Clock Controller and Symbols



## Timer 0

Timer 0 functions as either a Timer or event Counter in four modes of operation. Figure 29 through Figure 35 show the logical configuration of each mode.

Timer 0 is controlled by the four lower bits of TMOD register (see Table 61) and bits 0, 1, 4 and 5 of TCON register (see Table 60). TMOD register selects the method of Timer gating (GATE0), Timer or Counter operation (C/T0#) and mode of operation (M10 and M00). TCON register provides Timer 0 control functions: overflow flag (TF0), run control bit (TR0), interrupt flag (IE0) and interrupt type control bit (IT0).

For normal Timer operation (GATE0 = 0), setting TR0 allows TL0 to be incremented by the selected input. Setting GATE0 and TR0 allows external pin INT0 to control Timer operation.

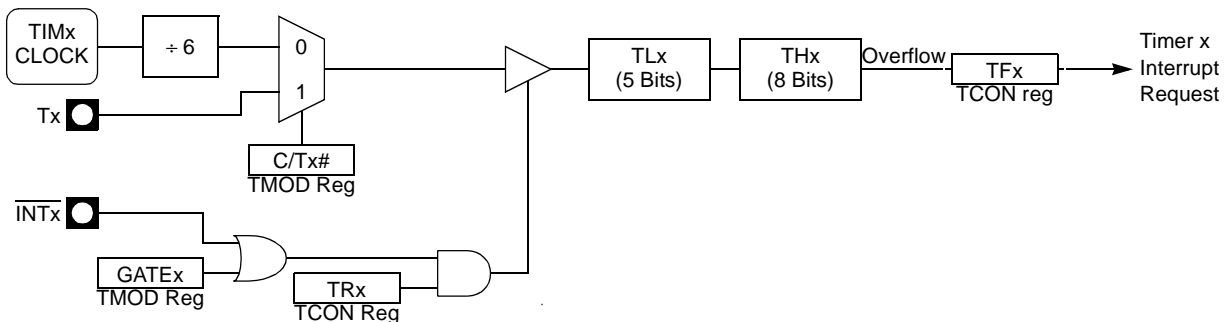
Timer 0 overflow (count rolls over from all 1s to all 0s) sets TF0 flag generating an interrupt request.

It is important to stop Timer/Counter before changing mode.

### Mode 0 (13-bit Timer)

Mode 0 configures Timer 0 as a 13-bit Timer which is set up as an 8-bit Timer (TH0 register) with a modulo 32 prescaler implemented with the lower five bits of TL0 register (see Figure 29). The upper three bits of TL0 register are indeterminate and should be ignored. Prescaler overflow increments TH0 register. Figure 30 gives the overflow period calculation formula.

**Figure 29.** Timer/Counter x (x = 0 or 1) in Mode 0



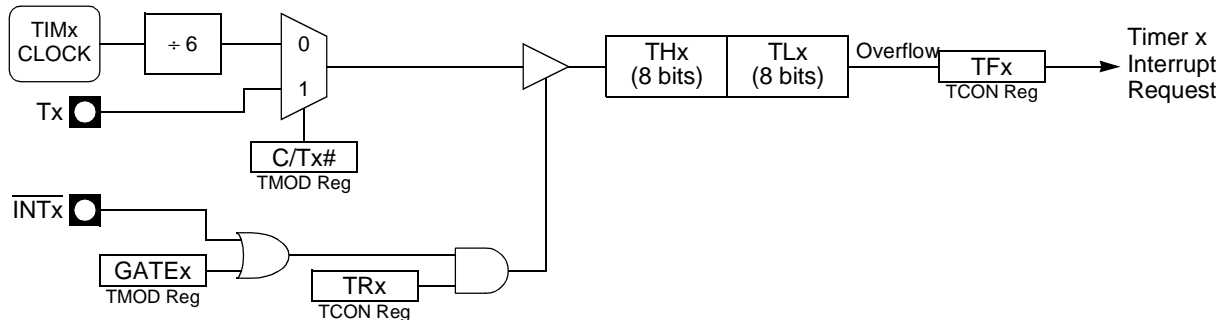
**Figure 30.** Mode 0 Overflow Period Formula

$$TF_{xPER} = \frac{6 \cdot (16384 - (TH_x, TL_x))}{F_{TIM_x}}$$

## Mode 1 (16-bit Timer)

Mode 1 configures Timer 0 as a 16-bit Timer with TH0 and TL0 registers connected in cascade (see Figure 31). The selected input increments TL0 register. Figure 32 gives the overflow period calculation formula when in timer mode.

**Figure 31.** Timer/Counter x (x = 0 or 1) in Mode 1



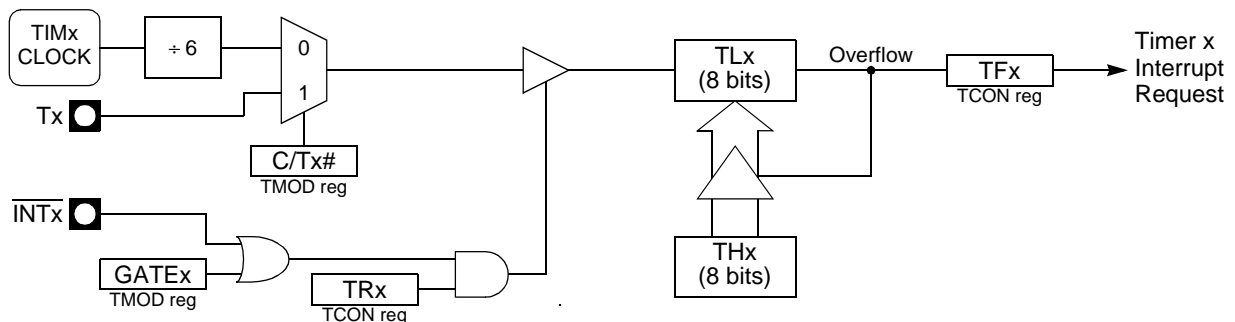
**Figure 32.** Mode 1 Overflow Period Formula

$$TF_{xPER} = \frac{6 \cdot (65536 - (TH_x, TL_x))}{F_{TIM_x}}$$

## Mode 2 (8-bit Timer with Auto-Reload)

Mode 2 configures Timer 0 as an 8-bit Timer (TL0 register) that automatically reloads from TH0 register (see Table 62). TL0 overflow sets TF0 flag in TCON register and reloads TL0 with the contents of TH0, which is preset by software. When the interrupt request is serviced, hardware clears TF0. The reload leaves TH0 unchanged. The next reload value may be changed at any time by writing it to TH0 register. Figure 34 gives the autoreload period calculation formula when in timer mode.

**Figure 33.** Timer/Counter x (x = 0 or 1) in Mode 2



**Figure 34.** Mode 2 Autoreload Period Formula

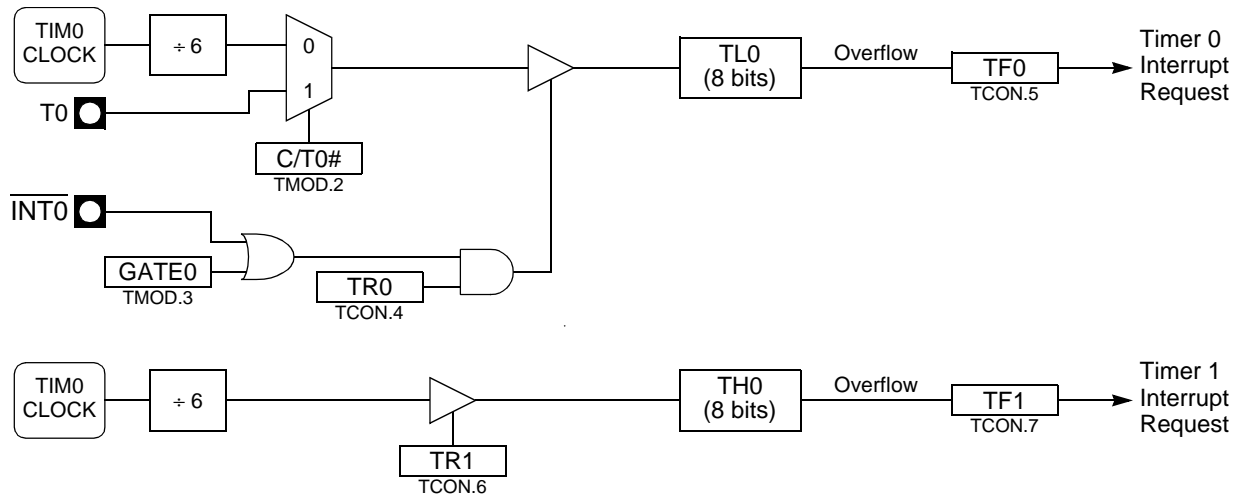
$$TF_{xPER} = \frac{6 \cdot (256 - TH_x)}{F_{TIM_x}}$$

## Mode 3 (2 8-bit Timers)

Mode 3 configures Timer 0 such that registers TL0 and TH0 operate as separate 8-bit Timers (see Figure 35). This mode is provided for applications requiring an additional 8-bit Timer or Counter. TL0 uses the Timer 0 control bits C/T0# and GATE0 in TMOD register, and TR0 and TF0 in TCON register in the normal manner. TH0 is locked into a Timer function (counting  $F_{TF1}/6$ ) and takes over use of the Timer 1 interrupt (TF1) and run control (TR1) bits. Thus, operation of Timer 1 is restricted when Timer 0 is in mode

3. Figure 34 gives the autoreload period calculation formulas for both TF0 and TF1 flags.

**Figure 35.** Timer/Counter 0 in Mode 3: 2 8-bit Counters



**Figure 36.** Mode 3 Overflow Period Formula

$$TF0_{PER} = \frac{6 \cdot (256 - TL0)}{F_{TIM0}}$$

$$TF1_{PER} = \frac{6 \cdot (256 - TH0)}{F_{TIM0}}$$

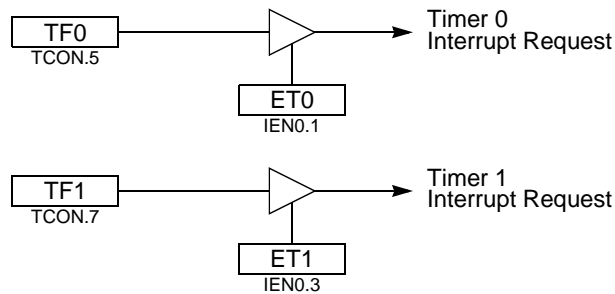
## Timer 1

Timer 1 is identical to Timer 0 except for Mode 3 which is a hold-count mode. The following comments help to understand the differences:

- Timer 1 functions as either a Timer or event Counter in three modes of operation. Figure 29 through Figure 33 show the logical configuration for modes 0, 1, and 2. Timer 1's mode 3 is a hold-count mode.
- Timer 1 is controlled by the four high-order bits of TMOD register (see Figure 61) and bits 2, 3, 6 and 7 of TCON register (see Figure 60). TMOD register selects the method of Timer gating (GATE1), Timer or Counter operation (C/T1#) and mode of operation (M11 and M01). TCON register provides Timer 1 control functions: overflow flag (TF1), run control bit (TR1), interrupt flag (IE1) and interrupt type control bit (IT1).
- Timer 1 can serve as the Baud Rate Generator for the Serial Port. Mode 2 is best suited for this purpose.
- For normal Timer operation (GATE1 = 0), setting TR1 allows TL1 to be incremented by the selected input. Setting GATE1 and TR1 allows external pin INT1 to control Timer operation.
- Timer 1 overflow (count rolls over from all 1s to all 0s) sets the TF1 flag generating an interrupt request.
- When Timer 0 is in mode 3, it uses Timer 1's overflow flag (TF1) and run control bit (TR1). For this situation, use Timer 1 only for applications that do not require an interrupt (such as a Baud Rate Generator for the Serial Port) and switch Timer 1 in and out of mode 3 to turn it off and on.
- It is important to stop the Timer/Counter before changing modes.

- Mode 0 (13-bit Timer)** Mode 0 configures Timer 1 as a 13-bit Timer, which is set up as an 8-bit Timer (TH1 register) with a modulo-32 prescaler implemented with the lower 5 bits of the TL1 register (see Figure 29). The upper 3 bits of TL1 register are ignored. Prescaler overflow increments TH1 register.
- Mode 1 (16-bit Timer)** Mode 1 configures Timer 1 as a 16-bit Timer with TH1 and TL1 registers connected in cascade (see Figure 31). The selected input increments TL1 register.
- Mode 2 (8-bit Timer with Auto-Reload)** Mode 2 configures Timer 1 as an 8-bit Timer (TL1 register) with automatic reload from TH1 register on overflow (see Figure 33). TL1 overflow sets TF1 flag in TCON register and reloads TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.
- Mode 3 (Halt)** Placing Timer 1 in mode 3 causes it to halt and hold its count. This can be used to halt Timer 1 when TR1 run control bit is not available i.e. when Timer 0 is in mode 3.
- Interrupt** Each Timer handles one interrupt source that is the timer overflow flag TF0 or TF1. This flag is set every time an overflow occurs. Flags are cleared when vectoring to the Timer interrupt routine. Interrupts are enabled by setting ETx bit in IEN0 register. This assumes interrupts are globally enabled by setting EA bit in IEN0 register.

**Figure 37.** Timer Interrupt System



## Registers

**Table 60.** TCON Register

TCON (S:88h) – Timer/Counter Control Register

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Bit Number	Bit Mnemonic	Description					
7	TF1	<b>Timer 1 Overflow Flag</b> Cleared by hardware when processor vectors to interrupt routine. Set by hardware on Timer/Counter overflow, when Timer 1 register overflows.					
6	TR1	<b>Timer 1 Run Control Bit</b> Clear to turn off Timer/Counter 1. Set to turn on Timer/Counter 1.					
5	TF0	<b>Timer 0 Overflow Flag</b> Cleared by hardware when processor vectors to interrupt routine. Set by hardware on Timer/Counter overflow, when Timer 0 register overflows.					
4	TR0	<b>Timer 0 Run Control Bit</b> Clear to turn off Timer/Counter 0. Set to turn on Timer/Counter 0.					
3	IE1	<b>Interrupt 1 Edge Flag</b> Cleared by hardware when interrupt is processed if edge-triggered (see IT1). Set by hardware when external interrupt is detected on INT1 pin.					
2	IT1	<b>Interrupt 1 Type Control Bit</b> Clear to select low level active (level triggered) for external interrupt 1 ( $\overline{\text{INT1}}$ ). Set to select falling edge active (edge triggered) for external interrupt 1.					
1	IE0	<b>Interrupt 0 Edge Flag</b> Cleared by hardware when interrupt is processed if edge-triggered (see IT0). Set by hardware when external interrupt is detected on INT0 pin.					
0	IT0	<b>Interrupt 0 Type Control Bit</b> Clear to select low level active (level triggered) for external interrupt 0 ( $\overline{\text{INT0}}$ ). Set to select falling edge active (edge triggered) for external interrupt 0.					

Reset Value = 0000 0000b



**Table 61.** TMOD Register  
 TMOD (S:89h) – Timer/Counter Mode Control Register

	7	6	5	4	3	2	1	0
	<b>GATE1</b>	<b>C/T1#</b>	<b>M11</b>	<b>M01</b>	<b>GATE0</b>	<b>C/T0#</b>	<b>M10</b>	<b>M00</b>

Bit Number	Bit Mnemonic	Description	
7	GATE1	<b>Timer 1 Gating Control Bit</b> Clear to enable Timer 1 whenever TR1 bit is set. Set to enable Timer 1 only while INT1 pin is high and TR1 bit is set.	
6	C/T1#	<b>Timer 1 Counter/Timer Select Bit</b> Clear for Timer operation: Timer 1 counts the divided-down system clock. Set for Counter operation: Timer 1 counts negative transitions on external pin T1.	
5	M11	<b>Timer 1 Mode Select Bits</b> <u>M11 M01</u> <u>Operating mode</u> 0 0      Mode 0: 8-bit Timer/Counter (TH1) with 5-bit prescaler (TL1). 0 1      Mode 1: 16-bit Timer/Counter. 1 0      Mode 2: 8-bit auto-reload Timer/Counter (TL1). <sup>(1)</sup> 1 1      Mode 3: Timer 1 halted. Retains count.	
4	M01		
3	GATE0		<b>Timer 0 Gating Control Bit</b> Clear to enable Timer 0 whenever TR0 bit is set. Set to enable Timer/Counter 0 only while INT0 pin is high and TR0 bit is set.
2	C/T0#		<b>Timer 0 Counter/Timer Select Bit</b> Clear for Timer operation: Timer 0 counts the divided-down system clock. Set for Counter operation: Timer 0 counts negative transitions on external pin T0.
1	M10	<b>Timer 0 Mode Select Bit</b> <u>M10 M00</u> <u>Operating mode</u> 0 0      Mode 0: 8-bit Timer/Counter (TH0) with 5-bit prescaler (TL0). 0 1      Mode 1: 16-bit Timer/Counter. 1 0      Mode 2: 8-bit auto-reload Timer/Counter (TL0). <sup>(2)</sup> 1 1      Mode 3: TL0 is an 8-bit Timer/Counter. TH0 is an 8-bit Timer using Timer 1's TR0 and TF0 bits.	
0	M00		

Notes: 1. Reloaded from TH1 at overflow.  
 2. Reloaded from TH0 at overflow.

Reset Value = 0000 0000b

**Table 62.** TH0 Register  
 TH0 (S:8Ch) – Timer 0 High Byte Register

	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	-

Bit Number	Bit Mnemonic	Description
7:0		<b>High Byte of Timer 0</b>

Reset Value = 0000 0000b

**Table 63.** TL0 Register

TL0 (S:8Ah) – Timer 0 Low Byte Register

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

Bit Number	Bit Mnemonic	Description
7:0		Low Byte of Timer 0

Reset Value = 0000 0000b

**Table 64.** TH1 Register

TH1 (S:8Dh) – Timer 1 High Byte Register

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

Bit Number	Bit Mnemonic	Description
7:0		High Byte of Timer 1

Reset Value = 0000 0000b

**Table 65.** TL1 Register

TL1 (S:8Bh) – Timer 1 Low Byte Register

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-

Bit Number	Bit Mnemonic	Description
7:0		Low Byte of Timer 1

Reset Value = 0000 0000b

## Watchdog Timer

The AT8xC51SND1C implement a hardware Watchdog Timer (WDT) that automatically resets the chip if it is allowed to time out. The WDT provides a means of recovering from routines that do not complete successfully due to software or hardware malfunctions.

## Description

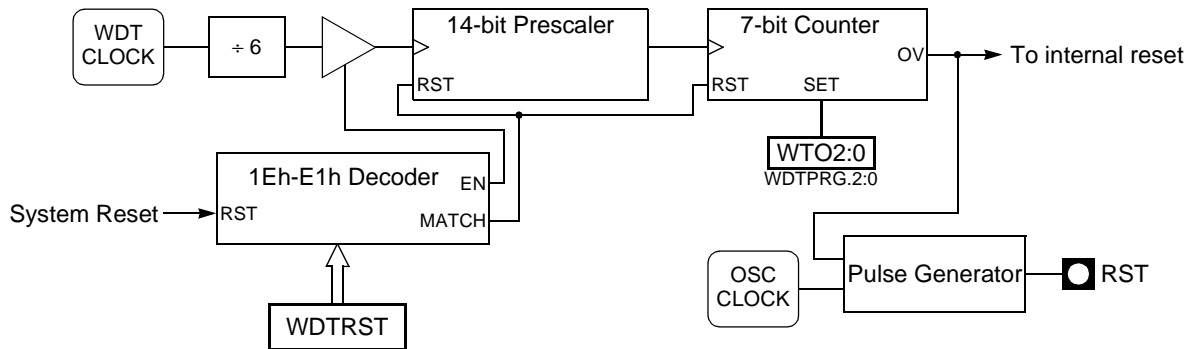
The WDT consists of a 14-bit prescaler followed by a 7-bit programmable counter. As shown in Figure 38, the 14-bit prescaler is fed by the WDT clock detailed in Section "Watchdog Clock Controller", page 59.

The Watchdog Timer Reset register (WDTRST, see Table 67) provides control access to the WDT, while the Watchdog Timer Program register (WDTPRG, see Figure 41) provides time-out period programming.

Three operations control the WDT:

- Chip reset clears and disables the WDT.
- Programming the time-out value to the WDTPRG register.
- Writing a specific 2-Byte sequence to the WDTRST register clears and enables the WDT.

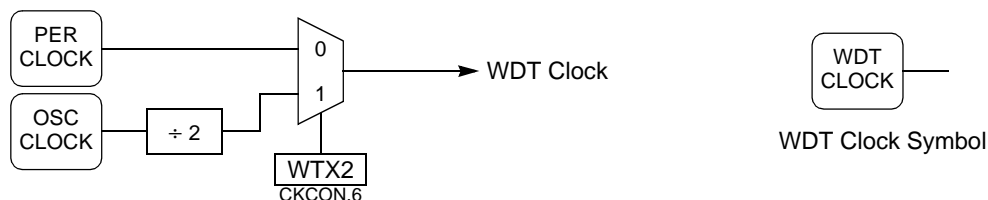
**Figure 38.** WDT Block Diagram



## Watchdog Clock Controller

As shown in Figure 39 the WDT clock ( $F_{WDT}$ ) is derived from either the peripheral clock ( $F_{PER}$ ) or the oscillator clock ( $F_{OSC}$ ) depending on the WTX2 bit in CKCON register. These clocks are issued from the Clock Controller block as detailed in Section "Clock Controller", page 12. When WTX2 bit is set, the WDT clock frequency is fixed and equal to the oscillator clock frequency divided by 2. When cleared, the WDT clock frequency is equal to the oscillator clock frequency divided by 2 in standard mode or to the oscillator clock frequency in X2 mode.

**Figure 39.** WDT Clock Controller and Symbol



## Watchdog Operation

After reset, the WDT is disabled. The WDT is enabled by writing the sequence 1Eh and E1h into the WDTRST register. As soon as it is enabled, there is no way except the chip reset to disable it. If it is not cleared using the previous sequence, the WDT overflows and forces a chip reset. This overflow generates a high level 96 oscillator periods pulse on the RST pin to globally reset the application (refer to Section “Power Management”, page 46).

The WDT time-out period can be adjusted using WTO2:0 bits located in the WDTPRG register accordingly to the formula shown in Figure 40. In this formula, WTOval represents the decimal value of WTO2:0 bits. Table 66 reports the time-out period depending on the WDT frequency.

**Figure 40.** WDT Time-Out Formula

$$WDT_{TO} = \frac{6 \cdot ((2^{14} \cdot 2^{WTOval}) - 1)}{F_{WDT}}$$

**Table 66.** WDT Time-Out Computation

WTO2	WTO1	WTO0	F <sub>WDT</sub> (ms)					
			6 MHz <sup>(1)</sup>	8 MHz <sup>(1)</sup>	10 MHz <sup>(1)</sup>	12 MHz <sup>(2)</sup>	16 MHz <sup>(2)</sup>	20 MHz <sup>(2)</sup>
0	0	0	16.38	12.28	9.83	8.19	6.14	4.92
0	0	1	32.77	24.57	19.66	16.38	12.28	9.83
0	1	0	65.54	49.14	39.32	32.77	24.57	19.66
0	1	1	131.07	98.28	78.64	65.54	49.14	39.32
1	0	0	262.14	196.56	157.29	131.07	98.28	78.64
1	0	1	524.29	393.1	314.57	262.14	196.56	157.29
1	1	0	1049	786.24	629.15	524.29	393.12	314.57
1	1	1	2097	1572	1258	1049	786.24	629.15

- Notes:
1. These frequencies are achieved in X1 mode or in X2 mode when WTX2 = 1:  
F<sub>WDT</sub> = F<sub>OSC</sub> ÷ 2.
  2. These frequencies are achieved in X2 mode when WTX2 = 0: F<sub>WDT</sub> = F<sub>OSC</sub>.

### WDT Behavior during Idle and Power-down Modes

Operation of the WDT during power reduction modes deserves special attention.

The WDT continues to count while the AT8xC51SND1C is in Idle mode. This means that you must dedicate some internal or external hardware to service the WDT during Idle mode. One approach is to use a peripheral Timer to generate an interrupt request when the Timer overflows. The interrupt service routine then clears the WDT, reloads the peripheral Timer for the next service period and puts the AT8xC51SND1C back into Idle mode.

The Power-down mode stops all phase clocks. This causes the WDT to stop counting and to hold its count. The WDT resumes counting from where it left off if the Power-down mode is terminated by INT0, INT1 or keyboard interrupt. To ensure that the WDT does not overflow shortly after exiting the Power-down mode, it is recommended to clear the WDT just before entering Power-down mode.

The WDT is cleared and disabled if the Power-down mode is terminated by a reset.

## Registers

**Table 67.** WDTRST Register

WDTRST (S:A6h Write only) – Watchdog Timer Reset Register

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-
Bit Number	Bit Mnemonic	Description					
7 - 0	-	Watchdog Control Value					

Reset Value = XXXX XXXXb

**Figure 41.** WDTPRG Register

WDTPRG (S:A7h) – Watchdog Timer Program Register

7	6	5	4	3	2	1	0
-	-	-	-	-	WTO2	WTO1	WTO0
Bit Number	Bit Mnemonic	Description					
7 - 3	-	<b>Reserved</b> The value read from these bits is indeterminate. Do not set these bits.					
2 - 0	WTO2:0	<b>Watchdog Timer Time-Out Selection Bits</b> Refer to Table 66 for time-out periods.					

Reset Value = XXXX X000b

## MP3 Decoder

The AT8xC51SND1C implement a MPEG I/II audio layer 3 decoder better known as MP3 decoder.

In MPEG I (ISO 11172-3) three layers of compression have been standardized supporting three sampling frequencies: 48, 44.1, and 32 kHz. Among these layers, layer 3 allows highest compression rate of about 12:1 while still maintaining CD audio quality. For example, 3 minutes of CD audio (16-bit PCM, 44.1 kHz) data, which needs about 32M bytes of storage, can be encoded into only 2.7M bytes of MPEG I audio layer 3 data.

In MPEG II (ISO 13818-3), three additional sampling frequencies: 24, 22.05, and 16 kHz are supported for low bit rates applications.

The AT8xC51SND1C can decode in real-time the MPEG I audio layer 3 encoded data into a PCM audio data, and also supports MPEG II audio layer 3 additional frequencies.

Additional features are supported by the AT8xC51SND1C MP3 decoder such as volume control, bass, medium, and treble controls, bass boost effect and ancillary data extraction.

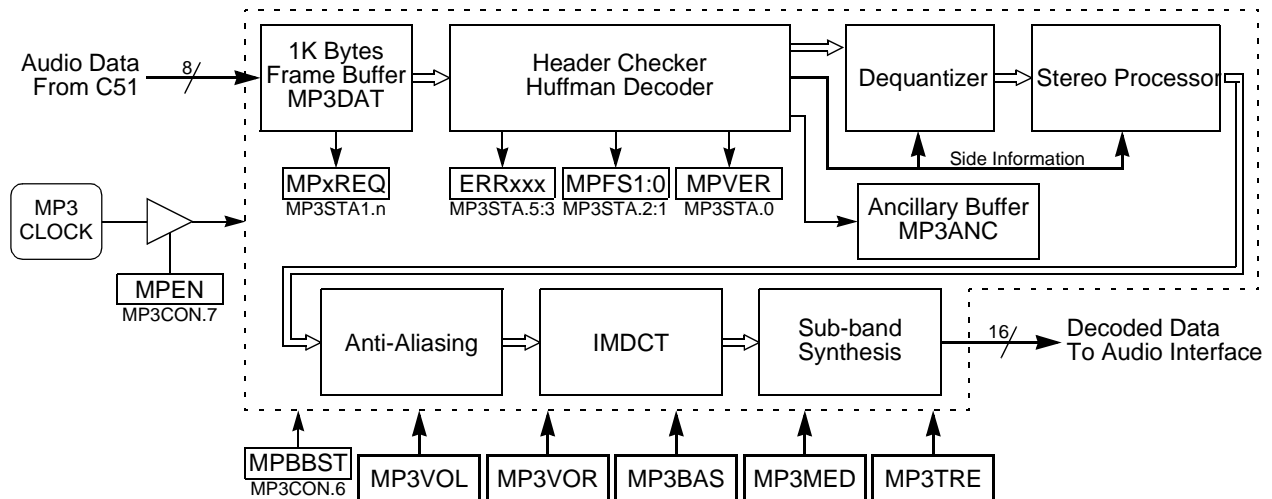
## Decoder

### Description

The C51 core interfaces to the MP3 decoder through nine special function registers: MP3CON, the MP3 Control register (see Table 72); MP3STA, the MP3 Status register (see Table 73); MP3DAT, the MP3 Data register (see Table 74); MP3ANC, the Ancillary Data register (see Table 76); MP3VOL and MP3VOR, the MP3 Volume Left and Right Control registers (see Table 77 and Table 78); MP3BAS, MP3MED, and MP3TRE, the MP3 Bass, Medium, and Treble Control registers (see Table 79, Table 80, and Table 81); and MPCLK, the MP3 Clock Divider register (see Table 82).

Figure 42 shows the MP3 decoder block diagram.

Figure 42. MP3 Decoder Block Diagram

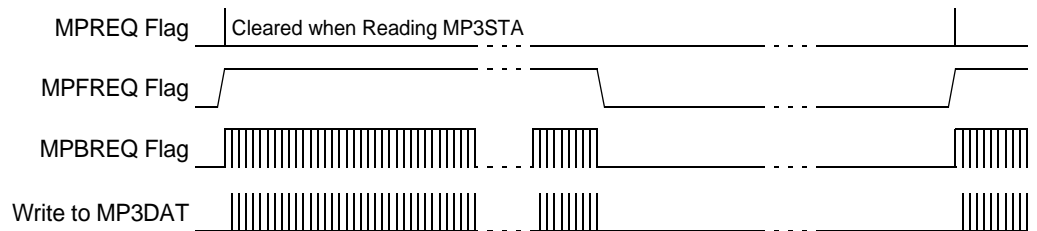


## MP3 Data

The MP3 decoder does not start any frame decoding before having a complete frame in its input buffer<sup>(1)</sup>. In order to manage the load of MP3 data in the frame buffer, a hardware handshake consisting of data request and data acknowledgment is implemented. Each time the MP3 decoder needs MP3 data, it sets the MPREQ, MPFREQ and MPBREQ flags respectively in MP3STA and MP3STA1 registers. MPREQ flag can generate an interrupt if enabled as explained in Section "Interrupt". The CPU must then load data in the buffer by writing it through MP3DAT register thus acknowledging the previous request. As shown in Figure 43, the MPFREQ flag remains set while data (i.e a frame) is requested by the decoder. It is cleared when no more data is requested and set again when new data are requested. MPBREQ flag toggles at every Byte writing.

Note: 1. The first request after enable, consists in 1024 Bytes of data to fill in the input buffer.

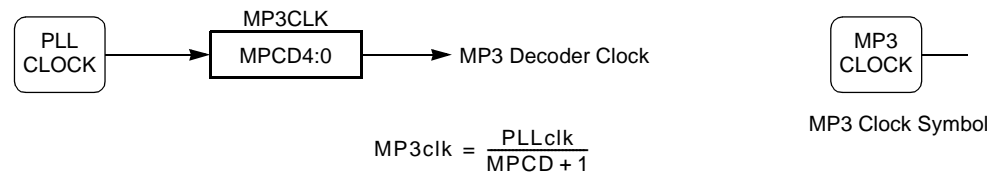
**Figure 43.** Data Timing Diagram



## MP3 Clock

The MP3 decoder clock is generated by division of the PLL clock. The division factor is given by MPCD4:0 bits in MP3CLK register. Figure 44 shows the MP3 decoder clock generator and its calculation formula. The MP3 decoder clock frequency depends only on the incoming MP3 frames.

**Figure 44.** MP3 Clock Generator and Symbol



As soon as the frame header has been decoded and the MPEG version extracted, the minimum MP3 input frequency must be programmed according to Table 68.

**Table 68.** MP3 Clock Frequency

MPEG Version	Minimum MP3 Clock (MHz)
I	21
II	10.5

## Audio Controls

### Volume Control

The MP3 decoder implements volume control on both right and left channels. The MP3VOR and MP3VOL registers allow a 32-step volume control according to Table 69.

**Table 69.** Volume Control

VOL4:0 or VOR4:0	Volume Gain (dB)
00000	Mute
00001	-33
00010	-27
11110	-1.5
11111	0

### Equalization Control

Sound can be adjusted using a 3-band equalizer: a bass band under 750 Hz, a medium band from 750 Hz to 3300 Hz and a treble band over 3300 Hz. The MP3BAS, MP3MED, and MP3TRE registers allow a 32-step gain control in each band according to Table 70.

**Table 70.** Bass, Medium, Treble Control

BAS4:0 or MED4:0 or TRE4:0	Gain (dB)
00000	-∞
00001	-14
00010	-10
11110	+1
11111	+1.5

### Special Effect

The MPBBST bit in MP3CON register allows enabling of a bass boost effect with the following characteristics: gain increase of +9 dB in the frequency under 375 Hz.

### Decoding Errors

The three different errors that can appear during frame processing are detailed in the following sections. All these errors can trigger an interrupt as explained in Section "Interrupt", page 66.

#### Layer Error

The ERRSYN flag in MP3STA is set when a non-supported layer is decoded in the header of the frame that has been sent to the decoder.

#### Synchronization Error

The ERRSYN flag in MP3STA is set when no synchronization pattern is found in the data that have been sent to the decoder.

#### CRC Error

When the CRC of a frame does not match the one calculated, the flag ERRCRC in MP3STA is set. In this case, depending on the CRCEN bit in MP3CON, the frame is played or rejected. In both cases, noise may appear at audio output.



## Frame Information

The MP3 frame header contains information on the audio data contained in the frame. These informations is made available in the MP3STA register for you information. MPVER and MPFS1:0 bits allow decoding of the sampling frequency according to Table 71. MPVER bit gives the MPEG version (2 or 1).

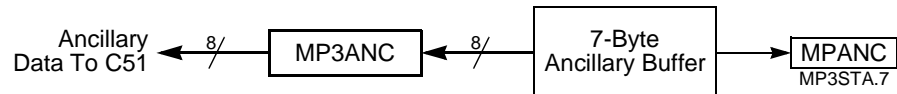
**Table 71.** MP3 Frame Frequency Sampling

MPVER	MPFS1	MPFS0	Fs (kHz)
0	0	0	22.05 (MPEG II)
0	0	1	24 (MPEG II)
0	1	0	16 (MPEG II)
0	1	1	Reserved
1	0	0	44.1 (MPEG I)
1	0	1	48 (MPEG I)
1	1	0	32 (MPEG I)
1	1	1	Reserved

## Ancillary Data

MP3 frames also contain data bits called ancillary data. These data are made available in the MP3ANC register for each frame. As shown in Figure 45, the ancillary data are available by Bytes when MPANC flag in MP3STA register is set. MPANC flag is set when the ancillary buffer is not empty (at least one ancillary data is available) and is cleared only when there is no more ancillary data in the buffer. This flag can generate an interrupt as explained in Section "Interrupt", page 66. When set, software must read all Bytes to empty the ancillary buffer.

**Figure 45.** Ancillary Data Block Diagram



## Interrupt

### Description

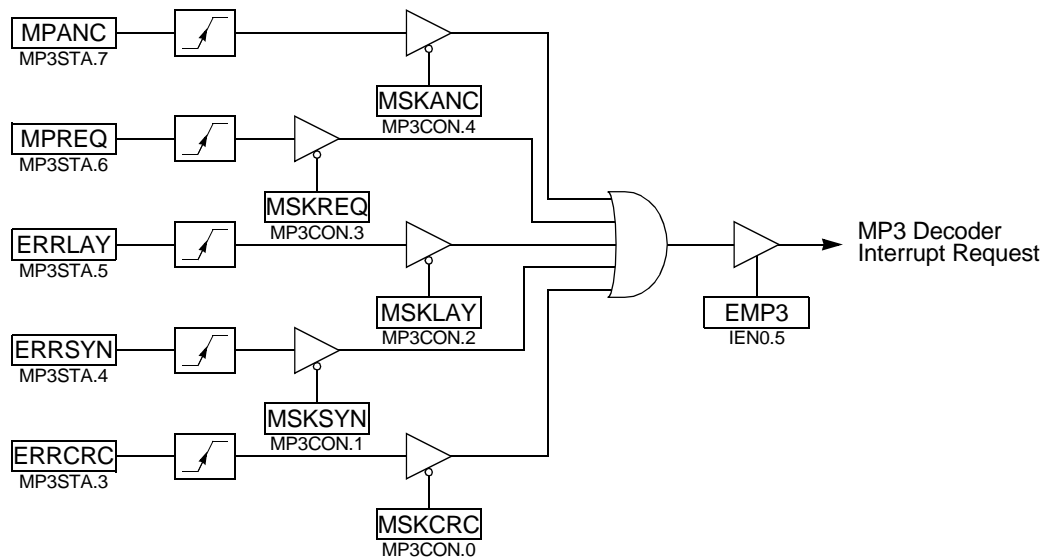
As shown in Figure 46, the MP3 decoder implements five interrupt sources reported in ERRCRC, ERRSYN, ERRLAY, MPREQ, and MPANC flags in MP3STA register.

All these sources are maskable separately using MSKCRC, MSKSYN, MSKLAY, MSKREQ, and MSKANC mask bits respectively in MP3CON register.

The MP3 interrupt is enabled by setting EMP3 bit in IEN0 register. This assumes interrupts are globally enabled by setting EA bit in IEN0 register.

All interrupt flags but MPREQ and MPANC are cleared when reading MP3STA register. The MPREQ flag is cleared by hardware when no more data is requested (see Figure 43) and MPANC flag is cleared by hardware when the ancillary buffer becomes empty.

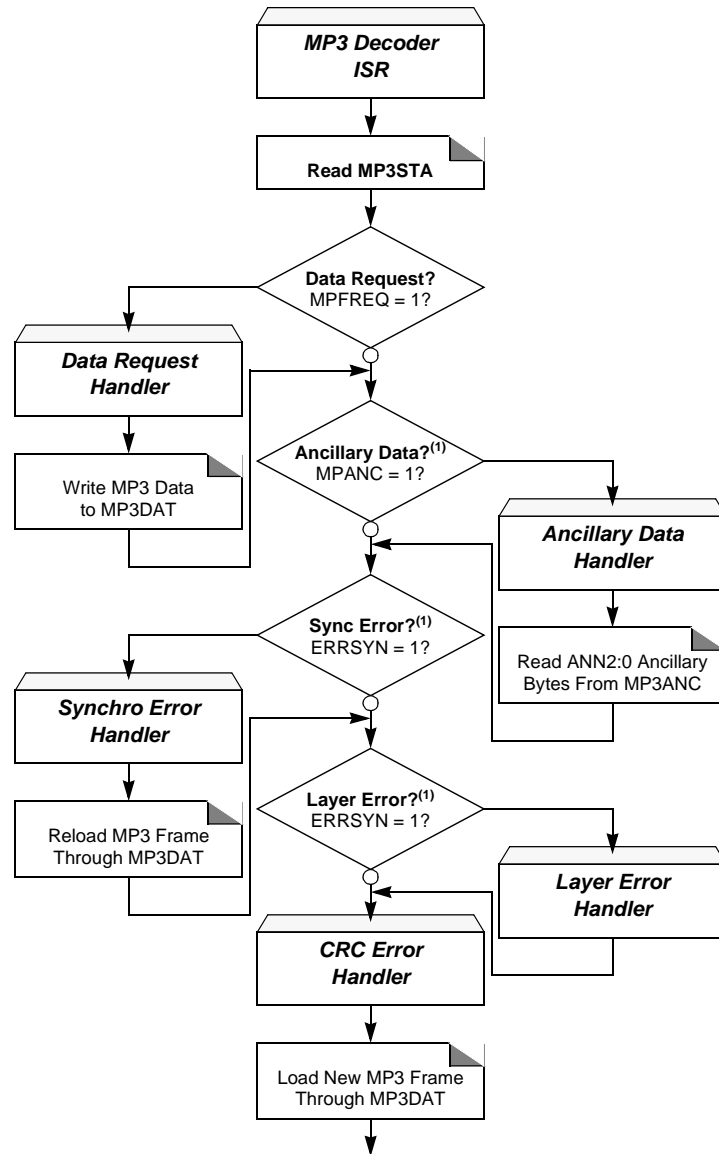
**Figure 46.** MP3 Decoder Interrupt System



### Management

Reading the MP3STA register automatically clears the interrupt flags (acknowledgment) except the MPREQ and MPANC flags. This implies that register content must be saved and tested, interrupt flag by interrupt flag to be sure not to forget any interrupts.

Figure 47. MP3 Interrupt Service Routine Flow



Note: 1. Test these bits only if needed (unmasked interrupt).

## Registers

**Table 72.** MP3CON Register  
MP3CON (S:AAh) – MP3 Decoder Control Register

7	6	5	4	3	2	1	0
MPEN	MPBBST	CRCEN	MSKANC	MSKREQ	MSKLAY	MSKSYN	MSKCRC
Bit Number	Bit Mnemonic	Description					
7	MPEN	<b>MP3 Decoder Enable Bit</b> Set to enable the MP3 decoder. Clear to disable the MP3 decoder.					
6	MPBBST	<b>Bass Boost Bit</b> Set to enable the bass boost sound effect. Clear to disable the bass boost sound effect.					
5	CRCEN	<b>CRC Check Enable Bit</b> Set to enable processing of frame that contains CRC error. Frame is played whatever the error. Clear to disable processing of frame that contains CRC error. Frame is skipped.					
4	MSKANC	<b>MPANC Flag Mask Bit</b> Set to prevent the MPANC flag from generating a MP3 interrupt. Clear to allow the MPANC flag to generate a MP3 interrupt.					
3	MSKREQ	<b>MPREQ Flag Mask Bit</b> Set to prevent the MPREQ flag from generating a MP3 interrupt. Clear to allow the MPREQ flag to generate a MP3 interrupt.					
2	MSKLAY	<b>ERRLAY Flag Mask Bit</b> Set to prevent the ERRLAY flag from generating a MP3 interrupt. Clear to allow the ERRLAY flag to generate a MP3 interrupt.					
1	MSKSYN	<b>ERRSYN Flag Mask Bit</b> Set to prevent the ERRSYN flag from generating a MP3 interrupt. Clear to allow the ERRSYN flag to generate a MP3 interrupt.					
0	MSKCRC	<b>ERRCRC Flag Mask Bit</b> Set to prevent the ERRCRC flag from generating a MP3 interrupt. Clear to allow the ERRCRC flag to generate a MP3 interrupt.					

Reset Value = 0011 1111b

**Table 73. MP3STA Register**  
MP3STA (S:C8h Read Only) – MP3 Decoder Status Register

7	6	5	4	3	2	1	0
MPANC	MPREQ	ERRLAY	ERRSYN	ERRCRC	MPFS1	MPFS0	MPVER
Bit Number	Bit Mnemonic	Description					
7	MPANC	<b>Ancillary Data Available Flag</b> Set by hardware as soon as one ancillary data is available (buffer not empty). Cleared by hardware when no more ancillary data is available (buffer empty).					
6	MPREQ	<b>MP3 Data Request Flag</b> Set by hardware when MP3 decoder request data. Cleared when reading MP3STA.					
5	ERRLAY	<b>Invalid Layer Error Flag</b> Set by hardware when an invalid layer is encountered. Cleared when reading MP3STA.					
4	ERRSYN	<b>Frame Synchronization Error Flag</b> Set by hardware when no synchronization pattern is encountered in a frame. Cleared when reading MP3STA.					
3	ERRCRC	<b>CRC Error Flag</b> Set by hardware when a frame handling CRC is corrupted. Cleared when reading MP3STA.					
2 - 1	MPFS1:0	<b>Frequency Sampling Bits</b> Refer to Table 71 for bits description.					
0	MPVER	<b>MPEG Version Bit</b> Set by the MP3 decoder when the loaded frame is a MPEG I frame. Cleared by the MP3 decoder when the loaded frame is a MPEG II frame.					

Reset Value = 0000 0001b

**Table 74. MP3DAT Register**  
MP3DAT (S:ACh) – MP3 Data Register

7	6	5	4	3	2	1	0
MPD7	MPD6	MPD5	MPD4	MPD3	MPD2	MPD1	MPD0
Bit Number	Bit Mnemonic	Description					
7 - 0	MPD7:0	<b>Input Stream Data Buffer</b> 8-bit MP3 stream data input buffer.					

Reset Value = 0000 0000b

**Table 75.** MP3STA1 Register  
MP3STA1 (S:AFh) – MP3 Decoder Status Register 1

7	6	5	4	3	2	1	0
-	-	-	MPFREQ	MPFREQ	-	-	-
Bit Number	Bit Mnemonic	Description					
7 - 5	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
4	MPFREQ	<b>MP3 Frame Data Request Flag</b> Set by hardware when MP3 decoder request data. Cleared when MP3 decoder no more request data .					
3	MPBREQ	<b>MP3 Byte Data Request Flag</b> Set by hardware when MP3 decoder request data. Cleared when writing to MP3DAT.					
2 - 0	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					

Reset Value = 0001 0001b

**Table 76.** MP3ANC Register  
MP3ANC (S:ADh Read Only) – MP3 Ancillary Data Register

7	6	5	4	3	2	1	0
AND7	AND6	AND5	AND4	AND3	AND2	AND1	AND0
Bit Number	Bit Mnemonic	Description					
7 - 0	AND7:0	<b>Ancillary Data Buffer</b> MP3 ancillary data Byte buffer.					

Reset Value = 0000 0000b

**Table 77.** MP3VOL Register  
MP3VOL (S:9Eh) – MP3 Volume Left Control Register

7	6	5	4	3	2	1	0
-	-	-	VOL4	VOL3	VOL2	VOL1	VOL0
Bit Number	Bit Mnemonic	Description					
7 - 5	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
4 - 0	VOL4:0	<b>Volume Left Value</b> Refer to Table 69 for the left channel volume control description.					

Reset Value = 0000 0000b

**Table 78.** MP3VOR Register  
MP3VOR (S:9Fh) – MP3 Volume Right Control Register

7	6	5	4	3	2	1	0
-	-	-	VOR4	VOR3	VOR2	VOR1	VOR0
Bit Number	Bit Mnemonic	Description					
7 - 5	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
4 - 0	VOR4:0	<b>Volume Right Value</b> Refer to Table 69 for the right channel volume control description.					

Reset Value = 0000 0000b

**Table 79.** MP3BAS Register  
MP3BAS (S:B4h) – MP3 Bass Control Register

7	6	5	4	3	2	1	0
-	-	-	BAS4	BAS3	BAS2	BAS1	BAS0
Bit Number	Bit Mnemonic	Description					
7 - 5	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
4 - 0	BAS4:0	<b>Bass Gain Value</b> Refer to Table 70 for the bass control description.					

Reset Value = 0000 0000b

**Table 80.** MP3MED Register  
MP3MED (S:B5h) – MP3 Medium Control Register

7	6	5	4	3	2	1	0
-	-	MED5	MED4	MED3	MED2	MED1	MED0
Bit Number	Bit Mnemonic	Description					
7 - 6	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
5-0	MED5:0	<b>Medium Gain Value</b> Refer to Table 70 for the medium control description.					

Reset Value = 0000 0000b

**Table 81.** MP3TRE Register  
MP3TRE (S:B6h) – MP3 Treble Control Register

7	6	5	4	3	2	1	0
-	-	TRE5	TRE4	TRE3	TRE2	TRE1	TRE0
Bit Number	Bit Mnemonic	Description					
7 - 6	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
5-0	TRE5:0	<b>Treble Gain Value</b> Refer to Table 70 for the treble control description.					

Reset Value = 0000 0000b

**Table 82.** MP3CLK Register  
MP3CLK (S:EBh) – MP3 Clock Divider Register

7	6	5	4	3	2	1	0
-	-	-	MPCD4	MPCD3	MPCD2	MPCD1	MPCD0
Bit Number	Bit Mnemonic	Description					
7 - 5	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
4-0	MPCD4:0	<b>MP3 Decoder Clock Divider</b> 5-bit divider for MP3 decoder clock generation.					

Reset Value = 0000 0000b



## Audio Output Interface

The AT8xC51SND1C implement an audio output interface allowing the audio bitstream to be output in various formats. It is compatible with right and left justification PCM and I<sup>2</sup>S formats and thanks to the on-chip PLL (see Section “Clock Controller”, page 12) allows connection of almost all of the commercial audio DAC families available on the market.

The audio bitstream can be from 2 different types:

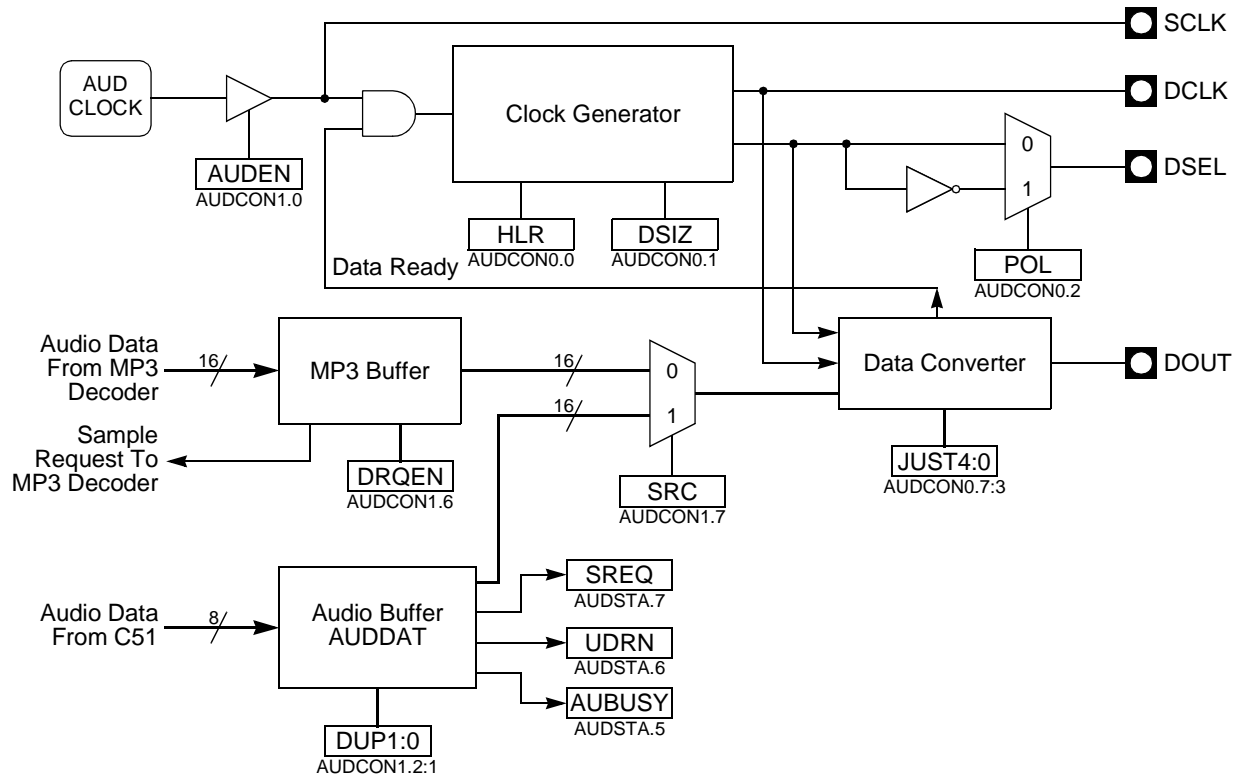
- The MP3 decoded bitstream coming from the MP3 decoder for playing songs.
- The audio bitstream coming from the MCU for outputting voice or sounds.

## Description

The C51 core interfaces to the audio interface through five special function registers: AUDCON0 and AUDCON1, the Audio Control registers (see Table 84 and Table 85); AUDSTA, the Audio Status register (see Table 86); AUDDAT, the Audio Data register (see Table 87); and AUDCLK, the Audio Clock Divider register (see Table 88).

Figure 48 shows the audio interface block diagram, blocks are detailed in the following sections.

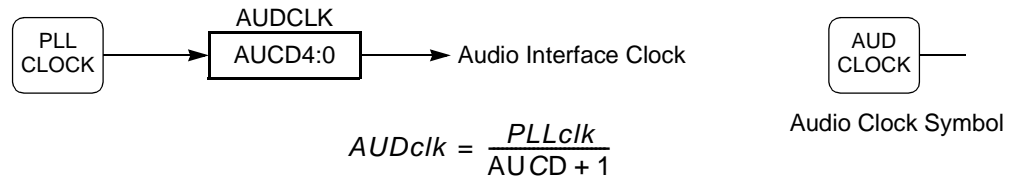
**Figure 48.** Audio Interface Block Diagram



## Clock Generator

The audio interface clock is generated by division of the PLL clock. The division factor is given by AUCD4:0 bits in CLKAUD register. Figure 49 shows the audio interface clock generator and its calculation formula. The audio interface clock frequency depends on the incoming MP3 frames and the audio DAC used.

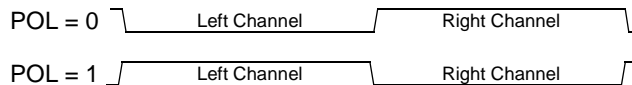
**Figure 49.** Audio Clock Generator and Symbol



As soon as audio interface is enabled by setting AUDEN bit in AUDCON1 register, the master clock generated by the PLL is output on the SCLK pin which is the DAC system clock. This clock is output at 256 or 384 times the sampling frequency depending on the DAC capabilities. HLR bit in AUDCON0 register must be set according to this rate for properly generating the audio bit clock on the DCLK pin and the word selection clock on the DSEL pin. These clocks are not generated when no data is available at the data converter input.

For DAC compatibility, the bit clock frequency is programmable for outputting 16 bits or 32 bits per channel using the DSIZ bit in AUDCON0 register (see Section "Data Converter", page 74), and the word selection signal is programmable for outputting left channel on low or high level according to POL bit in AUDCON0 register as shown in Figure 50.

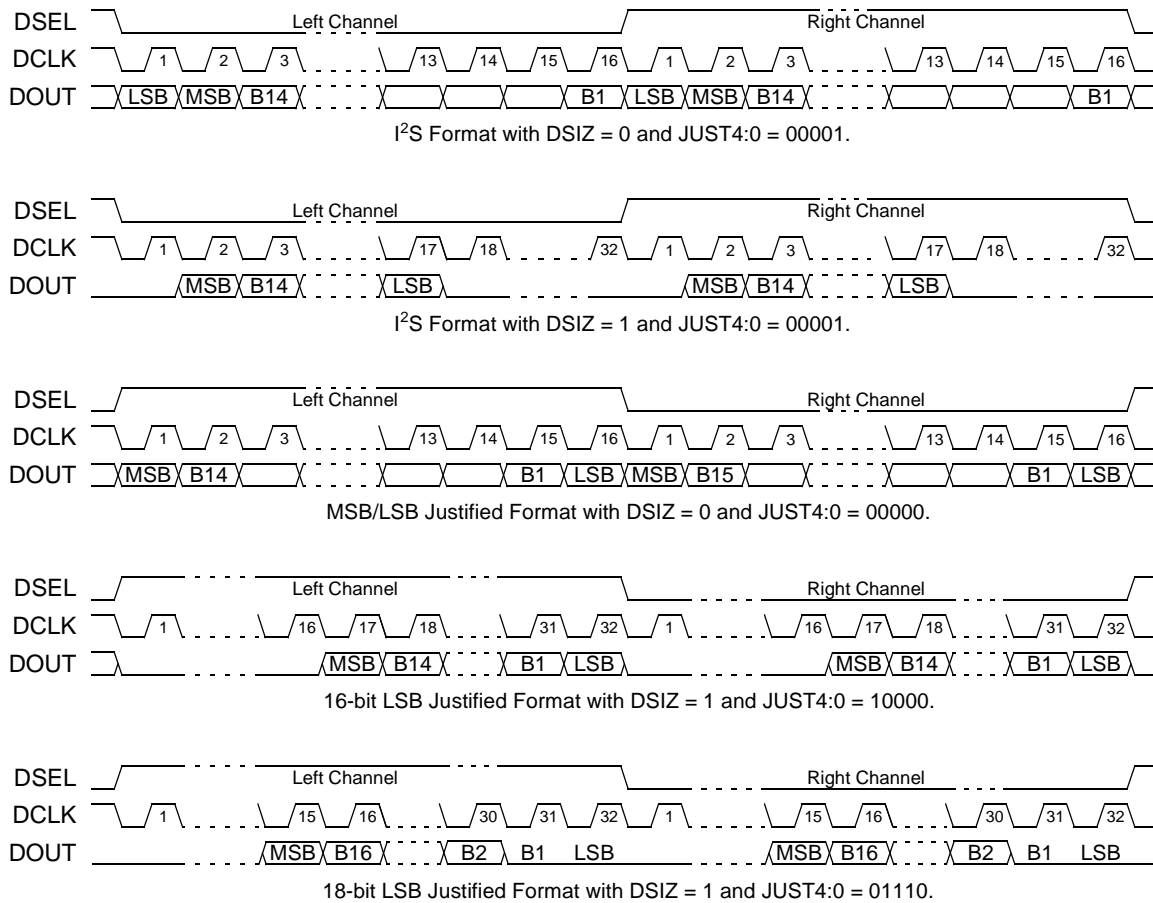
**Figure 50.** DSEL Output Polarity



## Data Converter

The data converter block converts the audio stream input from the 16-bit parallel format to a serial format. For accepting all PCM formats and I<sup>2</sup>S format, JUST4:0 bits in AUDCON0 register are used to shift the data output point. As shown in Figure 51, these bits allow MSB justification by setting JUST4:0 = 00000, LSB justification by setting JUST4:0 = 10000, I<sup>2</sup>S Justification by setting JUST4:0 = 00001, and more than 16-bit LSB justification by filling the low significant bits with logic 0.

**Figure 51. Audio Output Format**



The data converter receives its audio stream from 2 sources selected by the SRC bit in AUDCON1 register. When cleared, the audio stream comes from the MP3 decoder (see Section “MP3 Decoder”, page 62) for song playing. When set, the audio stream is coming from the C51 core for voice or sound playing.

As soon as first audio data is input to the data converter, it enables the clock generator for generating the bit and word clocks.

## Audio Buffer

In voice or sound playing mode, the audio stream comes from the C51 core through an audio buffer. The data is in 8-bit format and is sampled at 8 kHz. The audio buffer adapts the sample format and rate. The sample format is extended to 16 bits by filling the LSB to 00h. Rate is adapted to the DAC rate by duplicating the data using DUP1:0 bits in AUDCON1 register according to Table 83.

The audio buffer interfaces to the C51 core through three flags: the sample request flag (SREQ in AUDSTA register), the under-run flag (UNDR in AUDSTA register) and the busy flag (AUBUSY in AUDSTA register). SREQ and UNDR can generate an interrupt request as explained in Section "Interrupt Request", page 76. The buffer size is 8 Bytes large. SREQ is set when the samples number switches from 4 to 3 and reset when the samples number switches from 4 to 5; UNDR is set when the buffer becomes empty signaling that the audio interface ran out of samples; and AUBUSY is set when the buffer is full.

**Table 83.** Sample Duplication Factor

DUP1	DUP0	Factor
0	0	No sample duplication, DAC rate = 8 kHz (C51 rate).
0	1	One sample duplication, DAC rate = 16 kHz (2 x C51 rate).
1	0	2 samples duplication, DAC rate = 32 kHz (4 x C51 rate).
1	1	Three samples duplication, DAC rate = 48 kHz (6 x C51 rate).

## MP3 Buffer

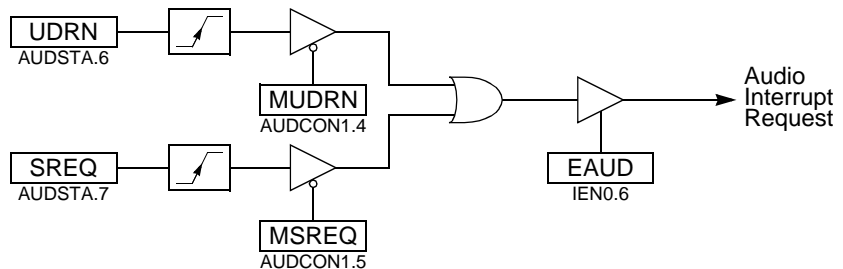
In song playing mode, the audio stream comes from the MP3 decoder through a buffer. The MP3 buffer is used to store the decoded MP3 data and interfaces to the decoder through a 16-bit data input and data request signal. This signal asks for data when the buffer has enough space to receive new data. Data request is conditioned by the DREQEN bit in AUDCON1 register. When set, the buffer requests data to the MP3 decoder. When cleared no more data is requested but data are output until the buffer is empty. This bit can be used to suspend the audio generation (pause mode).

## Interrupt Request

The audio interrupt request can be generated by 2 sources when in C51 audio mode: a sample request when SREQ flag in AUDSTA register is set to logic 1, and an under-run condition when UDRN flag in AUDSTA register is set to logic 1. Both sources can be enabled separately by masking one of them using the MSREQ and MUDRN bits in AUDCON1 register. A global enable of the audio interface is provided by setting the EAUD bit in IEN0 register.

The interrupt is requested each time one of the 2 sources is set to one. The source flags are cleared by writing some data in the audio buffer through AUDDAT, but the global audio interrupt flag is cleared by hardware when the interrupt service routine is executed.

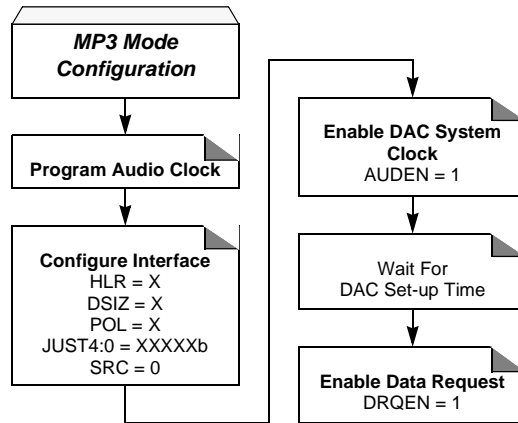
**Figure 52.** Audio Interface Interrupt System



## MP3 Song Playing

In MP3 song playing mode, the operations to do are to configure the PLL and the audio interface according to the DAC selected. The audio clock is programmed to generate the 256-Fs or 384-Fs as explained in Section "Clock Generator", page 74. Figure 53 shows the configuration flow of the audio interface when in MP3 song mode.

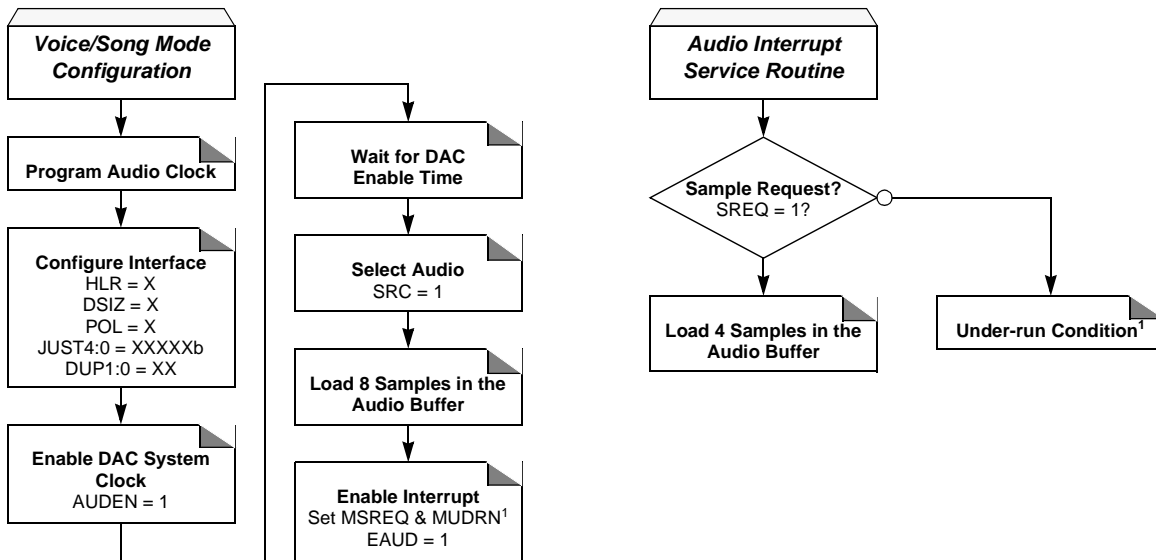
Figure 53. MP3 Mode Audio Configuration Flow



**Voice or Sound Playing**

In voice or sound playing mode, the operations required are to configure the PLL and the audio interface according to the DAC selected. The audio clock is programmed to generate the 256-Fs or 384-Fs as for the MP3 playing mode. The data flow sent by the C51 is then regulated by interrupt and data is loaded 4 Bytes by 4 Bytes. Figure 54 shows the configuration flow of the audio interface when in voice or sound mode.

Figure 54. Voice or Sound Mode Audio Flows



Note: 1. An under-run occurrence signifies that C51 core did not respond to the previous sample request interrupt. It may never occur for a correct voice/sound generation. It is the user's responsibility to mask it or not.

## Registers

**Table 84.** AUDCON0 Register

AUDCON0 (S:9Ah) – Audio Interface Control Register 0

7	6	5	4	3	2	1	0
JUST4	JUST3	JUST2	JUST1	JUST0	POL	DSIZ	HLR
Bit Number	Bit Mnemonic	Description					
7 - 3	JUST4:0	<b>Audio Stream Justification Bits</b> Refer to Section "Data Converter", page 74 for bits description.					
2	POL	<b>DSEL Signal Output Polarity</b> Set to output the left channel on high level of DSEL output (PCM mode). Clear to output the left channel on the low level of DSEL output (I <sup>2</sup> S mode).					
1	DSIZ	<b>Audio Data Size</b> Set to select 32-bit data output format. Clear to select 16-bit data output format.					
0	HLR	<b>High/Low Rate Bit</b> Set by software when the PLL clock frequency is 384-Fs. Clear by software when the PLL clock frequency is 256-Fs.					

Reset Value = 0000 1000b

**Table 85.** AUDCON1 Register

AUDCON1 (S:9Bh) – Audio Interface Control Register 1

7	6	5	4	3	2	1	0
SRC	DRQEN	MSREQ	MUDRN	-	DUP1	DUP0	AUDEN
Bit Number	Bit Mnemonic	Description					
7	SRC	<b>Audio Source Bit</b> Set to select C51 as audio source for voice or sound playing. Clear to select the MP3 decoder output as audio source for song playing.					
6	DRQEN	<b>MP3 Decoded Data Request Enable Bit</b> Set to enable data request to the MP3 decoder and to start playing song. Clear to disable data request to the MP3 decoder.					
5	MSREQ	<b>Audio Sample Request Flag Mask Bit</b> Set to prevent the SREQ flag from generating an audio interrupt. Clear to allow the SREQ flag to generate an audio interrupt.					
4	MUDRN	<b>Audio Sample Under-run Flag Mask Bit</b> Set to prevent the UDRN flag from generating an audio interrupt. Clear to allow the UDRN flag to generate an audio interrupt.					
3	-	<b>Reserved</b> The value read from this bit is always 0. Do not set this bit.					
2 - 1	DUP1:0	<b>Audio Duplication Factor</b> Refer to Table 83 for bits description.					
0	AUDEN	<b>Audio Interface Enable Bit</b> Set to enable the audio interface. Clear to disable the audio interface.					

Reset Value = 1011 0010b

**Table 86.** AUDSTA Register  
AUDSTA (S:9Ch Read Only) – Audio Interface Status Register

7	6	5	4	3	2	1	0	
SREQ	UDRN	AUBUSY	-	-	-	-	-	
Bit Number	Bit Mnemonic	Description						
7	SREQ	<b>Audio Sample Request Flag</b> Set in C51 audio source mode when the audio interface request samples (buffer half empty). This bit generates an interrupt if not masked and if enabled in IEN0. Cleared by hardware when samples are loaded in AUDDAT.						
6	UDRN	<b>Audio Sample Under-run Flag</b> Set in C51 audio source mode when the audio interface runs out of samples (buffer empty). This bit generates an interrupt if not masked and if enabled in IEN0. Cleared by hardware when samples are loaded in AUDDAT.						
5	AUBUSY	<b>Audio Interface Busy Bit</b> Set in C51 audio source mode when the audio interface can not accept more sample (buffer full). Cleared by hardware when buffer is no more full.						
4 - 0	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.						

Reset Value = 1100 0000b

**Table 87.** AUDDAT Register  
AUDDAT (S:9Dh) – Audio Interface Data Register

7	6	5	4	3	2	1	0	
AUD7	AUD6	AUD5	AUD4	AUD3	AUD2	AUD1	AUD0	
Bit Number	Bit Mnemonic	Description						
7 - 0	AUD7:0	<b>Audio Data</b> 8-bit sampling data for voice or sound playing.						

Reset Value = 1111 1111b

**Table 88.** AUDCLK Register  
AUDCLK (S:ECh) – Audio Clock Divider Register

7	6	5	4	3	2	1	0	
-	-	-	AUCD4	AUCD3	AUCD2	AUCD1	AUCD0	
Bit Number	Bit Mnemonic	Description						
7 - 5	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.						
4 - 0	AUCD4:0	<b>Audio Clock Divider</b> 5-bit divider for audio clock generation.						

Reset Value = 0000 0000b

## Universal Serial Bus

The AT8xC51SND1C implements a USB device controller supporting full speed data transfer. In addition to the default control endpoint 0, it provides 2 other endpoints, which can be configured in control, bulk, interrupt or isochronous modes:

- Endpoint 0: 32-Byte FIFO, default control endpoint
- Endpoint 1, 2: 64-Byte Ping-pong FIFO,

This allows the firmware to be developed conforming to most USB device classes, for example:

- USB Mass Storage Class Bulk-only Transport, Revision 1.0 - September 31, 1999
- USB Human Interface Device Class, Version 1.1 - April 7, 1999
- USB Device Firmware Upgrade Class, Revision 1.0 - May 13, 1999

### USB Mass Storage Class Bulk-Only Transport

Within the Bulk-only framework, the Control endpoint is only used to transport class-specific and standard USB requests for device set-up and configuration. One Bulk-out endpoint is used to transport commands and data from the host to the device. One Bulk-in endpoint is used to transport status and data from the device to the host.

The following AT8xC51SND1C configuration adheres to those requirements:

- Endpoint 0: 32 Bytes, Control In-Out
- Endpoint 1: 64 Bytes, Bulk-in
- Endpoint 2: 64 Bytes, Bulk-out

### USB Device Firmware Upgrade (DFU)

The USB Device Firmware Update (DFU) protocol can be used to upgrade the on-chip Flash memory of the AT89C51SND1C. This allows installing product enhancements and patches to devices that are already in the field. 2 different configurations and descriptor sets are used to support DFU functions. The Run-Time configuration co-exist with the usual functions of the device, which is USB Mass Storage for AT89C51SND1C. It is used to initiate DFU from the normal operating mode. The DFU configuration is used to perform the firmware update after device re-configuration and USB reset. It excludes any other function. Only the default control pipe (endpoint 0) is used to support DFU services in both configurations.

The only possible value for the MaxPacketSize in the DFU configuration is 32 Bytes, which is the size of the FIFO implemented for endpoint 0.



## Description

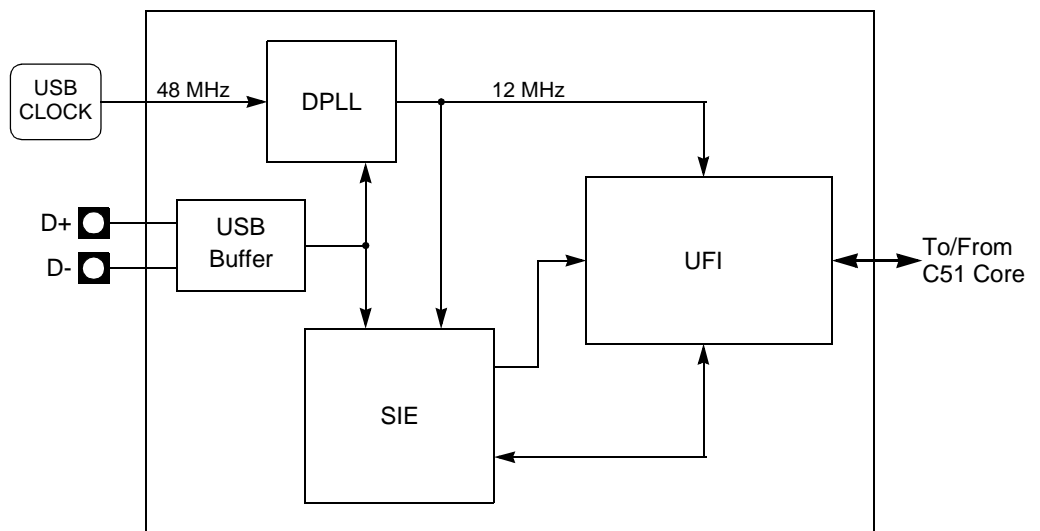
The USB device controller provides the hardware that the AT8xC51SND1C needs to interface a USB link to a data flow stored in a double port memory.

It requires a 48 MHz reference clock provided by the clock controller as detailed in Section "Clock Controller", page 81. This clock is used to generate a 12 MHz Full Speed bit clock from the received USB differential data flow and to transmit data according to full speed USB device tolerance. Clock recovery is done by a Digital Phase Locked Loop (DPLL) block.

The Serial Interface Engine (SIE) block performs NRZI encoding and decoding, bit stuffing, CRC generation and checking, and the serial-parallel data conversion.

The Universal Function Interface (UFI) controls the interface between the data flow and the Dual Port RAM, but also the interface with the C51 core itself.

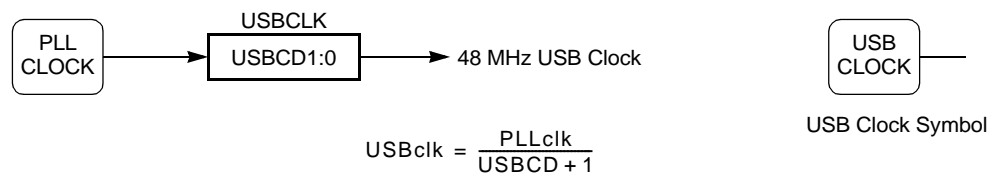
**Figure 55.** USB Device Controller Block Diagram



## Clock Controller

The USB controller clock is generated by division of the PLL clock. The division factor is given by USBCD1:0 bits in USBCLK register (see Table 104). Figure 56 shows the USB controller clock generator and its calculation formula. The USB controller clock frequency must always be 48 MHz.

**Figure 56.** USB Clock Generator and Symbol

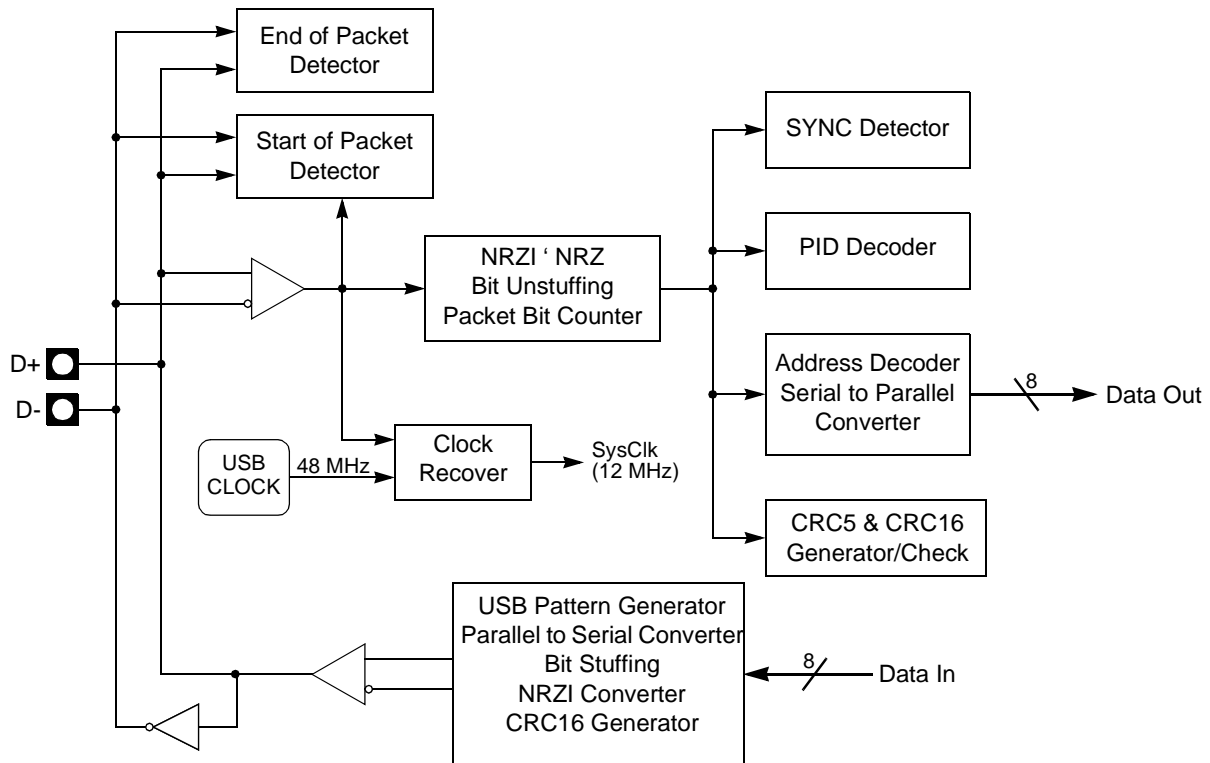


### Serial Interface Engine (SIE)

The SIE performs the following functions:

- NRZI data encoding and decoding.
- Bit stuffing and unstuffing.
- CRC generation and checking.
- ACKs and NACKs automatic generation.
- TOKEN type identifying.
- Address checking.
- Clock recovery (using DPLL).

**Figure 57.** SIE Block Diagram

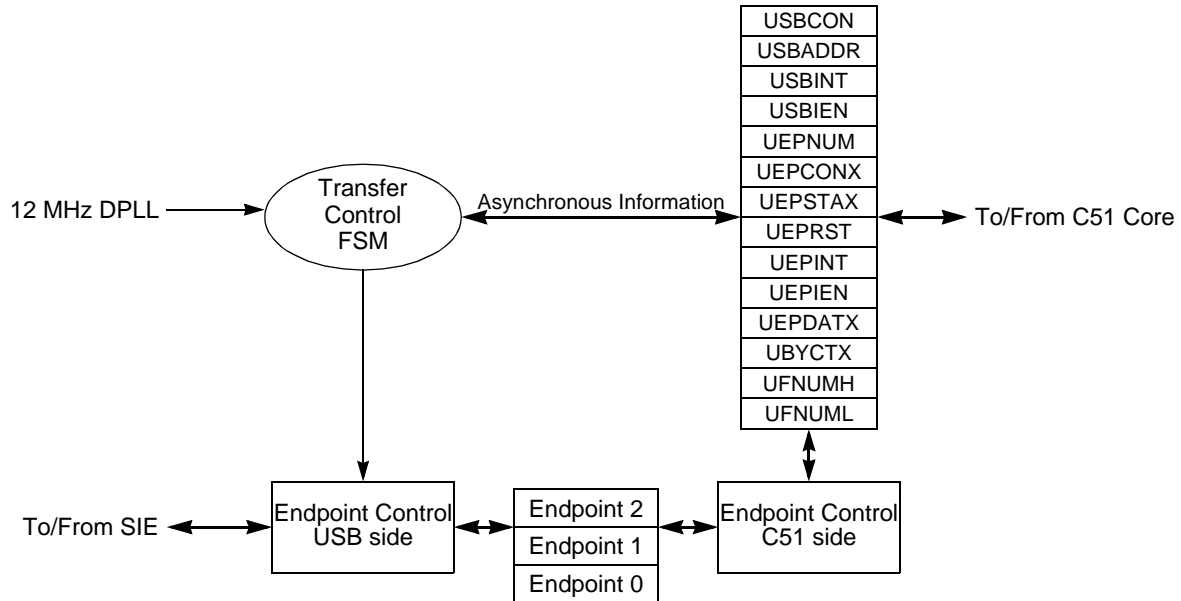


## Function Interface Unit (UFI)

The Function Interface Unit provides the interface between the AT8xC51SND1C and the SIE. It manages transactions at the packet level with minimal intervention from the device firmware, which reads and writes the endpoint FIFOs.

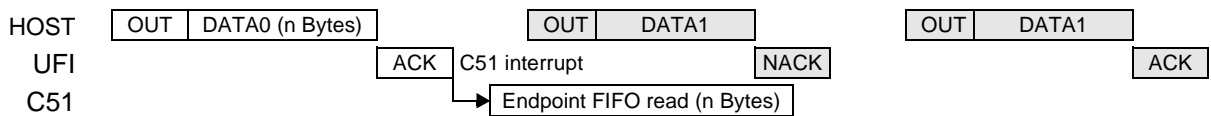
Figure 59 shows typical USB IN and OUT transactions reporting the split in the hardware (UFI) and software (C51) load.

**Figure 58.** UFI Block Diagram



**Figure 59.** USB Typical Transaction Load

### OUT Transactions:



### IN Transactions:



## Configuration

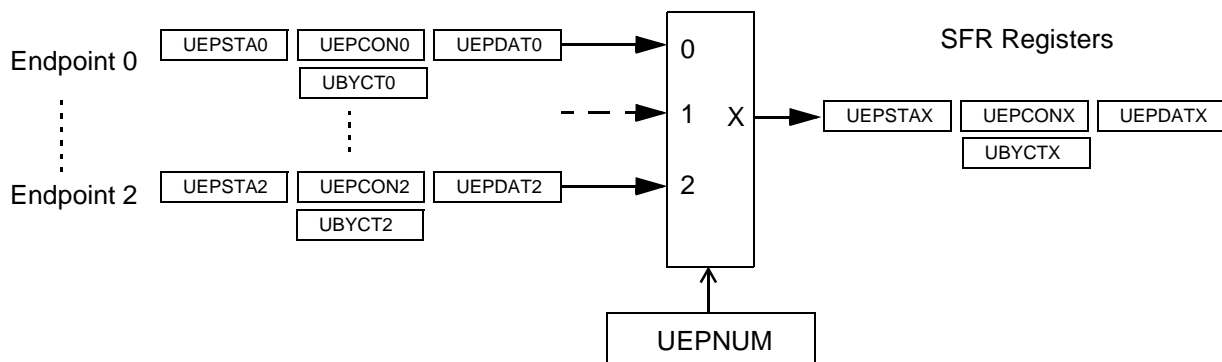
### General Configuration

- USB controller enable  
Before any USB transaction, the 48 MHz required by the USB controller must be correctly generated (See “Clock Controller” on page 19).  
The USB controller should be then enabled by setting the EUSB bit in the USBCON register.
- Set address  
After a Reset or a USB reset, the software has to set the FEN (Function Enable) bit in the USBADDR register. This action will allow the USB controller to answer to the requests sent at the address 0.  
When a SET\_ADDRESS request has been received, the USB controller must only answer to the address defined by the request. The new address should be stored in the USBADDR register. The FEN bit and the FADDEN bit in the USBCON register should be set to allow the USB controller to answer only to requests sent at the new address.
- Set configuration  
The CONFIG bit in the USBCON register should be set after a SET\_CONFIGURATION request with a non-zero value. Otherwise, this bit should be cleared.

### Endpoint Configuration

- Selection of an Endpoint  
The endpoint register access is performed using the UEPNUM register. The registers
  - UEPSTAX
  - UEPCONX
  - UEPDATX
  - UBYCTX
 These registers correspond to the endpoint whose number is stored in the UEPNUM register. To select an Endpoint, the firmware has to write the endpoint number in the UEPNUM register.

Figure 60. Endpoint Selection



- **Endpoint enable**  
 Before using an endpoint, this must be enabled by setting the EPEN bit in the UEPCONX register.  
 An endpoint which is not enabled won't answer to any USB request. The Default Control Endpoint (Endpoint 0) should always be enabled in order to answer to USB standard requests.
- **Endpoint type configuration**  
 All Standard Endpoints can be configured in Control, Bulk, Interrupt or Isochronous mode. The Ping-pong Endpoints can be configured in Bulk, Interrupt or Isochronous mode. The configuration of an endpoint is performed by setting the field EPTYPE with the following values:
  - Control: EPTYPE = 00b
  - Isochronous: EPTYPE = 01b
  - Bulk: EPTYPE = 10b
  - Interrupt: EPTYPE = 11b
 The Endpoint 0 is the Default Control Endpoint and should always be configured in Control type.
- **Endpoint direction configuration**  
 For Bulk, Interrupt and Isochronous endpoints, the direction is defined with the EPDIR bit of the UEPCONX register with the following values:
  - IN: EPDIR = 1b
  - OUT: EPDIR = 0b
 For Control endpoints, the EPDIR bit has no effect.
- **Summary of Endpoint Configuration:**  
 Do not forget to select the correct endpoint number in the UEPNUM register before accessing endpoint specific registers.

**Table 89.** Summary of Endpoint Configuration

Endpoint Configuration	EPEN	EPDIR	EPTYPE	UEPCONX
Disabled	0b	Xb	XXb	0XXX XXXb
Control	1b	Xb	00b	80h
Bulk-in	1b	1b	10b	86h
Bulk-out	1b	0b	10b	82h
Interrupt-In	1b	1b	11b	87h
Interrupt-Out	1b	0b	11b	83h
Isochronous-In	1b	1b	01b	85h
Isochronous-Out	1b	0b	01b	81h

- Endpoint FIFO reset

Before using an endpoint, its FIFO should be reset. This action resets the FIFO pointer to its original value, resets the Byte counter of the endpoint (UBYCTX register), and resets the data toggle bit (DTGL bit in UEPCONX).

The reset of an endpoint FIFO is performed by setting to 1 and resetting to 0 the corresponding bit in the UEPRST register.

For example, in order to reset the Endpoint number 2 FIFO, write 0000 0100b then 0000 0000b in the UEPRST register.

Note that the endpoint reset doesn't reset the bank number for ping-pong endpoints.

## Read/Write Data FIFO

### Read Data FIFO

The read access for each OUT endpoint is performed using the UEPDATX register.

After a new valid packet has been received on an Endpoint, the data are stored into the FIFO and the Byte counter of the endpoint is updated (UBYCTX registers). The firmware has to store the endpoint Byte counter before any access to the endpoint FIFO. The Byte counter is not updated when reading the FIFO.

To read data from an endpoint, select the correct endpoint number in UEPNUM and read the UEPDATX register. This action automatically decreases the corresponding address vector, and the next data is then available in the UEPDATX register.

### Write Data FIFO

The write access for each IN endpoint is performed using the UEPDATX register.

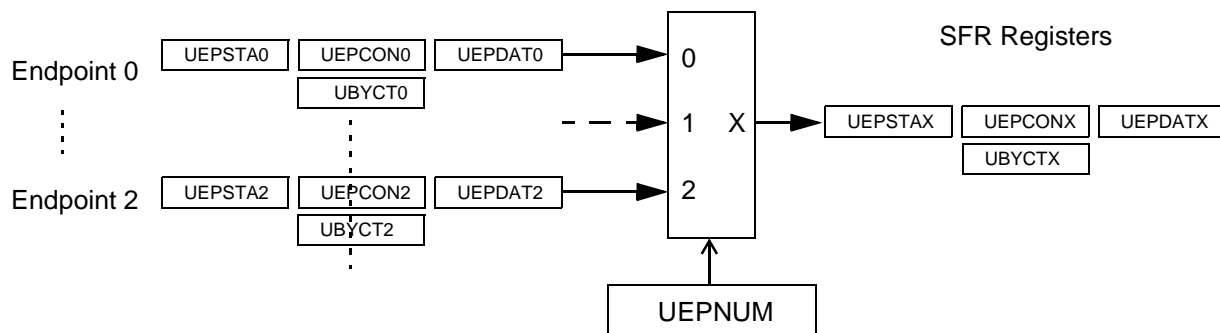
To write a Byte into an IN endpoint FIFO, select the correct endpoint number in UEPNUM and write into the UEPDATX register. The corresponding address vector is automatically increased, and another write can be carried out.

Warning 1: The Byte counter is not updated.

Warning 2: Do not write more Bytes than supported by the corresponding endpoint.

## FIFO Mapping

Figure 61. Endpoint FIFO Configuration

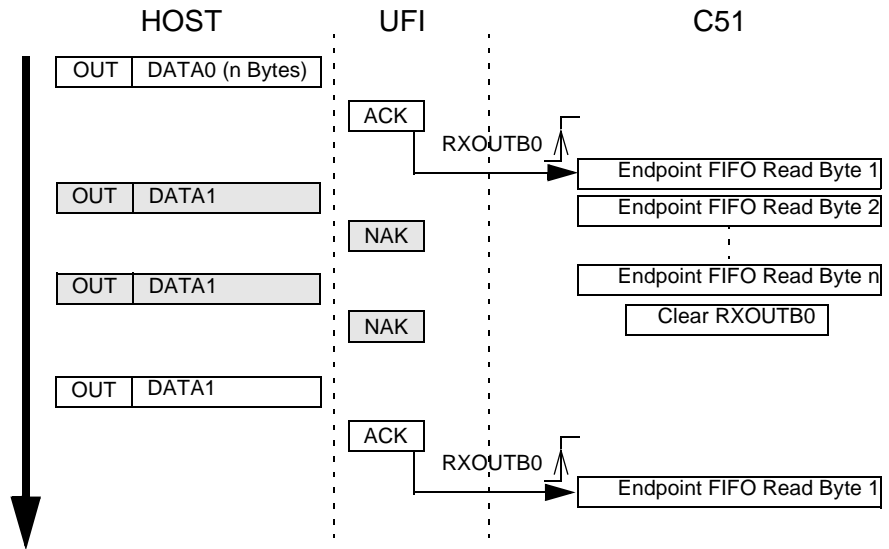


**Bulk/Interrupt Transactions**

Bulk and Interrupt transactions are managed in the same way.

**Bulk/Interrupt OUT Transactions in Standard Mode**

**Figure 62.** Bulk/Interrupt OUT transactions in Standard Mode



An endpoint should be first enabled and configured before being able to receive Bulk or Interrupt packets.

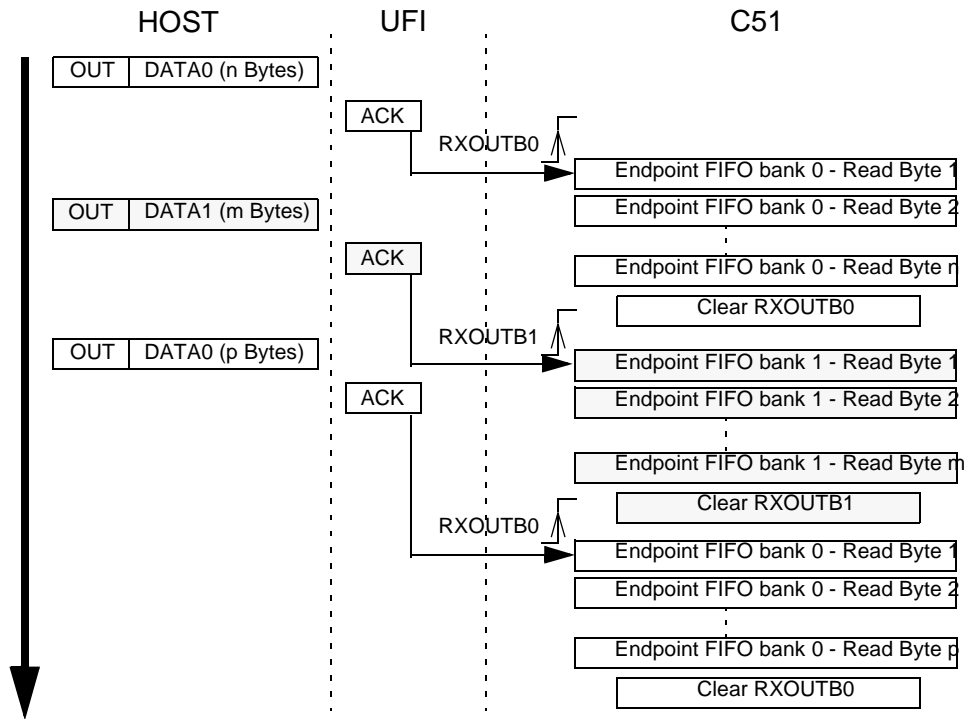
When a valid OUT packet is received on an endpoint, the RXOUTB0 bit is set by the USB controller. This triggers an interrupt if enabled. The firmware has to select the corresponding endpoint, store the number of data Bytes by reading the UBYCTX register. If the received packet is a ZLP (Zero Length Packet), the UBYCTX register value is equal to 0 and no data has to be read.

When all the endpoint FIFO Bytes have been read, the firmware should clear the RXOUTB0 bit to allow the USB controller to accept the next OUT packet on this endpoint. Until the RXOUTB0 bit has been cleared by the firmware, the USB controller will answer a NAK handshake for each OUT requests.

If the Host sends more Bytes than supported by the endpoint FIFO, the overflow data won't be stored, but the USB controller will consider that the packet is valid if the CRC is correct and the endpoint Byte counter contains the number of Bytes sent by the Host.

## Bulk/Interrupt OUT Transactions in Ping-pong Mode

Figure 63. Bulk/Interrupt OUT Transactions in Ping-pong Mode



An endpoint should be first enabled and configured before being able to receive Bulk or Interrupt packets.

When a valid OUT packet is received on the endpoint bank 0, the RXOUTB0 bit is set by the USB controller. This triggers an interrupt if enabled. The firmware has to select the corresponding endpoint, store the number of data Bytes by reading the UBYCTX register. If the received packet is a ZLP (Zero Length Packet), the UBYCTX register value is equal to 0 and no data has to be read.

When all the endpoint FIFO Bytes have been read, the firmware should clear the RXOUB0 bit to allow the USB controller to accept the next OUT packet on the endpoint bank 0. This action switches the endpoint bank 0 and 1. Until the RXOUTB0 bit has been cleared by the firmware, the USB controller will answer a NAK handshake for each OUT requests on the bank 0 endpoint FIFO.

When a new valid OUT packet is received on the endpoint bank 1, the RXOUTB1 bit is set by the USB controller. This triggers an interrupt if enabled. The firmware empties the bank 1 endpoint FIFO before clearing the RXOUTB1 bit. Until the RXOUTB1 bit has been cleared by the firmware, the USB controller will answer a NAK handshake for each OUT requests on the bank 1 endpoint FIFO.

The RXOUTB0 and RXOUTB1 bits are, alternatively, set by the USB controller at each new valid packet receipt.

The firmware has to clear one of these 2 bits after having read all the data FIFO to allow a new valid packet to be stored in the corresponding bank.

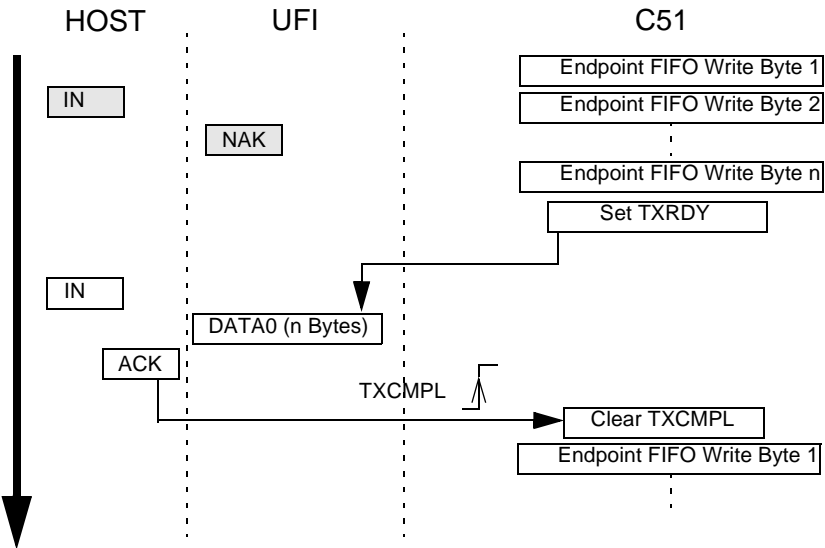
A NAK handshake is sent by the USB controller only if the banks 0 and 1 has not been released by the firmware.



If the Host sends more Bytes than supported by the endpoint FIFO, the overflow data won't be stored, but the USB controller will consider that the packet is valid if the CRC is correct.

## Bulk/Interrupt IN Transactions in Standard Mode

**Figure 64.** Bulk/Interrupt IN Transactions in Standard Mode



An endpoint should be first enabled and configured before being able to send Bulk or Interrupt packets.

The firmware should fill the FIFO with the data to be sent and set the TXRDY bit in the UEPSTAX register to allow the USB controller to send the data stored in FIFO at the next IN request concerning this endpoint. To send a Zero Length Packet, the firmware should set the TXRDY bit without writing any data into the endpoint FIFO.

Until the TXRDY bit has been set by the firmware, the USB controller will answer a NAK handshake for each IN requests.

To cancel the sending of this packet, the firmware has to reset the TXRDY bit. The packet stored in the endpoint FIFO is then cleared and a new packet can be written and sent.

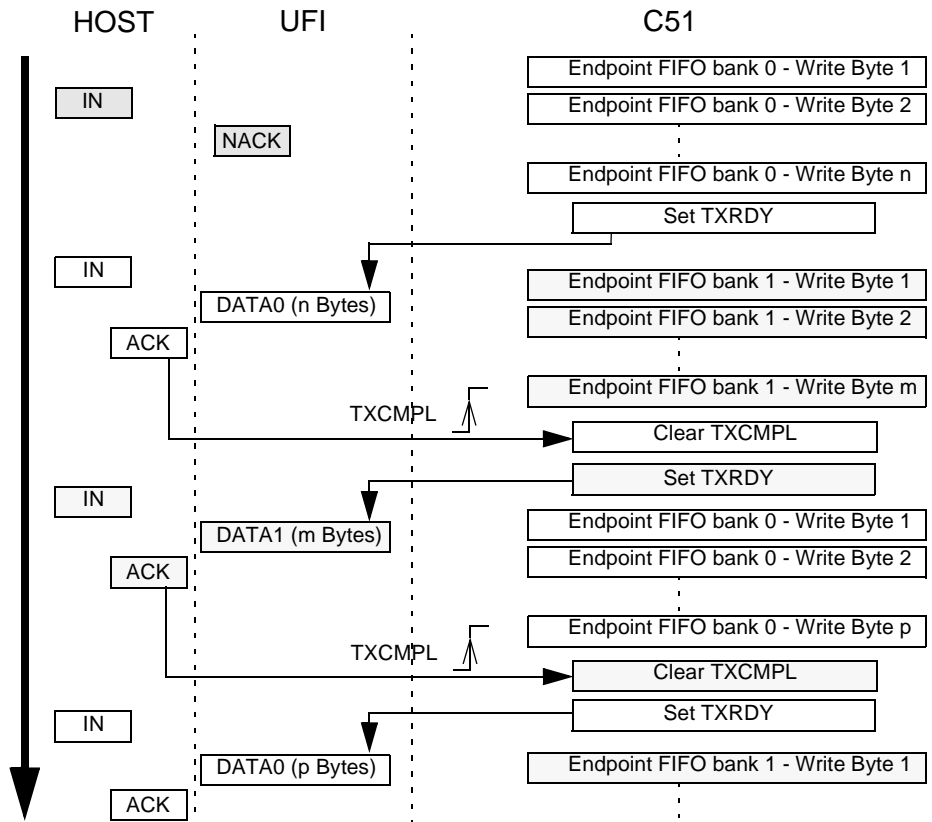
When the IN packet has been sent and acknowledged by the Host, the TXCMPL bit in the UEPSTAX register is set by the USB controller. This triggers a USB interrupt if enabled. The firmware should clear the TXCMPL bit before filling the endpoint FIFO with new data.

The firmware should never write more Bytes than supported by the endpoint FIFO.

All USB retry mechanisms are automatically managed by the USB controller.

## Bulk/Interrupt IN Transactions in Ping-pong Mode

Figure 65. Bulk/Interrupt IN transactions in Ping-pong mode



An endpoint should be first enabled and configured before being able to send Bulk or Interrupt packets.

The firmware should fill the FIFO bank 0 with the data to be sent and set the TXRDY bit in the UEPSTAX register to allow the USB controller to send the data stored in FIFO at the next IN request concerning the endpoint. The FIFO banks are automatically switched, and the firmware can immediately write into the endpoint FIFO bank 1.

When the IN packet concerning the bank 0 has been sent and acknowledged by the Host, the TXCMPL bit is set by the USB controller. This triggers a USB interrupt if enabled. The firmware should clear the TXCMPL bit before filling the endpoint FIFO bank 0 with new data. The FIFO banks are then automatically switched.

When the IN packet concerning the bank 1 has been sent and acknowledged by the Host, the TXCMPL bit is set by the USB controller. This triggers a USB interrupt if enabled. The firmware should clear the TXCMPL bit before filling the endpoint FIFO bank 1 with new data.

The bank switch is performed by the USB controller each time the TXRDY bit is set by the firmware. Until the TXRDY bit has been set by the firmware for an endpoint bank, the USB controller will answer a NAK handshake for each IN requests concerning this bank.

Note that in the example above, the firmware clears the Transmit Complete bit (TXCBulk-outMPL) before setting the Transmit Ready bit (TXRDY). This is done in order to avoid the firmware to clear at the same time the TXCMPL bit for for bank 0 and the bank 1.

The firmware should never write more Bytes than supported by the endpoint FIFO.

## Control Transactions

### Setup Stage

The DIR bit in the UEPSTAX register should be at 0.

Receiving Setup packets is the same as receiving Bulk Out packets, except that the RXSETUP bit in the UEPSTAX register is set by the USB controller instead of the RXOUTB0 bit to indicate that an Out packet with a Setup PID has been received on the Control endpoint. When the RXSETUP bit has been set, all the other bits of the UEPSTAX register are cleared and an interrupt is triggered if enabled.

The firmware has to read the Setup request stored in the Control endpoint FIFO before clearing the RXSETUP bit to free the endpoint FIFO for the next transaction.

### Data Stage: Control Endpoint Direction

The data stage management is similar to Bulk management.

A Control endpoint is managed by the USB controller as a full-duplex endpoint: IN and OUT. All other endpoint types are managed as half-duplex endpoint: IN or OUT. The firmware has to specify the control endpoint direction for the data stage using the DIR bit in the UEPSTAX register.

- If the data stage consists of INs, the firmware has to set the DIR bit in the UEPSTAX register before writing into the FIFO and sending the data by setting to 1 the TXRDY bit in the UEPSTAX register. The IN transaction is complete when the TXCMPL has been set by the hardware. The firmware should clear the TXCMPL bit before any other transaction.
- If the data stage consists of OUTs, the firmware has to leave the DIR bit at 0. The RXOUTB0 bit is set by hardware when a new valid packet has been received on the endpoint. The firmware must read the data stored into the FIFO and then clear the RXOUTB0 bit to reset the FIFO and to allow the next transaction.

To send a STALL handshake, see “STALL Handshake” on page 94.

### Status Stage

The DIR bit in the UEPSTAX register should be reset at 0 for IN and OUT status stage.

The status stage management is similar to Bulk management.

- For a Control Write transaction or a No-Data Control transaction, the status stage consists of a IN Zero Length Packet (see “Bulk/Interrupt IN Transactions in Standard Mode” on page 89). To send a STALL handshake, see “STALL Handshake” on page 94.
- For a Control Read transaction, the status stage consists of a OUT Zero Length Packet (see “Bulk/Interrupt OUT Transactions in Standard Mode” on page 87).

## Isochronous Transactions

### Isochronous OUT Transactions in Standard Mode

An endpoint should be first enabled and configured before being able to receive Isochronous packets.

When an OUT packet is received on an endpoint, the RXOUTB0 bit is set by the USB controller. This triggers an interrupt if enabled. The firmware has to select the correct Bulk-outspending endpoint, store the number of data Bytes by reading the UBYCTX register. If the received packet is a ZLP (Zero Length Packet), the UBYCTX register value is equal to 0 and no data has to be read.

The STLCRC bit in the UEPSTAX register is set by the USB controller if the packet stored in FIFO has a corrupted CRC. This bit is updated after each new packet receipt.

When all the endpoint FIFO Bytes have been read, the firmware should clear the RXOUTB0 bit to allow the USB controller to store the next OUT packet data into the endpoint FIFO. Until the RXOUTB0 bit has been cleared by the firmware, the data sent by the Host at each OUT transaction will be lost.

If the RXOUTB0 bit is cleared while the Host is sending data, the USB controller will store only the remaining Bytes into the FIFO.

If the Host sends more Bytes than supported by the endpoint FIFO, the overflow data won't be stored, but the USB controller will consider that the packet is valid if the CRC is correct.

### Isochronous OUT Transactions in Ping-pong Mode

An endpoint should be first enabled and configured before being able to receive Isochronous packets.

When a OUT packet is received on the endpoint bank 0, the RXOUTB0 bit is set by the USB controller. This triggers an interrupt if enabled. The firmware has to select the corresponding endpoint, store the number of data Bytes by reading the UBYCTX register. If the received packet is a ZLP (Zero Length Packet), the UBYCTX register value is equal to 0 and no data has to be read.

The STLCRC bit in the UEPSTAX register is set by the USB controller if the packet stored in FIFO has a corrupted CRC. This bit is updated after each new packet receipt.

When all the endpoint FIFO Bytes have been read, the firmware should clear the RXOUB0 bit to allow the USB controller to store the next OUT packet data into the endpoint FIFO bank 0. This action switches the endpoint bank 0 and 1. Until the RXOUTB0 bit has been cleared by the firmware, the data sent by the Host on the bank 0 endpoint FIFO will be lost.

If the RXOUTB0 bit is cleared while the Host is sending data on the endpoint bank 0, the USB controller will store only the remaining Bytes into the FIFO.

When a new OUT packet is received on the endpoint bank 1, the RXOUTB1 bit is set by the USB controller. This triggers an interrupt if enabled. The firmware empties the bank 1 endpoint FIFO before clearing the RXOUTB1 bit. Until the RXOUTB1 bit has been cleared by the firmware, the data sent by the Host on the bank 1 endpoint FIFO will be lost.

The RXOUTB0 and RXOUTB1 bits are alternatively set by the USB controller at each new packet receipt.

The firmware has to clear one of these 2 bits after having read all the data FIFO to allow a new packet to be stored in the corresponding bank.

If the Host sends more Bytes than supported by the endpoint FIFO, the overflow data won't be stored, but the USB controller will consider that the packet is valid if the CRC is correct.

**Isochronous IN Transactions in Standard Mode**

An endpoint should be first enabled and configured before being able to send Isochronous packets.

The firmware should fill the FIFO with the data to be sent and set the TXRDY bit in the UEPSTAX register to allow the USB controller to send the data stored in FIFO at the next IN request concerning this endpoint.

If the TXRDY bit is not set when the IN request occurs, nothing will be sent by the USB controller.

When the IN packet has been sent, the TXCMPL bit in the UEPSTAX register is set by the USB controller. This triggers a USB interrupt if enabled. The firmware should clear the TXCMPL bit before filling the endpoint FIFO with new data.

The firmware should never write more Bytes than supported by the endpoint FIFO

**Isochronous IN Transactions in Ping-pong Mode**

An endpoint should be first enabled and configured before being able to send Isochronous packets.

The firmware should fill the FIFO bank 0 with the data to be sent and set the TXRDY bit in the UEPSTAX register to allow the USB controller to send the data stored in FIFO at the next IN request concerning the endpoint. The FIFO banks are automatically switched, and the firmware can immediately write into the endpoint FIFO bank 1.

If the TXRDY bit is not set when the IN request occurs, nothing will be sent by the USB controller.

When the IN packet concerning the bank 0 has been sent, the TXCMPL bit is set by the USB controller. This triggers a USB interrupt if enabled. The firmware should clear the TXCMPL bit before filling the endpoint FIFO bank 0 with new data. The FIFO banks are then automatically switched.

When the IN packet concerning the bank 1 has been sent, the TXCMPL bit is set by the USB controller. This triggers a USB interrupt if enabled. The firmware should clear the TXCMPL bit before filling the endpoint FIFO bank 1 with new data.

The bank switch is performed by the USB controller each time the TXRDY bit is set by the firmware. Until the TXRDY bit has been set by the firmware for an endpoint bank, the USB controller won't send anything at each IN requests concerning this bank.

The firmware should never write more Bytes than supported by the endpoint FIFO.

## Miscellaneous

### USB Reset

The EORINT bit in the USBINT register is set by hardware when a End Of Reset has been detected on the USB bus. This triggers a USB interrupt if enabled. The USB controller is still enabled, but all the USB registers are reset by hardware. The firmware should clear the EORINT bit to allow the next USB reset detection.

### STALL Handshake

This function is only available for Control, Bulk, and Interrupt endpoints.

The firmware has to set the STALLRQ bit in the UEPSTAX register to send a STALL handshake at the next request of the Host on the endpoint selected with the UEPNUM register. The RXSETUP, TXRDY, TXCMPL, RXOUTB0 and RXOUTB1 bits must be first resetted to 0. The bit STLCRC is set at 1 by the USB controller when a STALL has been sent. This triggers an interrupt if enabled.

The firmware should clear the STALLRQ and STLCRC bits after each STALL sent. The STALLRQ bit is cleared automatically by hardware when a valid SETUP PID is received on a CONTROL type endpoint.

Important note: when a Clear Halt Feature occurs for an endpoint, the firmware should reset this endpoint using the UEPRST register in order to reset the data toggle management.

### Start of Frame Detection

The SOFINT bit in the USBINT register is set when the USB controller detects a Start Of Frame PID. This triggers an interrupt if enabled. The firmware should clear the SOFINT bit to allow the next Start of Frame detection.

### Frame Number

When receiving a Start Of Frame, the frame number is automatically stored in the UFNUML and UFNUMH registers. The CRCOK and CRCERR bits indicate if the CRC of the last Start Of Frame is valid (CRCOK set at 1) or corrupted (CRCERR set at 1). The UFNUML and UFNUMH registers are automatically updated when receiving a new Start of Frame.

### Data Toggle Bit

The Data Toggle bit is set by hardware when a DATA0 packet is received and accepted by the USB controller and cleared by hardware when a DATA1 packet is received and accepted by the USB controller. This bit is reset when the firmware resets the endpoint FIFO using the UEPRST register.

For Control endpoints, each SETUP transaction starts with a DATA0 and data toggling is then used as for Bulk endpoints until the end of the Data stage (for a control write transfer). The Status stage completes the data transfer with a DATA1 (for a control read transfer).

For Isochronous endpoints, the device firmware should ignore the data-toggle.

## Suspend/Resume Management

### Suspend

The Suspend state can be detected by the USB controller if all the clocks are enabled and if the USB controller is enabled. The bit SPINT is set by hardware when an idle state is detected for more than 3 ms. This triggers a USB interrupt if enabled.

In order to reduce current consumption, the firmware can stop the clocks and put the C51 in Idle or Power-down mode. The Resume detection is still active.

The stop of the 48 MHz clock from the PLL should be done in the following order:

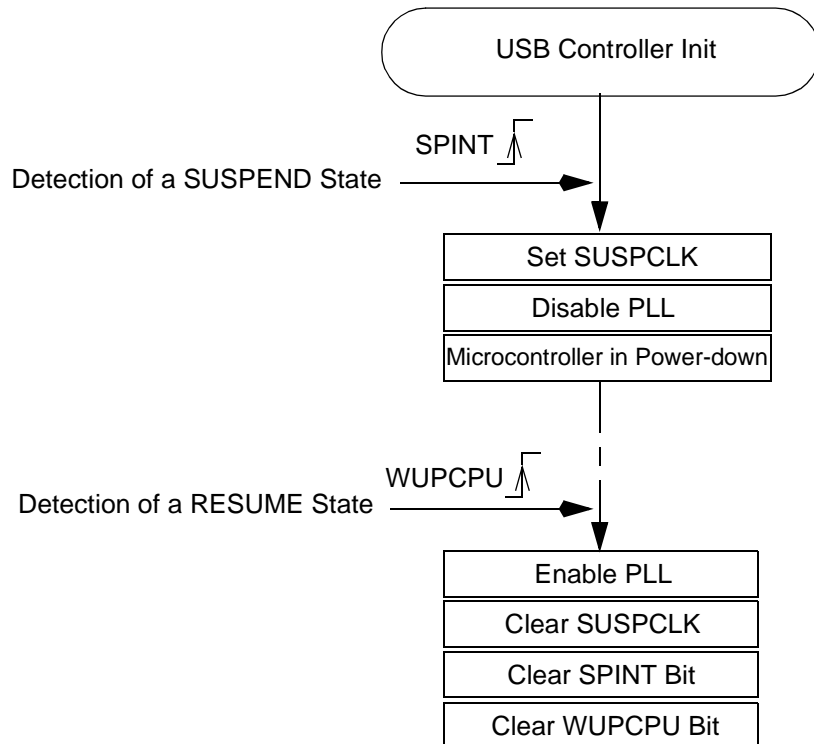
1. Disable of the 48 MHz clock input of the USB controller by setting to 1 the SUSPCLK bit in the USBCON register.
2. Disable the PLL by clearing the PLEN bit in the PLLCON register.

### Resume

When the USB controller is in Suspend state, the Resume detection is active even if all the clocks are disabled and if the C51 is in Idle or Power-down mode. The WUPCPU bit is set by hardware when a non-idle state occurs on the USB bus. This triggers an interrupt if enabled. This interrupt wakes up the CPU from its Idle or Power-down state and the interrupt function is then executed. The firmware should first enable the 48 MHz generation and then reset to 0 the SUSPCLK bit in the USBCON register if needed. The firmware has to clear the SPINT bit in the USBINT register before any other USB operation in order to wake up the USB controller from its Suspend mode.

The USB controller is then re-activated.

**Figure 66.** Example of a Suspend/Resume management



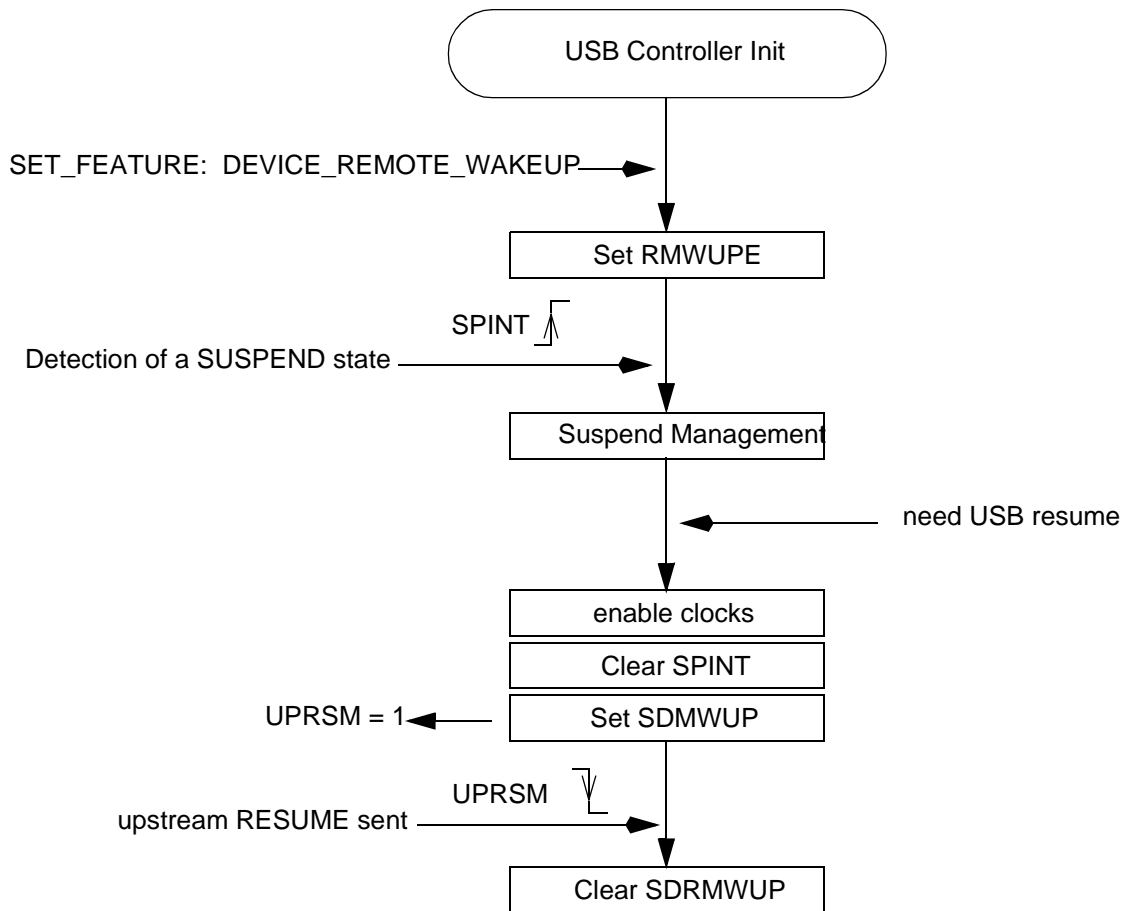
## Upstream Resume

A USB device can be allowed by the Host to send an upstream resume for Remote Wake-up purpose.

When the USB controller receives the SET\_FEATURE request: DEVICE\_REMOTE\_WAKEUP, the firmware should set to 1 the RMWUPE bit in the USBCON register to enable this functionality. RMWUPE value should be 0 in the other cases.

If the device is in SUSPEND mode, the USB controller can send an upstream resume by clearing first the SPINT bit in the USBINT register and by setting then to 1 the SDRMWUP bit in the USBCON register. The USB controller sets to 1 the UPRSM bit in the USBCON register. All clocks must be enabled first. The Remote Wake is sent only if the USB bus was in Suspend state for at least 5ms. When the upstream resume is completed, the UPRSM bit is reset to 0 by hardware. The firmware should then clear the SDRMWUP bit.

**Figure 67.** Example of REMOTE WAKEUP Management





USB Interrupt System

Interrupt System Priorities

Figure 68. USB Interrupt Control System

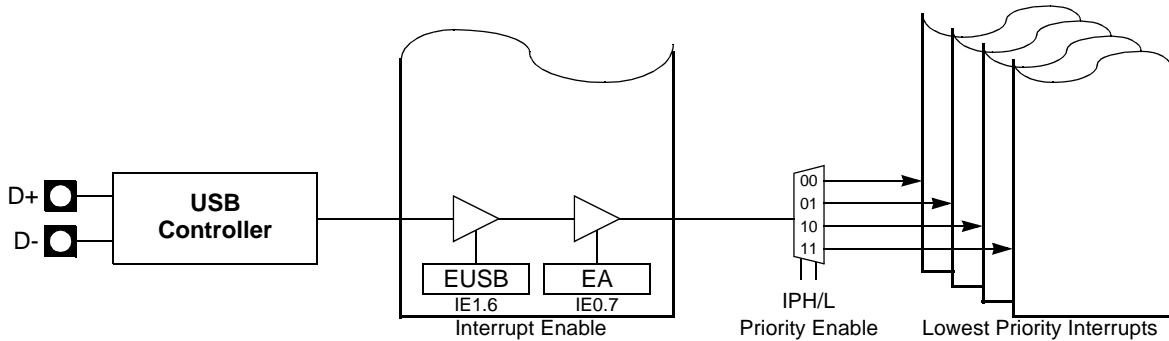


Table 1. Priority Levels

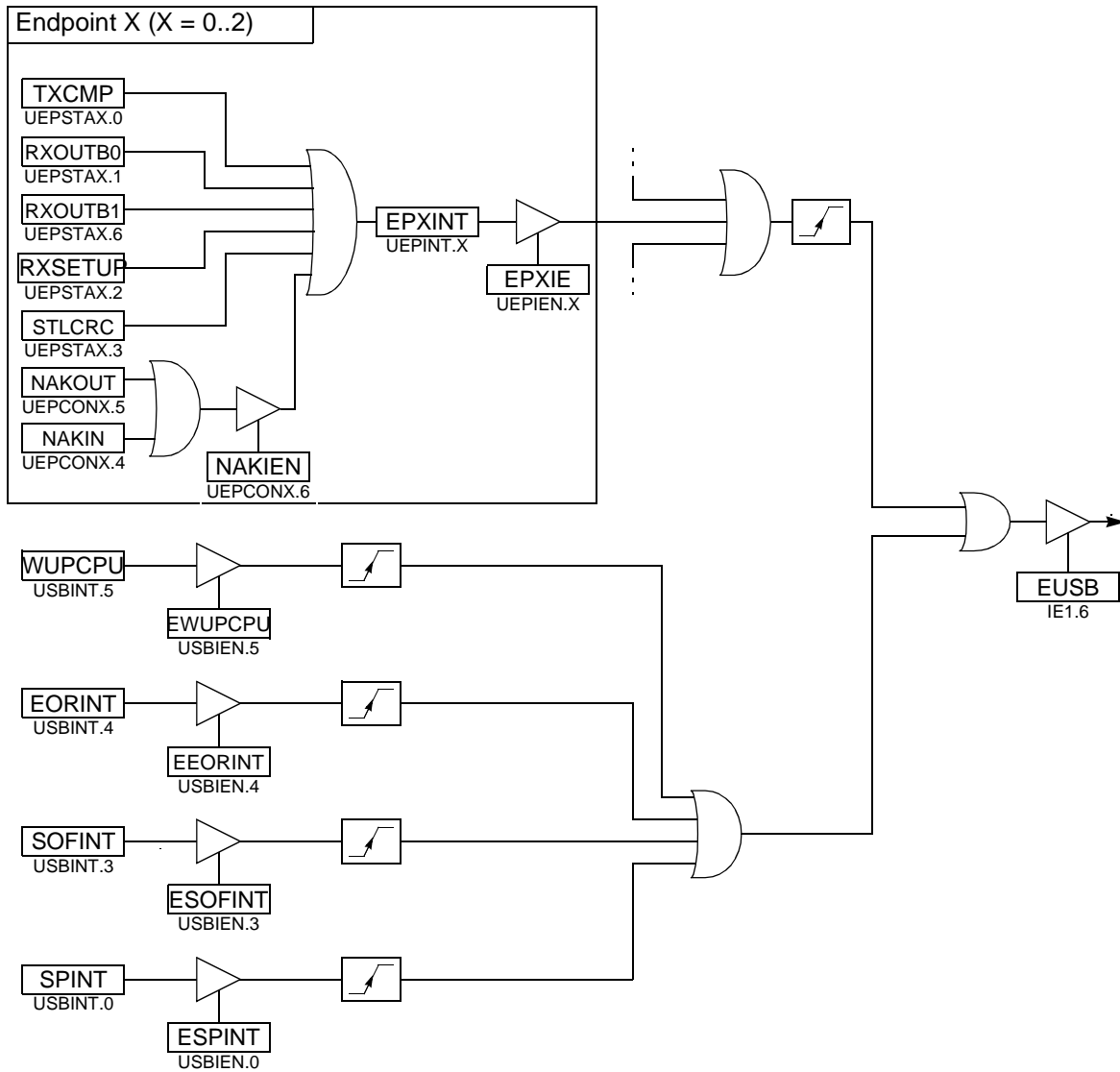
IPHUSB	IPLUSB	USB Priority Level
0	0	0.....Lowest
0	1	1
1	0	2
1	1	3.....Highest

USB Interrupt Control System

As shown in Figure 69, many events can produce a USB interrupt:

- TXCMPL: Transmitted In Data (Table 96 on page 103). This bit is set by hardware when the Host accept a In packet.
- RXOUTB0: Received Out Data Bank 0 (Table 96 on page 103). This bit is set by hardware when an Out packet is accepted by the endpoint and stored in bank 0.
- RXOUTB1: Received Out Data Bank 1 (only for Ping-pong endpoints) (Table 96 on page 103). This bit is set by hardware when an Out packet is accepted by the endpoint and stored in bank 1.
- RXSETUP: Received Setup (Table 96 on page 103). This bit is set by hardware when an SETUP packet is accepted by the endpoint.
- STLCRC: STALLED (only for Control, Bulk and Interrupt endpoints) (Table 96 on page 103). This bit is set by hardware when a STALL handshake has been sent as requested by STALLRQ, and is reset by hardware when a SETUP packet is received.
- SOFINT: Start of Frame Interrupt (Table 92 on page 100). This bit is set by hardware when a USB start of frame packet has been received.
- WUPCPU: Wake-Up CPU Interrupt (Table 92 on page 100). This bit is set by hardware when a USB resume is detected on the USB bus, after a SUSPEND state.
- SPINT: Suspend Interrupt (Table 92 on page 100). This bit is set by hardware when a USB suspend is detected on the USB bus.

Figure 69. USB Interrupt Control Block Diagram



## Registers

**Table 90.** USBCON Register  
 USBCON (S:BCh) – USB Global Control Register

7	6	5	4	3	2	1	0
USBE	SUSPCLK	SDRMWUP	-	UPRSM	RMWUPE	CONFG	FADDEN
Bit Number	Bit Mnemonic	Description					
7	USBE	<b>USB Enable Bit</b> Set this bit to enable the USB controller. Clear this bit to disable and reset the USB controller, to disable the USB transceiver and to disable the USB controller clock inputs.					
6	SUSPCLK	<b>Suspend USB Clock Bit</b> Set to disable the 48 MHz clock input (Resume Detection is still active). Clear to enable the 48 MHz clock input.					
5	SDRMWUP	<b>Send Remote Wake-Up Bit</b> Set to force an external interrupt on the USB controller for Remote Wake UP purpose. An upstream resume is send only if the bit RMWUPE is set, all USB clocks are enabled AND the USB bus was in SUSPEND state for at least 5 ms. See UPRSM below. Cleared by software.					
4	-	<b>Reserved</b> The value read from this bit is always 0. Do not set this bit.					
3	UPRSM	<b>Upstream Resume Bit (read only)</b> Set by hardware when SDRMWUP has been set and if RMWUPE is enabled. Cleared by hardware after the upstream resume has been sent.					
2	RMWUPE	<b>Remote Wake-Up Enable Bit</b> Set to enabled request an upstream resume signaling to the host. Clear after the upstream resume has been indicated by RSMINPR. Note: Do not set this bit if the host has not set the DEVICE_REMOTE_WAKEUP feature for the device.					
1	CONFG	<b>Configuration Bit</b> This bit should be set by the device firmware after a SET_CONFIGURATION request with a non-zero value has been correctly processed. It should be cleared by the device firmware when a SET_CONFIGURATION request with a zero value is received. It is cleared by hardware on hardware reset or when an USB reset is detected on the bus (SE0 state for at least 32 Full Speed bit times: typically 2.7 μs).					
0	FADDEN	<b>Function Address Enable Bit</b> This bit should be set by the device firmware after a successful status phase of a SET_ADDRESS transaction. It should not be cleared afterwards by the device firmware. It is cleared by hardware on hardware reset or when an USB reset is received (see above). When this bit is cleared, the default function address is used (0).					

Reset Value = 0000 0000b

**Table 91.** USBADDR Register  
USBADDR (S:C6h) – USB Address Register

7	6	5	4	3	2	1	0
FEN	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0
Bit Number	Bit Mnemonic	Description					
7	FEN	<b>Function Enable Bit</b> Set to enable the function. The device firmware should set this bit after it has received a USB reset and participate in the following configuration process with the default address (FEN is reset to 0). Cleared by hardware at power-up, should not be cleared by the device firmware once set.					
6 - 0	UADD6:0	<b>USB Address Bits</b> This field contains the default address (0) after power-up or USB bus reset. It should be written with the value set by a SET_ADDRESS request received by the device firmware.					

Reset Value = 0000 0000b

**Table 92.** USBINT Register  
USBINT (S:BDh) – USB Global Interrupt Register

7	6	5	4	3	2	1	0
-	-	WUPCPU	EORINT	SOFINT	-	-	SPINT
Bit Number	Bit Mnemonic	Description					
7 - 6	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
5	WUPCPU	<b>Wake Up CPU Interrupt Flag</b> Set by hardware when the USB controller is in SUSPEND state and is re-activated by a non-idle signal from USB line (not by an upstream resume). This triggers a USB interrupt when EWUPCPU is set in the USBIEN. Cleared by software after re-enabling all USB clocks.					
4	EORINT	<b>End of Reset Interrupt Flag</b> Set by hardware when a End of Reset has been detected by the USB controller. This triggers a USB interrupt when EEORINT is set in USBIEN. Cleared by software.					
3	SOFINT	<b>Start of Frame Interrupt Flag</b> Set by hardware when an USB Start of Frame packet (SOF) has been properly received. This triggers a USB interrupt when ESOFINT is set in USBIEN. Cleared by software.					
2 - 1	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
0	SPINT	<b>Suspend Interrupt Flag</b> Set by hardware when a USB Suspend (Idle bus for three frame periods: a J state for 3 ms) is detected. This triggers a USB interrupt when ESPINT is set in USBIEN. Cleared by software.					

Reset Value = 0000 0000b

**Table 93.** USBIEN Register  
USBIEN (S:BEh) – USB Global Interrupt Enable Register

7	6	5	4	3	2	1	0
-	-	EWUPCPU	EORINT	ESOFINT	-	-	ESPINT
Bit Number	Bit Mnemonic	Description					
7 - 6	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
5	EWUPCPU	<b>Wake Up CPU Interrupt Enable Bit</b> Set to enable the Wake Up CPU interrupt. Clear to disable the Wake Up CPU interrupt.					
4	EORINT	<b>End Of Reset Interrupt Enable Bit</b> Set to enable the End Of Reset interrupt. This bit is set after reset. Clear to disable End Of Reset interrupt.					
3	ESOFINT	<b>Start Of Frame Interrupt Enable Bit</b> Set to enable the SOF interrupt. Clear to disable the SOF interrupt.					
2 - 1	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
0	ESPINT	<b>Suspend Interrupt Enable Bit</b> Set to enable Suspend interrupt. Clear to disable Suspend interrupt.					

Reset Value = 0001 0000b

**Table 94.** UEPNUM Register  
UEPNUM (S:C7h) – USB Endpoint Number

7	6	5	4	3	2	1	0
-	-	-	-	-	-	EPNUM1	EPNUM0
Bit Number	Bit Mnemonic	Description					
7 - 2	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
1 - 0	EPNUM1:0	<b>Endpoint Number Bits</b> Set this field with the number of the endpoint which should be accessed when reading or writing to registers UEPSTAX, UEPDATX, UBYCTX or UEPCONX.					

Reset Value = 0000 0000b

**Table 95.** UEPCONX Register  
 UEPCONX (S:D4h) – USB Endpoint X Control Register (X = EPNUM set in UEPNUM)

7	6	5	4	3	2	1	0
EPEN	NAKIEN	NAKOUT	NAKIN	DTGL	EPDIR	EPTYPE1	EPTYPE0
Bit Number	Bit Mnemonic	Description					
7	EPEN	<b>Endpoint Enable Bit</b> Set to enable the endpoint according to the device configuration. Endpoint 0 should always be enabled after a hardware or USB bus reset and participate in the device configuration. Clear to disable the endpoint according to the device configuration.					
6	NAKIEN	<b>NAK Interrupt enable</b> Set this bit to enable NAK IN or NAK OUT interrupt. Clear this bit to disable NAK IN or NAK OUT Interrupt.					
5	NAKOUT	<b>NAK OUT received</b> This bit is set by hardware when an NAK handshake has been sent in response of a OUT request from the Host. This triggers a USB interrupt when NAKIEN is set. This bit should be cleared by software.					
4	NAKIN	<b>NAK IN received</b> This bit is set by hardware when an NAK handshake has been sent in response of a IN request from the Host. This triggers a USB interrupt when NAKIEN is set. This bit should be cleared by software.					
3	DTGL	<b>Data Toggle Status Bit (Read-only)</b> Set by hardware when a DATA1 packet is received. Cleared by hardware when a DATA0 packet is received.					
2	EPDIR	<b>Endpoint Direction Bit</b> Set to configure IN direction for Bulk, Interrupt and Isochronous endpoints. Clear to configure OUT direction for Bulk, Interrupt and Isochronous endpoints. This bit has no effect for Control endpoints.					
1-0	EPTYPE1:0	<b>Endpoint Type Bits</b> Set this field according to the endpoint configuration (Endpoint 0 should always be configured as Control): 00 Control endpoint 01 Isochronous endpoint 10 Bulk endpoint 11 Interrupt endpoint					

Reset Value = 1000 0000b

**Table 96.** UEPSTAX Register

UEPSTAX (S:CEh) – USB Endpoint X Status and Control Register (X = EPNUM set in UEPNUM)

7	6	5	4	3	2	1	0
DIR	RXOUTB1	STALLRQ	TXRDY	STLCRC	RXSETUP	RXOUTB0	TXCMP
Bit Number	Bit Mnemonic	Description					
7	DIR	<p><b>Control Endpoint Direction Bit</b>                      This bit is relevant only if the endpoint is configured in Control type. Set for the data stage. Clear otherwise.                      Note: This bit should be configured on RXSETUP interrupt before any other bit is changed. This also determines the status phase (IN for a control write and OUT for a control read). This bit should be cleared for status stage of a Control Out transaction.</p>					
6	RXOUTB1	<p><b>Received OUT Data Bank 1 for Endpoints 1 and 2 (Ping-pong mode)</b>                      This bit is set by hardware after a new packet has been stored in the endpoint FIFO data bank 1 (only in Ping-pong mode). Then, the endpoint interrupt is triggered if enabled and all the following OUT packets to the endpoint bank 1 are rejected (NAK'ed) until this bit has been cleared, excepted for Isochronous Endpoints.                      This bit should be cleared by the device firmware after reading the OUT data from the endpoint FIFO.</p>					

Bit Number	Bit Mnemonic	Description
5	STALLRQ	<b>Stall Handshake Request Bit</b> Set to send a STALL answer to the host for the next handshake. Clear otherwise.
4	TXRDY	<b>TX Packet Ready Control Bit</b> Set after a packet has been written into the endpoint FIFO for IN data transfers. Data should be written into the endpoint FIFO only after this bit has been cleared. Set this bit without writing data to the endpoint FIFO to send a Zero Length Packet, which is generally recommended and may be required to terminate a transfer when the length of the last data packet is equal to MaxPacketSize (e.g. for control read transfers). Cleared by hardware, as soon as the packet has been sent for Isochronous endpoints, or after the host has acknowledged the packet for Control, Bulk and Interrupt endpoints.
3	STLCRC	<b>Stall Sent Interrupt Flag/CRC Error Interrupt Flag</b> <b>For Control, Bulk and Interrupt Endpoints:</b> Set by hardware after a STALL handshake has been sent as requested by STALLRQ. Then, the endpoint interrupt is triggered if enabled in UEPIEN. Cleared by hardware when a SETUP packet is received (see RXSETUP). <b>For Isochronous Endpoints:</b> Set by hardware if the last data received is corrupted (CRC error on data). Then, the endpoint interrupt is triggered if enabled in UEPIEN. Cleared by hardware when a non corrupted data is received.
2	RXSETUP	<b>Received SETUP Interrupt Flag</b> Set by hardware when a valid SETUP packet has been received from the host. Then, all the other bits of the register are cleared by hardware and the endpoint interrupt is triggered if enabled in UEPIEN. Clear by software after reading the SETUP data from the endpoint FIFO.
1	RXOUTB0	<b>Received OUT Data Bank 0</b> (see also RXOUTB1 bit for Ping-pong Endpoints) This bit is set by hardware after a new packet has been stored in the endpoint FIFO data bank 0. Then, the endpoint interrupt is triggered if enabled and all the following OUT packets to the endpoint bank 0 are rejected (NAK'ed) until this bit has been cleared, excepted for Isochronous Endpoints. However, for control endpoints, an early SETUP transaction may overwrite the content of the endpoint FIFO, even if its Data packet is received while this bit is set. This bit should be cleared by the device firmware after reading the OUT data from the endpoint FIFO.
0	TXCMP	<b>Transmitted IN Data Complete Interrupt Flag</b> Set by hardware after an IN packet has been transmitted for Isochronous endpoints and after it has been accepted (ACK'ed) by the host for Control, Bulk and Interrupt endpoints. Then, the endpoint interrupt is triggered if enabled in UEPIEN. Clear by software before setting again TXRDY.

Reset Value = 0000 0000b



**Table 97. UEPRST Register**  
 UEPRST (S:D5h) – USB Endpoint FIFO Reset Register

7	6	5	4	3	2	1	0
-	-	-	-	-	EP2RST	EP1RST	EP0RST

Bit Number	Bit Mnemonic	Description
7 - 3	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.
2	EP2RST	<b>Endpoint 2 FIFO Reset</b> Set and clear to reset the endpoint 2 FIFO prior to any other operation, upon hardware reset or when an USB bus reset has been received.
1	EP1RST	<b>Endpoint 1 FIFO Reset</b> Set and clear to reset the endpoint 1 FIFO prior to any other operation, upon hardware reset or when an USB bus reset has been received.
0	EP0RST	<b>Endpoint 0 FIFO Reset</b> Set and clear to reset the endpoint 0 FIFO prior to any other operation, upon hardware reset or when an USB bus reset has been received.

Reset Value = 0000 0000b

**Table 98. UEPINT Register**  
 UEPINT (S:F8h Read-only) – USB Endpoint Interrupt Register

7	6	5	4	3	2	1	0
-	-	-	-	-	EP2INT	EP1INT	EP0INT

Bit Number	Bit Mnemonic	Description
7 - 3	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.
2	EP2INT	<b>Endpoint 2 Interrupt Flag</b> Set by hardware when an interrupt is triggered in UEPSTAX and the endpoint 2 interrupt is enabled in UEPIEN. Must be cleared by software.
1	EP1INT	<b>Endpoint 1 Interrupt Flag</b> Set by hardware when an interrupt is triggered in UEPSTAX and the endpoint 1 interrupt is enabled in UEPIEN. Must be cleared by software.
0	EP0INT	<b>Endpoint 0 Interrupt Flag</b> Set by hardware when an interrupt is triggered in UEPSTAX and the endpoint 0 interrupt is enabled in UEPIEN. Must be cleared by software.

Reset Value = 0000 0000b

**Table 99.** UEPIEN Register  
 UEPIEN (S:C2h) – USB Endpoint Interrupt Enable Register

7	6	5	4	3	2	1	0
-	-	-	-	-	EP2INTE	EP1INTE	EP0INTE

Bit Number	Bit Mnemonic	Description
7 - 3	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.
2	EP2INTE	<b>Endpoint 2 Interrupt Enable Bit</b> Set to enable the interrupts for endpoint 2. Clear this bit to disable the interrupts for endpoint 2.
1	EP1INTE	<b>Endpoint 1 Interrupt Enable Bit</b> Set to enable the interrupts for the endpoint 1. Clear to disable the interrupts for the endpoint 1.
0	EP0INTE	<b>Endpoint 0 Interrupt Enable Bit</b> Set to enable the interrupts for the endpoint 0. Clear to disable the interrupts for the endpoint 0.

Reset Value = 0000 0000b

**Table 100.** UEPDATX Register  
 UEPDATX (S:CFh) – USB Endpoint X FIFO Data Register (X = EPNUM set in UEPNUM)

7	6	5	4	3	2	1	0
FDAT7	FDAT6	FDAT5	FDAT4	FDAT3	FDAT2	FDAT1	FDAT0

Bit Number	Bit Mnemonic	Description
7 - 0	FDAT7:0	<b>Endpoint X FIFO Data</b> Data Byte to be written to FIFO or data Byte to be read from the FIFO, for the Endpoint X (see EPNUM).

Reset Value = XXh

**Table 101.** UBYCTX Register

UBYCTX (S:E2h) – USB Endpoint X Byte Count Register (X = EPNUM set in UEPNUM)

	7	6	5	4	3	2	1	0
	-	BYCT6	BYCT5	BYCT4	BYCT3	BYCT2	BYCT1	BYCT0

Bit Number	Bit Mnemonic	Description
7	-	<b>Reserved</b> The value read from this bits is always 0. Do not set this bit.
6 - 0	BYCT7:0	<b>Byte Count</b> Byte count of a received data packet. This Byte count is equal to the number of data Bytes received after the Data PID.

Reset Value = 0000 0000b

**Table 102.** UFNUML Register

UFNUML (S:BAh, Read-only) – USB Frame Number Low Register

	7	6	5	4	3	2	1	0
	FNUM7	FNUM6	FNUM5	FNUM4	FNUM3	FNUM2	FNUM1	FNUM0

Bit Number	Bit Mnemonic	Description
7 - 0	FNUM7:0	<b>Frame Number</b> Lower 8 bits of the 11-bit Frame Number.

Reset Value = 00h

**Table 103.** UFNUMH Register  
UFNUMH (S:BBh, Read-only) – USB Frame Number High Register

7	6	5	4	3	2	1	0
-	-	CRCOK	CRCERR	-	FNUM10	FNUM9	FNUM8
Bit Number	Bit Mnemonic	Description					
7 - 3	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
5	CRCOK	<b>Frame Number CRC OK Bit</b> Set by hardware after a non corrupted Frame Number in Start of Frame Packet is received. Updated after every Start Of Frame packet reception. Note: The Start Of Frame interrupt is generated just after the PID receipt.					
4	CRCERR	<b>Frame Number CRC Error Bit</b> Set by hardware after a corrupted Frame Number in Start of Frame Packet is received. Updated after every Start Of Frame packet reception. Note: The Start Of Frame interrupt is generated just after the PID receipt.					
3	-	<b>Reserved</b> The value read from this bits is always 0. Do not set this bit.					
2-0	FNUM10:8	<b>Frame Number</b> Upper 3 bits of the 11-bit Frame Number. It is provided in the last received SOF packet. FNUM does not change if a corrupted SOF is received.					

Reset Value = 00h

**Table 104.** USBCLK Register  
USBCLK (S:EAh) – USB Clock Divider Register

7	6	5	4	3	2	1	0
-	-	-	-	-	-	USBCD1	USBCD0
Bit Number	Bit Mnemonic	Description					
7 - 2	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
1 - 0	USBCD1:0	<b>USB Controller Clock Divider</b> 2-bit divider for USB controller clock generation.					

Reset Value = 0000 0000b

## MultiMedia Card Controller

The AT8xC51SND1C implements a MultiMedia Card (MMC) controller. The MMC is used to store MP3 encoded audio files in removable Flash memory cards that can be easily plugged or removed from the application.

### Card Concept

The basic MultiMedia Card concept is based on transferring data via a minimum number of signals.

### Card Signals

The communication signals are:

- CLK: with each cycle of this signal a one bit transfer on the command and data lines is done. The frequency may vary from zero to the maximum clock frequency.
- CMD: is a bi-directional command channel used for card initialization and data transfer commands. The CMD signal has 2 operation modes: open-drain for initialization mode and push-pull for fast command transfer. Commands are sent from the MultiMedia Card bus master to the card and responses from the cards to the host.
- DAT: is a bi-directional data channel. The DAT signal operates in push-pull mode. Only one card or the host is driving this signal at a time.

### Card Registers

Within the card interface five registers are defined: OCR, CID, CSD, RCA and DSR. These can be accessed only by the corresponding commands.

The 32-bit Operation Conditions Register (OCR) stores the  $V_{DD}$  voltage profile of the card. The register is optional and can be read only.

The 128-bit wide CID register carries the card identification information (Card ID) used during the card identification procedure.

The 128-bit wide Card-Specific Data register (CSD) provides information on how to access the card contents. The CSD defines the data format, error correction type, maximum data access time, data transfer speed, and whether the DSR register can be used. The 16-bit Relative Card Address register (RCA) carries the card address assigned by the host during the card identification. This address is used for the addressed host-card communication after the card identification procedure.

The 16-bit Driver Stage Register (DSR) can be optionally used to improve the bus performance for extended operating conditions (depending on parameters like bus length, transfer rate or number of cards).

## Bus Concept

The MultiMedia Card bus is designed to connect either solid-state mass-storage memory or I/O-devices in a card format to multimedia applications. The bus implementation allows the coverage of application fields from low-cost systems to systems with a fast data transfer rate. It is a single master bus with a variable number of slaves. The MultiMedia Card bus master is the bus controller and each slave is either a single mass storage card (with possibly different technologies such as ROM, OTP, Flash etc.) or an I/O-card with its own controlling unit (on card) to perform the data transfer.

The MultiMedia Card bus also includes power connections to supply the cards.

The bus communication uses a special protocol (MultiMedia Card bus protocol) which is applicable for all devices. Therefore, the payload data transfer between the host and the cards can be bi-directional.

## Bus Lines

The MultiMedia Card bus architecture requires all cards to be connected to the same set of lines. No card has an individual connection to the host or other devices, which reduces the connection costs of the MultiMedia Card system.

The bus lines can be divided into three groups:

- Power supply:  $V_{SS1}$  and  $V_{SS2}$ ,  $V_{DD}$  – used to supply the cards.
- Data transfer: MCMD, MDAT – used for bi-directional communication.
- Clock: MCLK – used to synchronize data transfer across the bus.

## Bus Protocol

After a power-on reset, the host must initialize the cards by a special message-based MultiMedia Card bus protocol. Each message is represented by one of the following tokens:

- Command: a command is a token which starts an operation. A command is transferred serially from the host to the card on the MCMD line.
- Response: a response is a token which is sent from an addressed card (or all connected cards) to the host as an answer to a previously received command. It is transferred serially on the MCMD line.
- Data: data can be transferred from the card to the host or vice-versa. Data is transferred serially on the MDAT line.

Card addressing is implemented using a session address assigned during the initialization phase, by the bus controller to all currently connected cards. Individual cards are identified by their CID number. This method requires that every card will have a unique CID number. To ensure uniqueness of CIDs the CID register contains 24 bits (MID and OID fields) which are defined by the MMCA. Every card manufacturers is required to apply for an unique MID (and optionally OID) number.

MultiMedia Card bus data transfers are composed of these tokens. One data transfer is a bus operation. There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token, the others transfer their information directly within the command or response structure. In this case no data token is present in an operation. The bits on the MDAT and the MCMD lines are transferred synchronous to the host clock.

2 types of data transfer commands are defined:

- Sequential commands: These commands initiate a continuous data stream, they are terminated only when a stop command follows on the MCMD line. This mode reduces the command overhead to an absolute minimum.
- Block-oriented commands: These commands send a data block succeeded by CRC bits. Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the MCMD line similarly to the stream read.

Figure 70 through Figure 74 show the different types of operations, on these figures, grayed tokens are from host to card(s) while white tokens are from card(s) to host.

**Figure 70.** Sequential Read Operation

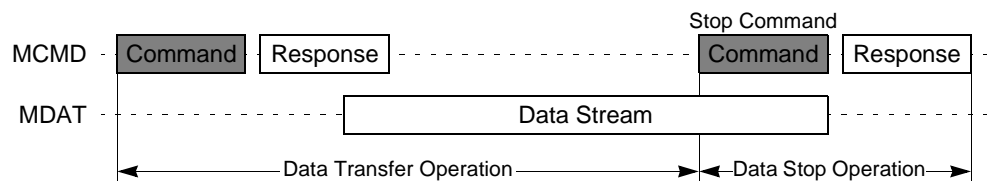
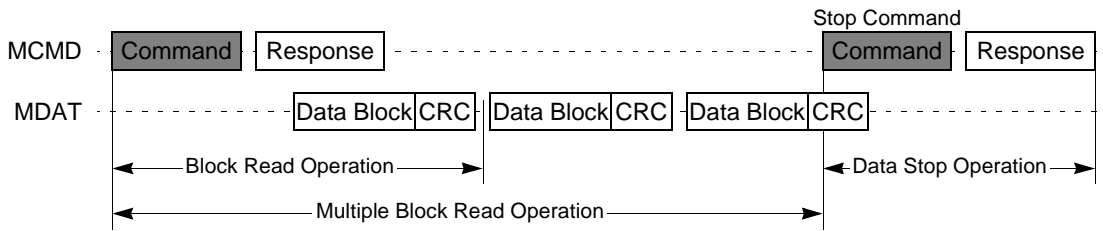


Figure 71. (Multiple) Block Read Operation



As shown in Figure 72 and Figure 73 the data write operation uses a simple busy signaling of the write operation duration on the data line (MDAT).

Figure 72. Sequential Write Operation

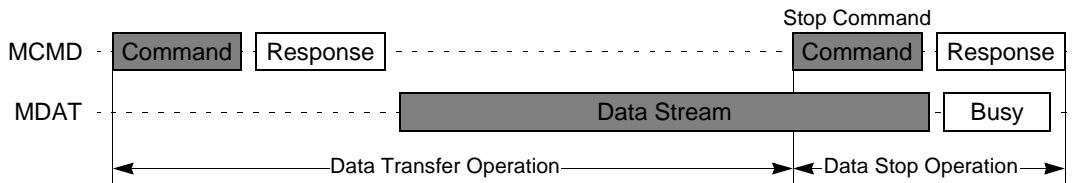


Figure 73. Multiple Block Write Operation

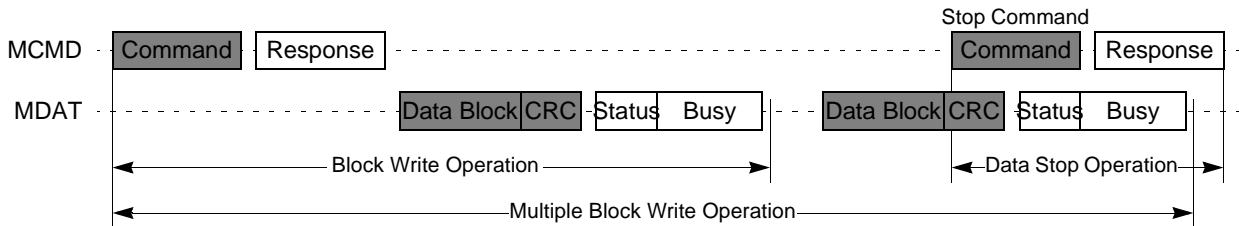
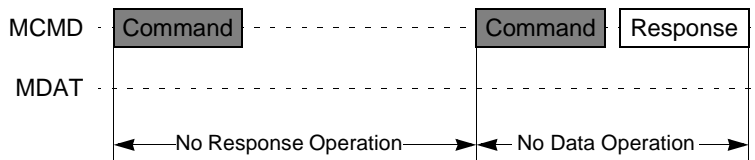


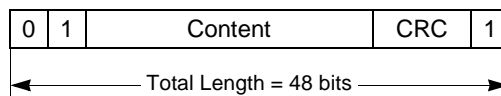
Figure 74. No Response and No Data Operation



**Command Token Format**

As shown in Figure 75, commands have a fixed code length of 48 bits. Each command token is preceded by a Start bit: a low level on MCMD line and succeeded by an End bit: a high level on MCMD line. The command content is preceded by a Transmission bit: a high level on MCMD line for a command token (host to card) and succeeded by a 7-bit CRC so that transmission errors can be detected and the operation may be repeated. Command content contains the command index and address information or parameters.

Figure 75. Command Token Format



**Table 105.** Command Token Format

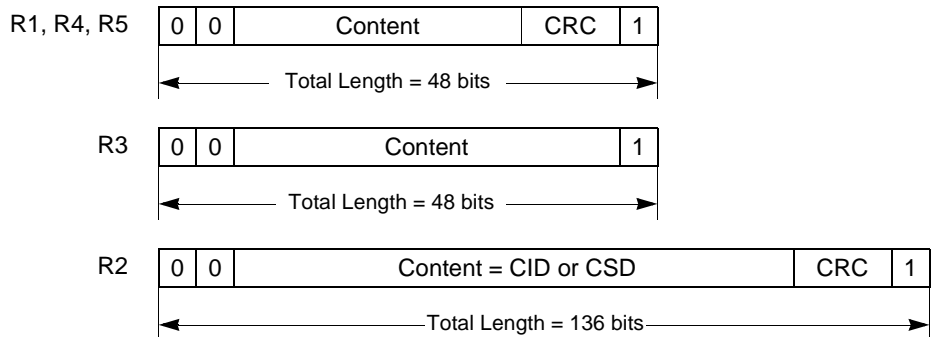
<b>Bit Position</b>	<b>47</b>	<b>46</b>	<b>45:40</b>	<b>39:8</b>	<b>7:1</b>	<b>0</b>
<b>Width (Bits)</b>	1	1	6	32	7	1
<b>Value</b>	'0'	'1'	-	-	-	'1'
<b>Description</b>	Start bit	Transmission bit	Command Index	Argument	CRC7	End bit

**Response Token Format**

There are five types of response tokens (R1 to R5). As shown in Figure 76, responses have a code length of 48 bits or 136 bits. A response token is preceded by a Start bit: a low level on MCMD line and succeeded by an End bit: a high level on MCMD line. The command content is preceded by a Transmission bit: a low level on MCMD line for a response token (card to host) and succeeded (R1,R2,R4,R5) or not (R3) by a 7 - bit CRC.

Response content contains mirrored command and status information (R1 response), CID register or CSD register (R2 response), OCR register (R3 response), or RCA register (R4 and R5 response).

**Figure 76.** Response Token Format



**Table 106.** R1 Response Format (Normal Response)

<b>Bit Position</b>	<b>47</b>	<b>46</b>	<b>45:40</b>	<b>39:8</b>	<b>7:1</b>	<b>0</b>
<b>Width (bits)</b>	1	1	6	32	7	1
<b>Value</b>	'0'	'0'	-	-	-	'1'
<b>Description</b>	Start bit	Transmission bit	Command Index	Card Status	CRC7	End bit

**Table 107.** R2 Response Format (CID and CSD registers)

<b>Bit Position</b>	<b>135</b>	<b>134</b>	<b>[133:128]</b>	<b>[127:1]</b>	<b>0</b>
<b>Width (bits)</b>	1	1	6	32	1
<b>Value</b>	'0'	'0'	'111111'	-	'1'
<b>Description</b>	Start bit	Transmission bit	Reserved	Argument	End bit



**Table 108.** R3 Response Format (OCR Register)

<b>Bit Position</b>	47	46	[45:40]	[39:8]	[7:1]	0
<b>Width (bits)</b>	1	1	6	32	7	1
<b>Value</b>	'0'	'0'	'111111'	-	'1111111'	'1'
<b>Description</b>	Start bit	Transmission bit	Reserved	OCR register	Reserved	End bit

**Table 109.** R4 Response Format (Fast I/O)

<b>Bit Position</b>	47	46	[45:40]	[39:8]	[7:1]	0
<b>Width (bits)</b>	1	1	6	32	7	1
<b>Value</b>	'0'	'0'	'100111'	-	-	'1'
<b>Description</b>	Start bit	Transmission bit	Command Index	Argument	CRC7	End bit

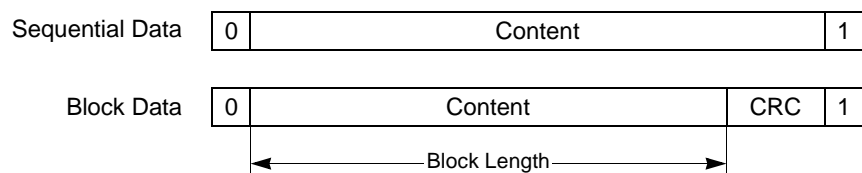
**Table 110.** R5 Response Format

<b>Bit Position</b>	47	46	[45:40]	[39:8]	[7:1]	0
<b>Width (bits)</b>	1	1	6	32	7	1
<b>Value</b>	'0'	'0'	'101000'	-	-	'1'
<b>Description</b>	Start bit	Transmission bit	Command Index	Argument	CRC7	End bit

## Data Packet Format

There are 2 types of data packets: stream and block. As shown in Figure 77, stream data packets have an indeterminate length while block packets have a fixed length depending on the block length. Each data packet is preceded by a Start bit: a low level on MCMD line and succeeded by an End bit: a high level on MCMD line. Due to the fact that there is no predefined end in stream packets, CRC protection is not included in this case. The CRC protection algorithm for block data is a 16-bit CCITT polynomial.

**Figure 77.** Data Token Format



## Clock Control

The MMC bus clock signal can be used by the host to turn the cards into energy saving mode or to control the data flow (to avoid under-run or over-run conditions) on the bus. The host is allowed to lower the clock frequency or shut it down.

There are a few restrictions the host must follow:

- The bus frequency can be changed at any time (under the restrictions of maximum data transfer frequency, defined by the cards, and the identification frequency defined by the specification document).
- It is an obvious requirement that the clock must be running for the card to output data or response tokens. After the last MultiMedia Card bus transaction, the host is

required, to provide 8 (eight) clock cycles for the card to complete the operation before shutting down the clock. Following is a list of the various bus transactions:

- A command with no response. 8 clocks after the host command End bit.
- A command with response. 8 clocks after the card command End bit.
- A read data transaction. 8 clocks after the End bit of the last data block.
- A write data transaction. 8 clocks after the CRC status token.
- The host is allowed to shut down the clock of a “busy” card. The card will complete the programming operation regardless of the host clock. However, the host must provide a clock edge for the card to turn off its busy signal. Without a clock edge the card (unless previously disconnected by a deselect command-CMD7) will force the MDAT line down, forever.

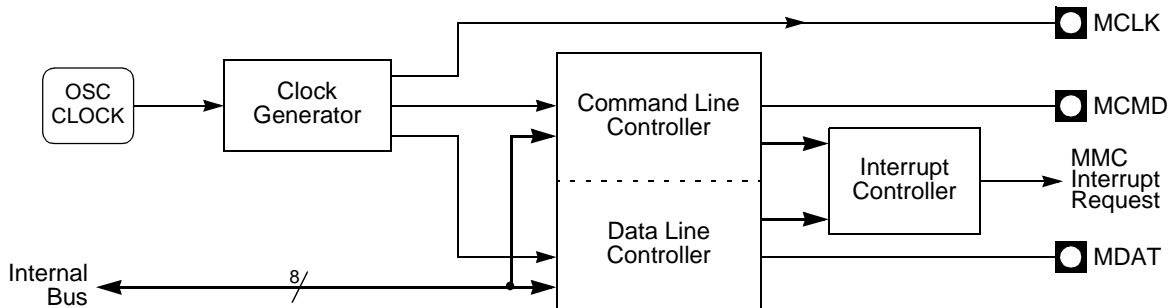
## Description

The MMC controller interfaces to the C51 core through the following eight special function registers:

MMCON0, MMCON1, MMCON2, the three MMC control registers (see Table 112 to Table 120); MMSTA, the MMC status register (see Table 115); MMINT, the MMC interrupt register (see Table 116); MMMSK, the MMC interrupt mask register (see Table 117); MMCMD, the MMC command register (see Table 118); MMDAT, the MMC data register (see Table 119); and MMCLK, the MMC clock register (see Table 120).

As shown in Figure 78, the MMC controller is divided in four blocks: the clock generator that handles the MCLK (formally the MMC CLK) output to the card, the command line controller that handles the MCMD (formally the MMC CMD) line traffic to or from the card, the data line controller that handles the MDAT (formally the MMC DAT) line traffic to or from the card, and the interrupt controller that handles the MMC controller interrupt sources. These blocks are detailed in the following sections.

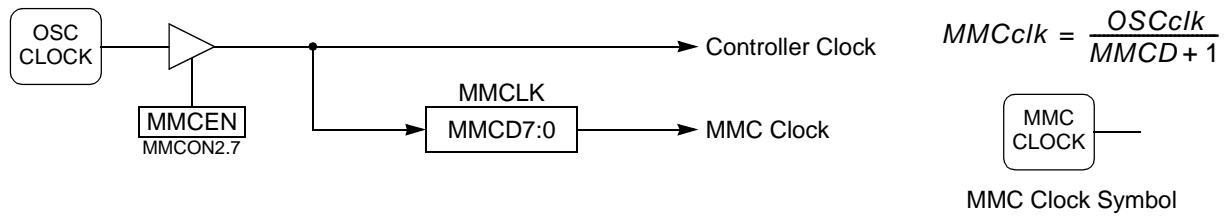
**Figure 78.** MMC Controller Block Diagram



## Clock Generator

The MMC clock is generated by division of the oscillator clock ( $F_{OSC}$ ) issued from the Clock Controller block as detailed in Section "Oscillator", page 12. The division factor is given by MMCD7:0 bits in MMCLK register, a value of 0x00 stops the MMC clock. Figure 79 shows the MMC clock generator and its output clock calculation formula.

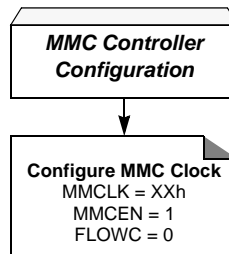
Figure 79. MMC Clock Generator and Symbol



As soon as MMCEN bit in MMCON2 is set, the MMC controller receives its system clock. The MMC command and data clock is generated on MCLK output and sent to the command line and data line controllers. Figure 80 shows the MMC controller configuration flow.

As exposed in Section “Clock Control”, page 113, MMCD7:0 bits can be used to dynamically increase or reduce the MMC clock.

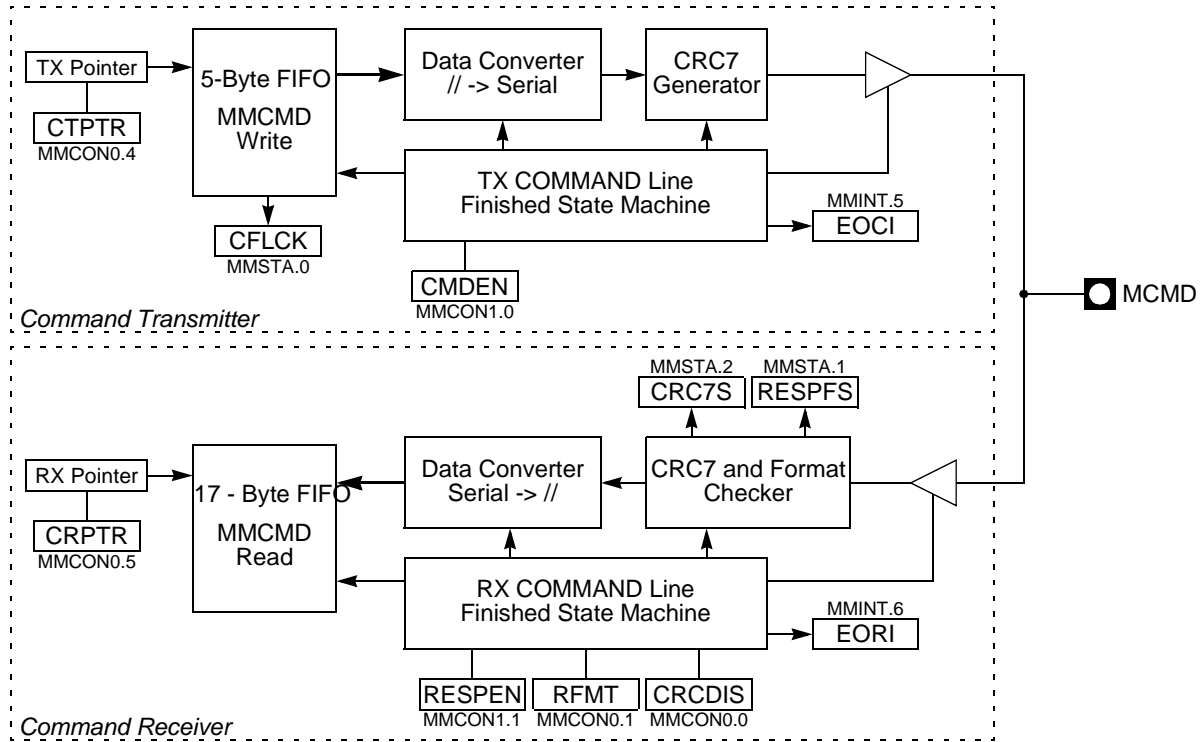
Figure 80. Configuration Flow



## Command Line Controller

As shown in Figure 81, the command line controller is divided in 2 channels: the command transmitter channel that handles the command transmission to the card through the MCMD line and the command receiver channel that handles the response reception from the card through the MCMD line. These channels are detailed in the following sections.

**Figure 81.** Command Line Controller Block Diagram



### Command Transmitter

For sending a command to the card, user must load the command index (1 Byte) and argument (4 Bytes) in the command transmit FIFO using the MMCMD register. Before starting transmission by setting and clearing the CMDEN bit in MMCON1 register, user must first configure:

- RESPEN bit in MMCON1 register to indicate whether a response is expected or not.
- RFMT bit in MMCON0 register to indicate the response size expected.
- CRCDIS bit in MMCON0 register to indicate whether the CRC7 included in the response will be computed or not. In order to avoid CRC error, CRCDIS may be set for response that do not include CRC7.

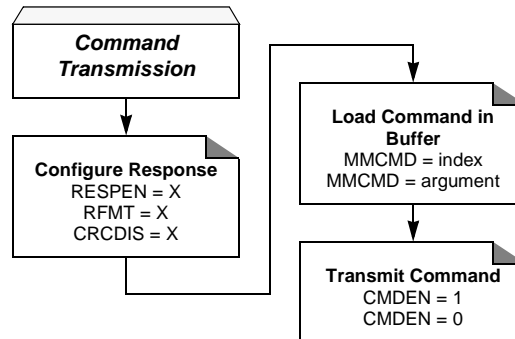
Figure 82 summarizes the command transmission flow.

As soon as command transmission is enabled, the CFLCK flag in MMSTA is set indicating that write to the FIFO is locked. This mechanism is implemented to avoid command overrun.

The end of the command transmission is signalled to you by the EOCI flag in MMINT register becoming set. This flag may generate an MMC interrupt request as detailed in Section "Interrupt", page 124. The end of the command transmission also resets the CFLCK flag.

User may abort command loading by setting and clearing the CTPTR bit in MMCON0 register which resets the write pointer to the transmit FIFO.

**Figure 82.** Command Transmission Flow



## Command Receiver

The end of the response reception is signalled to you by the EORI flag in MMINT register. This flag may generate an MMC interrupt request as detailed in Section "Interrupt", page 124. When this flag is set, 2 other flags in MMSTA register: RESPFS and CRC7S give a status on the response received. RESPFS indicates if the response format is correct or not: the size is the one expected (48 bits or 136 bits) and a valid End bit has been received, and CRC7S indicates if the CRC7 computation is correct or not. These Flags are cleared when a command is sent to the card and updated when the response has been received.

User may abort response reading by setting and clearing the CRPTR bit in MMCON0 register which resets the read pointer to the receive FIFO.

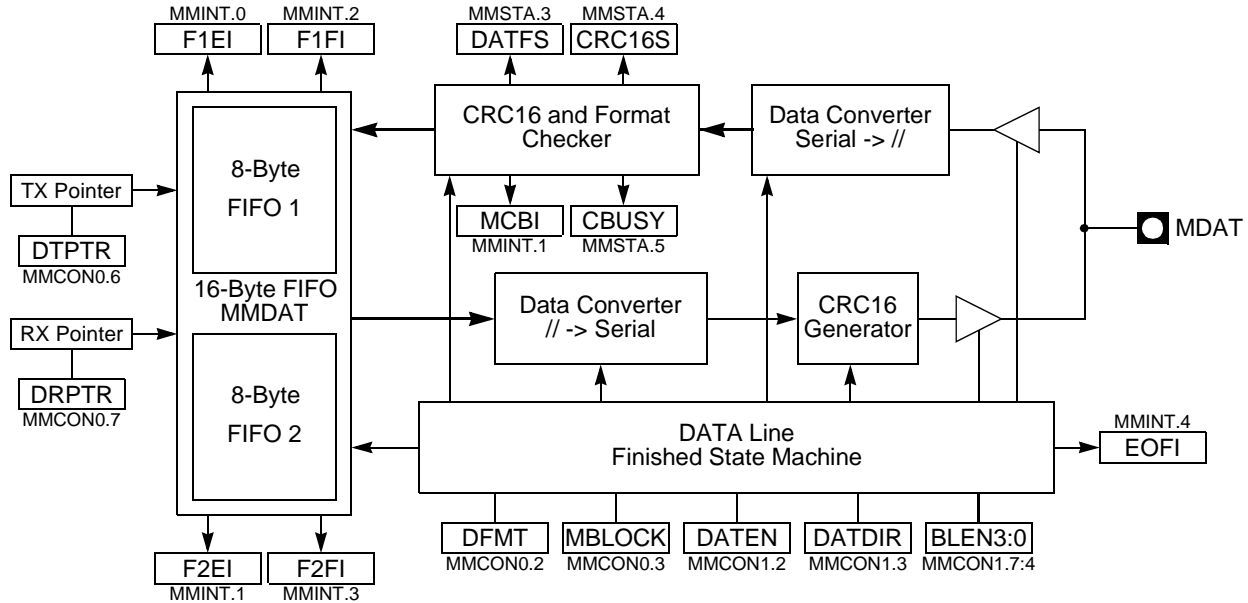
According to the MMC specification delay between a command and a response (formally  $N_{CR}$  parameter) can not exceed 64 MMC clock periods. To avoid any locking of the MMC controller when card does not send its response (e.g. physically removed from the bus), user must launch a time-out period to exit from such situation. In case of time-out user may reset the command controller and its internal state machine by setting and clearing the CCR bit in MMCON2 register.

This time-out may be disarmed when receiving the response.

## Data Line Controller

The data line controller is based on a 16-Byte FIFO used both by the data transmitter channel and by the data receiver channel.

Figure 83. Data Line Controller Block Diagram



### FIFO Implementation

The 16-Byte FIFO is based on a dual 8-Byte FIFOs managed using 2 pointers and four flags indicating the status full and empty of each FIFO.

Pointers are not accessible to user but can be reset at any time by setting and clearing DRPTR and DTPTR bits in MMCON0 register. Resetting the pointers is equivalent to abort the writing or reading of data.

F1EI and F2EI flags in MMINT register signal when set that respectively FIFO1 and FIFO2 are empty. F1FI and F2FI flags in MMINT register signal when set that respectively FIFO1 and FIFO2 are full. These flags may generate an MMC interrupt request as detailed in Section "Interrupt".

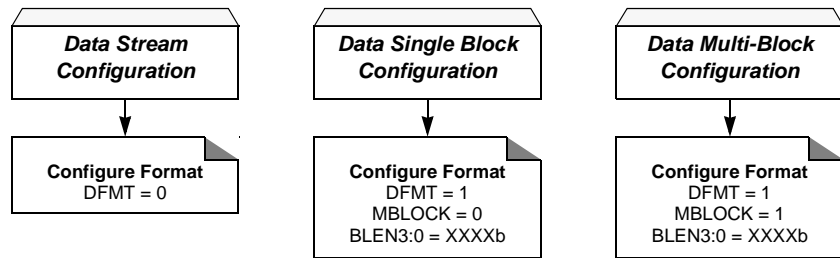
### Data Configuration

Before sending or receiving any data, the data line controller must be configured according to the type of the data transfer considered. This is achieved using the Data Format bit: DFMT in MMCON0 register. Clearing DFMT bit enables the data stream format while setting DFMT bit enables the data block format. In data block format, user must also configure the single or multi-block mode by clearing or setting the MBLOCK bit in MMCON0 register and the block length using BLEN3:0 bits in MMCON1 according to Table 111. Figure 84 summarizes the data modes configuration flows.

Table 111. Block Length Programming

BLEN3:0	Block Length (Byte)
BLEN = 0000 to 1011	Length = $2^{BLEN}$ : 1 to 2048
> 1011	Reserved: do not program BLEN3:0 > 1011

**Figure 84. Data Controller Configuration Flows**



## Data Transmitter

### Configuration

For transmitting data to the card user must first configure the data controller in transmission mode by setting the DATDIR bit in MMCON1 register.

Figure 85 summarizes the data stream transmission flows in both polling and interrupt modes while Figure 86 summarizes the data block transmission flows in both polling and interrupt modes, these flows assume that block length is greater than 16 data.

### Data Loading

Data is loaded in the FIFO by writing to MMDAT register. Number of data loaded may vary from 1 to 16 Bytes. Then if necessary (more than 16 Bytes to send) user must wait that one FIFO becomes empty (F1EI or F2EI set) before loading 8 new data.

### Data Transmission

Transmission is enabled by setting and clearing DATEN bit in MMCON1 register. Data is transmitted immediately if the response has already been received, or is delayed after the response reception if its status is correct. In both cases transmission is delayed if a card sends a busy state on the data line until the end of this busy condition.

According to the MMC specification, the data transfer from the host to the card may not start sooner than 2 MMC clock periods after the card response was received (formally  $N_{WR}$  parameter). To address all card types, this delay can be programmed using DATD1:0 bits in MMCON2 register from 3 MMC clock periods when DATD1:0 bits are cleared to 9 MMC clock periods when DATD1:0 bits are set, by step of 2 MMC clock periods.

### End of Transmission

The end of a data frame (block or stream) transmission is signalled to you by the EOFI flag in MMINT register. This flag may generate an MMC interrupt request as detailed in Section "Interrupt", page 124.

In data stream mode, EOFI flag is set, after reception of the End bit. This assumes user has previously sent the STOP command to the card, which is the only way to stop stream transfer.

In data block mode, EOFI flag is set, after reception of the CRC status token (see Figure 76). 2 other flags in MMSTA register: DATFS and CRC16S report a status on the frame sent. DATFS indicates if the CRC status token format is correct or not, and CRC16S indicates if the card has found the CRC16 of the block correct or not.

### Busy Status

As shown in Figure 76 the card uses a busy token during a block write operation. This busy status is reported to you by the CBUSY flag in MMSTA register and by the MCBI flag in MMINT which is set every time CBUSY toggles, i.e. when the card enters and exits its busy state. This flag may generate an MMC interrupt request as detailed in Section "Interrupt", page 124.

Figure 85. Data Stream Transmission Flows

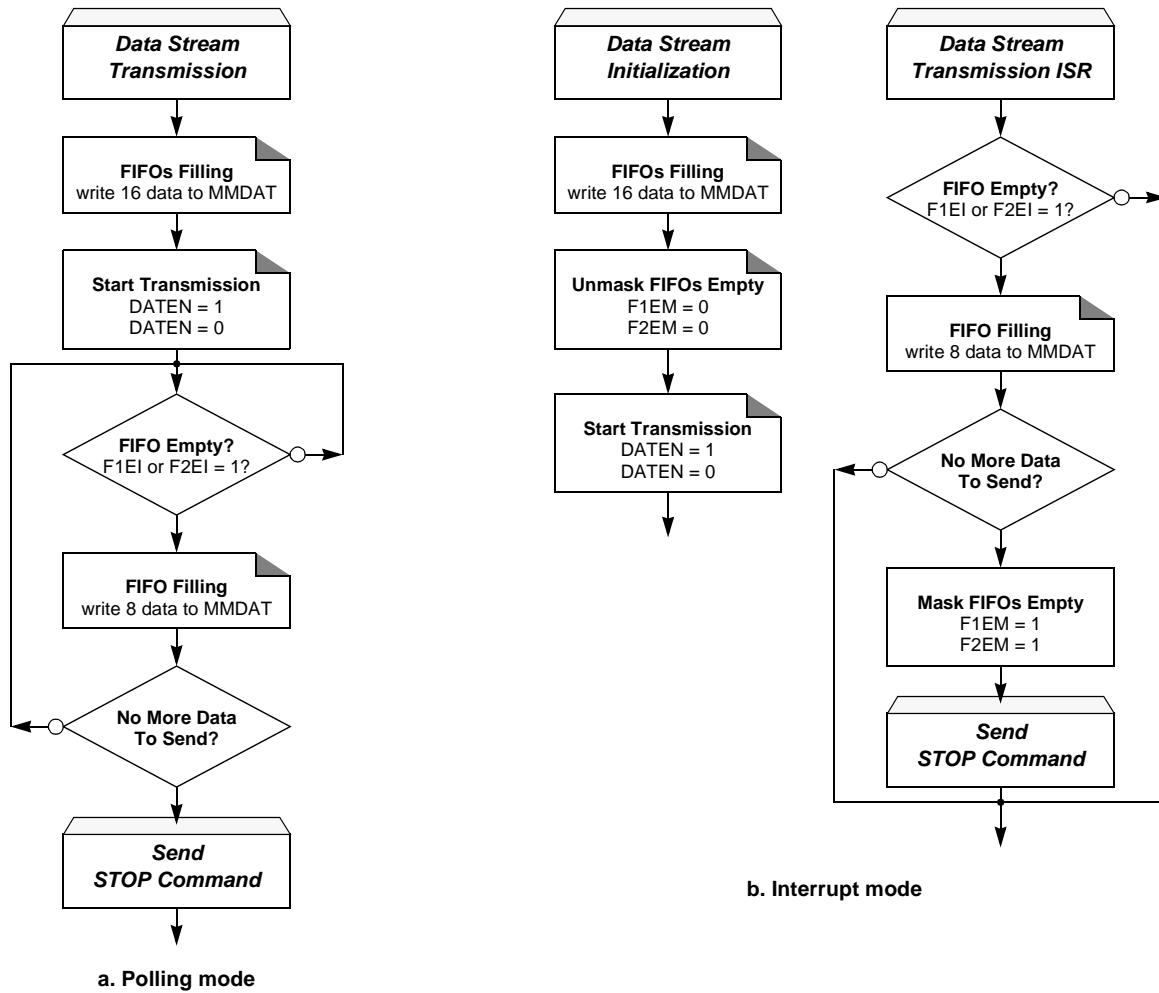
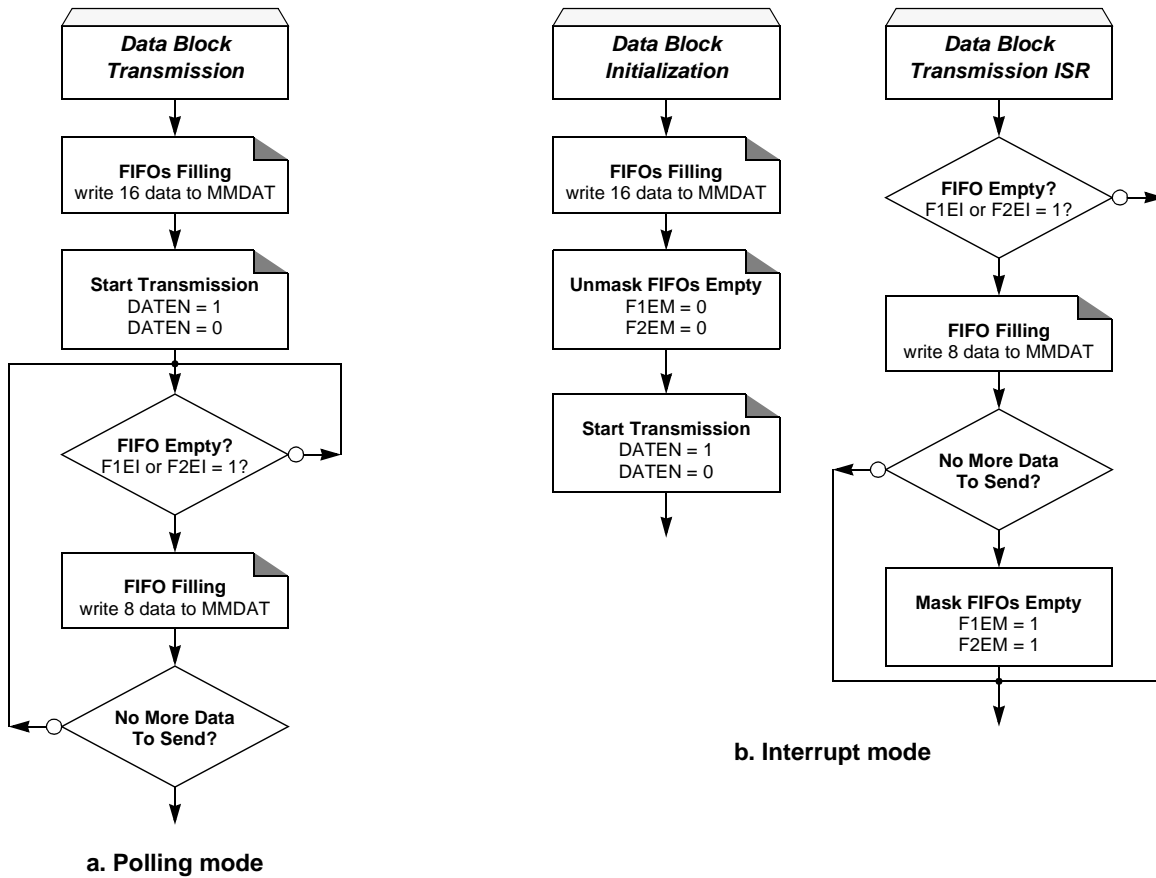




Figure 86. Data Block Transmission Flows



**Data Receiver**

*Configuration*

To receive data from the card you must first configure the data controller in reception mode by clearing the DATDIR bit in MMCON1 register.

Figure 87 summarizes the data stream reception flows in both polling and interrupt modes while Figure 88 summarizes the data block reception flows in both polling and interrupt modes, these flows assume that block length is greater than 16 Bytes.

*Data Reception*

The end of a data frame (block or stream) reception is signalled to you by the EOFI flag in MMINT register. This flag may generate an MMC interrupt request as detailed in Section "Interrupt", page 124. When this flag is set, 2 other flags in MMSTA register: DATFS and CRC16S give a status on the frame received. DATFS indicates if the frame format is correct or not: a valid End bit has been received, and CRC16S indicates if the CRC16 computation is correct or not. In case of data stream CRC16S has no meaning and stays cleared.

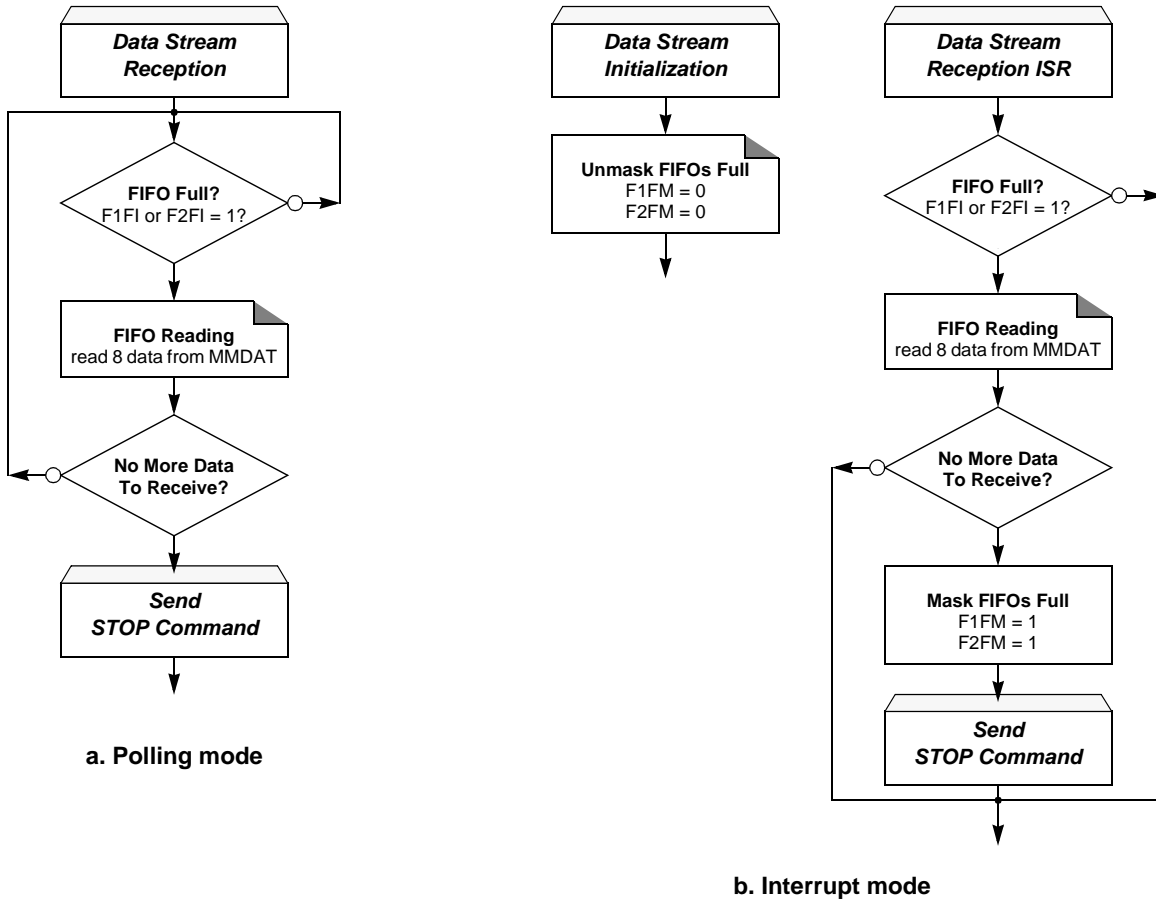
According to the MMC specification data transmission from the card starts after the access time delay (formally  $N_{AC}$  parameter) beginning from the End bit of the read command. To avoid any locking of the MMC controller when card does not send its data (e.g. physically removed from the bus), you must launch a time-out period to exit from such situation. In case of time-out you may reset the data controller and its internal state machine by setting and clearing the DCR bit in MMCON2 register.

This time-out may be disarmed after receiving 8 data (F1FI flag set) or after receiving end of frame (EOF1 flag set) in case of block length less than 8 data (1, 2 or 4).

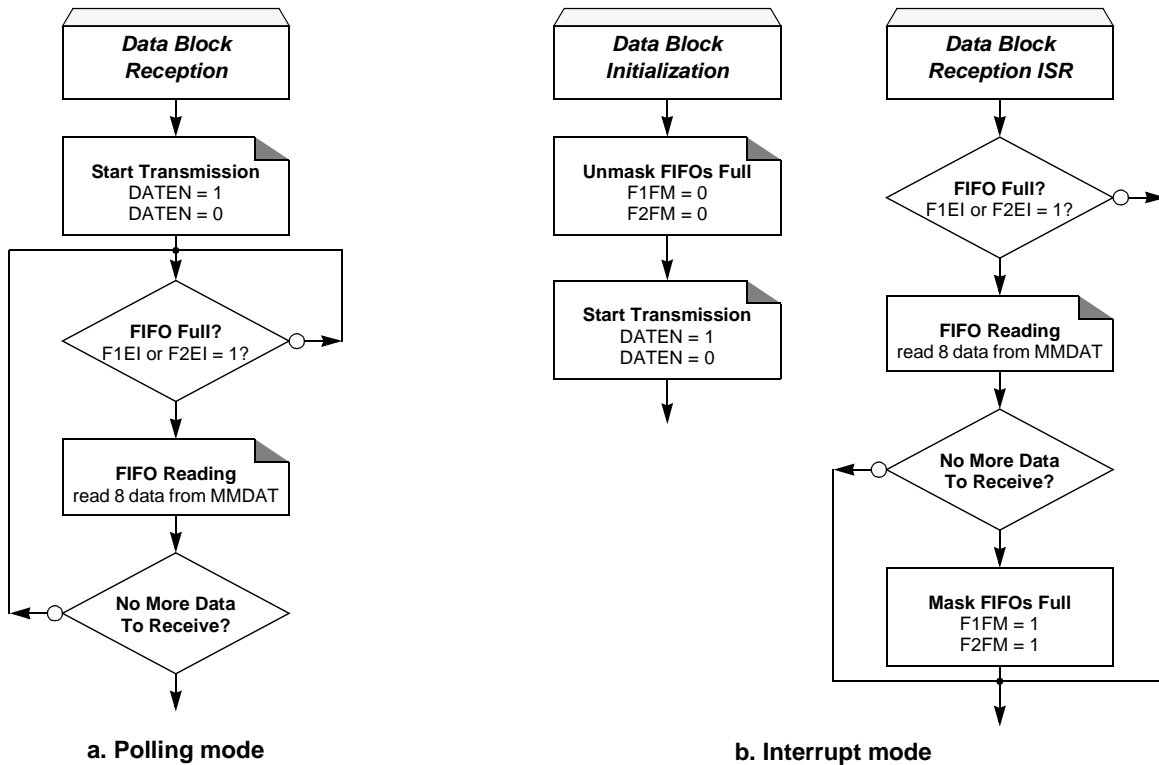
*Data Reading*

Data is read from the FIFO by reading to MMDAT register. Each time one FIFO becomes full (F1FI or F2FI set), user is requested to flush this FIFO by reading 8 data.

**Figure 87.** Data Stream Reception Flows



**Figure 88. Data Block Reception Flows**



## Flow Control

To allow transfer at high speed without taking care of CPU oscillator frequency, the FLOWC bit in MMCON2 allows control of the data flow in both transmission and reception.

During transmission, setting the FLOWC bit has the following effects:

- MMCLK is stopped when both FIFOs become empty: F1EI and F2EI set.
- MMCLK is restarted when one of the FIFOs becomes full: F1EI or F2EI cleared.

During reception, setting the FLOWC bit has the following effects:

- MMCLK is stopped when both FIFOs become full: F1FI and F2FI set.
- MMCLK is restarted when one of the FIFOs becomes empty: F1FI or F2FI cleared.

As soon as the clock is stopped, the MMC bus is frozen and remains in its state until the clock is restored by writing or reading data in MMDAT.

## Interrupt

### Description

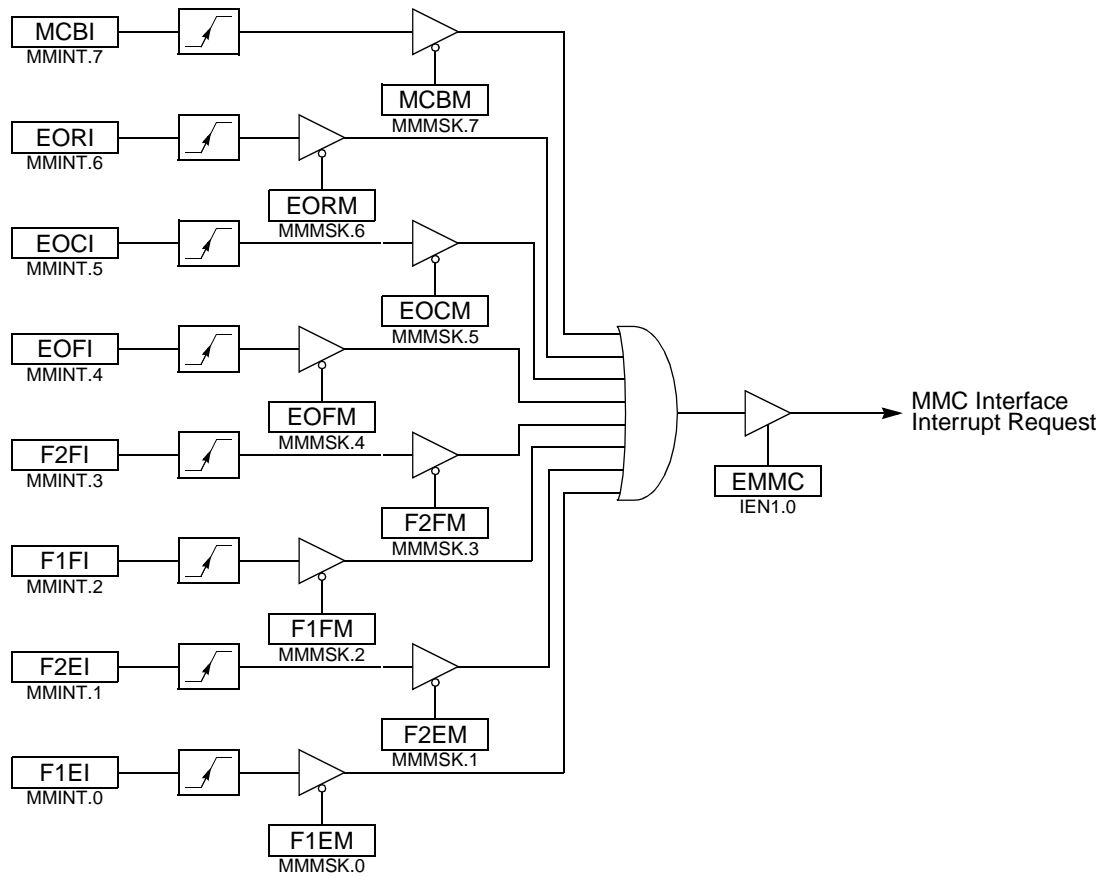
As shown in Figure 89, the MMC controller implements eight interrupt sources reported in MCB1, EORI, EOCl, EOFI, F2FI, F1FI, and F2EI flags in MMCINT register. These flags are detailed in the previous sections.

All these sources are maskable separately using MCBM, EORM, EOCM, EOFM, F2FM, F1FM, and F2EM mask bits respectively in MMMSK register.

The interrupt request is generated each time an unmasked flag is set, and the global MMC controller interrupt enable bit is set (EMMC in IEN1 register).

Reading the MMINT register automatically clears the interrupt flags (acknowledgment). This implies that register content must be saved and tested interrupt flag by interrupt flag to be sure not to forget any interrupts.

**Figure 89.** MMC Controller Interrupt System



## Registers

**Table 112.** MMCON0 Register

MMCON0 (S:E4h) – MMC Control Register 0

7	6	5	4	3	2	1	0
DRPTR	DTPTR	CRPTR	CTPTR	MBLOCK	DFMT	RFMT	CRCDIS
Bit Number	Bit Mnemonic	Description					
7	DRPTR	<b>Data Receive Pointer Reset Bit</b> Set to reset the read pointer of the data FIFO. Clear to release the read pointer of the data FIFO.					
6	DTPTR	<b>Data Transmit Pointer Reset Bit</b> Set to reset the write pointer of the data FIFO. Clear to release the write pointer of the data FIFO.					
5	CRPTR	<b>Command Receive Pointer Reset Bit</b> Set to reset the read pointer of the receive command FIFO. Clear to release the read pointer of the receive command FIFO.					
4	CTPTR	<b>Command Transmit Pointer Reset Bit</b> Set to reset the write pointer of the transmit command FIFO. Clear to release the read pointer of the transmit command FIFO.					
3	MBLOCK	<b>Multi-block Enable Bit</b> Set to select multi-block data format. Clear to select single block data format.					
2	DFMT	<b>Data Format Bit</b> Set to select the block-oriented data format. Clear to select the stream data format.					
1	RFMT	<b>Response Format Bit</b> Set to select the 48-bit response format. Clear to select the 136-bit response format.					
0	CRCDIS	<b>CRC7 Disable Bit</b> Set to disable the CRC7 computation when receiving a response. Clear to enable the CRC7 computation when receiving a response.					

Reset Value = 0000 0000b

**Table 113.** MMCON1 Register

MMCON1 (S:E5h) – MMC Control Register 1

7	6	5	4	3	2	1	0
BLEN3	BLEN2	BLEN1	BLEN0	DATDIR	DATEN	RESPEN	CMDEN
Bit Number	Bit Mnemonic	Description					
7 - 4	BLEN3:0	<b>Block Length Bits</b> Refer to Table 111 for bits description. Do not program value > 1011b					
3	DATDIR	<b>Data Direction Bit</b> Set to select data transfer from host to card (write mode). Clear to select data transfer from card to host (read mode).					
2	DATEN	<b>Data Transmission Enable Bit</b> Set and clear to enable data transmission immediately or after response has been received.					
1	RESPEN	<b>Response Enable Bit</b> Set and clear to enable the reception of a response following a command transmission.					
0	CMDEN	<b>Command Transmission Enable Bit</b> Set and clear to enable transmission of the command FIFO to the card.					

Reset Value = 0000 0000b

**Table 114.** MMCON2 Register

MMCON2 (S:E6h) – MMC Control Register 2

7	6	5	4	3	2	1	0
MMCEN	DCR	CCR	-	-	DATD1	DATD0	FLOWC
Bit Number	Bit Mnemonic	Description					
7	MMCEN	<b>MMC Clock Enable Bit</b> Set to enable the MCLK clocks and activate the MMC controller. Clear to disable the MMC clocks and freeze the MMC controller.					
6	DCR	<b>Data Controller Reset Bit</b> Set and clear to reset the data line controller in case of transfer abort.					
5	CCR	<b>Command Controller Reset Bit</b> Set and clear to reset the command line controller in case of transfer abort.					
4-3	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
2-1	DATD1:0	<b>Data Transmission Delay Bits</b> Used to delay the data transmission after a response from 3 MMC clock periods (all bits cleared) to 9 MMC clock periods (all bits set) by step of 2 MMC clock periods.					
0	FLOWC	<b>MMC Flow Control Bit</b> Set to enable the flow control during data transfers. Clear to disable the flow control during data transfers.					

Reset Value = 0000 0000b

**Table 115.** MMSTA Register

MMSTA (S:DEh Read Only) – MMC Control and Status Register

7	6	5	4	3	2	1	0
-	-	CBUSY	CRC16S	DATFS	CRC7S	RESPFS	CFLCK
Bit Number	Bit Mnemonic	Description					
7 - 6	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
5	CBUSY	<b>Card Busy Flag</b> Set by hardware when the card sends a busy state on the data line. Cleared by hardware when the card no more sends a busy state on the data line.					
4	CRC16S	<b>CRC16 Status Bit</b> <b>Transmission mode</b> Set by hardware when the token response reports a good CRC. Cleared by hardware when the token response reports a bad CRC. <b>Reception mode</b> Set by hardware when the CRC16 received in the data block is correct. Cleared by hardware when the CRC16 received in the data block is not correct.					
3	DATFS	<b>Data Format Status Bit</b> <b>Transmission mode</b> Set by hardware when the format of the token response is correct. Cleared by hardware when the format of the token response is not correct. <b>Reception mode</b> Set by hardware when the format of the frame is correct. Cleared by hardware when the format of the frame is not correct.					
2	CRC7S	<b>CRC7 Status Bit</b> Set by hardware when the CRC7 computed in the response is correct. Cleared by hardware when the CRC7 computed in the response is not correct. This bit is not relevant when CRCDIS is set.					
1	RESPFS	<b>Response Format Status Bit</b> Set by hardware when the format of a response is correct. Cleared by hardware when the format of a response is not correct.					
0	CFLCK	<b>Command FIFO Lock Bit</b> Set by hardware to signal user not to write in the transmit command FIFO: busy state. Cleared by hardware to signal user the transmit command FIFO is available: idle state.					

Reset Value = 0000 0000b

**Table 116.** MMINT Register

MMINT (S:E7h Read Only) – MMC Interrupt Register

7	6	5	4	3	2	1	0
MCBI	EORI	EOCI	EOFI	F2FI	F1FI	F2EI	F1EI
Bit Number	Bit Mnemonic	Description					
7	MCBI	<b>MMC Card Busy Interrupt Flag</b> Set by hardware when the card enters or exits its busy state (when the busy signal is asserted or deasserted on the data line). Cleared when reading MMINT.					
6	EORI	<b>End of Response Interrupt Flag</b> Set by hardware at the end of response reception. Cleared when reading MMINT.					
5	EOCI	<b>End of Command Interrupt Flag</b> Set by hardware at the end of command transmission. Clear when reading MMINT.					
4	EOFI	<b>End of Frame Interrupt Flag</b> Set by hardware at the end of frame (stream or block) transfer. Clear when reading MMINT.					
3	F2FI	<b>FIFO 2 Full Interrupt Flag</b> Set by hardware when second FIFO becomes full. Cleared by hardware when second FIFO becomes empty.					
2	F1FI	<b>FIFO 1 Full Interrupt Flag</b> Set by hardware when first FIFO becomes full. Cleared by hardware when first FIFO becomes empty.					
1	F2EI	<b>FIFO 2 Empty Interrupt Flag</b> Set by hardware when second FIFO becomes empty. Cleared by hardware when second FIFO becomes full.					
0	F1EI	<b>FIFO 1 Empty Interrupt Flag</b> Set by hardware when first FIFO becomes empty. Cleared by hardware when first FIFO becomes full.					

Reset Value = 0000 0011b



**Table 117. MMMSK Register**

MMMSK (S:DFh) – MMC Interrupt Mask Register

7	6	5	4	3	2	1	0
MCB <sub>M</sub>	EOR <sub>M</sub>	EOC <sub>M</sub>	EOF <sub>M</sub>	F2F <sub>M</sub>	F1F <sub>M</sub>	F2E <sub>M</sub>	F1E <sub>M</sub>
Bit Number	Bit Mnemonic	Description					
7	MCB <sub>M</sub>	<b>MMC Card Busy Interrupt Mask Bit</b> Set to prevent MCB <sub>I</sub> flag from generating an MMC interrupt. Clear to allow MCB <sub>I</sub> flag to generate an MMC interrupt.					
6	EOR <sub>M</sub>	<b>End Of Response Interrupt Mask Bit</b> Set to prevent EOR <sub>I</sub> flag from generating an MMC interrupt. Clear to allow EOR <sub>I</sub> flag to generate an MMC interrupt.					
5	EOC <sub>M</sub>	<b>End Of Command Interrupt Mask Bit</b> Set to prevent EOC <sub>I</sub> flag from generating an MMC interrupt. Clear to allow EOC <sub>I</sub> flag to generate an MMC interrupt.					
4	EOF <sub>M</sub>	<b>End Of Frame Interrupt Mask Bit</b> Set to prevent EOF <sub>I</sub> flag from generating an MMC interrupt. Clear to allow EOF <sub>I</sub> flag to generate an MMC interrupt.					
3	F2F <sub>M</sub>	<b>FIFO 2 Full Interrupt Mask Bit</b> Set to prevent F2F <sub>I</sub> flag from generating an MMC interrupt. Clear to allow F2F <sub>I</sub> flag to generate an MMC interrupt.					
2	F1F <sub>M</sub>	<b>FIFO 1 Full Interrupt Mask Bit</b> Set to prevent F1F <sub>I</sub> flag from generating an MMC interrupt. Clear to allow F1F <sub>I</sub> flag to generate an MMC interrupt.					
1	F2E <sub>M</sub>	<b>FIFO 2 Empty Interrupt Mask Bit</b> Set to prevent F2E <sub>I</sub> flag from generating an MMC interrupt. Clear to allow F2E <sub>I</sub> flag to generate an MMC interrupt.					
0	F1E <sub>M</sub>	<b>FIFO 1 Empty Interrupt Mask Bit</b> Set to prevent F1E <sub>I</sub> flag from generating an MMC interrupt. Clear to allow F1E <sub>I</sub> flag to generate an MMC interrupt.					

Reset Value = 1111 1111b

**Table 118. MMCMD Register**

MMCMD (S:DDh) – MMC Command Register

7	6	5	4	3	2	1	0
MC7	MC6	MC5	MC4	MC3	MC2	MC1	MC0
Bit Number	Bit Mnemonic	Description					
7 - 0	MC7:0	<b>MMC Command Receive Byte</b> Output (read) register of the response FIFO. <b>MMC Command Transmit Byte</b> Input (write) register of the command FIFO.					

Reset Value = 1111 1111b

**Table 119.** MMDAT Register

MMDAT (S:DCh) – MMC Data Register

7	6	5	4	3	2	1	0
MD7	MD6	MD5	MD4	MD3	MD2	MD1	MD0
Bit Number	Bit Mnemonic	Description					
7 - 0	MD7:0	<b>MMC Data Byte</b> Input (write) or output (read) register of the data FIFO.					

Reset Value = 1111 1111b

**Table 120.** MMCLK Register

MMCLK (S:EDh) – MMC Clock Divider Register

7	6	5	4	3	2	1	0
MMCD7	MMCD6	MMCD5	MMCD4	MMCD3	MMCD2	MMCD1	MMCD0
Bit Number	Bit Mnemonic	Description					
7 - 0	MMCD7:0	<b>MMC Clock Divider</b> 8-bit divider for MMC clock generation.					

Reset Value = 0000 0000b

## IDE/ATAPI Interface

The AT8xC51SND1C provides an IDE/ATAPI interface allowing connection of devices such as CD-ROM reader, CompactFlash cards, Hard Disk Drive, etc. It consists of a 16-bit data transfer (read or write) between the AT8xC51SND1C and the IDE device.

## Description

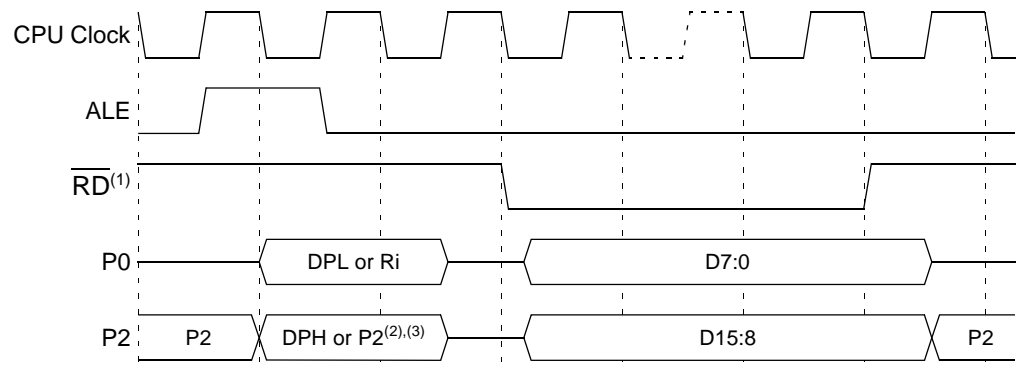
The IDE interface mode is enabled by setting the EXT16 bit in AUXR (see Figure 29, page 29). As soon as this bit is set, all MOVX instructions read or write are done in a 16-bit mode compare to the standard 8-bit mode. P0 carries the low order multiplexed address and data bus (A7:0, D7:0) while P2 carries the high order multiplexed address and data bus (A15:8, D15:8). When writing data in IDE mode, the ACC contains D7:0 data (as in 8-bit mode) while DAT16H register (see Table 122) contains D15:8 data. When reading data in IDE mode, D7:0 data is returned in ACC while D15:8 data is returned in DAT16H.

Figure 90 shows the IDE read bus cycle while Figure 91 shows the IDE write bus cycle. For simplicity, these figures depict the bus cycle waveforms in idealized form and do not provide precise timing information. For IDE bus cycle timing parameters refer to the Section "AC Characteristics".

IDE cycle takes 6 CPU clock periods which is equivalent to 12 oscillator clock periods in standard mode or 6 oscillator clock periods in X2 mode. For further information on X2 mode, refer to the Section "X2 Feature", page 12.

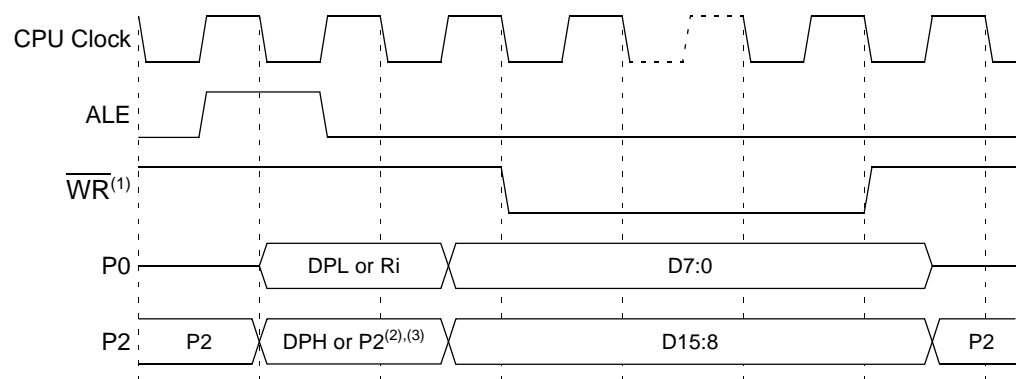
Slow IDE devices can be accessed by stretching the read and write cycles. This is done using the M0 bit in AUXR. Setting this bit changes the width of the RD and WR signals from 3 to 15 CPU clock periods.

**Figure 90.** IDE Read Waveforms



- Notes:
1. RD signal may be stretched using M0 bit in AUXR register.
  2. When executing MOVX @Ri instruction, P2 outputs SFR content.
  3. When executing MOVX @DPTR instruction, if DPHDIS is set (Page Access Mode), P2 outputs SFR content instead of DPH.

**Figure 91. IDE Write Waveforms**

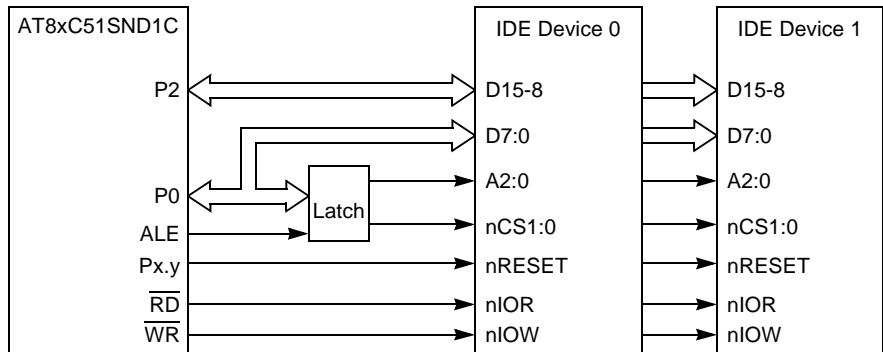


- Notes:
1.  $\overline{WR}$  signal may be stretched using M0 bit in AUXR register.
  2. When executing MOVX @Ri instruction, P2 outputs SFR content.
  3. When executing MOVX @DPTR instruction, if DPHDIS is set (Page Access Mode), P2 outputs SFR content instead of DPH.

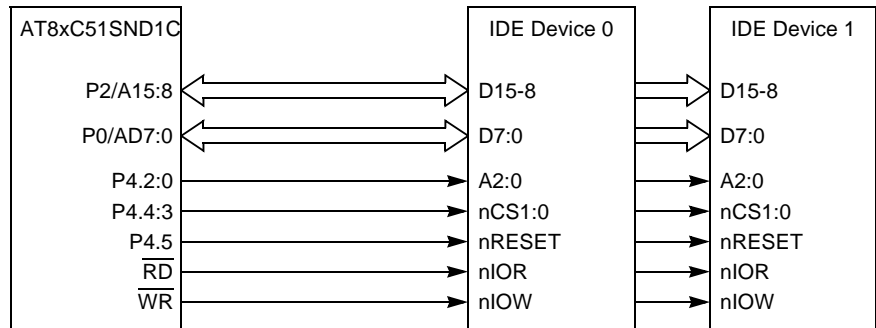
## IDE Device Connection

Figure 92 and Figure 93 show 2 examples on how to interface up to 2 IDE devices to the AT8xC51SND1C. In both examples P0 carries IDE low order data bits D7:0, P2 carries IDE high order data bits D15:8, while  $\overline{RD}$  and  $\overline{WR}$  signals are respectively connected to the IDE nIOR and nIOW signals. Other IDE control signals are generated by the external address latch outputs in the first example while they are generated by some port I/Os in the second one. Using an external latch will achieve higher transfer rate.

**Figure 92. IDE Device Connection Example 1**



**Figure 93. IDE Device Connection Example 2**



**Table 121.** External Data Memory Interface Signals

Signal Name	Type	Description	Alternate Function
A15:8	I/O	<b>Address Lines</b> Upper address lines for the external bus. Multiplexed higher address and data lines for the IDE interface.	P2.7:0
AD7:0	I/O	<b>Address/Data Lines</b> Multiplexed lower address and data lines for the IDE interface.	P0.7:0
ALE	O	<b>Address Latch Enable</b> ALE signals indicates that valid address information is available on lines AD7:0.	-
$\overline{\text{RD}}$	O	<b>Read</b> Read signal output to external data memory.	P3.7
$\overline{\text{WR}}$	O	<b>Write</b> Write signal output to external memory.	P3.6

## Registers

**Table 122.** DAT16H Register

DAT16H (S:F9h) – Data 16 High Order Byte

7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8
Bit Number	Bit Mnemonic	Description					
7 - 0	D15:8	<b>Data 16 High Order Byte</b> When EXT16 bit is set, DAT16H is set by software with the high order data Byte prior any MOVX write instruction. When EXT16 bit is set, DAT16H contains the high order data Byte after any MOVX read instruction.					

Reset Value =XXXX XXXXb

## Serial I/O Port

The serial I/O port in the AT8xC51SND1C provides both synchronous and asynchronous communication modes. It operates as a Synchronous Receiver and Transmitter in one single mode (Mode 0) and operates as an Universal Asynchronous Receiver and Transmitter (UART) in three full-duplex modes (Modes 1, 2 and 3). Asynchronous modes support framing error detection and multiprocessor communication with automatic address recognition.

## Mode Selection

SM0 and SM1 bits in SCON register (see Figure 125) are used to select a mode among the single synchronous and the three asynchronous modes according to Table 123.

**Table 123.** Serial I/O Port Mode Selection

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Synchronous Shift Register	Fixed/Variable
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	Fixed
1	1	3	9-bit UART	Variable

## Baud Rate Generator

Depending on the mode and the source selection, the baud rate can be generated from either the Timer 1 or the Internal Baud Rate Generator. The Timer 1 can be used in Modes 1 and 3 while the Internal Baud Rate Generator can be used in Modes 0, 1 and 3.

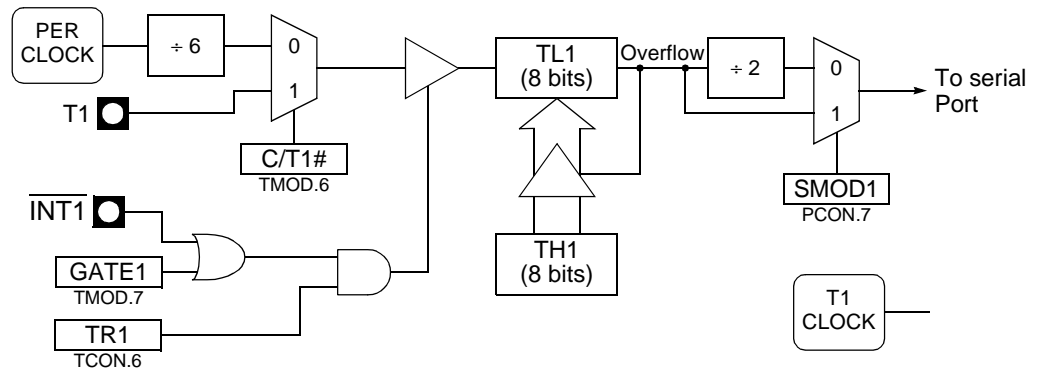
The addition of the Internal Baud Rate Generator allows freeing of the Timer 1 for other purposes in the application. It is highly recommended to use the Internal Baud Rate Generator as it allows higher and more accurate baud rates than Timer 1.

Baud rate formulas depend on the modes selected and are given in the following mode sections.

## Timer 1

When using Timer 1, the Baud Rate is derived from the overflow of the timer. As shown in Figure 94 Timer 1 is used in its 8-bit auto-reload mode (detailed in Section "Mode 2 (8-bit Timer with Auto-Reload)", page 53). SMOD1 bit in PCON register allows doubling of the generated baud rate.

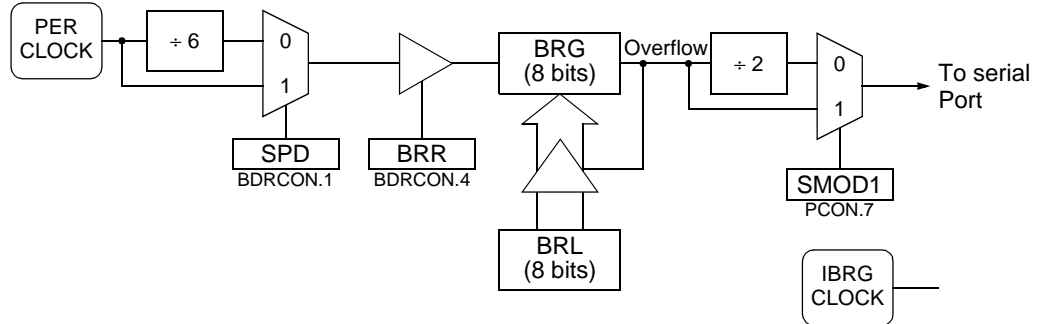
**Figure 94.** Timer 1 Baud Rate Generator Block Diagram



## Internal Baud Rate Generator

When using the Internal Baud Rate Generator, the Baud Rate is derived from the overflow of the timer. As shown in Figure 95 the Internal Baud Rate Generator is an 8-bit auto-reload timer fed by the peripheral clock or by the peripheral clock divided by 6 depending on the SPD bit in BDRCON register (see Table 129). The Internal Baud Rate Generator is enabled by setting BBR bit in BDRCON register. SMOD1 bit in PCON register allows doubling of the generated baud rate.

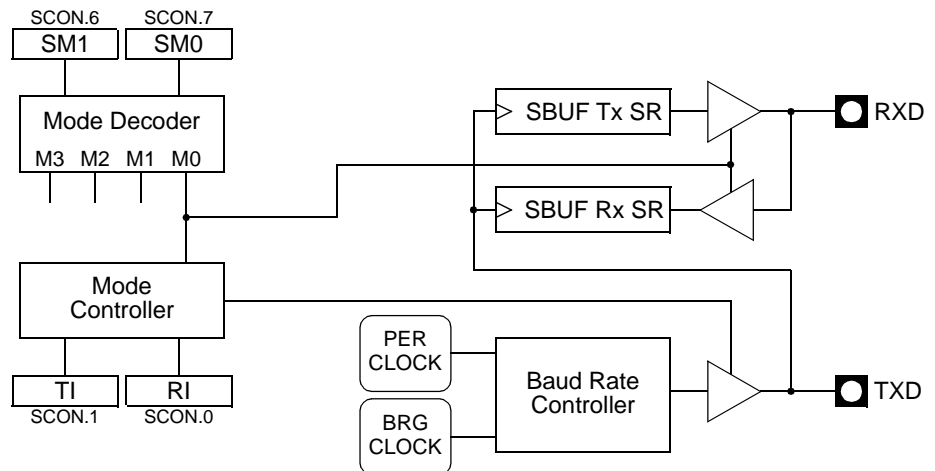
**Figure 95.** Internal Baud Rate Generator Block Diagram



## Synchronous Mode (Mode 0)

Mode 0 is a half-duplex, synchronous mode, which is commonly used to expand the I/O capabilities of a device with shift registers. The transmit data (TXD) pin outputs a set of eight clock pulses while the receive data (RXD) pin transmits or receives a Byte of data. The 8-bit data are transmitted and received least-significant bit (LSB) first. Shifts occur at a fixed Baud Rate (see Section "Baud Rate Selection (Mode 0)", page 136). Figure 96 shows the serial port block diagram in Mode 0.

**Figure 96.** Serial I/O Port Block Diagram (Mode 0)

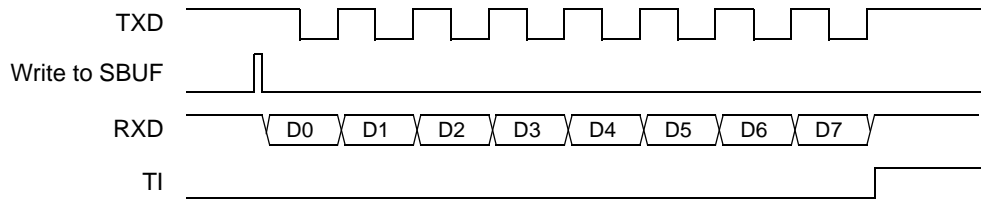


## Transmission (Mode 0)

To start a transmission mode 0, write to SCON register clearing bits SM0, SM1.

As shown in Figure 97, writing the Byte to transmit to SBUF register starts the transmission. Hardware shifts the LSB (D0) onto the RXD pin during the first clock cycle composed of a high level then low level signal on TXD. During the eighth clock cycle the MSB (D7) is on the RXD pin. Then, hardware drives the RXD pin high and asserts TI to indicate the end of the transmission.

**Figure 97.** Transmission Waveforms (Mode 0)

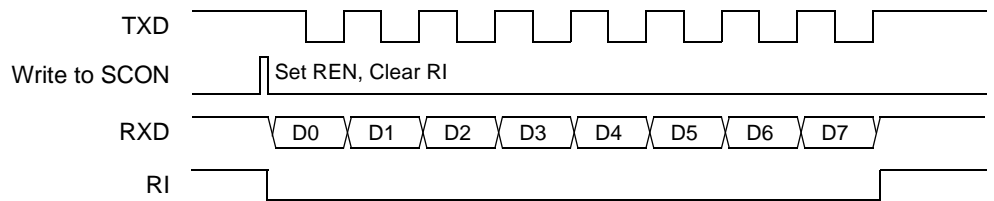


**Reception (Mode 0)**

To start a reception in mode 0, write to SCON register clearing SM0, SM1 and RI bits and setting the REN bit.

As shown in Figure 98, Clock is pulsed and the LSB (D0) is sampled on the RXD pin. The D0 bit is then shifted into the shift register. After eight samplings, the MSB (D7) is shifted into the shift register, and hardware asserts RI bit to indicate a completed reception. Software can then read the received Byte from SBUF register.

**Figure 98.** Reception Waveforms (Mode 0)

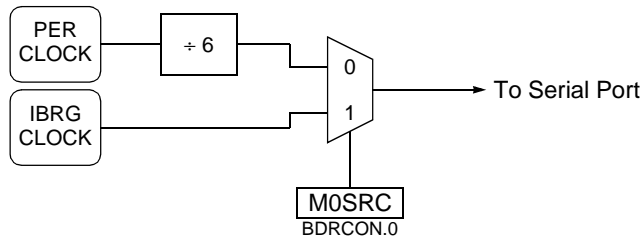


**Baud Rate Selection (Mode 0)**

In mode 0, the baud rate can be either, fixed or variable.

As shown in Figure 99, the selection is done using M0SRC bit in BDRCON register. Figure 100 gives the baud rate calculation formulas for each baud rate source.

**Figure 99.** Baud Rate Source Selection (mode 0)



**Figure 100.** Baud Rate Formulas (Mode 0)

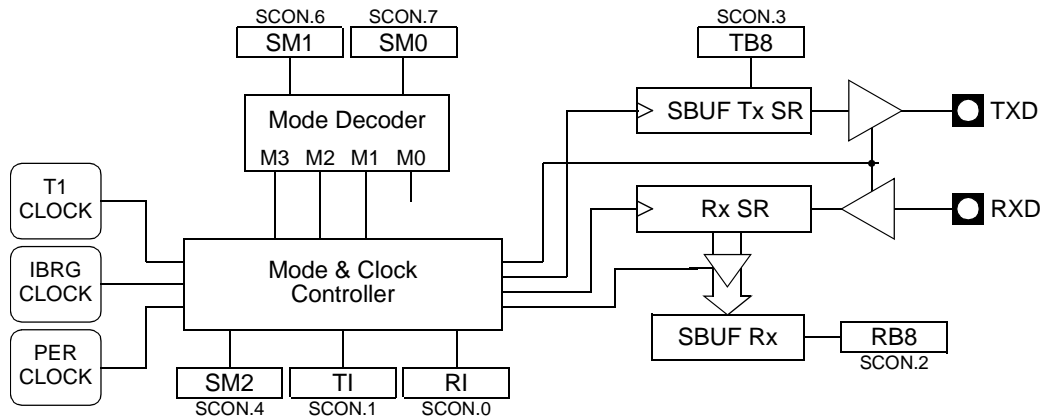
$\text{Baud\_Rate} = \frac{F_{\text{PER}}}{6}$ <p><b>a. Fixed Formula</b></p>	$\text{Baud\_Rate} = \frac{2^{\text{SMOD}1} \cdot F_{\text{PER}}}{6^{(1-\text{SPD})} \cdot 32 \cdot (256 - \text{BRL})}$ $\text{BRL} = 256 - \frac{2^{\text{SMOD}1} \cdot F_{\text{PER}}}{6^{(1-\text{SPD})} \cdot 32 \cdot \text{Baud\_Rate}}$ <p><b>b. Variable Formula</b></p>
---	---



## Asynchronous Modes (Modes 1, 2 and 3)

The Serial Port has one 8-bit and 2 9-bit asynchronous modes of operation. Figure 101 shows the Serial Port block diagram in such asynchronous modes.

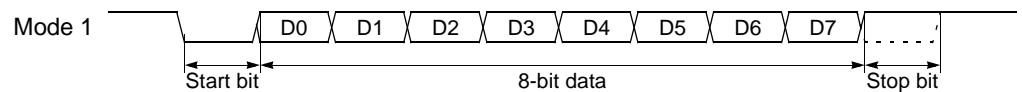
**Figure 101.** Serial I/O Port Block Diagram (Modes 1, 2 and 3)



### Mode 1

Mode 1 is a full-duplex, asynchronous mode. The data frame (see Figure 102) consists of 10 bits: one start, eight data bits and one stop bit. Serial data is transmitted on the TXD pin and received on the RXD pin. When a data is received, the stop bit is read in the RB8 bit in SCON register.

**Figure 102.** Data Frame Format (Mode 1)



### Modes 2 and 3

Modes 2 and 3 are full-duplex, asynchronous modes. The data frame (see Figure 103) consists of 11 bits: one start bit, eight data bits (transmitted and received LSB first), one programmable ninth data bit and one stop bit. Serial data is transmitted on the TXD pin and received on the RXD pin. On receive, the ninth bit is read from RB8 bit in SCON register. On transmit, the ninth data bit is written to TB8 bit in SCON register. Alternatively, you can use the ninth bit can be used as a command/data flag.

**Figure 103.** Data Frame Format (Modes 2 and 3)



### Transmission (Modes 1, 2 and 3)

To initiate a transmission, write to SCON register, set the SM0 and SM1 bits according to Table 123, and set the ninth bit by writing to TB8 bit. Then, writing the Byte to be transmitted to SBUF register starts the transmission.

### Reception (Modes 1, 2 and 3)

To prepare for reception, write to SCON register, set the SM0 and SM1 bits according to Table 123, and set the REN bit. The actual reception is then initiated by a detected high-to-low transition on the RXD pin.

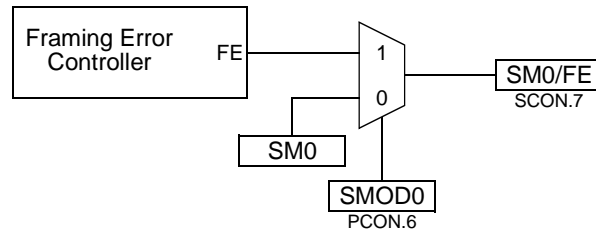
### Framing Error Detection (Modes 1, 2 and 3)

Framing error detection is provided for the three asynchronous modes. To enable the framing bit error detection feature, set SMOD0 bit in PCON register as shown in Figure 104.

When this feature is enabled, the receiver checks each incoming data frame for a valid stop bit. An invalid stop bit may result from noise on the serial lines or from simultaneous transmission by 2 devices. If a valid stop bit is not found, the software sets FE bit in SCON register.

Software may examine FE bit after each reception to check for data errors. Once set, only software or a chip reset clear FE bit. Subsequently received frames with valid stop bits cannot clear FE bit. When the framing error detection feature is enabled, RI rises on stop bit instead of the last data bit as detailed in Figure 110.

**Figure 104.** Framing Error Block Diagram

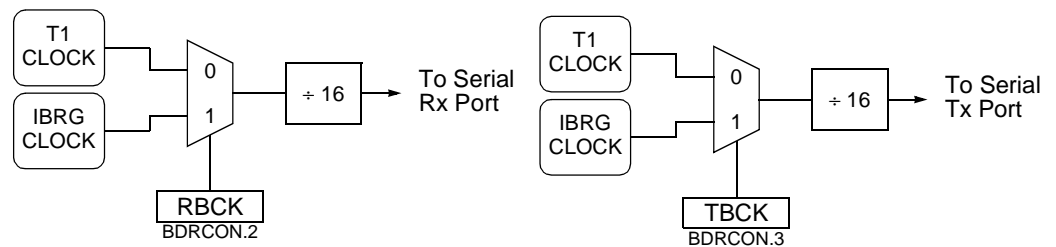


### Baud Rate Selection (Modes 1 and 3)

In modes 1 and 3, the Baud Rate is derived either from the Timer 1 or the Internal Baud Rate Generator and allows different baud rate in reception and transmission. As shown in Figure 105 the selection is done using RBCK and TBCK bits in BDRCON register.

Figure 106 gives the baud rate calculation formulas for each baud rate source while Table 124 details Internal Baud Rate Generator configuration for different peripheral clock frequencies and giving baud rates closer to the standard baud rates.

**Figure 105.** Baud Rate Source Selection (Modes 1 and 3)



**Figure 106.** Baud Rate Formulas (Modes 1 and 3)

$$\text{Baud\_Rate} = \frac{2^{\text{SMOD}1} \cdot F_{\text{PER}}}{6^{(1-\text{SPD})} \cdot 32 \cdot (256 - \text{BRL})}$$

$$\text{BRL} = 256 - \frac{2^{\text{SMOD}1} \cdot F_{\text{PER}}}{6^{(1-\text{SPD})} \cdot 32 \cdot \text{Baud\_Rate}}$$

**a. IBRG Formula**

$$\text{Baud\_Rate} = \frac{2^{\text{SMOD}1} \cdot F_{\text{PER}}}{6 \cdot 32 \cdot (256 - \text{TH1})}$$

$$\text{TH1} = 256 - \frac{2^{\text{SMOD}1} \cdot F_{\text{PER}}}{192 \cdot \text{Baud\_Rate}}$$

**b. T1 Formula**

**Table 124.** Internal Baud Rate Generator Value

Baud Rate	$F_{PER} = 6 \text{ MHz}^{(1)}$				$F_{PER} = 8 \text{ MHz}^{(1)}$				$F_{PER} = 10 \text{ MHz}^{(1)}$			
	SPD	SMOD1	BRL	Error %	SPD	SMOD1	BRL	Error %	SPD	SMOD1	BRL	Error %
115200	-	-	-	-	-	-	-	-	-	-	-	-
57600	-	-	-	-	1	1	247	3.55	1	1	245	1.36
38400	1	1	246	2.34	1	1	243	0.16	1	1	240	1.73
19200	1	1	236	2.34	1	1	230	0.16	1	1	223	1.36
9600	1	1	217	0.16	1	1	204	0.16	1	1	191	0.16
4800	1	1	178	0.16	1	1	152	0.16	1	1	126	0.16

Baud Rate	$F_{PER} = 12 \text{ MHz}^{(2)}$				$F_{PER} = 16 \text{ MHz}^{(2)}$				$F_{PER} = 20 \text{ MHz}^{(2)}$			
	SPD	SMOD1	BRL	Error %	SPD	SMOD1	BRL	Error %	SPD	SMOD1	BRL	Error %
115200	-	-	-	-	1	1	247	3.55	1	1	245	1.36
57600	1	1	243	0.16	1	1	239	2.12	1	1	234	1.36
38400	1	1	236	2.34	1	1	230	0.16	1	1	223	1.36
19200	1	1	217	0.16	1	1	204	0.16	1	1	191	0.16
9600	1	1	178	0.16	1	1	152	0.16	1	1	126	0.16
4800	1	1	100	0.16	1	1	48	0.16	1	0	126	0.16

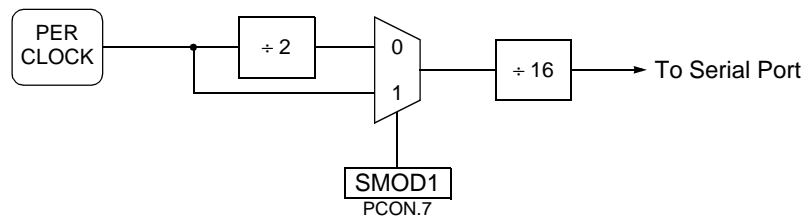
Notes: 1. These frequencies are achieved in X1 mode,  $F_{PER} = F_{OSC} \div 2$ .  
 2. These frequencies are achieved in X2 mode,  $F_{PER} = F_{OSC}$ .

**Baud Rate Selection (Mode 2)** In mode 2, the baud rate can only be programmed to 2 fixed values: 1/16 or 1/32 of the peripheral clock frequency.

As shown in Figure 107 the selection is done using SMOD1 bit in PCON register.

Figure 108 gives the baud rate calculation formula depending on the selection.

**Figure 107.** Baud Rate Generator Selection (Mode 2)



**Figure 108.** Baud Rate Formula (Mode 2)

$$\text{Baud\_Rate} = \frac{2^{\text{SMOD}1} \cdot F_{\text{PER}}}{32}$$

## Multiprocessor Communication (Modes 2 and 3)

Modes 2 and 3 provide a ninth-bit mode to facilitate multiprocessor communication. To enable this feature, set SM2 bit in SCON register. When the multiprocessor communication feature is enabled, the serial Port can differentiate between data frames (ninth bit clear) and address frames (ninth bit set). This allows the AT8xC51SND1C to function as a slave processor in an environment where multiple slave processors share a single serial line.

When the multiprocessor communication feature is enabled, the receiver ignores frames with the ninth bit clear. The receiver examines frames with the ninth bit set for an address match. If the received address matches the slaves address, the receiver hardware sets RB8 and RI bits in SCON register, generating an interrupt.

The addressed slave's software then clears SM2 bit in SCON register and prepares to receive the data Bytes. The other slaves are unaffected by these data Bytes because they are waiting to respond to their own addresses.

## Automatic Address Recognition

The automatic address recognition feature is enabled when the multiprocessor communication feature is enabled (SM2 bit in SCON register is set).

Implemented in hardware, automatic address recognition enhances the multiprocessor communication feature by allowing the Serial Port to examine the address of each incoming command frame. Only when the Serial Port recognizes its own address, the receiver sets RI bit in SCON register to generate an interrupt. This ensures that the CPU is not interrupted by command frames addressed to other devices.

If desired, the automatic address recognition feature in mode 1 may be enabled. In this configuration, the stop bit takes the place of the ninth data bit. Bit RI is set only when the received command frame address matches the device's address and is terminated by a valid stop bit.

To support automatic address recognition, a device is identified by a given address and a broadcast address.

Note: The multiprocessor communication and automatic address recognition features cannot be enabled in mode 0 (i.e., setting SM2 bit in SCON register in mode 0 has no effect).

## Given Address

Each device has an individual address that is specified in SADDR register; the SADEN register is a mask Byte that contains don't care bits (defined by zeros) to form the device's given address. The don't care bits provide the flexibility to address one or more slaves at a time. The following example illustrates how a given address is formed. To address a device by its individual address, the SADEN mask Byte must be 1111 1111b.

For example:

```
SADDR = 0101 0110b
SADEN = 1111 1100b
Given  = 0101 01XXb
```

The following is an example of how to use given addresses to address different slaves:

```
Slave A:SADDR = 1111 0001b
SADEN = 1111 1010b
Given = 1111 0X0Xb
Slave B:SADDR = 1111 0011b
SADEN = 1111 1001b
Given = 1111 0XX1b
Slave C:SADDR = 1111 0010b
SADEN = 1111 1101b
Given = 1111 00X1b
```

The SADEN Byte is selected so that each slave may be addressed separately. For slave A, bit 0 (the LSB) is a don't-care bit; for slaves B and C, bit 0 is a 1. To communicate with slave A only, the master must send an address where bit 0 is clear (e.g. 1111 0000B).

For slave A, bit 1 is a 0; for slaves B and C, bit 1 is a don't care bit. To communicate with slaves A and B, but not slave C, the master must send an address with bits 0 and 1 both set (e.g. 1111 0011B).

To communicate with slaves A, B and C, the master must send an address with bit 0 set, bit 1 clear, and bit 2 clear (e.g. 1111 0001B).

### Broadcast Address

A broadcast address is formed from the logical OR of the SADDR and SADEN registers with zeros defined as don't-care bits, e.g.:

```
SADDR = 0101 0110b
SADEN = 1111 1100b
(SADDR | SADEN)=1111 111Xb
```

The use of don't-care bits provides flexibility in defining the broadcast address, however in most applications, a broadcast address is FFh.

The following is an example of using broadcast addresses:

```
Slave A:SADDR = 1111 0001b
SADEN = 1111 1010b
Given = 1111 1X11b,

Slave B:SADDR = 1111 0011b
SADEN = 1111 1001b
Given = 1111 1X11b,

Slave C:SADDR = 1111 0010b
SADEN = 1111 1101b
Given = 1111 1111b,
```

For slaves A and B, bit 2 is a don't care bit; for slave C, bit 2 is set. To communicate with all of the slaves, the master must send the address FFh.

To communicate with slaves A and B, but not slave C, the master must send the address FBh.

### Reset Address

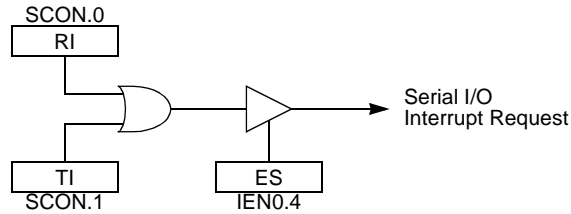
On reset, the SADDR and SADEN registers are initialized to 00h, i.e. the given and broadcast addresses are XXXX XXXXb (all don't care bits). This ensures that the Serial Port is backwards compatible with the 80C51 microcontrollers that do not support automatic address recognition.

## Interrupt

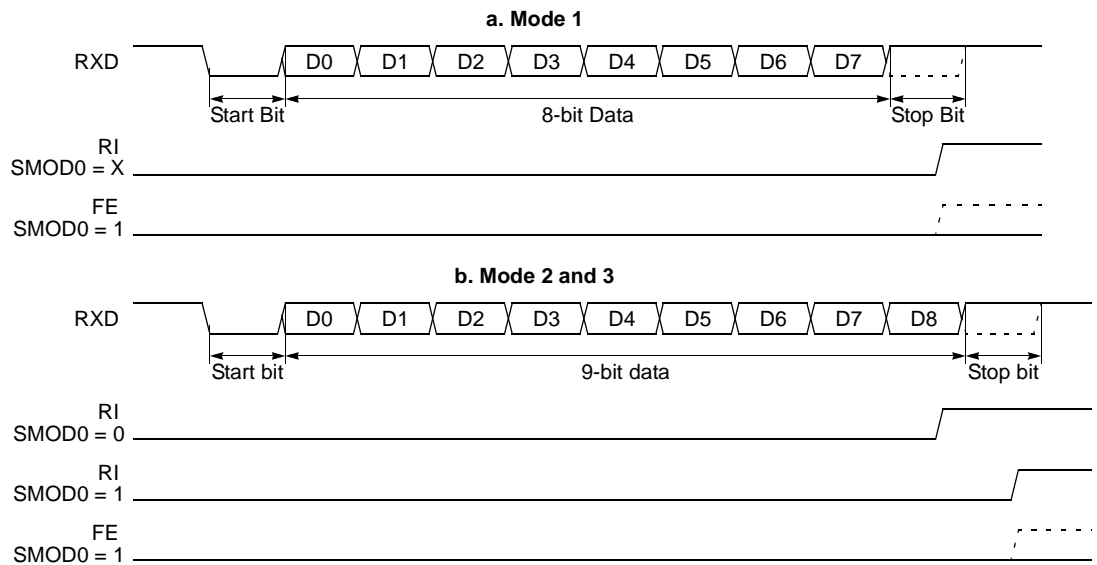
The Serial I/O Port handles 2 interrupt sources that are the “end of reception” (RI in SCON) and “end of transmission” (TI in SCON) flags. As shown in Figure 109 these flags are combined together to appear as a single interrupt source for the C51 core. Flags must be cleared by software when executing the serial interrupt service routine. The serial interrupt is enabled by setting ES bit in IEN0 register. This assumes interrupts are globally enabled by setting EA bit in IEN0 register.

Depending on the selected mode and whether the framing error detection is enabled or disabled, RI flag is set during the stop bit or during the ninth bit as detailed in Figure 110.

**Figure 109.** Serial I/O Interrupt System



**Figure 110.** Interrupt Waveforms



Registers

**Table 125.** SCON Register  
 SCON (S:98h) – Serial Control Register

7	6	5	4	3	2	1	0
FE/SM0	OVR/SM1	SM2	REN	TB8	RB8	TI	RI
Bit Number	Bit Mnemonic	Description					
7	FE	<b>Framing Error Bit</b> To select this function, set SMOD0 bit in PCON register. Set by hardware to indicate an invalid stop bit. Must be cleared by software.					
	SM0	<b>Serial Port Mode Bit 0</b> Refer to Table 123 for mode selection.					
6	SM1	<b>Serial Port Mode Bit 1</b> Refer to Table 123 for mode selection.					
5	SM2	<b>Serial Port Mode Bit 2</b> Set to enable the multiprocessor communication and automatic address recognition features. Clear to disable the multiprocessor communication and automatic address recognition features.					
4	REN	<b>Receiver Enable Bit</b> Set to enable reception. Clear to disable reception.					
3	TB8	<b>Transmit Bit 8</b> Modes 0 and 1: Not used. Modes 2 and 3: Software writes the ninth data bit to be transmitted to TB8.					
2	RB8	<b>Receiver Bit 8</b> Mode 0: Not used. Mode 1 (SM2 cleared): Set or cleared by hardware to reflect the stop bit received. Modes 2 and 3 (SM2 set): Set or cleared by hardware to reflect the ninth bit received.					
1	TI	<b>Transmit Interrupt Flag</b> Set by the transmitter after the last data bit is transmitted. Must be cleared by software.					
0	RI	<b>Receive Interrupt Flag</b> Set by the receiver after the stop bit of a frame has been received. Must be cleared by software.					

Reset Value = 0000 0000b

**Table 126.** SBUF Register

SBUF (S:99h) – Serial Buffer Register

7	6	5	4	3	2	1	0
SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
Bit Number	Bit Mnemonic	Description					
7 - 0	SD7:0	<b>Serial Data Byte</b> Read the last data received by the serial I/O Port. Write the data to be transmitted by the serial I/O Port.					

Reset value = XXXX XXXXb

**Table 127.** SADDR Register

SADDR (S:A9h) – Slave Individual Address Register

7	6	5	4	3	2	1	0
SAD7	SAD6	SAD5	SAD4	SAD3	SAD2	SAD1	SAD0
Bit Number	Bit Mnemonic	Description					
7 - 0	SAD7:0	<b>Slave Individual Address</b>					

Reset Value = 0000 0000b

**Table 128.** SADEN Register

SADEN (S:B9h) – Slave Individual Address Mask Byte Register

7	6	5	4	3	2	1	0
SAE7	SAE6	SAE5	SAE4	SAE3	SAE2	SAE1	SAE0
Bit Number	Bit Mnemonic	Description					
7 - 0	SAE7:0	<b>Slave Address Mask Byte</b>					

Reset Value = 0000 0000b



**Table 129.** BDRCON Register

BDRCON (S:92h) – Baud Rate Generator Control Register

7	6	5	4	3	2	1	0
-	-	-	BRR	TBCK	RBCK	SPD	M0SRC

Bit Number	Bit Mnemonic	Description
7 - 5	-	<b>Reserved</b> The value read from these bits are indeterminate. Do not set these bits.
4	BRR	<b>Baud Rate Run Bit</b> Set to enable the baud rate generator. Clear to disable the baud rate generator.
3	TBCK	<b>Transmission Baud Rate Selection Bit</b> Set to select the baud rate generator as transmission baud rate generator. Clear to select the Timer 1 as transmission baud rate generator.
2	RBCK	<b>Reception Baud Rate Selection Bit</b> Set to select the baud rate generator as reception baud rate generator. Clear to select the Timer 1 as reception baud rate generator.
1	SPD	<b>Baud Rate Speed Bit</b> Set to select high speed baud rate generation. Clear to select low speed baud rate generation.
0	M0SRC	<b>Mode 0 Baud Rate Source Bit</b> Set to select the variable baud rate generator in Mode 0. Clear to select fixed baud rate in Mode 0.

Reset Value = XXX0 0000b

**Table 130.** BRL Register

BRL (S:91h) – Baud Rate Generator Reload Register

7	6	5	4	3	2	1	0
BRL7	BRL6	BRL5	BRL4	BRL3	BRL2	BRL1	BRL0

Bit Number	Bit Mnemonic	Description
7 - 0	BRL7:0	<b>Baud Rate Reload Value</b>

Reset Value = 0000 0000b

## Synchronous Peripheral Interface

The AT8xC51SND1C implements a Synchronous Peripheral Interface with master and slave modes capability.

Figure 111 shows an SPI bus configuration using the AT8xC51SND1C as master connected to slave peripherals while Figure 112 shows an SPI bus configuration using the AT8xC51SND1C as slave of an other master.

The bus is made of three wires connecting all the devices together:

- Master Output Slave Input (MOSI): it is used to transfer data in series from the master to a slave. It is driven by the master.
- Master Input Slave Output (MISO): it is used to transfer data in series from a slave to the master. It is driven by the selected slave.
- Serial Clock (SCK): it is used to synchronize the data transmission both in and out the devices through their MOSI and MISO lines. It is driven by the master for eight clock cycles which allows to exchange one Byte on the serial lines.

Each slave peripheral is selected by one Slave Select pin ( $\overline{SS}$ ). If there is only one slave, it may be continuously selected with  $\overline{SS}$  tied to a low level. Otherwise, the AT8xC51SND1C may select each device by software through port pins (Pn.x). Special care should be taken not to select 2 slaves at the same time to avoid bus conflicts.

Figure 111. Typical Master SPI Bus Configuration

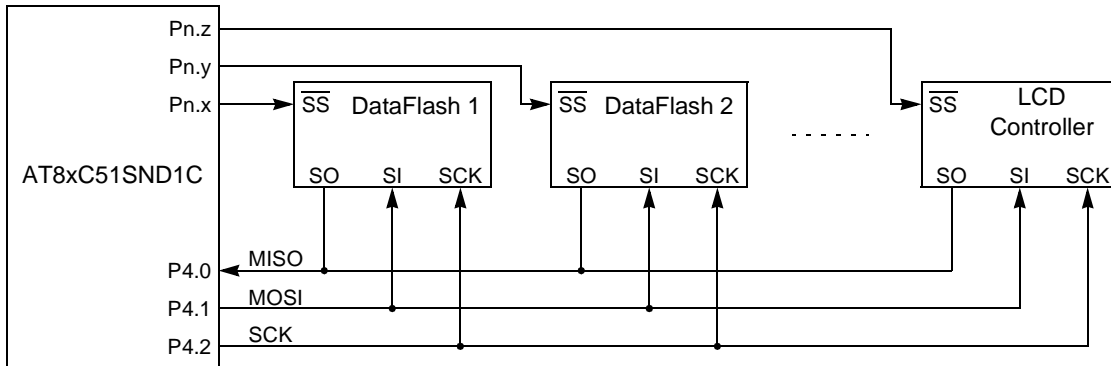
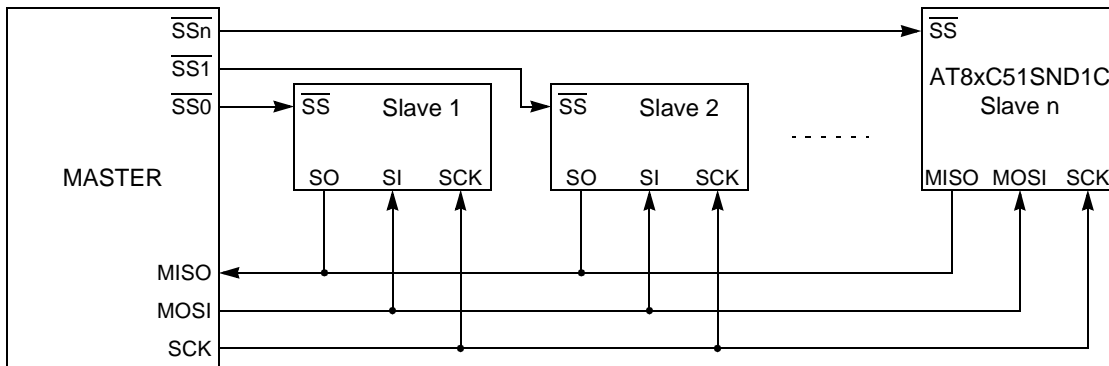


Figure 112. Typical Slave SPI Bus Configuration



## Description

The SPI controller interfaces with the C51 core through three special function registers: SPCON, the SPI control register (see Table 132); SPSTA, the SPI status register (see Table 133); and SPDAT, the SPI data register (see Table 134).

## Master Mode

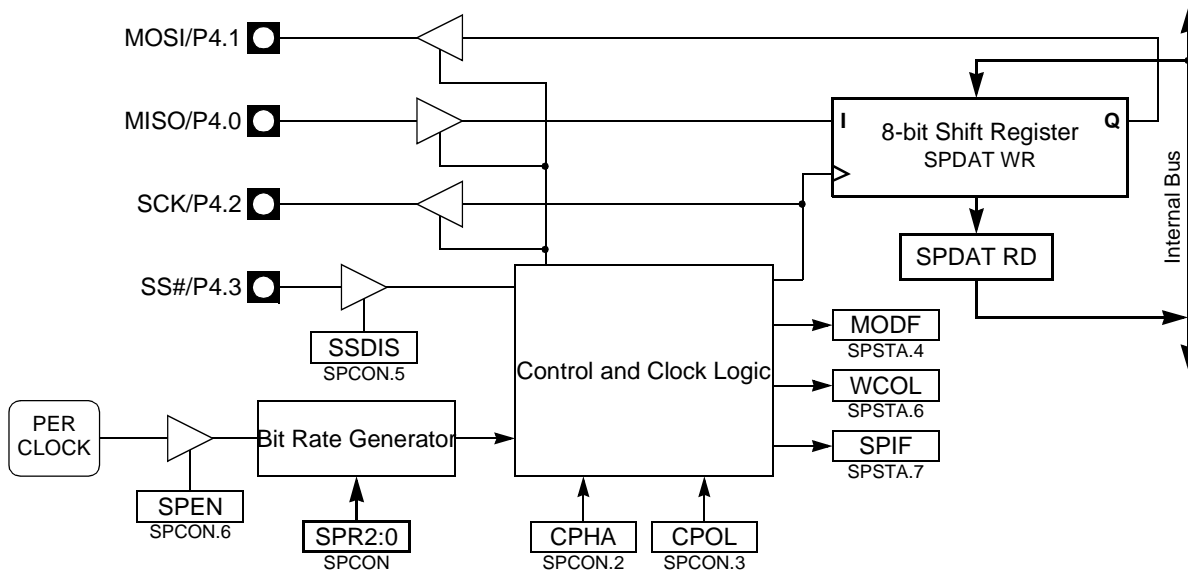
The SPI operates in master mode when the MSTR bit in SPCON is set.

Figure 113 shows the SPI block diagram in master mode. Only a master SPI module can initiate transmissions. Software begins the transmission by writing to SPDAT. Writing to SPDAT writes to the shift register while reading SPDAT reads an intermediate register updated at the end of each transfer.

The Byte begins shifting out on the MOSI pin under the control of the bit rate generator. This generator also controls the shift register of the slave peripheral through the SCK output pin. As the Byte shifts out, another Byte shifts in from the slave peripheral on the MISO pin. The Byte is transmitted most significant bit (MSB) first. The end of transfer is signaled by SPIF being set.

When the AT8xC51SND1C is the only master on the bus, it can be useful not to use SS# pin and get it back to I/O functionality. This is achieved by setting SSDIS bit in SPCON.

**Figure 113.** SPI Master Mode Block Diagram



Note: MSTR bit in SPCON is set to select master mode.

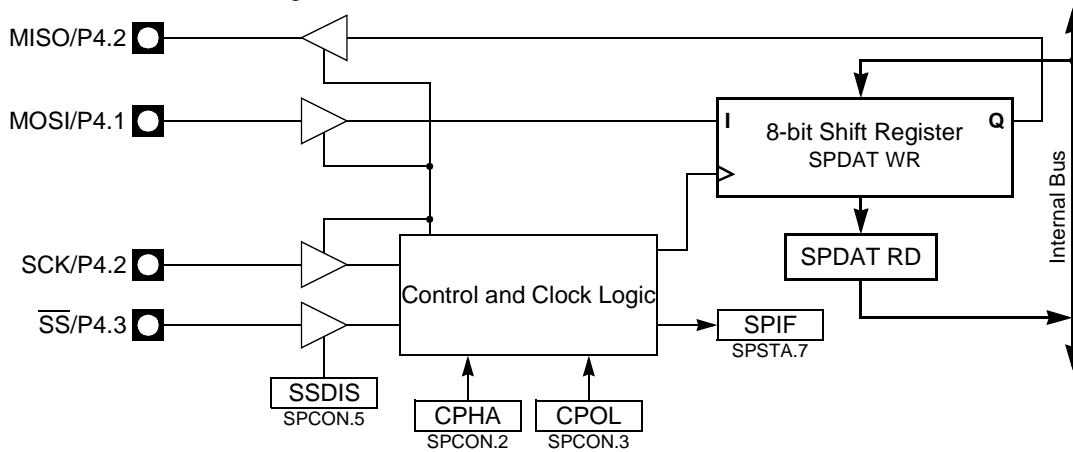
## Slave Mode

The SPI operates in slave mode when the MSTR bit in SPCON is cleared and data has been loaded in SPDAT.

Figure 114 shows the SPI block diagram in slave mode. In slave mode, before a data transmission occurs, the  $\overline{SS}$  pin of the slave SPI must be asserted to low level.  $\overline{SS}$  must remain low until the transmission of the Byte is complete. In the slave SPI module, data enters the shift register through the MOSI pin under the control of the serial clock provided by the master SPI module on the SCK input pin. When the master starts a transmission, the data in the shift register begins shifting out on the MISO pin. The end of transfer is signaled by SPIF being set.

When the AT8xC51SND1C is the only slave on the bus, it can be useful not to use SS# pin and get it back to I/O functionality. This is achieved by setting SSDIS bit in SPCON. This bit has no effect when CPHA is cleared (see Section "SS Management", page 150).

**Figure 114.** SPI Slave Mode Block Diagram



Note: 1. MSTR bit in SPCON is cleared to select slave mode.

## Bit Rate

The bit rate can be selected from seven predefined bit rates using the SPR2, SPR1 and SPR0 control bits in SPCON according to Table 131. These bit rates are derived from the peripheral clock ( $F_{PER}$ ) issued from the Clock Controller block as detailed in Section "Oscillator", page 12.

**Table 131. Serial Bit Rates**

SPR2	SPR1	SPR0	Bit Rate (kHz) Vs F <sub>PER</sub>						F <sub>PER</sub> Divider
			6 MHz <sup>(1)</sup>	8 MHz <sup>(1)</sup>	10 MHz <sup>(1)</sup>	12 MHz <sup>(2)</sup>	16 MHz <sup>(2)</sup>	20 MHz <sup>(2)</sup>	
0	0	0	3000	4000	5000	6000	8000	10000	2
0	0	1	1500	2000	2500	3000	4000	5000	4
0	1	0	750	1000	1250	1500	2000	2500	8
0	1	1	375	500	625	750	1000	1250	16
1	0	0	187.5	250	312.5	375	500	625	32
1	0	1	93.75	125	156.25	187.5	250	312.5	64
1	1	0	46.875	62.5	78.125	93.75	125	156.25	128
1	1	1	6000	8000	10000	12000	16000	20000	1

Notes: 1. These frequencies are achieved in X1 mode,  $F_{PER} = F_{OSC} \div 2$ .  
 2. These frequencies are achieved in X2 mode,  $F_{PER} = F_{OSC}$ .

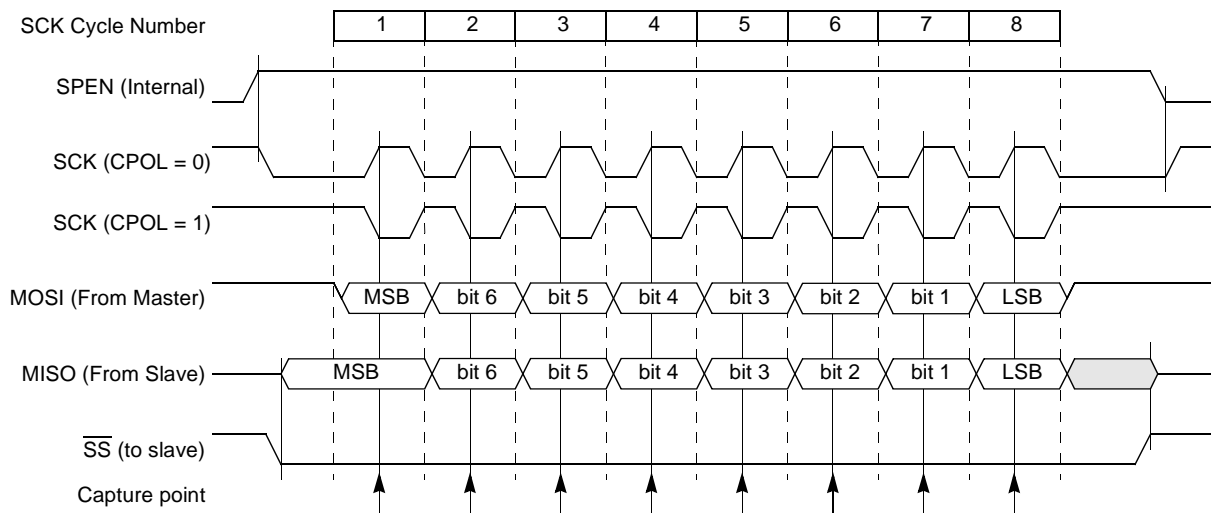
## Data Transfer

The Clock Polarity bit (CPOL in SPCON) defines the default SCK line level in idle state<sup>(1)</sup> while the Clock Phase bit (CPHA in SPCON) defines the edges on which the input data are sampled and the edges on which the output data are shifted (see Figure 115 and Figure 116). The SI signal is output from the selected slave and the SO signal is the output from the master. The AT8xC51SND1C captures data from the SI line while the selected slave captures data from the SO line.

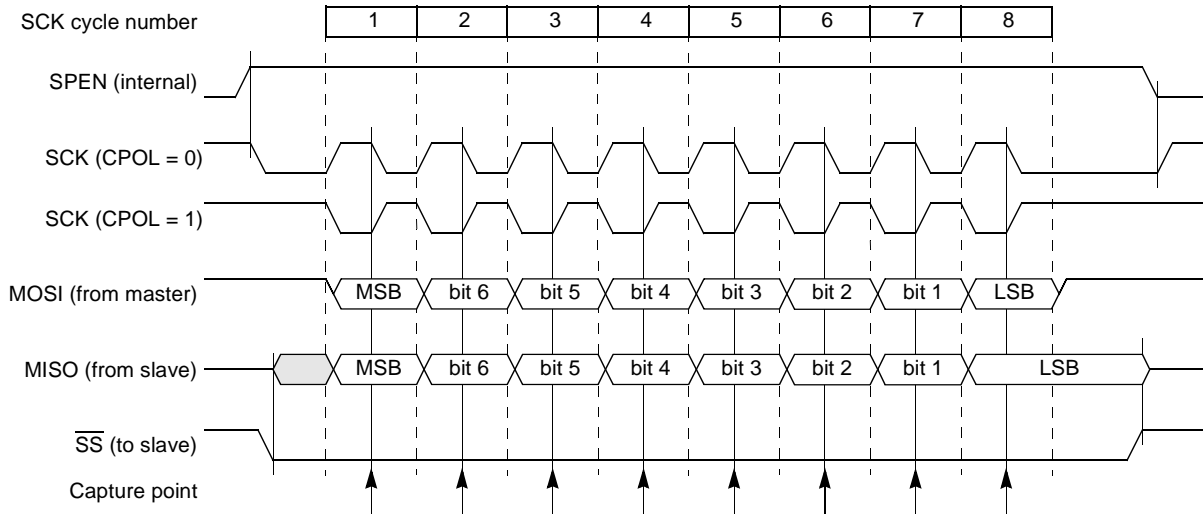
For simplicity, Figure 115 and Figure 116 depict the SPI waveforms in idealized form and do not provide precise timing information. For timing parameters refer to the Section "AC Characteristics".

Note: 1. When the peripheral is disabled (SPEN = 0), default SCK line is high level.

**Figure 115. Data Transmission Format (CPHA = 0)**



**Figure 116.** Data Transmission Format (CPHA = 1)

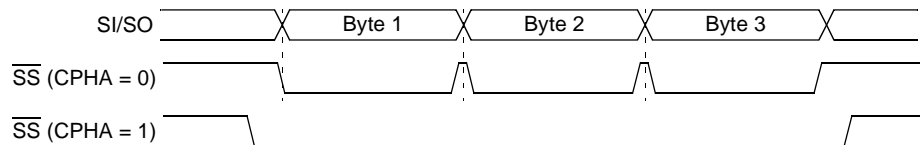


### SS Management

Figure 115 shows an SPI transmission with CPHA = 0, where the first SCK edge is the MSB capture point. Therefore the slave starts to output its MSB as soon as it is selected:  $\overline{SS}$  asserted to low level.  $\overline{SS}$  must then be deasserted between each Byte transmission (see Figure 117). SPDAT must be loaded with a data before  $\overline{SS}$  is asserted again.

Figure 116 shows an SPI transmission with CPHA = 1, where the first SCK edge is used by the slave as a start of transmission signal. Therefore,  $\overline{SS}$  may remain asserted between each Byte transmission (see Figure 117).

**Figure 117.**  $\overline{SS}$  Timing Diagram



### Error Conditions

The following flags signal the SPI error conditions:

- MODF in SPSTA signals a mode fault.  
MODF flag is relevant only in master mode when  $\overline{SS}$  usage is enabled (SSDIS bit cleared). It signals when set that an other master on the bus has asserted  $\overline{SS}$  pin and so, may create a conflict on the bus with 2 master sending data at the same time.
- A mode fault automatically disables the SPI (SPEN cleared) and configures the SPI in slave mode (MSTR cleared).  
MODF flag can trigger an interrupt as explained in Section "Interrupt", page 151. MODF flag is cleared by reading SPSTA and re-configuring SPI by writing to SPCON.
- WCOL in SPSTA signals a write collision.  
WCOL flag is set when SPDAT is loaded while a transfer is on-going. In this case data is not written to SPDAT and transfer continue uninterrupted. WCOL flag does not trigger any interrupt and is relevant jointly with SPIF flag.  
WCOL flag is cleared after reading SPSTA and writing new data to SPDAT while no transfer is on-going.

**Interrupt**

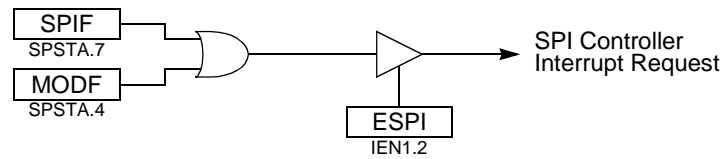
The SPI handles 2 interrupt sources that are the “end of transfer” and the “mode fault” flags.

As shown in Figure 118, these flags are combined together to appear as a single interrupt source for the C51 core. The SPIF flag is set at the end of an 8-bit shift in and out and is cleared by reading SPSTA and then reading from or writing to SPDAT.

The MODF flag is set in case of mode fault error and is cleared by reading SPSTA and then writing to SPCON.

The SPI interrupt is enabled by setting ESPI bit in IEN1 register. This assumes interrupts are globally enabled by setting EA bit in IEN0 register.

**Figure 118.** SPI Interrupt System



## Configuration

The SPI configuration is made through SPCON.

## Master Configuration

The SPI operates in master mode when the MSTR bit in SPCON is set.

## Slave Configuration

The SPI operates in slave mode when the MSTR bit in SPCON is cleared and data has been loaded in SPDAT.

## Data Exchange

There are 2 possible methods to exchange data in master and slave modes:

- polling
- interrupts

## Master Mode with Polling Policy

Figure 119 shows the initialization phase and the transfer phase flows using the polling method. Using this flow prevents any overrun error occurrence.

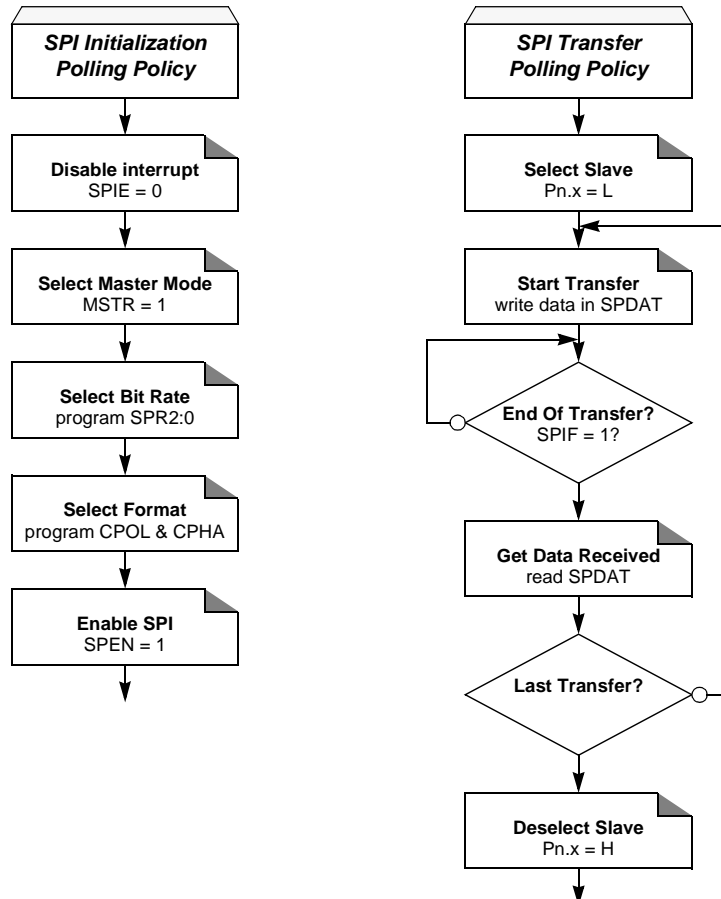
The bit rate is selected according to Table 131. The transfer format depends on the slave peripheral.

$\overline{SS}$  may be deasserted between transfers depending also on the slave peripheral.

SPIF flag is cleared when reading SPDAT (SPSTA has been read before by the “end of transfer” check).

This polling method provides the fastest effective transmission and is well adapted when communicating at high speed with other microcontrollers. However, the procedure may then be interrupted at any time by higher priority tasks.

**Figure 119.** Master SPI Polling Flows





**Master Mode with Interrupt**

Figure 120 shows the initialization phase and the transfer phase flows using the interrupt. Using this flow prevents any overrun error occurrence.

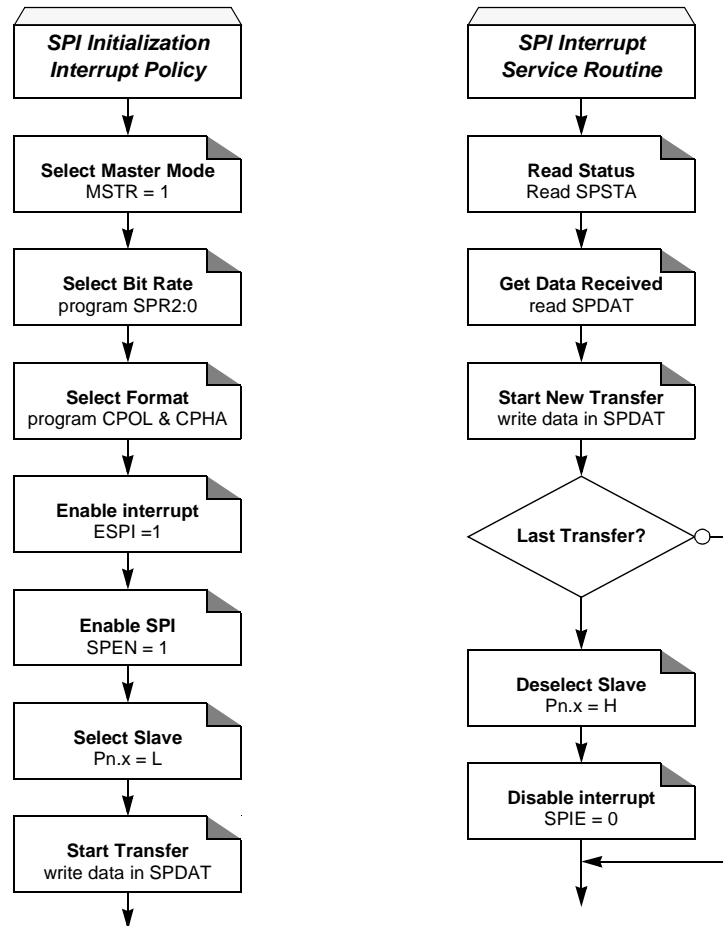
The bit rate is selected according to Table 131.

The transfer format depends on the slave peripheral.

$\overline{SS}$  may be deasserted between transfers depending also on the slave peripheral.

Reading SPSTA at the beginning of the ISR is mandatory for clearing the SPIF flag. Clear is effective when reading SPDAT.

**Figure 120.** Master SPI Interrupt Flows



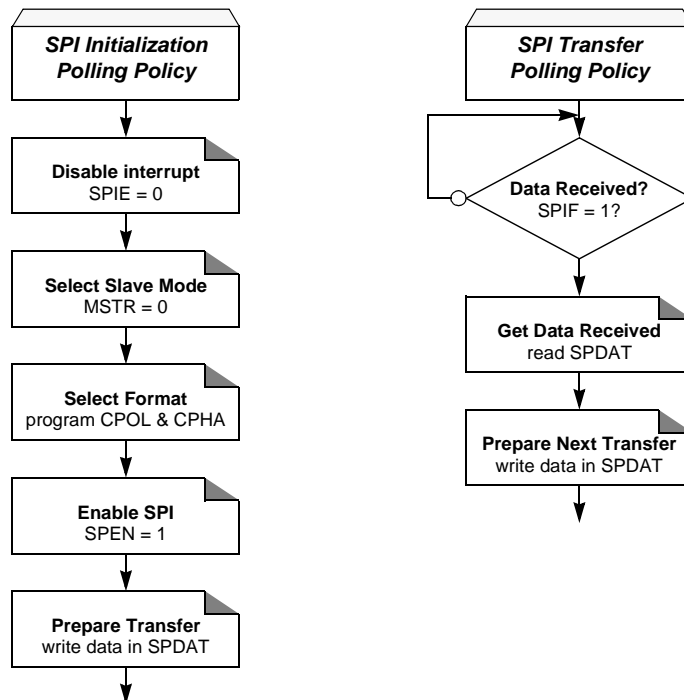
## Slave Mode with Polling Policy

Figure 121 shows the initialization phase and the transfer phase flows using the polling. The transfer format depends on the master controller.

SPIF flag is cleared when reading SPDAT (SPSTA has been read before by the “end of reception” check).

This provides the fastest effective transmission and is well adapted when communicating at high speed with other Microcontrollers. However, the process may then be interrupted at any time by higher priority tasks.

**Figure 121.** Slave SPI Polling Flows



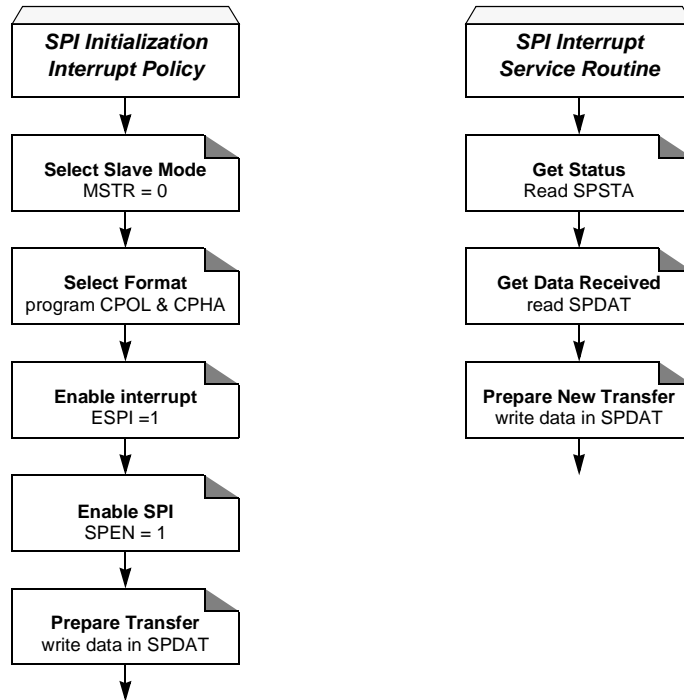
**Slave Mode with Interrupt Policy**

Figure 120 shows the initialization phase and the transfer phase flows using the interrupt.

The transfer format depends on the master controller.

Reading SPSTA at the beginning of the ISR is mandatory for clearing the SPIF flag. Clear is effective when reading SPDAT.

**Figure 122.** Slave SPI Interrupt Policy Flows



## Registers

**Table 132.** SPCON Register

SPCON (S:C3h) – SPI Control Register

7	6	5	4	3	2	1	0
SPR2	SPEN	SSDIS	MSTR	CPOL	CPHA	SPR1	SPR0
Bit Number	Bit Mnemonic	Description					
7	SPR2	<b>SPI Rate Bit 2</b> Refer to Table 131 for bit rate description.					
6	SPEN	<b>SPI Enable Bit</b> Set to enable the SPI interface. Clear to disable the SPI interface.					
5	SSDIS	<b>Slave Select Input Disable Bit</b> Set to disable $\overline{SS}$ in both master and slave modes. In slave mode this bit has no effect if CPHA = 0. Clear to enable $\overline{SS}$ in both master and slave modes.					
4	MSTR	<b>Master Mode Select</b> Set to select the master mode. Clear to select the slave mode.					
3	CPOL	<b>SPI Clock Polarity Bit<sup>(1)</sup></b> Set to have the clock output set to high level in idle state. Clear to have the clock output set to low level in idle state.					
2	CPHA	<b>SPI Clock Phase Bit</b> Set to have the data sampled when the clock returns to idle state (see CPOL). Clear to have the data sampled when the clock leaves the idle state (see CPOL).					
1 - 0	SPR1:0	<b>SPI Rate Bits 0 and 1</b> Refer to Table 131 for bit rate description.					

Reset Value = 0001 0100b

Note: 1. When the SPI is disabled, SCK outputs high level.

**Table 133.** SPSTA Register

SPSTA (S:C4h) – SPI Status Register

	7	6	5	4	3	2	1	0
	SPIF	WCOL	-	MODF	-	-	-	-

Bit Number	Bit Mnemonic	Description
7	SPIF	<b>SPI Interrupt Flag</b> Set by hardware when an 8-bit shift is completed. Cleared by hardware when reading or writing SPDAT after reading SPSTA.
6	WCOL	<b>Write Collision Flag</b> Set by hardware to indicate that a collision has been detected. Cleared by hardware to indicate that no collision has been detected.
5	-	<b>Reserved</b> The value read from this bit is indeterminate. Do not set this bit.
4	MODF	<b>Mode Fault</b> Set by hardware to indicate that the $\overline{SS}$ pin is at an appropriate level. Cleared by hardware to indicate that the $\overline{SS}$ pin is at an inappropriate level.
3 - 0	-	<b>Reserved</b> The value read from these bits is indeterminate. Do not set these bits.

Reset Value = 00000 0000b

**Table 134.** SPDAT Register

SPDAT (S:C5h) – Synchronous Serial Data Register

	7	6	5	4	3	2	1	0
	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0

Bit Number	Bit Mnemonic	Description
7 - 0	SPD7:0	<b>Synchronous Serial Data.</b>

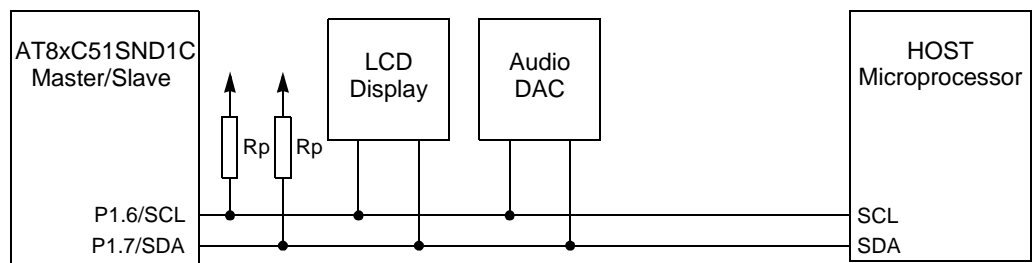
Reset Value = XXXX XXXXb

## Two-wire Interface (TWI) Controller

The AT8xC51SND1C implements a TWI controller supporting the four standard master and slave modes with multimaster capability. Thus, it allows connection of slave devices like LCD controller, audio DAC, etc., but also external master controlling where the AT8xC51SND1C is used as a peripheral of a host.

The TWI bus is a bi-directional TWI serial communication standard. It is designed primarily for simple but efficient integrated circuit control. The system is comprised of 2 lines, SCL (Serial Clock) and SDA (Serial Data) that carry information between the ICs connected to them. The serial data transfer is limited to 100 Kbit/s in low speed mode, however, some higher bit rates can be achieved depending on the oscillator frequency. Various communication configurations can be designed using this bus. Figure 123 shows a typical TWI bus configuration using the AT8xC51SND1C in master and slave modes. All the devices connected to the bus can be master and slave.

**Figure 123.** Typical TWI Bus Configuration



### Description

The CPU interfaces to the TWI logic via the following four 8-bit special function registers: the Synchronous Serial Control register (SSCON SFR, see Table 142), the Synchronous Serial Data register (SSDAT SFR, see Table 144), the Synchronous Serial Status register (SSSTA SFR, see Table 143) and the Synchronous Serial Address register (SSADR SFR, see Table 145).

SSCON is used to enable the controller, to program the bit rate (see Table 142), to enable slave modes, to acknowledge or not a received data, to send a START or a STOP condition on the TWI bus, and to acknowledge a serial interrupt. A hardware reset disables the TWI controller.

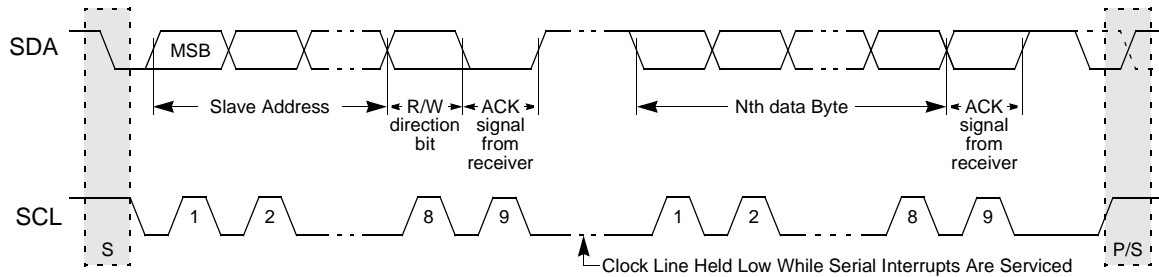
SSSTA contains a status code which reflects the status of the TWI logic and the TWI bus. The three least significant bits are always zero. The five most significant bits contains the status code. There are 26 possible status codes. When SSSTA contains F8h, no relevant state information is available and no serial interrupt is requested. A valid status code is available in SSSTA after SSI is set by hardware and is still present until SSI has been reset by software. Table 136 to Table 128 give the status for both master and slave modes and miscellaneous states.

SSDAT contains a Byte of serial data to be transmitted or a Byte which has just been received. It is addressable while it is not in process of shifting a Byte. This occurs when TWI logic is in a defined state and the serial interrupt flag is set. Data in SSDAT remains stable as long as SSI is set. While data is being shifted out, data on the bus is simultaneously shifted in; SSDAT always contains the last Byte present on the bus.

SSADR may be loaded with the 7 - bit slave address (7 most significant bits) to which the controller will respond when programmed as a slave transmitter or receiver. The LSB is used to enable general call address (00h) recognition.

Figure 124 shows how a data transfer is accomplished on the TWI bus.

**Figure 124. Complete Data Transfer on TWI Bus**



The four operating modes are:

- Master transmitter
- Master receiver
- Slave transmitter
- Slave receiver

Data transfer in each mode of operation are shown in Figure 125 through Figure 128. These figures contain the following abbreviations:

A	Acknowledge bit (low level at SDA)
$\bar{A}$	Not acknowledge bit (high level on SDA)
Data	8-bit data Byte
S	START condition
P	STOP condition
MR	Master Receive
MT	Master Transmit
SLA	Slave Address
GCA	General Call Address (00h)
R	Read bit (high level at SDA)
W	Write bit (low level at SDA)

In Figure 125 through Figure 128, circles are used to indicate when the serial interrupt flag is set. The numbers in the circles show the status code held in SSSTA. At these points, a service routine must be executed to continue or complete the serial transfer. These service routines are not critical since the serial transfer is suspended until the serial interrupt flag is cleared by software.

When the serial interrupt routine is entered, the status code in SSSTA is used to branch to the appropriate service routine. For each status code, the required software action and details of the following serial transfer are given in Table 136 through Table 128.

## Bit Rate

The bit rate can be selected from seven predefined bit rates or from a programmable bit rate generator using the SSCR2, SSCR1, and SSCR0 control bits in SSCR0 (see Table 142). The predefined bit rates are derived from the peripheral clock ( $F_{PER}$ ) issued from the Clock Controller block as detailed in section "Oscillator", page 12, while bit rate generator is based on timer 1 overflow output.

**Table 135.** Serial Clock Rates

SSCRx			Bit Frequency (kHz)			$F_{PER}$ Divided By
2	1	0	$F_{PER} = 6 \text{ MHz}$	$F_{PER} = 8 \text{ MHz}$	$F_{PER} = 10 \text{ MHz}$	
0	0	0	47	62.5	78.125	256
0	0	1	53.5	71.5	89.3	224
0	1	0	62.5	83	104.2 <sup>(1)</sup>	192
0	1	1	75	100	125 <sup>(1)</sup>	160
1	0	0	12.5	16.5	20.83	960
1	0	1	100	133.3 <sup>(1)</sup>	166.7 <sup>(1)</sup>	120
1	1	0	200 <sup>(1)</sup>	266.7 <sup>(1)</sup>	333.3 <sup>(1)</sup>	60
1	1	1	$0.5 < \cdot < 125^{(1)}$	$0.67 < \cdot < 166.7^{(1)}$	$0.81 < \cdot < 208.3^{(1)}$	$96 \cdot (256 - \text{reload value Timer 1})$

Note: 1. These bit rates are outside of the low speed standard specification limited to 100 kHz but can be used with high speed TWI components limited to 400 kHz.

## Master Transmitter Mode

In the master transmitter mode, a number of data Bytes are transmitted to a slave receiver (see Figure 125). Before the master transmitter mode can be entered, SSCR0 must be initialized as follows:

SSCR2	SSPE	SSSTA	SSSTO	SSI	SSAA	SSCR1	SSCR0
Bit Rate	1	0	0	0	X	Bit Rate	Bit Rate

SSCR2:0 define the serial bit rate (see Table 135). SSPE must be set to enable the controller. SSSTA, SSSTO and SSI must be cleared.

The master transmitter mode may now be entered by setting the SSSTA bit. The TWI logic will now monitor the TWI bus and generate a START condition as soon as the bus becomes free. When a START condition is transmitted, the serial interrupt flag (SSI bit in SSCR0) is set, and the status code in SSSTA is 08h. This status must be used to vector to an interrupt routine that loads SSDAT with the slave address and the data direction bit (SLA+W). The serial interrupt flag (SSI) must then be cleared before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, SSI is set again and a number of status code in SSSTA are possible. There are 18h, 20h or 38h for the master mode and also 68h, 78h or B0h if the slave mode was enabled (SSAA = logic 1). The appropriate action to be taken for each of these status code is detailed in Table 136. This scheme is repeated until a STOP condition is transmitted.

SSPE and SSCR2:0 are not affected by the serial transfer and are not referred to in Table 136. After a repeated START condition (state 10h) the controller may switch to the master receiver mode by loading SSDAT with SLA+R.



## Master Receiver Mode

In the master receiver mode, a number of data Bytes are received from a slave transmitter (see Figure 126). The transfer is initialized as in the master transmitter mode. When the START condition has been transmitted, the interrupt routine must load SSDAT with the 7 - bit slave address and the data direction bit (SLA+R). The serial interrupt flag (SSI) must then be cleared before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag is set again and a number of status code in SSSTA are possible. There are 40h, 48h or 38h for the master mode and also 68h, 78h or B0h if the slave mode was enabled (SSAA = logic 1). The appropriate action to be taken for each of these status code is detailed in Table 128. This scheme is repeated until a STOP condition is transmitted.

SSPE and SSCR2:0 are not affected by the serial transfer and are not referred to in Table 128. After a repeated START condition (state 10h) the controller may switch to the master transmitter mode by loading SSDAT with SLA+W.

## Slave Receiver Mode

In the slave receiver mode, a number of data Bytes are received from a master transmitter (see Figure 127). To initiate the slave receiver mode, SSADR and SSCR2:0 must be loaded as follows:

SSA6	SSA5	SSA4	SSA3	SSA2	SSA1	SSA0	SSGC
← Own Slave Address →							X

The upper 7 bits are the addresses to which the controller will respond when addressed by a master. If the LSB (SSGC) is set, the controller will respond to the general call address (00h); otherwise, it ignores the general call address.

SSCR2	SSPE	SSSTA	SSSTO	SSI	SSAA	SSCR1	SSCR0
X	1	0	0	0	1	X	X

SSCR2:0 have no effect in the slave mode. SSPE must be set to enable the controller. The SSAA bit must be set to enable the own slave address or the general call address acknowledgment. SSSTA, SSSTO and SSI must be cleared.

When SSADR and SSCR2:0 have been initialized, the controller waits until it is addressed by its own slave address followed by the data direction bit which must be logic 0 (W) for operating in the slave receiver mode. After its own slave address and the W bit has been received, the serial interrupt flag is set and a valid status code can be read from SSSTA. This status code is used to vector to an interrupt service routine, and the appropriate action to be taken for each of these status code is detailed in Table 128 and Table 140. The slave receiver mode may also be entered if arbitration is lost while the controller is in the master mode (see states 68h and 78h).

If the SSAA bit is reset during a transfer, the controller will return a not acknowledge (logic 1) to SDA after the next received data Byte. While SSAA is reset, the controller does not respond to its own slave address. However, the TWI bus is still monitored and address recognition may be resumed at any time by setting SSAA. This means that the SSAA bit may be used to temporarily isolate the controller from the TWI bus.

## Slave Transmitter Mode

In the slave transmitter mode, a number of data Bytes are transmitted to a master receiver (see Figure 128). Data transfer is initialized as in the slave receiver mode. When SSADR and SSSCON have been initialized, the controller waits until it is addressed by its own slave address followed by the data direction bit which must be logic 1 (R) for operating in the slave transmitter mode. After its own slave address and the R bit have been received, the serial interrupt flag is set and a valid status code can be read from SSSTA. This status code is used to vector to an interrupt service routine, and the appropriate action to be taken for each of these status code is detailed in Table 140. The slave transmitter mode may also be entered if arbitration is lost while the controller is in the master mode (see state B0h).

If the SSAA bit is reset during a transfer, the controller will transmit the last Byte of the transfer and enter state C0h or C8h. The controller is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1's as serial data. While SSAA is reset, the controller does not respond to its own slave address. However, the TWI bus is still monitored and address recognition may be resumed at any time by setting SSAA. This means that the SSAA bit may be used to temporarily isolate the controller from the TWI bus.

## Miscellaneous States

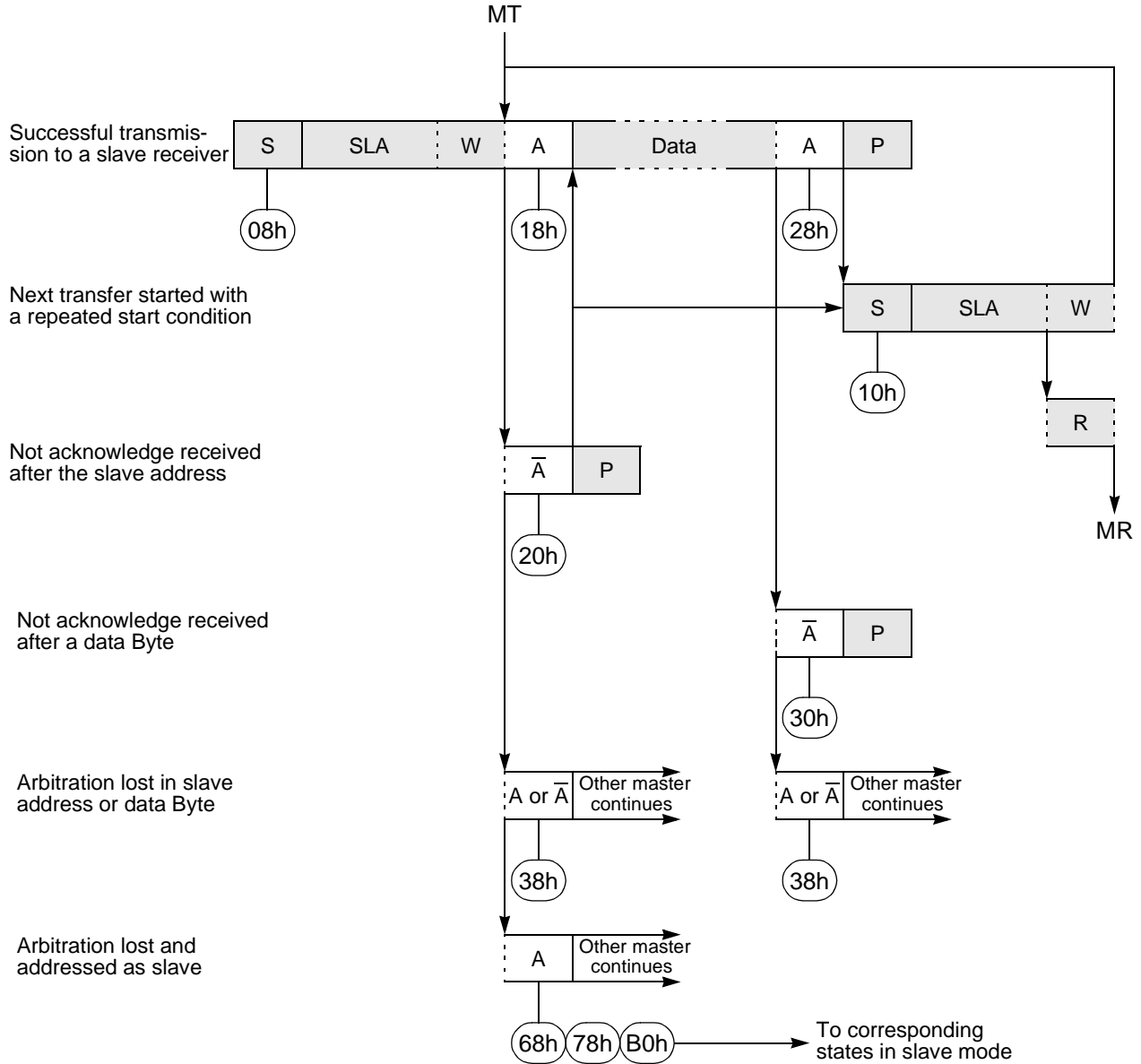
There are 2 SSSTA codes that do not correspond to a defined TWI hardware state (see Table 141). These are discussed below.

Status F8h indicates that no relevant information is available because the serial interrupt flag is not yet set. This occurs between other states and when the controller is not involved in a serial transfer.

Status 00h indicates that a bus error has occurred during a serial transfer. A bus error is caused when a START or a STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address Byte, a data Byte, or an acknowledge bit. When a bus error occurs, SSI is set. To recover from a bus error, the SSSTO flag must be set and SSI must be cleared. This causes the controller to enter the not addressed slave mode and to clear the SSSTO flag (no other bits in S1CON are affected). The SDA and SCL lines are released and no STOP condition is transmitted.

Note: The TWI controller interfaces to the external TWI bus via 2 port 1 pins: P1.6/SCL (serial clock line) and P1.7/SDA (serial data line). To avoid low level asserting and conflict on these lines when the TWI controller is enabled, the output latches of P1.6 and P1.7 must be set to logic 1.

Figure 125. Format and States in the Master Transmitter Mode



From master to slave
  From slave to master

Data
  A
 Any number of data Bytes and their associated acknowledge bits

nnh
This number (contained in SSSTA) corresponds to a defined state of the TWI bus

**Figure 126. Format and States in the Master Receiver Mode**

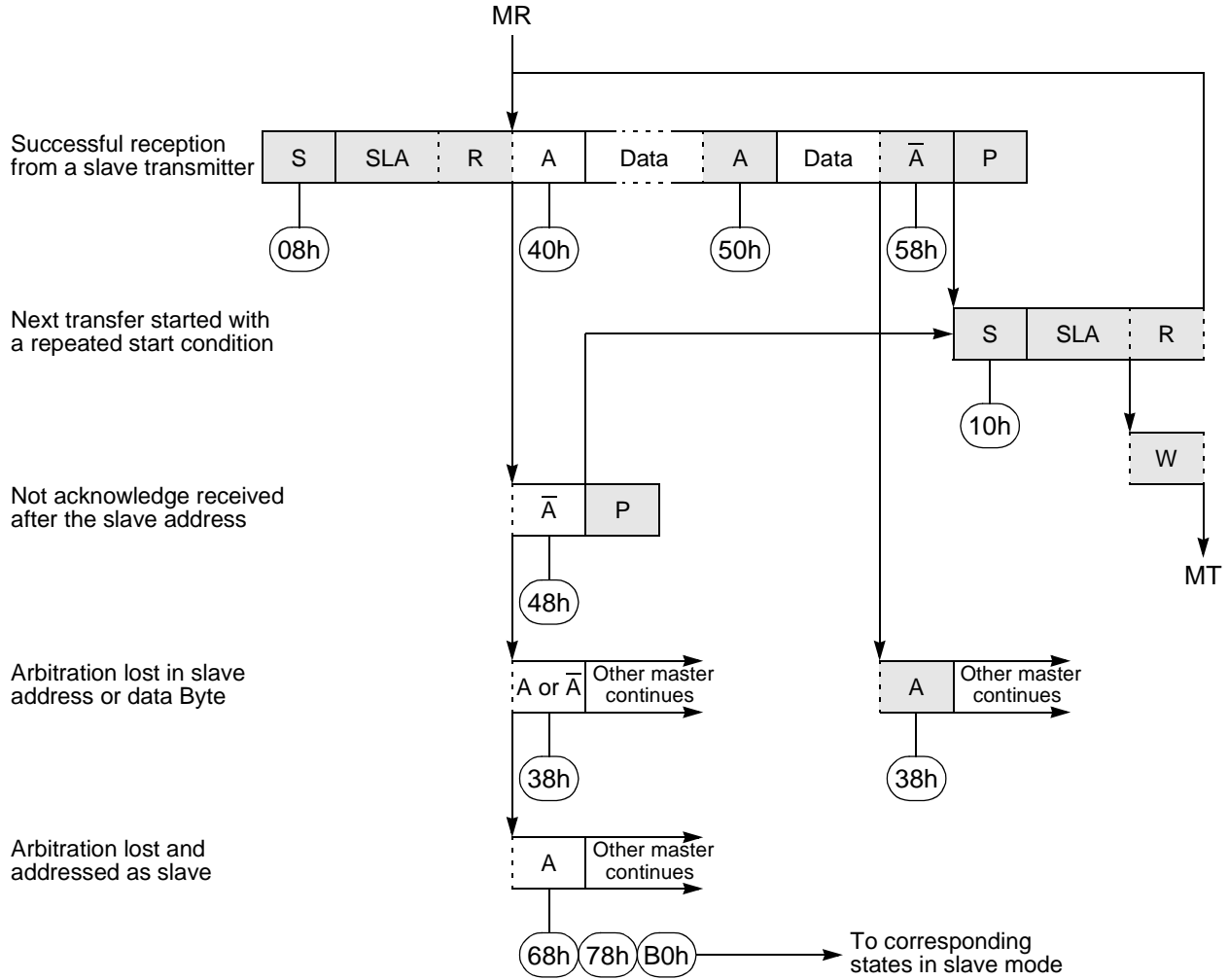


Figure 127. Format and States in the Slave Receiver Mode

Reception of the own slave address and one or more data Bytes. All are acknowledged

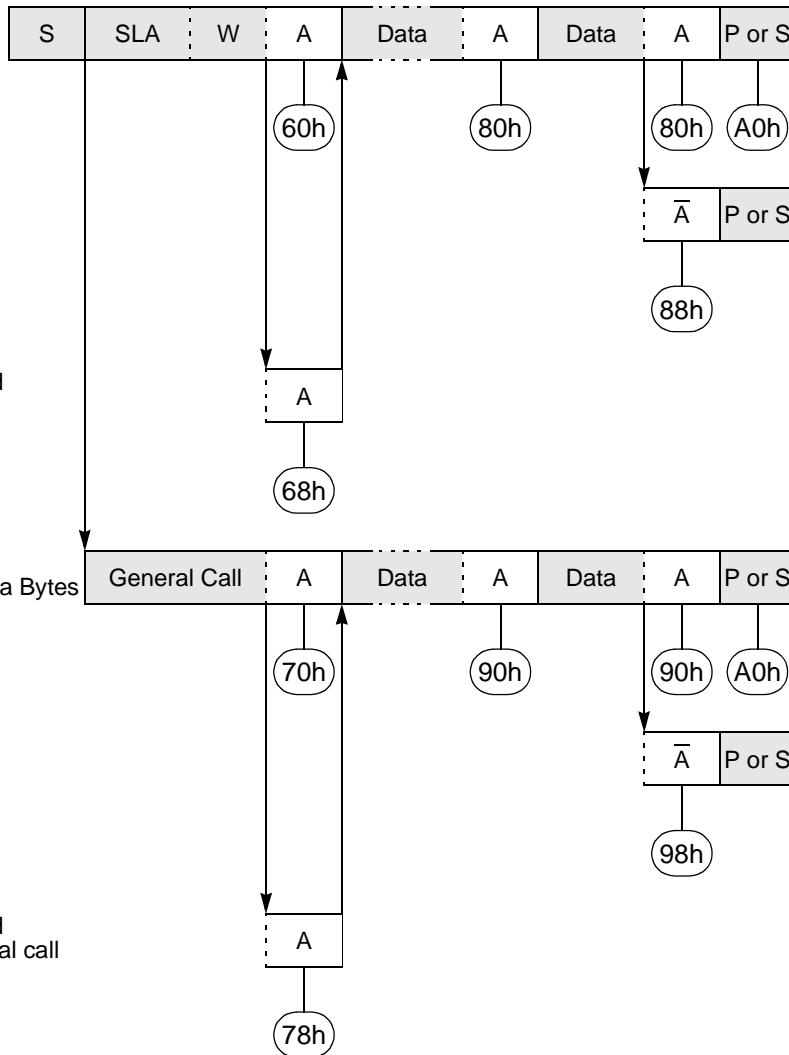
Last data Byte received is not acknowledged



Arbitration lost as master and addressed as slave

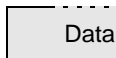
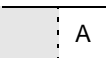

Reception of the general call address and one or more data Bytes

Last data Byte received is not acknowledged

Arbitration lost as master and addressed as slave by general call



 From master to slave  
 From slave to master

 Data     A  
 (nnh)

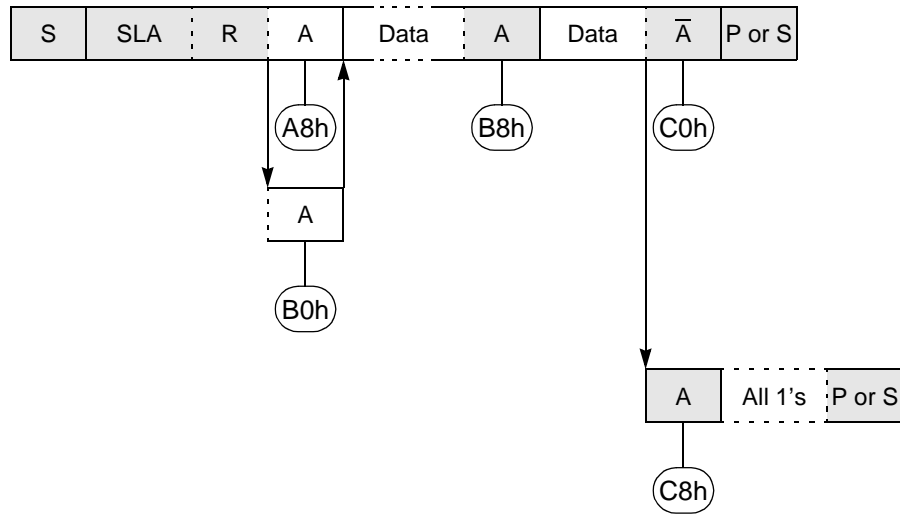
Any number of data Bytes and their associated acknowledge bits  
 This number (contained in SSSTA) corresponds to a defined state of the TWI bus

**Figure 128. Format and States in the Slave Transmitter Mode**

Reception of the own slave address and transmission of one or more data Bytes.

Arbitration lost as master and addressed as slave

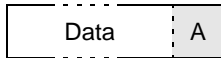
Last data Byte transmitted. Switched to not addressed slave (SSAA = 0).



From master to slave



From slave to master



Data

A

Any number of data Bytes and their associated acknowledge bits

(nnh)

This number (contained in SSSTA) corresponds to a defined state of the TWI bus

**Table 136. Status for Master Transmitter Mode**

Status Code SSSTA	Status of the TWI Bus and TWI Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/From SSDAT	To SSSCON				
			SSSTA	SSSTO	SSI	SSAA	
08h	A START condition has been transmitted	Write SLA+W	X	0	0	X	SLA+W will be transmitted.
10h	A repeated START condition has been transmitted	Write SLA+W	X	0	0	X	SLA+W will be transmitted.
		Write SLA+R	X	0	0	X	SLA+R will be transmitted. Logic will switch to master receiver mode
18h	SLA+W has been transmitted; ACK has been received	Write data Byte	0	0	0	X	Data Byte will be transmitted.
		No SSDAT action	1	0	0	X	Repeated START will be transmitted.
		No SSDAT action	0	1	0	X	STOP condition will be transmitted and SSSTO flag will be reset.
		No SSDAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted and SSSTO flag will be reset.
20h	SLA+W has been transmitted; NOT ACK has been received	Write data Byte	0	0	0	X	Data Byte will be transmitted.
		No SSDAT action	1	0	0	X	Repeated START will be transmitted.
		No SSDAT action	0	1	0	X	STOP condition will be transmitted and SSSTO flag will be reset.
		No SSDAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted and SSSTO flag will be reset.
28h	Data Byte has been transmitted; ACK has been received	Write data Byte	0	0	0	X	Data Byte will be transmitted.
		No SSDAT action	1	0	0	X	Repeated START will be transmitted.
		No SSDAT action	0	1	0	X	STOP condition will be transmitted and SSSTO flag will be reset.
		No SSDAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted and SSSTO flag will be reset.
30h	Data Byte has been transmitted; NOT ACK has been received	Write data Byte	0	0	0	X	Data Byte will be transmitted.
		No SSDAT action	1	0	0	X	Repeated START will be transmitted.
		No SSDAT action	0	1	0	X	STOP condition will be transmitted and SSSTO flag will be reset.
		No SSDAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted and SSSTO flag will be reset.
38h	Arbitration lost in SLA+W or data Bytes	No SSDAT action	0	0	0	X	TWI bus will be released and not addressed slave mode will be entered.
		No SSDAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.

**Table 137. Status for Master Receiver Mode**

Status Code SSSTA	Status of the TWI Bus and TWI Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/From SSDAT	To SSSCON				
			SSSTA	SSSTO	SSI	SSAA	
08h	A START condition has been transmitted	Write SLA+R	X	0	0	X	SLA+R will be transmitted.
10h	A repeated START condition has been transmitted	Write SLA+R	X	0	0	X	SLA+R will be transmitted.
		Write SLA+W	X	0	0	X	SLA+W will be transmitted. Logic will switch to master transmitter mode.
38h	Arbitration lost in SLA+R or NOT ACK bit	No SSDAT action	0	0	0	X	TWI bus will be released and not addressed slave mode will be entered.
		No SSDAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
40h	SLA+R has been transmitted; ACK has been received	No SSDAT action	0	0	0	0	Data Byte will be received and NOT ACK will be returned.
		No SSDAT action	0	0	0	1	Data Byte will be received and ACK will be returned.
48h	SLA+R has been transmitted; NOT ACK has been received	No SSDAT action	1	0	0	X	Repeated START will be transmitted.
		No SSDAT action	0	1	0	X	STOP condition will be transmitted and SSSTO flag will be reset.
		No SSDAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted and SSSTO flag will be reset.
50h	Data Byte has been received; ACK has been returned	Read data Byte	0	0	0	0	Data Byte will be received and NOT ACK will be returned.
		Read data Byte	0	0	0	1	Data Byte will be received and ACK will be returned.
58h	Data Byte has been received; NOT ACK has been returned	Read data Byte	1	0	0	X	Repeated START will be transmitted.
		Read data Byte	0	1	0	X	STOP condition will be transmitted and SSSTO flag will be reset.
		Read data Byte	1	1	0	X	STOP condition followed by a START condition will be transmitted and SSSTO flag will be reset.



**Table 138.** Status for Slave Receiver Mode with Own Slave Address

Status Code SSSTA	Status of the TWI Bus and TWI Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/From SSDAT	To SSSCON				
			SSSTA	SSSTO	SSI	SSAA	
60h	Own SLA+W has been received; ACK has been returned	No SSDAT action	X	0	0	0	Data Byte will be received and NOT ACK will be returned.
		No SSDAT action	X	0	0	1	Data Byte will be received and ACK will be returned.
68h	Arbitration lost in SLA+R/W as master; own SLA+W has been received; ACK has been returned	No SSDAT action	X	0	0	0	Data Byte will be received and NOT ACK will be returned.
		No SSDAT action	X	0	0	1	Data Byte will be received and ACK will be returned.
80h	Previously addressed with own SLA+W; data has been received; ACK has been returned	Read data Byte	X	0	0	0	Data Byte will be received and NOT ACK will be returned.
		Read data Byte	X	0	0	1	Data Byte will be received and ACK will be returned.
88h	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data Byte	0	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA.
		Read data Byte	0	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1.
		Read data Byte	1	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA. A START condition will be transmitted when the bus becomes free.
		Read data Byte	1	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1. A START condition will be transmitted when the bus becomes free.
A0h	A STOP condition or repeated START condition has been received while still addressed as slave	No SSDAT action	0	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA.
		No SSDAT action	0	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1.
		No SSDAT action	1	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA. A START condition will be transmitted when the bus becomes free.
		No SSDAT action	1	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1. A START condition will be transmitted when the bus becomes free.

**Table 139. Status for Slave Receiver Mode with General Call Address**

Status Code SSSTA	Status of the TWI Bus and TWI Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/From SSDAT	To SSSCON				
			SSSTA	SSSTO	SSI	SSAA	
70h	General call address has been received; ACK has been returned	No SSDAT action	X	0	0	0	Data Byte will be received and NOT ACK will be returned.
		No SSDAT action	X	0	0	1	Data Byte will be received and ACK will be returned.
78h	Arbitration lost in SLA+R/W as master; general call address has been received; ACK has been returned	No SSDAT action	X	0	0	0	Data Byte will be received and NOT ACK will be returned.
		No SSDAT action	X	0	0	1	Data Byte will be received and ACK will be returned.
90h	Previously addressed with general call; data has been received; ACK has been returned	Read data Byte	X	0	0	0	Data Byte will be received and NOT ACK will be returned.
		Read data Byte	X	0	0	1	Data Byte will be received and ACK will be returned.
98h	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data Byte	0	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA.
		Read data Byte	0	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1.
		Read data Byte	1	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA. A START condition will be transmitted when the bus becomes free.
		Read data Byte	1	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1. A START condition will be transmitted when the bus becomes free.
A0h	A STOP condition or repeated START condition has been received while still addressed as slave	No SSDAT action	0	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA.
		No SSDAT action	0	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1.
		No SSDAT action	1	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA. A START condition will be transmitted when the bus becomes free.
		No SSDAT action	1	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1. A START condition will be transmitted when the bus becomes free.

**Table 140. Status for Slave Transmitter Mode**

Status Code SSSTA	Status of the TWI Bus and TWI Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/From SSDAT	To SSSCON				
			SSSTA	SSSTO	SSI	SSAA	
A8h	Own SLA+R has been received; ACK has been returned	Write data Byte	X	0	0	0	Last data Byte will be transmitted.
		Write data Byte	X	0	0	1	Data Byte will be transmitted.
B0h	Arbitration lost in SLA+R/W as master; own SLA+R has been received; ACK has been returned	Write data Byte	X	0	0	0	Last data Byte will be transmitted.
		Write data Byte	X	0	0	1	Data Byte will be transmitted.
B8h	Data Byte in SSDAT has been transmitted; ACK has been received	Write data Byte	X	0	0	0	Last data Byte will be transmitted.
		Write data Byte	X	0	0	1	Data Byte will be transmitted.
C0h	Data Byte in SSDAT has been transmitted; NOT ACK has been received	No SSDAT action	0	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA.
		No SSDAT action	0	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1.
		No SSDAT action	1	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA. A START condition will be transmitted when the bus becomes free.
		No SSDAT action	1	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1. A START condition will be transmitted when the bus becomes free.
C8h	Last data Byte in SSDAT has been transmitted (SSAA= 0); ACK has been received	No SSDAT action	0	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA.
		No SSDAT action	0	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1.
		No SSDAT action	1	0	0	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA. A START condition will be transmitted when the bus becomes free.
		No SSDAT action	1	0	0	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if SSGC = logic 1. A START condition will be transmitted when the bus becomes free.

**Table 141. Status for Miscellaneous States**

Status Code SSSTA	Status of the TWI Bus and TWI Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/From SSDAT	To SSSCON				
			SSSTA	SSSTO	SSI	SSAA	
F8h	No relevant state information available; SSI = 0	No SSDAT action	No SSSCON action				Wait or proceed current transfer.
00h	Bus error due to an illegal START or STOP condition	No SSDAT action	0	1	0	X	Only the internal hardware is affected, no STOP condition is sent on the bus. In all cases, the bus is released and SSSTO is reset.

## Registers

**Table 142.** SSCR0 Register  
SSCR0 (S:93h) – Synchronous Serial Control Register

7	6	5	4	3	2	1	0
SSCR2	SSPE	SSSTA	SSSTO	SSI	SSAA	SSCR1	SSCR0
Bit Number	Bit Mnemonic	Description					
7	SSCR2	<b>Synchronous Serial Control Rate Bit 2</b> Refer to Table 135 for rate description.					
6	SSPE	<b>Synchronous Serial Peripheral Enable Bit</b> Set to enable the controller. Clear to disable the controller.					
5	SSSTA	<b>Synchronous Serial Start Flag</b> Set to send a START condition on the bus. Clear not to send a START condition on the bus.					
4	SSSTO	<b>Synchronous Serial Stop Flag</b> Set to send a STOP condition on the bus. Clear not to send a STOP condition on the bus.					
3	SSI	<b>Synchronous Serial Interrupt Flag</b> Set by hardware when a serial interrupt is requested. Must be cleared by software to acknowledge interrupt.					
2	SSAA	<b>Synchronous Serial Assert Acknowledge Flag</b> Set to enable slave modes. Slave modes are entered when SLA or GCA (if SSGC set) is recognized. Clear to disable slave modes. <u>Master Receiver Mode in progress</u> Clear to force a not acknowledge (high level on SDA). Set to force an acknowledge (low level on SDA). <u>Master Transmitter Mode in progress</u> This bit has no specific effect when in master transmitter mode. <u>Slave Receiver Mode in progress</u> Clear to force a not acknowledge (high level on SDA). Set to force an acknowledge (low level on SDA). <u>Slave Transmitter Mode in progress</u> Clear to isolate slave from the bus after last data Byte transmission. Set to enable slave mode.					
1	SSCR1	<b>Synchronous Serial Control Rate Bit 1</b> Refer to Table 135 for rate description.					
0	SSCR0	<b>Synchronous Serial Control Rate Bit 0</b> Refer to Table 135 for rate description.					

Reset Value = 0000 0000b

**Table 143.** SSSTA Register

SSSTA (S:94h) – Synchronous Serial Status Register

7	6	5	4	3	2	1	0
<b>SSC4</b>	<b>SSC3</b>	<b>SSC2</b>	<b>SSC1</b>	<b>SSC0</b>	<b>0</b>	<b>0</b>	<b>0</b>

Bit Number	Bit Mnemonic	Description
7:3	SSC4:0	<b>Synchronous Serial Status Code Bits 0 to 4</b> Refer to Table 136 to Table 128 for status description.
2:0	0	Always 0.

Reset Value = F8h

**Table 144.** SSDAT Register

SSDAT (S:95h) – Synchronous Serial Data Register

7	6	5	4	3	2	1	0
<b>SSD7</b>	<b>SSD6</b>	<b>SSD5</b>	<b>SSD4</b>	<b>SSD3</b>	<b>SSD2</b>	<b>SSD1</b>	<b>SSD0</b>

Bit Number	Bit Mnemonic	Description
7:1	SSD7:1	<b>Synchronous Serial Address bits 7 to 1 or Synchronous Serial Data Bits 7 to 1</b>
0	SSD0	<b>Synchronous Serial Address bit 0 (R/W) or Synchronous Serial Data Bit 0</b>

Reset Value = 1111 1111b

**Table 145.** SSADR Register

SSADR (S:96h) – Synchronous Serial Address Register

7	6	5	4	3	2	1	0
<b>SSA7</b>	<b>SSA6</b>	<b>SSA5</b>	<b>SSA4</b>	<b>SSA3</b>	<b>SSA2</b>	<b>SSA1</b>	<b>SSGC</b>

Bit Number	Bit Mnemonic	Description
7:1	SSA7:1	<b>Synchronous Serial Slave Address Bits 7 to 1</b>
0	SSGC	<b>Synchronous Serial General Call Bit</b> Set to enable the general call address recognition. Clear to disable the general call address recognition.

Reset Value = 1111 1110b

# Analog to Digital Converter

## Description

The AT8xC51SND1C implement a 2-channel 10-bit (8 true bits) analog to digital converter (ADC). First channel of this ADC can be used for battery monitoring while the second one can be used for voice sampling at 8 kHz.

The A/D converter interfaces with the C51 core through four special function registers: ADCON, the ADC control register (see Table 147); ADDH and ADDL, the ADC data registers (see Table 149 and Table 150); and ADCLK, the ADC clock register (see Table 148).

As shown in Figure 129, the ADC is composed of a 10-bit cascaded potentiometric digital to analog converter, connected to the negative input of a comparator. The output voltage of this DAC is compared to the analog voltage stored in the Sample and Hold and coming from AIN0 or AIN1 input depending on the channel selected (see Table 146). The 10-bit ADDAT converted value (see formula in Figure 129) is delivered in ADDH and ADDL registers, ADDH is giving the 8 most significant bits while ADDL is giving the 2 least significant bits. ADDAT

**Figure 129.** ADC Structure

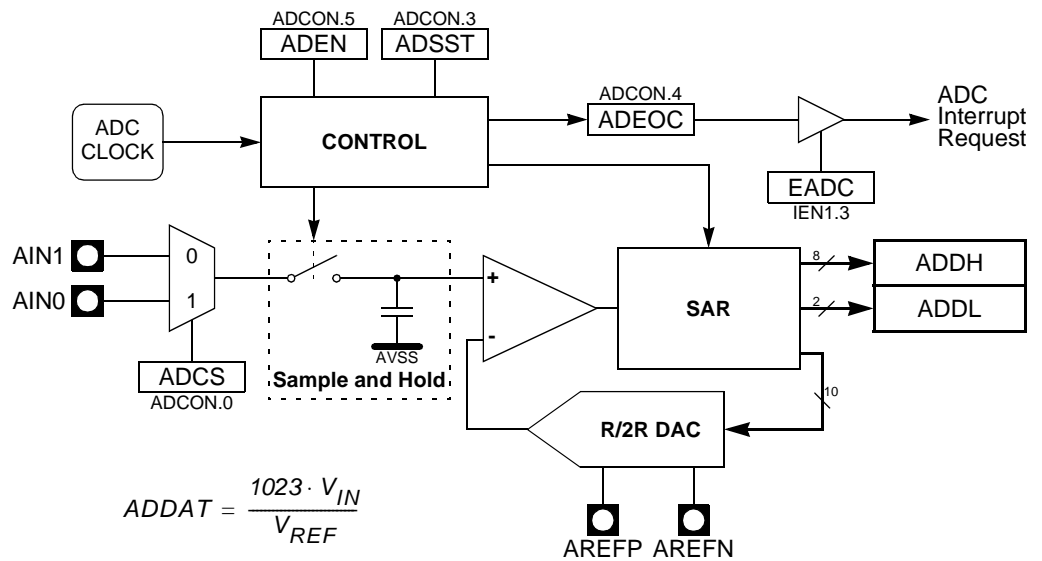
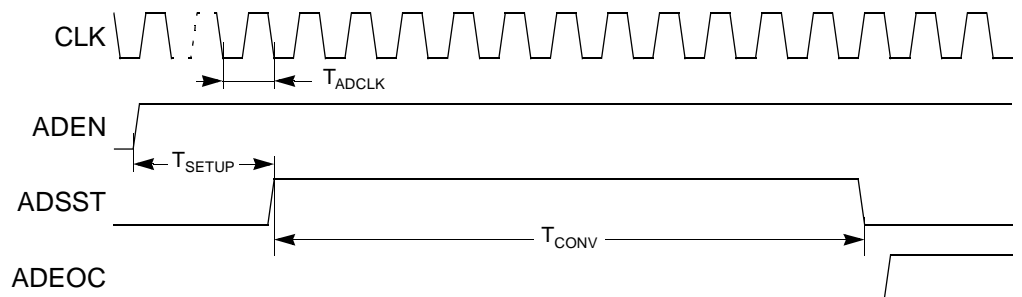


Figure 130 shows the timing diagram of a complete conversion. For simplicity, the figure depicts the waveforms in idealized form and do not provide precise timing information. For ADC characteristics and timing parameters refer to the section “AC Characteristics”.

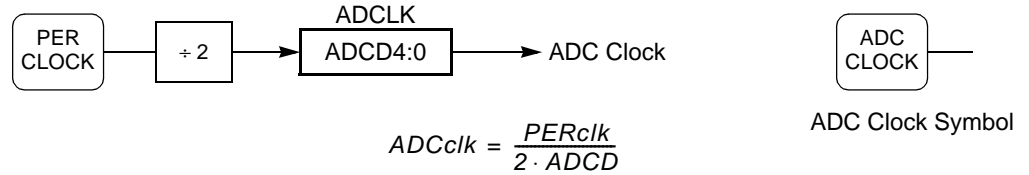
**Figure 130.** Timing Diagram



## Clock Generator

The ADC clock is generated by division of the peripheral clock (see details in section "X2 Feature", page 12). The division factor is then given by ADPC4:0 bits in ADCLK register. Figure 131 shows the ADC clock generator and its calculation formula<sup>(1)</sup>.

**Figure 131.** ADC Clock Generator and Symbol Caution:



- Note:
1. In all cases, the ADC clock frequency may be higher than the maximum  $F_{ADCLK}$  parameter reported in the section "Analog to Digital Converter", page 197.
  2. The ADCD value of 0 is equivalent to an ADCD value of 32.

## Channel Selection

The channel on which conversion is performed is selected by the ADCS bit in ADCON register according to Table 146.

**Table 146.** ADC Channel Selection

ADCS	Channel
0	AIN1
1	AIN0

## Conversion Precision

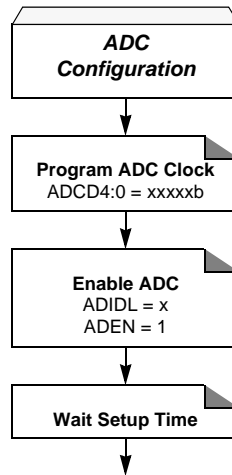
The 10-bit precision conversion is achieved by stopping the CPU core activity during conversion for limiting the digital noise induced by the core. This mode called the Pseudo-Idle mode<sup>(1),(2)</sup> is enabled by setting the ADIDL bit in ADCON register<sup>(3)</sup>. Thus, when conversion is launched (see Section "Conversion Launching", page 176), the CPU core is stopped until the end of the conversion (see Section "End Of Conversion", page 176). This bit is cleared by hardware at the end of the conversion.

- Notes:
1. Only the CPU activity is frozen, peripherals are not affected by the Pseudo-Idle mode.
  2. If some interrupts occur during the Pseudo-Idle mode, they will be delayed and processed, according to their priority after the end of the conversion.
  3. Concurrently with ADSST bit.

## Configuration

The ADC configuration consists in programming the ADC clock as detailed in the Section "Clock Generator", page 175. The ADC is enabled using the ADEN bit in ADCON register. As shown in Figure 93, user must wait the setup time ( $T_{SETUP}$ ) before launching any conversion.

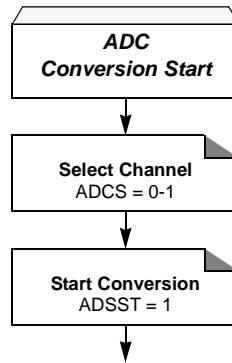
**Figure 132.** ADC Configuration Flow



**Conversion Launching**

The conversion is launched by setting the ADSST bit in ADCON register, this bit remains set during the conversion. As soon as the conversion is started, it takes 11 clock periods ( $T_{CONV}$ ) before the data is available in ADDH and ADDL registers.

**Figure 133.** ADC Conversion Launching Flow



**End Of Conversion**

The end of conversion is signalled by the ADEOC flag in ADCON register becoming set or by the ADSST bit in ADCON register becoming cleared. ADEOC flag can generate an interrupt if enabled by setting EADC bit in IEN1 register. This flag is set by hardware and must be reset by software.



## Registers

**Table 147.** ADCON Register

ADCON (S:F3h) – ADC Control Register

	7	6	5	4	3	2	1	0
	-	ADIDL	ADEN	ADEOC	ADSST	-	-	ADCS

Bit Number	Bit Mnemonic	Description
7	-	<b>Reserved</b> The value read from this bit is always 0. Do not set this bit.
6	ADIDL	<b>ADC Pseudo-Idle Mode</b> Set to suspend the CPU core activity (pseudo-idle mode) during conversion. Clear by hardware at the end of conversion.
5	ADEN	<b>ADC Enable Bit</b> Set to enable the A to D converter. Clear to disable the A to D converter and put it in low power stand by mode.
4	ADEOC	<b>End Of Conversion Flag</b> Set by hardware when ADC result is ready to be read. This flag can generate an interrupt. Must be cleared by software.
3	ADSST	<b>Start and Status Bit</b> Set to start an A to D conversion on the selected channel. Cleared by hardware at the end of conversion.
2 - 1	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.
0	ADCS	<b>Channel Selection Bit</b> Set to select channel 0 for conversion. Clear to select channel 1 for conversion.

Reset Value = 0000 0000b

**Table 148.** ADCLK Register

ADCLK (S:F2h) – ADC Clock Divider Register

	7	6	5	4	3	2	1	0
	-	-	-	ADCD4	ADCD3	ADCD2	ADCD1	ADCD0

Bit Number	Bit Mnemonic	Description
7 - 5	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.
4 - 0	ADCD4:0	<b>ADC Clock Divider</b> 5-bit divider for ADC clock generation.

Reset Value = 0000 0000b

**Table 149.** ADDH Register

ADDH (S:F5h Read Only) – ADC Data High Byte Register

7	6	5	4	3	2	1	0
<b>ADAT9</b>	<b>ADAT8</b>	<b>ADAT7</b>	<b>ADAT6</b>	<b>ADAT5</b>	<b>ADAT4</b>	<b>ADAT3</b>	<b>ADAT2</b>
Bit Number	Bit Mnemonic	Description					
7 - 0	ADAT9:2	<b>ADC Data</b> 8 Most Significant Bits of the 10-bit ADC data.					

Reset Value = 0000 0000b

**Table 150.** ADDL Register

ADDL (S:F4h Read Only) – ADC Data Low Byte Register

7	6	5	4	3	2	1	0
-	-	-	-	-	-	<b>ADAT1</b>	<b>ADAT0</b>
Bit Number	Bit Mnemonic	Description					
7 - 2	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.					
1 - 0	ADAT1:0	<b>ADC Data</b> 2 Least Significant Bits of the 10-bit ADC data.					

Reset Value = 0000 0000b

## Keyboard Interface

The AT8xC51SND1C implement a keyboard interface allowing the connection of a 4 x n matrix keyboard. It is based on 4 inputs with programmable interrupt capability on both high or low level. These inputs are available as alternate function of P1.3:0 and allow exit from idle and power down modes.

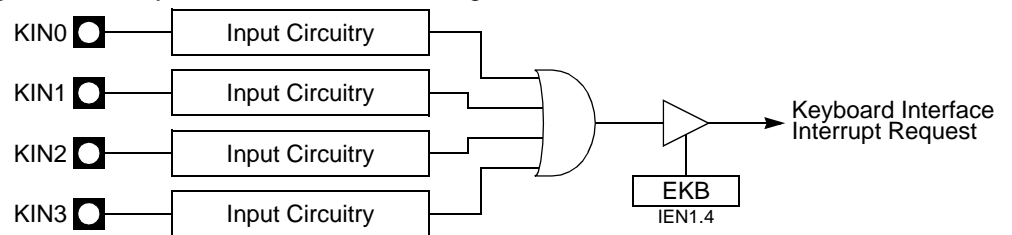
### Description

The keyboard interfaces with the C51 core through 2 special function registers: KBCON, the keyboard control register (see Table 151); and KBSTA, the keyboard control and status register (see Table 152).

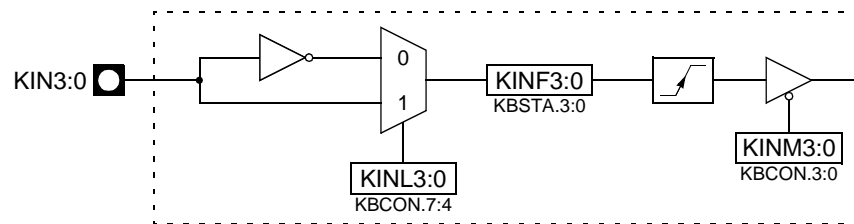
The keyboard inputs are considered as 4 independent interrupt sources sharing the same interrupt vector. An interrupt enable bit (EKB in IEN1 register) allows global enable or disable of the keyboard interrupt (see Figure 134). As detailed in Figure 135 each keyboard input has the capability to detect a programmable level according to KINL3:0 bit value in KBCON register. Level detection is then reported in interrupt flags KINF3:0 in KBSTA register.

A keyboard interrupt is requested each time one of the four flags is set, i.e. the input level matches the programmed one. Each of these four flags can be masked by software using KINM3:0 bits in KBCON register and is cleared by reading KBSTA register. This structure allows keyboard arrangement from 1 by n to 4 by n matrix and allow usage of KIN inputs for any other purposes.

**Figure 134.** Keyboard Interface Block Diagram



**Figure 135.** Keyboard Input Circuitry



### Power Reduction Mode

KIN3:0 inputs allow exit from idle and power-down modes as detailed in section “Power Management”, page 46. To enable this feature, KPDE bit in KBSTA register must be set to logic 1.

Due to the asynchronous keypad detection in power down mode (all clocks are stopped), exit may happen on parasitic key press. In this case, no key is detected and software must enter power down again.

## Registers

**Table 151.** KBCON Register

KBCON (S:A3h) – Keyboard Control Register

	7	6	5	4	3	2	1	0
	KINL3	KINL2	KINL1	KINL0	KINM3	KINM2	KINM1	KINM0

Bit Number	Bit Mnemonic	Description
7 - 4	KINL3:0	<b>Keyboard Input Level Bit</b> Set to enable a high level detection on the respective KIN3:0 input. Clear to enable a low level detection on the respective KIN3:0 input.
3 - 0	KINM3:0	<b>Keyboard Input Mask Bit</b> Set to prevent the respective KINF3:0 flag from generating a keyboard interrupt. Clear to allow the respective KINF3:0 flag to generate a keyboard interrupt.

Reset Value = 0000 1111b

**Table 152.** KBSTA Register

KBSTA (S:A4h) – Keyboard Control and Status Register

	7	6	5	4	3	2	1	0
	KPDE	-	-	-	KINF3	KINF2	KINF1	KINF0

Bit Number	Bit Mnemonic	Description
7	KPDE	<b>Keyboard Power Down Enable Bit</b> Set to enable exit of power down mode by the keyboard interrupt. Clear to disable exit of power down mode by the keyboard interrupt.
6 - 4	-	<b>Reserved</b> The value read from these bits is always 0. Do not set these bits.
3 - 0	KINF3:0	<b>Keyboard Input Interrupt Flag</b> Set by hardware when the respective KIN3:0 input detects a programmed level. Cleared when reading KBSTA.

Reset Value = 0000 0000b

## Electrical Characteristics

### Absolute Maximum Rating

Storage Temperature .....	-65 to +150°C	<b>*NOTICE:</b> Stressing the device beyond the “Absolute Maximum Ratings” may cause permanent damage. These are stress ratings only. Operation beyond the “operating conditions” is not recommended and extended exposure beyond the “Operating Conditions” may affect device reliability.
Voltage on any other Pin to $V_{SS}$ .....	-0.3 to +4.0 V	
$I_{OL}$ per I/O Pin .....	5 mA	
Power Dissipation .....	1 W	
Operating Conditions		
Ambient Temperature Under Bias.....	-40 to +85°C	
$V_{DD}$ .....	2.7 to 3.3V	

### DC Characteristics

#### Digital Logic

**Table 153.** Digital DC Characteristics  
 **$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$**

Symbol	Parameter	Min	Typ <sup>(1)</sup>	Max	Units	Test Conditions
$V_{IL}$	Input Low Voltage	-0.5		$0.2 \cdot V_{DD} - 0.1$	V	
$V_{IH1}^{(2)}$	Input High Voltage (except RST, X1)	$0.2 \cdot V_{DD} + 1.1$		$V_{DD}$	V	
$V_{IH2}$	Input High Voltage (RST, X1)	$0.7 \cdot V_{DD}$		$V_{DD} + 0.5$	V	
$V_{OL1}$	Output Low Voltage (except P0, ALE, MCMD, MDAT, MCLK, SCLK, DCLK, DSEL, DOUT)			0.45	V	$I_{OL} = 1.6$ mA
$V_{OL2}$	Output Low Voltage (P0, ALE, MCMD, MDAT, MCLK, SCLK, DCLK, DSEL, DOUT)			0.45	V	$I_{OL} = 3.2$ mA
$V_{OH1}$	Output High Voltage (P1, P2, P3, P4 and P5)	$V_{DD} - 0.7$			V	$I_{OH} = -30$ $\mu\text{A}$
$V_{OH2}$	Output High Voltage (P0, P2 address mode, ALE, MCMD, MDAT, MCLK, SCLK, DCLK, DSEL, DOUT, D+, D-)	$V_{DD} - 0.7$			V	$I_{OH} = -3.2$ mA
$I_{IL}$	Logical 0 Input Current (P1, P2, P3, P4 and P5)			-50	$\mu\text{A}$	$V_{in} = 0.45$ V
$I_{LI}$	Input Leakage Current (P0, ALE, MCMD, MDAT, MCLK, SCLK, DCLK, DSEL, DOUT)			10	$\mu\text{A}$	$0.45 < V_{IN} < V_{DD}$
$I_{TL}$	Logical 1 to 0 Transition Current (P1, P2, P3, P4 and P5)			-650	$\mu\text{A}$	$V_{in} = 2.0$ V
$R_{RST}$	Pull-Down Resistor	50	90	200	k $\Omega$	
$C_{IO}$	Pin Capacitance		10		pF	$T_A = 25^\circ\text{C}$
$V_{RET}$	$V_{DD}$ Data Retention Limit			1.8	V	

**Table 153.** Digital DC Characteristics  
**V<sub>DD</sub> = 2.7 to 3.3 V, T<sub>A</sub> = -40 to +85°C**

Symbol	Parameter	Min	Typ <sup>(1)</sup>	Max	Units	Test Conditions
I <sub>DD</sub>	AT89C51SND1C Operating Current		(3)	X1 / X2 mode 6.5 / 10.5 8 / 13.5 9.5 / 17	mA	V <sub>DD</sub> < 3.3 V 12 MHz 16 MHz 20 MHz
	AT83C51SND1C Operating Current		TBD	TBD	mA	V <sub>DD</sub> < 3.3 V 12 MHz 16 MHz 20 MHz
I <sub>DL</sub>	AT89C51SND1C Idle Mode Current		(3)	X1 / X2 mode 5.3 / 8.1 6.4 / 10.3 7.5 / 13	mA	V <sub>DD</sub> < 3.3 V 12 MHz 16 MHz 20 MHz
	AT83C51SND1C Idle Mode Current		TBD	TBD	mA	V <sub>DD</sub> < 3.3 V 12 MHz 16 MHz 20 MHz
I <sub>PD</sub>	AT89C51SND1C Power-Down Mode Current		20	500	μA	V <sub>RET</sub> < V <sub>DD</sub> < 3.3 V
	AT83C51SND1C Power-Down Mode Current		TBD	TBD	μA	V <sub>RET</sub> < V <sub>DD</sub> < 3.3 V
I <sub>FP</sub>	AT89C51SND1C Flash Programming Current			15	mA	V <sub>DD</sub> < 3.3 V

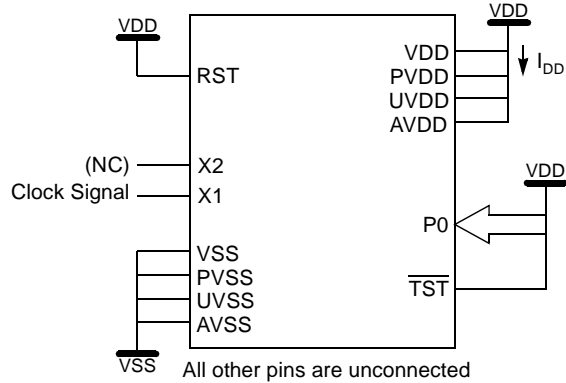
- Notes: 1. Typical values are obtained using V<sub>DD</sub> = 3 V and T<sub>A</sub> = 25°C. They are not tested and there is no guarantee on these values.  
2. Flash retention is guaranteed with the same formula for V<sub>DD</sub> min down to 0V.  
3. See Table 154 for typical consumption in player mode.

**Table 154.** Typical Reference Design AT89C51SND1C Power Consumption

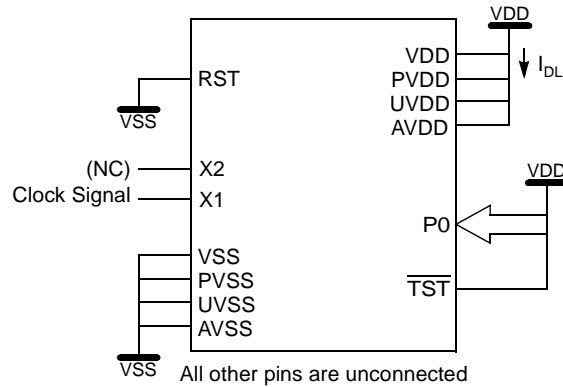
Player Mode	I <sub>DD</sub>	Test Conditions
Stop	10 mA	AT89C51SND1C at 16 MHz, X2 mode, V <sub>DD</sub> = 3 V No song playing
Playing	30 mA	AT89C51SND1C at 16 MHz, X2 mode, V <sub>DD</sub> = 3 V MP3 Song with F <sub>s</sub> = 44.1 KHz, at any bit rates (Variable Bit Rate)

$I_{DD}$ ,  $I_{DL}$  and  $I_{PD}$  Test Conditions

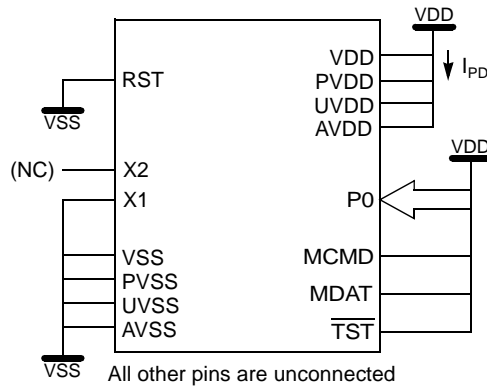
**Figure 136.**  $I_{DD}$  Test Condition, Active Mode



**Figure 137.**  $I_{DL}$  Test Condition, Idle Mode



**Figure 138.**  $I_{PD}$  Test Condition, Power-Down Mode



## A to D Converter

**Table 155.** A to D Converter DC Characteristics

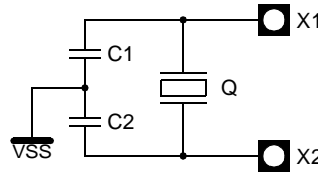
$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
$AV_{DD}$	Analog Supply Voltage	2.7		3.3	V	
$AI_{DD}$	Analog Operating Supply Current			600	$\mu\text{A}$	$AV_{DD} = 3.3\text{V}$ $AIN1:0 = 0$ to $AV_{DD}$ $ADEN = 1$
$AI_{PD}$	Analog Standby Current			2	$\mu\text{A}$	$AV_{DD} = 3.3\text{V}$ $ADEN = 0$ or $PD = 1$
$AV_{IN}$	Analog Input Voltage	$AV_{SS}$		$AV_{DD}$	V	
$AV_{REF}$	Reference Voltage $A_{REFN}$ $A_{REFP}$	$AV_{SS}$ 2.4		$AV_{DD}$	V	
$R_{REF}$	AREF Input Resistance	10		30	$\text{K}\Omega$	$T_A = 25^\circ\text{C}$
$C_{IA}$	Analog Input capacitance			10	$\text{pF}$	$T_A = 25^\circ\text{C}$

## Oscillator & Crystal

### Schematic

**Figure 139.** Crystal Connection



Note: For operation with most standard crystals, no external components are needed on X1 and X2. It may be necessary to add external capacitors on X1 and X2 to ground in special cases (max 10 pF). X1 and X2 may not be used to drive other circuits.

### Parameters

**Table 156.** Oscillator & Crystal Characteristics

$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$

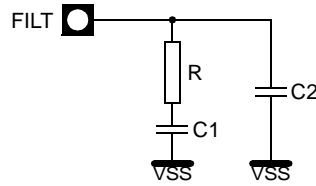
Symbol	Parameter	Min	Typ	Max	Unit
$C_{X1}$	Internal Capacitance (X1 - VSS)		10		$\text{pF}$
$C_{X2}$	Internal Capacitance (X2 - VSS)		10		$\text{pF}$
$C_L$	Equivalent Load Capacitance (X1 - X2)		5		$\text{pF}$
DL	Drive Level			50	$\mu\text{W}$
F	Crystal Frequency			20	MHz
RS	Crystal Series Resistance			40	$\Omega$
CS	Crystal Shunt Capacitance			6	$\text{pF}$



Phase Lock Loop

Schematic

Figure 140. PLL Filter Connection



Parameters

Table 157. PLL Filter Characteristics

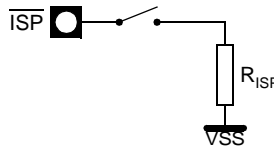
$V_{DD} = 2.7 \text{ to } 3.3 \text{ V}$ ,  $T_A = -40 \text{ to } +85^\circ\text{C}$

Symbol	Parameter	Min	Typ	Max	Unit
R	Filter Resistor		100		$\Omega$
C1	Filter Capacitance 1		10		nF
C2	Filter Capacitance 2		2.2		nF

In System Programming

Schematic

Figure 141. ISP Pull-Down Connection



Parameters

Table 158. ISP Pull-Down Characteristics

$V_{DD} = 2.7 \text{ to } 3.3 \text{ V}$ ,  $T_A = -40 \text{ to } +85^\circ\text{C}$

Symbol	Parameter	Min	Typ	Max	Unit
$R_{ISP}$	ISP Pull-Down Resistor		2.2		$K\Omega$

## AC Characteristics

### External 8-bit Bus Cycles

#### Definition of Symbols

**Table 159.** External 8-bit Bus Cycles Timing Symbol Definitions

Signals	
A	Address
D	Data In
L	ALE
Q	Data Out
R	$\overline{RD}$
W	$\overline{WR}$

Conditions	
H	High
L	Low
V	Valid
X	No Longer Valid
Z	Floating

#### Timings

Test conditions: capacitive load on all pins= 50 pF.

**Table 160.** External 8-bit Bus Cycle - Data Read AC Timings

$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$

Symbol	Parameter	Variable Clock Standard Mode		Variable Clock X2 Mode		Unit
		Min	Max	Min	Max	
$T_{CLCL}$	Clock Period	50		50		ns
$T_{LHLL}$	ALE Pulse Width	$2 \cdot T_{CLCL} - 15$		$T_{CLCL} - 15$		ns
$T_{AVLL}$	Address Valid to ALE Low	$T_{CLCL} - 20$		$0.5 \cdot T_{CLCL} - 20$		ns
$T_{LLAX}$	Address hold after ALE Low	$T_{CLCL} - 20$		$0.5 \cdot T_{CLCL} - 20$		ns
$T_{LLRL}$	ALE Low to $\overline{RD}$ Low	$3 \cdot T_{CLCL} - 30$		$1.5 \cdot T_{CLCL} - 30$		ns
$T_{RLRH}$	$\overline{RD}$ Pulse Width	$6 \cdot T_{CLCL} - 25$		$3 \cdot T_{CLCL} - 25$		ns
$T_{RHLH}$	$\overline{RD}$ high to ALE High	$T_{CLCL} - 20$	$T_{CLCL} + 20$	$0.5 \cdot T_{CLCL} - 20$	$0.5 \cdot T_{CLCL} + 20$	ns
$T_{AVDV}$	Address Valid to Valid Data In		$9 \cdot T_{CLCL} - 65$		$4.5 \cdot T_{CLCL} - 65$	ns
$T_{AVRL}$	Address Valid to $\overline{RD}$ Low	$4 \cdot T_{CLCL} - 30$		$2 \cdot T_{CLCL} - 30$		ns
$T_{RLDV}$	$\overline{RD}$ Low to Valid Data		$5 \cdot T_{CLCL} - 30$		$2.5 \cdot T_{CLCL} - 30$	ns
$T_{RLAZ}$	$\overline{RD}$ Low to Address Float		0		0	ns
$T_{RHDX}$	Data Hold After $\overline{RD}$ High	0		0		ns
$T_{RHDZ}$	Instruction Float After $\overline{RD}$ High		$2 \cdot T_{CLCL} - 25$		$T_{CLCL} - 25$	ns

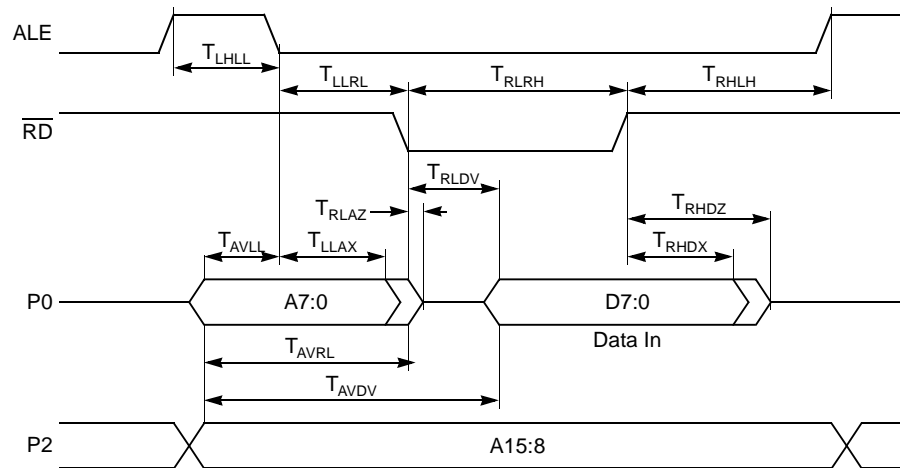
**Table 161.** External 8-bit Bus Cycle - Data Write AC Timings

$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$

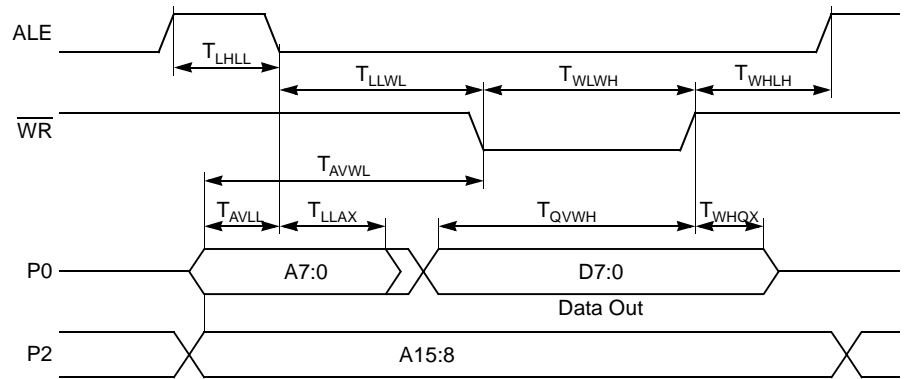
Symbol	Parameter	Variable Clock Standard Mode		Variable Clock X2 Mode		Unit
		Min	Max	Min	Max	
$T_{CLCL}$	Clock Period	50		50		ns
$T_{LHLL}$	ALE Pulse Width	$2 \cdot T_{CLCL} - 15$		$T_{CLCL} - 15$		ns
$T_{AVLL}$	Address Valid to ALE Low	$T_{CLCL} - 20$		$0.5 \cdot T_{CLCL} - 20$		ns
$T_{LLAX}$	Address hold after ALE Low	$T_{CLCL} - 20$		$0.5 \cdot T_{CLCL} - 20$		ns
$T_{LLWL}$	ALE Low to $\overline{WR}$ Low	$3 \cdot T_{CLCL} - 30$		$1.5 \cdot T_{CLCL} - 30$		ns
$T_{WLWH}$	$\overline{WR}$ Pulse Width	$6 \cdot T_{CLCL} - 25$		$3 \cdot T_{CLCL} - 25$		ns
$T_{WHLH}$	$\overline{WR}$ High to ALE High	$T_{CLCL} - 20$	$T_{CLCL} + 20$	$0.5 \cdot T_{CLCL} - 20$	$0.5 \cdot T_{CLCL} + 20$	ns
$T_{AVWL}$	Address Valid to $\overline{WR}$ Low	$4 \cdot T_{CLCL} - 30$		$2 \cdot T_{CLCL} - 30$		ns
$T_{QVWH}$	Data Valid to $\overline{WR}$ High	$7 \cdot T_{CLCL} - 20$		$3.5 \cdot T_{CLCL} - 20$		ns
$T_{WHQX}$	Data Hold after $\overline{WR}$ High	$T_{CLCL} - 15$		$0.5 \cdot T_{CLCL} - 15$		ns

Waveforms

**Figure 142.** External 8-bit Bus Cycle - Data Read Waveforms



**Figure 143.** External 8-bit Bus Cycle - Data Write Waveforms



**External IDE 16-bit Bus Cycles**

*Definition of Symbols*

**Table 162.** External IDE 16-bit Bus Cycles Timing Symbol Definitions

Signals	
A	Address
D	Data In
L	ALE
Q	Data Out
R	$\overline{RD}$
W	$\overline{WR}$

Conditions	
H	High
L	Low
V	Valid
X	No Longer Valid
Z	Floating

## Timings

Test conditions: capacitive load on all pins= 50 pF.

**Table 163.** External IDE 16-bit Bus Cycle - Data Read AC Timings

$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$

Symbol	Parameter	Variable Clock Standard Mode		Variable Clock X2 Mode		Unit
		Min	Max	Min	Max	
$T_{CLCL}$	Clock Period	50		50		ns
$T_{LHLL}$	ALE Pulse Width	$2 \cdot T_{CLCL} - 15$		$T_{CLCL} - 15$		ns
$T_{AVLL}$	Address Valid to ALE Low	$T_{CLCL} - 20$		$0.5 \cdot T_{CLCL} - 20$		ns
$T_{LLAX}$	Address hold after ALE Low	$T_{CLCL} - 20$		$0.5 \cdot T_{CLCL} - 20$		ns
$T_{LLRL}$	ALE Low to $\overline{RD}$ Low	$3 \cdot T_{CLCL} - 30$		$1.5 \cdot T_{CLCL} - 30$		ns
$T_{RLRH}$	$\overline{RD}$ Pulse Width	$6 \cdot T_{CLCL} - 25$		$3 \cdot T_{CLCL} - 25$		ns
$T_{RHLH}$	$\overline{RD}$ high to ALE High	$T_{CLCL} - 20$	$T_{CLCL} + 20$	$0.5 \cdot T_{CLCL} - 20$	$0.5 \cdot T_{CLCL} + 20$	ns
$T_{AVDV}$	Address Valid to Valid Data In		$9 \cdot T_{CLCL} - 65$		$4.5 \cdot T_{CLCL} - 65$	ns
$T_{AVRL}$	Address Valid to $\overline{RD}$ Low	$4 \cdot T_{CLCL} - 30$		$2 \cdot T_{CLCL} - 30$		ns
$T_{RLDV}$	$\overline{RD}$ Low to Valid Data		$5 \cdot T_{CLCL} - 30$		$2.5 \cdot T_{CLCL} - 30$	ns
$T_{RLAZ}$	$\overline{RD}$ Low to Address Float		0		0	ns
$T_{RHDX}$	Data Hold After $\overline{RD}$ High	0		0		ns
$T_{RHDZ}$	Instruction Float After $\overline{RD}$ High		$2 \cdot T_{CLCL} - 25$		$T_{CLCL} - 25$	ns

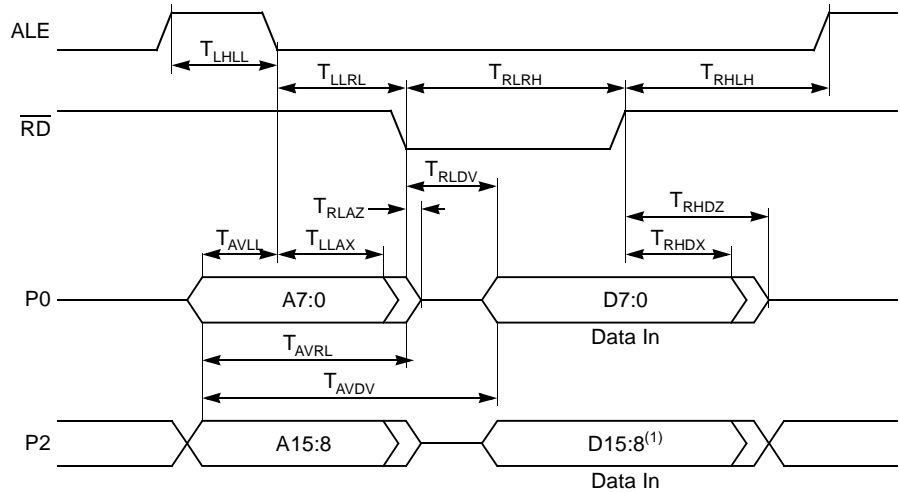
**Table 164.** External IDE 16-bit Bus Cycle - Data Write AC Timings

$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$

Symbol	Parameter	Variable Clock Standard Mode		Variable Clock X2 Mode		Unit
		Min	Max	Min	Max	
$T_{CLCL}$	Clock Period	50		50		ns
$T_{LHLL}$	ALE Pulse Width	$2 \cdot T_{CLCL} - 15$		$T_{CLCL} - 15$		ns
$T_{AVLL}$	Address Valid to ALE Low	$T_{CLCL} - 20$		$0.5 \cdot T_{CLCL} - 20$		ns
$T_{LLAX}$	Address hold after ALE Low	$T_{CLCL} - 20$		$0.5 \cdot T_{CLCL} - 20$		ns
$T_{LLWL}$	ALE Low to $\overline{WR}$ Low	$3 \cdot T_{CLCL} - 30$		$1.5 \cdot T_{CLCL} - 30$		ns
$T_{WLWH}$	$\overline{WR}$ Pulse Width	$6 \cdot T_{CLCL} - 25$		$3 \cdot T_{CLCL} - 25$		ns
$T_{WHLH}$	$\overline{WR}$ High to ALE High	$T_{CLCL} - 20$	$T_{CLCL} + 20$	$0.5 \cdot T_{CLCL} - 20$	$0.5 \cdot T_{CLCL} + 20$	ns
$T_{AVWL}$	Address Valid to $\overline{WR}$ Low	$4 \cdot T_{CLCL} - 30$		$2 \cdot T_{CLCL} - 30$		ns
$T_{QVWH}$	Data Valid to $\overline{WR}$ High	$7 \cdot T_{CLCL} - 20$		$3.5 \cdot T_{CLCL} - 20$		ns
$T_{WHQX}$	Data Hold after $\overline{WR}$ High	$T_{CLCL} - 15$		$0.5 \cdot T_{CLCL} - 15$		ns

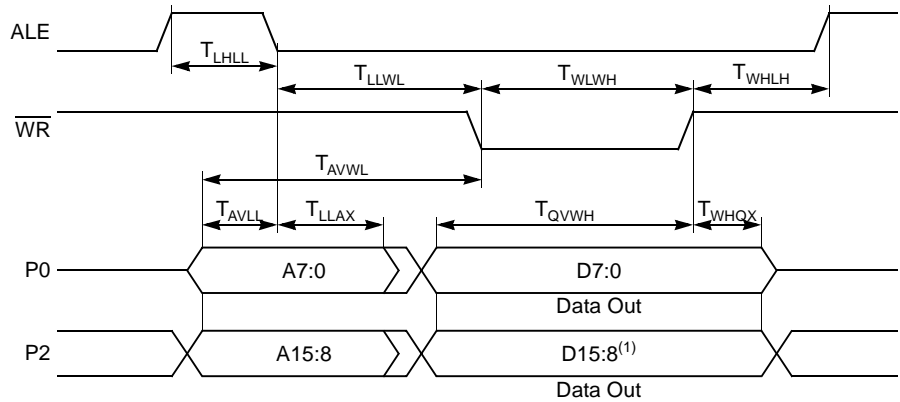
Waveforms

**Figure 144.** External IDE 16-bit Bus Cycle - Data Read Waveforms



Note: 1. D15:8 is written in DAT16H SFR.

**Figure 145.** External IDE 16-bit Bus Cycle - Data Write Waveforms



Note: 1. D15:8 is the content of DAT16H SFR.

**SPI Interface**

Definition of Symbols

**Table 165.** SPI Interface Timing Symbol Definitions

Signals	
C	Clock
I	Data In
O	Data Out

Conditions	
H	High
L	Low
V	Valid
X	No Longer Valid
Z	Floating

## Timings

Test conditions: capacitive load on all pins= 50 pF.

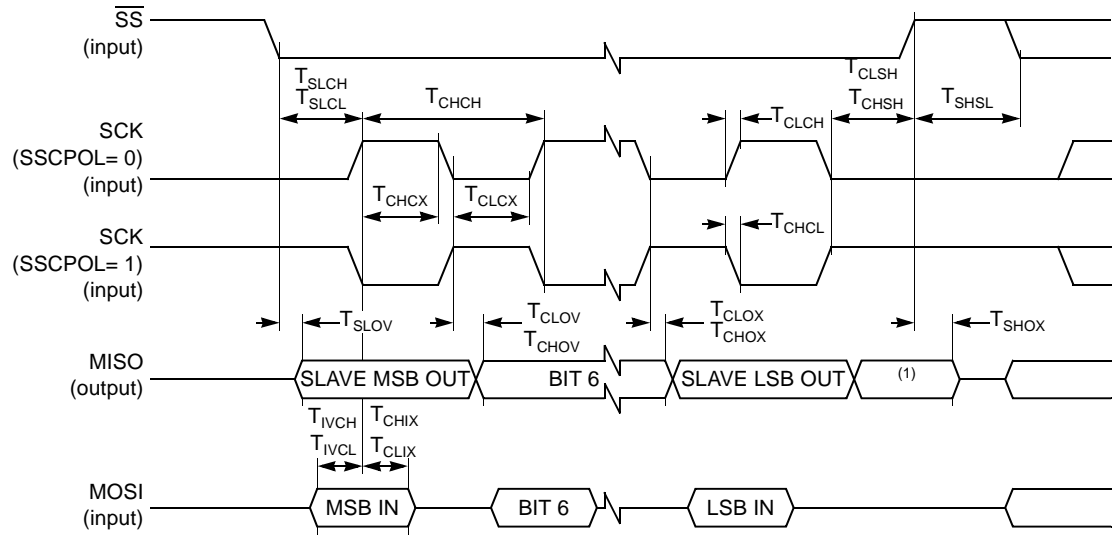
**Table 166.** SPI Interface Master AC Timing

$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$

Symbol	Parameter	Min	Max	Unit
<b>Slave Mode</b>				
$T_{CHCH}$	Clock Period	2		$T_{PER}$
$T_{CHCX}$	Clock High Time	0.8		$T_{PER}$
$T_{CLCX}$	Clock Low Time	0.8		$T_{PER}$
$T_{SLCH}, T_{SLCL}$	$\overline{SS}$ Low to Clock edge	100		ns
$T_{IVCL}, T_{IVCH}$	Input Data Valid to Clock Edge	40		ns
$T_{CLIX}, T_{CHIX}$	Input Data Hold after Clock Edge	40		ns
$T_{CLOV}, T_{CHOV}$	Output Data Valid after Clock Edge		40	ns
$T_{CLOX}, T_{CHOX}$	Output Data Hold Time after Clock Edge	0		ns
$T_{CLSH}, T_{CHSH}$	$\overline{SS}$ High after Clock Edge	0		ns
$T_{SLOV}$	$\overline{SS}$ Low to Output Data Valid		50	ns
$T_{SHOX}$	Output Data Hold after $\overline{SS}$ High		50	ns
$T_{SHSL}$	$\overline{SS}$ High to $\overline{SS}$ Low	(1)		
$T_{ILIH}$	Input Rise Time		2	$\mu\text{s}$
$T_{IHIL}$	Input Fall Time		2	$\mu\text{s}$
$T_{OLOH}$	Output Rise time		100	ns
$T_{OHOL}$	Output Fall Time		100	ns
<b>Master Mode</b>				
$T_{CHCH}$	Clock Period	2		$T_{PER}$
$T_{CHCX}$	Clock High Time	0.8		$T_{PER}$
$T_{CLCX}$	Clock Low Time	0.8		$T_{PER}$
$T_{IVCL}, T_{IVCH}$	Input Data Valid to Clock Edge	20		ns
$T_{CLIX}, T_{CHIX}$	Input Data Hold after Clock Edge	20		ns
$T_{CLOV}, T_{CHOV}$	Output Data Valid after Clock Edge		40	ns
$T_{CLOX}, T_{CHOX}$	Output Data Hold Time after Clock Edge	0		ns
$T_{ILIH}$	Input Data Rise Time		2	$\mu\text{s}$
$T_{IHIL}$	Input Data Fall Time		2	$\mu\text{s}$
$T_{OLOH}$	Output Data Rise time		50	ns
$T_{OHOL}$	Output Data Fall Time		50	ns

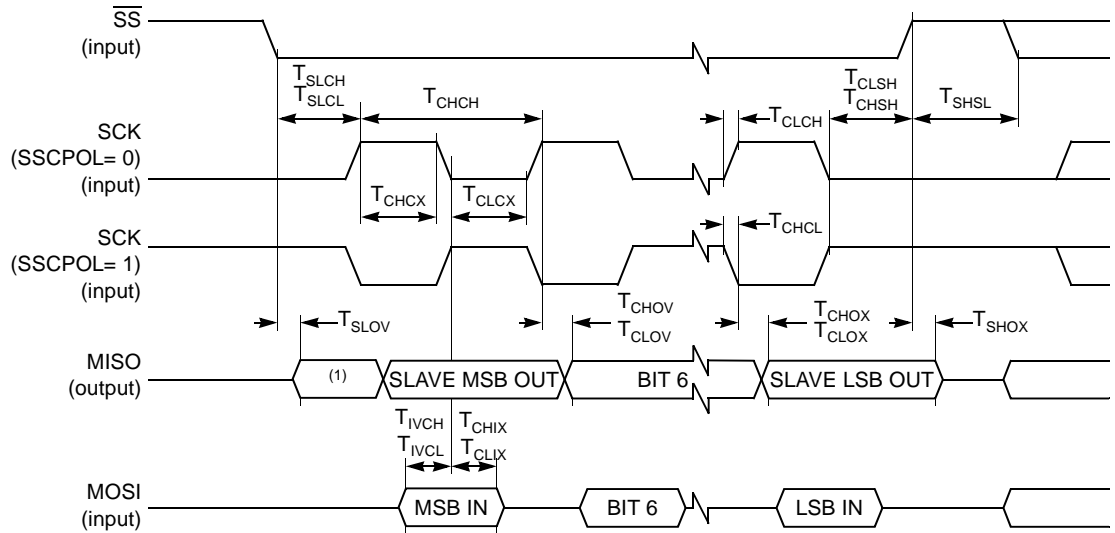
Note: 1. Value of this parameter depends on software.

**Figure 146. SPI Slave Waveforms (SSCPHA= 0)**



Note: 1. Not Defined but generally the MSB of the character which has just been received.

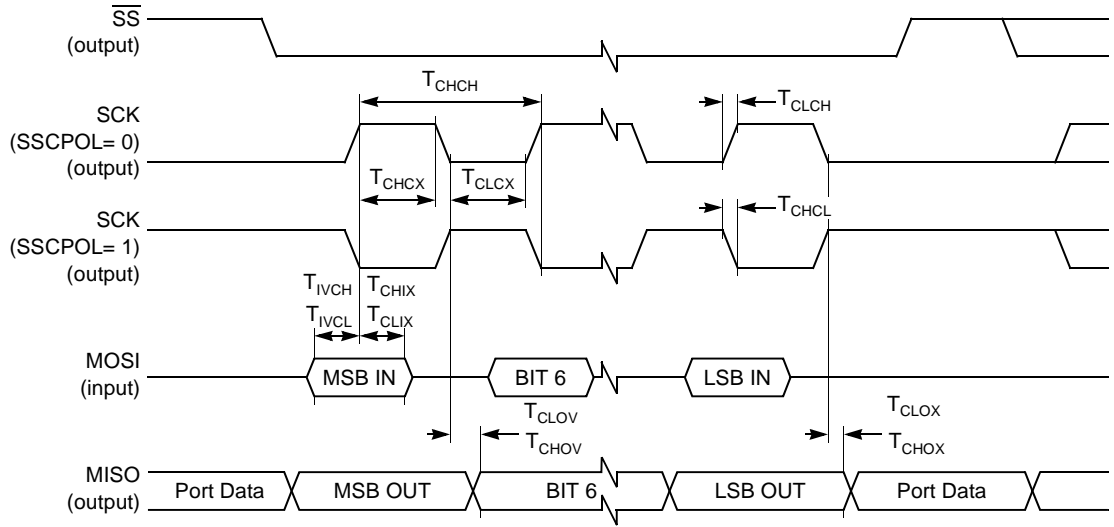
**Figure 147. SPI Slave Waveforms (SSCPHA= 1)**



Note: 1. Not Defined but generally the LSB of the character which has just been received.

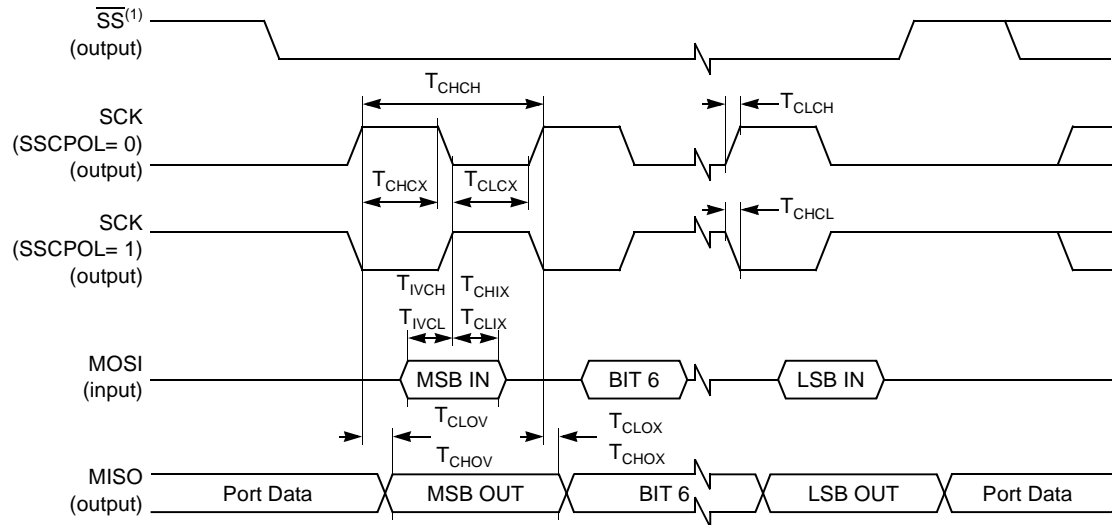


Figure 148. SPI Master Waveforms (SSCPHA= 0)



Note: 1.  $\overline{SS}$  handled by software using general purpose port pin.

Figure 149. SPI Master Waveforms (SSCPHA= 1)



Note: 1.  $\overline{SS}$  handled by software using general purpose port pin.

## Two-wire Interface

### Timings

**Table 167.** TWI Interface AC Timing

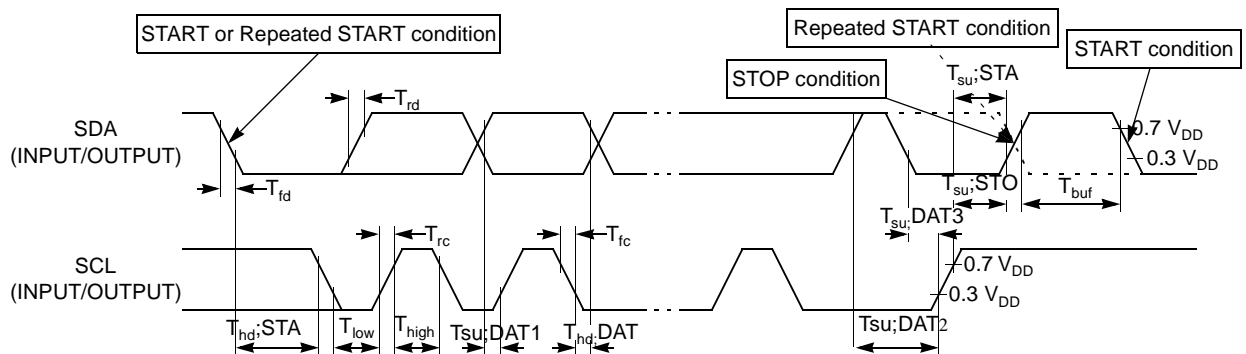
$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$

Symbol	Parameter	INPUT Min Max	OUTPUT Min Max
$T_{HD}; STA$	Start condition hold time	$14 \cdot T_{CLCL}^{(4)}$	$4.0 \mu\text{s}^{(1)}$
$T_{LOW}$	SCL low time	$16 \cdot T_{CLCL}^{(4)}$	$4.7 \mu\text{s}^{(1)}$
$T_{HIGH}$	SCL high time	$14 \cdot T_{CLCL}^{(4)}$	$4.0 \mu\text{s}^{(1)}$
$T_{RC}$	SCL rise time	$1 \mu\text{s}$	.. <sup>(2)</sup>
$T_{FC}$	SCL fall time	$0.3 \mu\text{s}$	$0.3 \mu\text{s}^{(3)}$
$T_{SU}; DAT1$	Data set-up time	$250 \text{ ns}$	$20 \cdot T_{CLCL}^{(4)} - T_{RD}$
$T_{SU}; DAT2$	SDA set-up time (before repeated START condition)	$250 \text{ ns}$	$1 \mu\text{s}^{(1)}$
$T_{SU}; DAT3$	SDA set-up time (before STOP condition)	$250 \text{ ns}$	$8 \cdot T_{CLCL}^{(4)}$
$T_{HD}; DAT$	Data hold time	$0 \text{ ns}$	$8 \cdot T_{CLCL}^{(4)} - T_{FC}$
$T_{SU}; STA$	Repeated START set-up time	$14 \cdot T_{CLCL}^{(4)}$	$4.7 \mu\text{s}^{(1)}$
$T_{SU}; STO$	STOP condition set-up time	$14 \cdot T_{CLCL}^{(4)}$	$4.0 \mu\text{s}^{(1)}$
$T_{BUF}$	Bus free time	$14 \cdot T_{CLCL}^{(4)}$	$4.7 \mu\text{s}^{(1)}$
$T_{RD}$	SDA rise time	$1 \mu\text{s}$	.. <sup>(2)</sup>
$T_{FD}$	SDA fall time	$0.3 \mu\text{s}$	$0.3 \mu\text{s}^{(3)}$

- Notes:
1. At 100 kbit/s. At other bit-rates this value is inversely proportional to the bit-rate of 100 kbit/s.
  2. Determined by the external bus-line capacitance and the external bus-line pull-up resistor, this must be  $< 1 \mu\text{s}$ .
  3. Spikes on the SDA and SCL lines with a duration of less than  $3 \cdot T_{CLCL}$  will be filtered out. Maximum capacitance on bus-lines SDA and SCL = 400 pF.
  4.  $T_{CLCL} = T_{OSC}$  = one oscillator clock period.

### Waveforms

**Figure 150.** Two Wire Waveforms



## MMC Interface

### Definition of symbols

**Table 168.** MMC Interface Timing Symbol Definitions

Signals	
C	Clock
D	Data In
O	Data Out

Conditions	
H	High
L	Low
V	Valid
X	No Longer Valid

### Timings

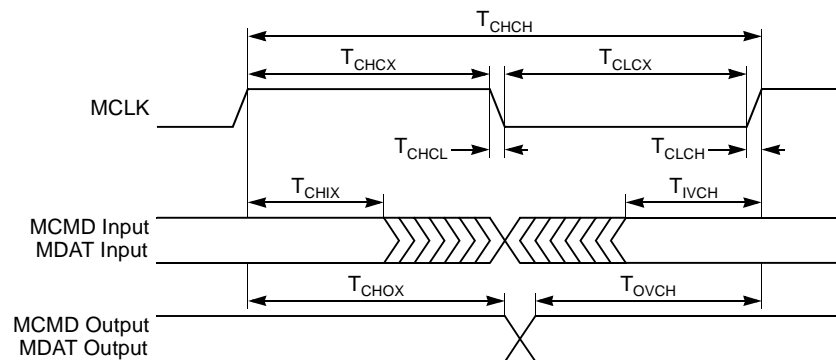
**Table 169.** MMC Interface AC timings

$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$ ,  $CL \leq 100\text{pF}$  (10 cards)

Symbol	Parameter	Min	Max	Unit
$T_{CHCH}$	Clock Period	50		ns
$T_{CHCX}$	Clock High Time	10		ns
$T_{CLCX}$	Clock Low Time	10		ns
$T_{CLCH}$	Clock Rise Time		10	ns
$T_{CHCL}$	Clock Fall Time		10	ns
$T_{DVCH}$	Input Data Valid to Clock High	3		ns
$T_{CHDX}$	Input Data Hold after Clock High	3		ns
$T_{CHOX}$	Output Data Hold after Clock High	5		ns
$T_{OVCH}$	Output Data Valid to Clock High	5		ns

### Waveforms

**Figure 151.** MMC Input-Output Waveforms



## Audio Interface

### Definition of symbols

**Table 170.** Audio Interface Timing Symbol Definitions

Signals	
C	Clock
O	Data Out
S	Data Select

Conditions	
H	High
L	Low
V	Valid
X	No Longer Valid

### Timings

**Table 171.** Audio Interface AC timings

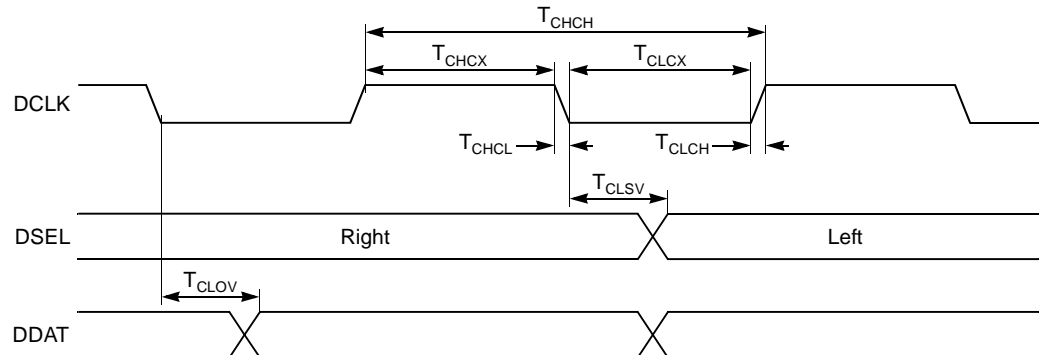
$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$ ,  $CL \leq 30\text{pF}$

Symbol	Parameter	Min	Max	Unit
$T_{CHCH}$	Clock Period		325.5 <sup>(1)</sup>	ns
$T_{CHCX}$	Clock High Time	30		ns
$T_{CLCX}$	Clock Low Time	30		ns
$T_{CLCH}$	Clock Rise Time		10	ns
$T_{CHCL}$	Clock Fall Time		10	ns
$T_{CLSV}$	Clock Low to Select Valid		10	ns
$T_{CLOV}$	Clock Low to Data Valid		10	ns

Note: 1. 32-bit format with  $F_s = 48$  KHz.

### Waveforms

**Figure 152.** Audio Interface Waveforms



## Analog to Digital Converter

### Definition of symbols

**Table 172.** Analog to Digital Converter Timing Symbol Definitions

Signals		Conditions	
C	Clock	H	High
E	Enable (ADEN bit)	L	Low
S	Start Conversion (ADSST bit)		

### Characteristics

**Table 173.** Analog to Digital Converter AC Characteristics

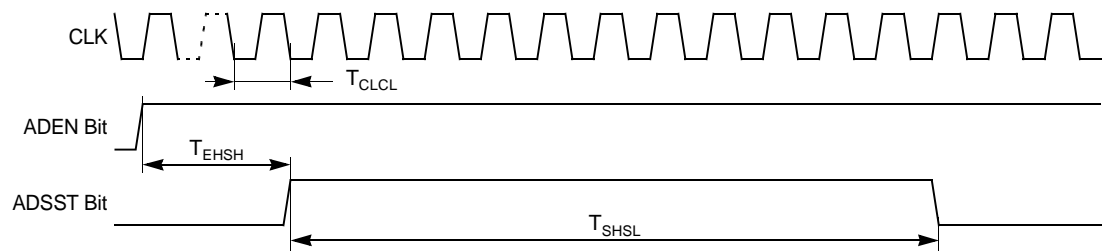
$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$

Symbol	Parameter	Min	Max	Unit
$T_{CLCL}$	Clock Period	4		$\mu\text{s}$
$T_{EHS}$	Start-up Time		4	$\mu\text{s}$
$T_{SHSL}$	Conversion Time		$11 \cdot T_{CLCL}$	$\mu\text{s}$
DLe	Differential non-linearity error <sup>(1)(2)</sup>		1	LSB
ILe	Integral non-linearity error <sup>(1)(3)</sup>		2	LSB
OSe	Offset error <sup>(1)(4)</sup>		4	LSB
Ge	Gain error <sup>(1)(5)</sup>		4	LSB

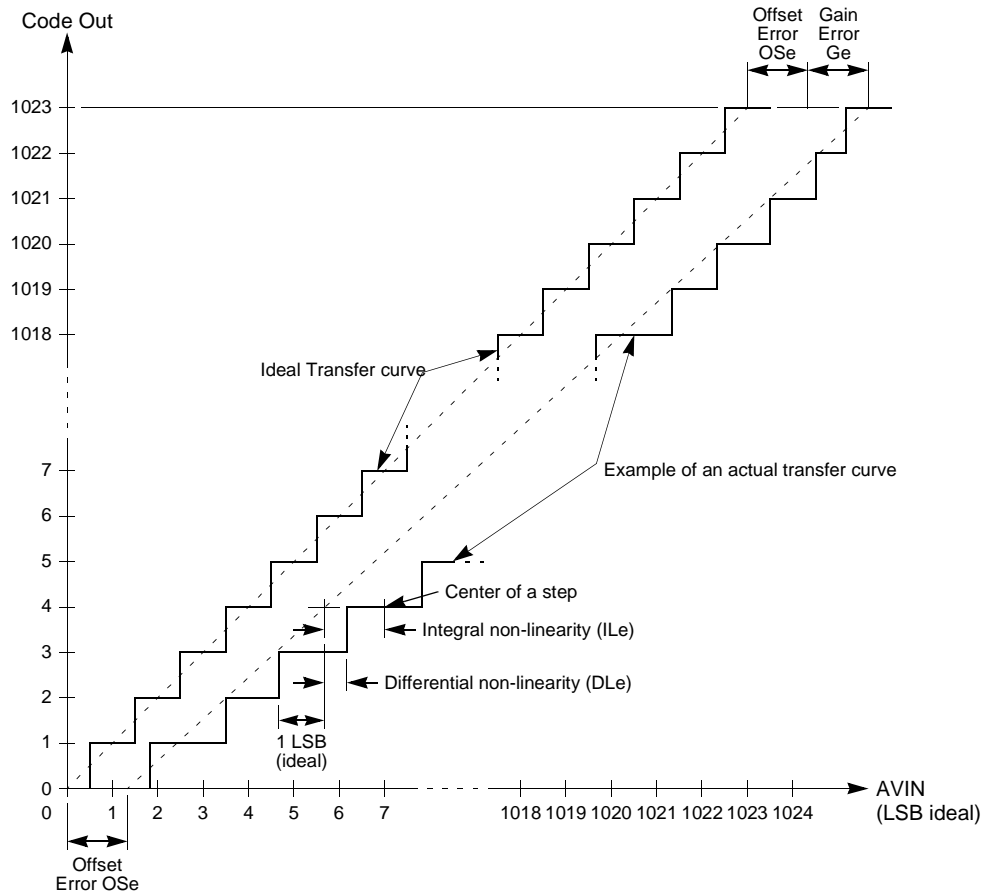
- Notes:
- $AV_{DD} = AV_{REFP} = 3.0$  V,  $AV_{SS} = AV_{REFN} = 0$  V. ADC is monotonic with no missing code.
  - The differential non-linearity is the difference between the actual step width and the ideal step width (see Figure 154).
  - The integral non-linearity is the peak difference between the center of the actual step and the ideal transfer curve after appropriate adjustment of gain and offset errors (see Figure 154).
  - The offset error is the absolute difference between the straight line which fits the actual transfer curve (after removing of gain error), and the straight line which fits the ideal transfer curve (see Figure 154).
  - The gain error is the relative difference in percent between the straight line which fits the actual transfer curve (after removing of offset error), and the straight line which fits the ideal transfer curve (see Figure 154).

### Waveforms

**Figure 153.** Analog to Digital Converter Internal Waveforms



**Figure 154.** Analog to Digital Converter Characteristics



## Flash Memory

### Definition of symbols

**Table 174.** Flash Memory Timing Symbol Definitions

Signals	
S	$\overline{\text{ISP}}$
R	RST
B	FBUSY flag

Conditions	
L	Low
V	Valid
X	No Longer Valid

### Timings

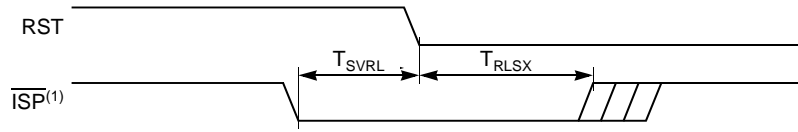
**Table 175.** Flash Memory AC Timing

$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$

Symbol	Parameter	Min	Typ	Max	Unit
$T_{SVRL}$	Input $\overline{\text{ISP}}$ Valid to RST Edge	50			ns
$T_{RLSX}$	Input $\overline{\text{ISP}}$ Hold after RST Edge	50			ns
$T_{BHBL}$	FLASH Internal Busy (Programming) Time		10		ms
$N_{FCY}$	Number of Flash Write Cycles	100K			Cycle
$T_{FDR}$	Flash Data Retention Time	10			Years

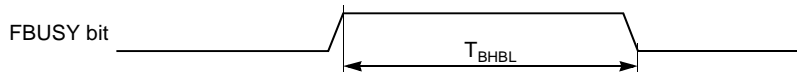
## Waveforms

**Figure 155.** FLASH Memory - ISP Waveforms



Note: 1.  $\overline{\text{ISP}}$  must be driven through a pull-down resistor (see Section "In System Programming", page 185).

**Figure 156.** FLASH Memory - Internal Busy Waveforms



## External Clock Drive and Logic Level References

### Definition of symbols

**Table 176.** External Clock Timing Symbol Definitions

Signals	
C	Clock

Conditions	
H	High
L	Low
X	No Longer Valid

### Timings

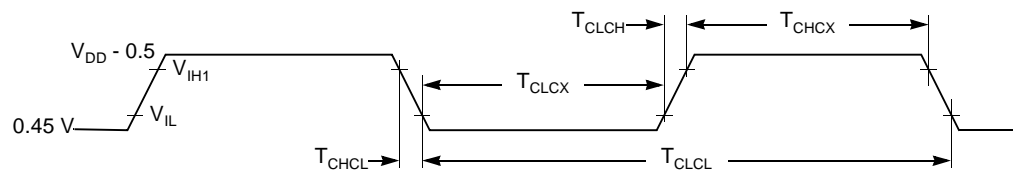
**Table 177.** External Clock AC Timings

$V_{DD} = 2.7$  to  $3.3$  V,  $T_A = -40$  to  $+85^\circ\text{C}$

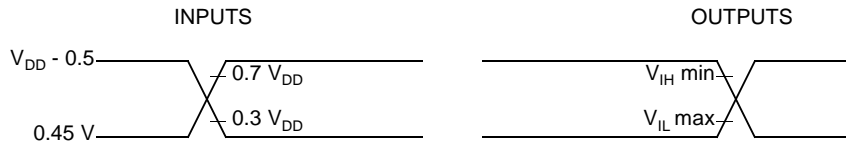
Symbol	Parameter	Min	Max	Unit
$T_{CLCL}$	Clock Period	50		ns
$T_{CHCX}$	High Time	10		ns
$T_{CLCX}$	Low Time	10		ns
$T_{CLCH}$	Rise Time	3		ns
$T_{CHCL}$	Fall Time	3		ns
$T_{CR}$	Cyclic Ratio in X2 mode	40	60	%

## Waveforms

**Figure 157.** External Clock Waveform

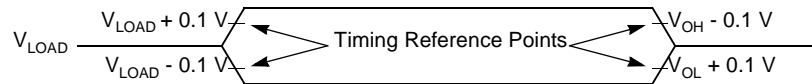


**Figure 158.** AC Testing Input/Output Waveforms



- Note:
1. During AC testing, all inputs are driven at  $V_{DD} - 0.5 V$  for a logic 1 and  $0.45 V$  for a logic 0.
  2. Timing measurements are made on all outputs at  $V_{IH} \text{ min}$  for a logic 1 and  $V_{IL} \text{ max}$  for a logic 0.

**Figure 159.** Float Waveforms



- Note:
- For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs and begins to float when a 100 mV change from the loading  $V_{OH}/V_{OL}$  level occurs with  $I_{OL}/I_{OH} = \pm 20 \text{ mA}$ .



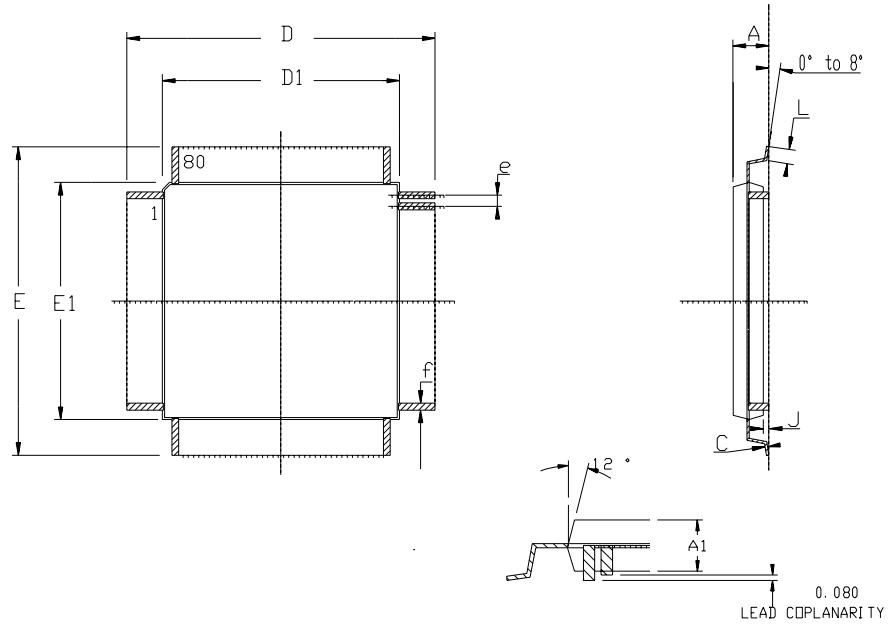
## Ordering Information

Part Number	Memory Size	Supply Voltage	Temperature Range	Max Frequency	Package <sup>(2)</sup>	Packing
AT89C51SND1C-ROTIL	64K Flash	3V	Industrial	40 MHz	TQFP80	Tray
AT89C51SND1C-7HTIL	64K Flash	3V	Industrial	40 MHz	BGA81	Tray
AT83SND1Cxxx <sup>(1)</sup> -ROTIL	64K ROM	3V	Industrial	40 MHz	TQFP80	Tray
AT83SND1Cxxx <sup>(1)</sup> -7HTIL	64K ROM	3V	Industrial	40 MHz	BGA81	Tray
AT80SND1C-ROTIL	ROMless	3V	Industrial	40 MHz	TQFP80	Tray
AT80SND1C-7HTIL	ROMless	3V	Industrial	40 MHz	BGA81	Tray

- Notes: 1. Refers to ROM code.  
 2. PLCC84 package only available for development board.

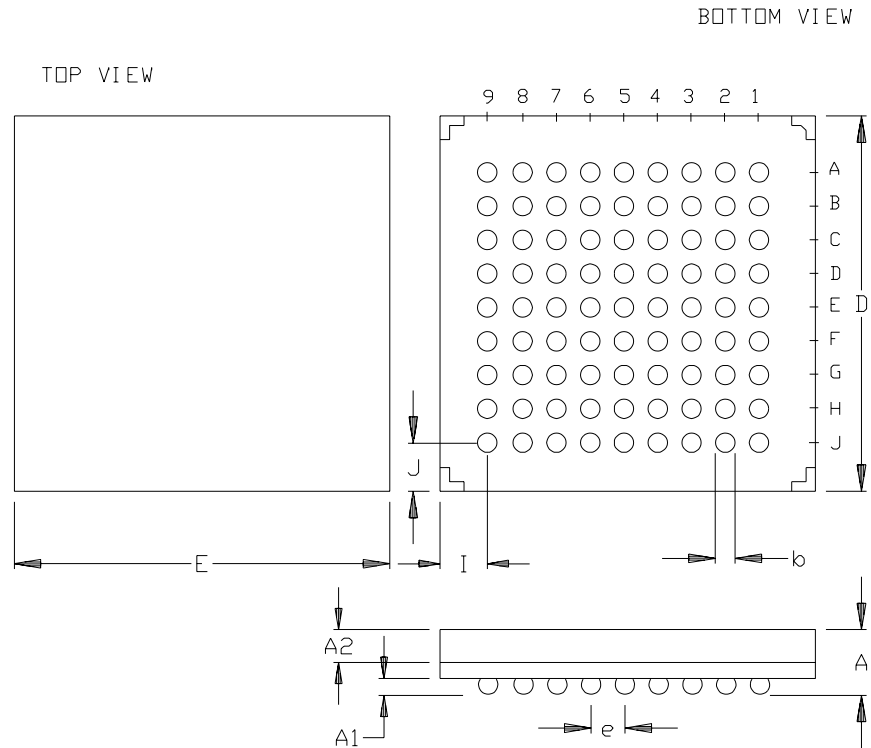
# Package Information

## TQFP80



	MM		INCH	
	Min	Max	Min	Max
A	1.40	1.60	.055	.063
A1	1.35	1.45	.053	.057
C	0.17 BSC		.007 BSC	
D	15.80	16.20	.622	.638
D1	13.90	14.10	.547	.555
E	15.80	16.20	.622	.638
E1	13.90	14.10	.547	.555
J	0.05	0.15	.002	.006
L	0.45	0.75	.018	.030
e	0.65 BSC		.0256 BSC	
f	0.30 BSC		.012 BSC	

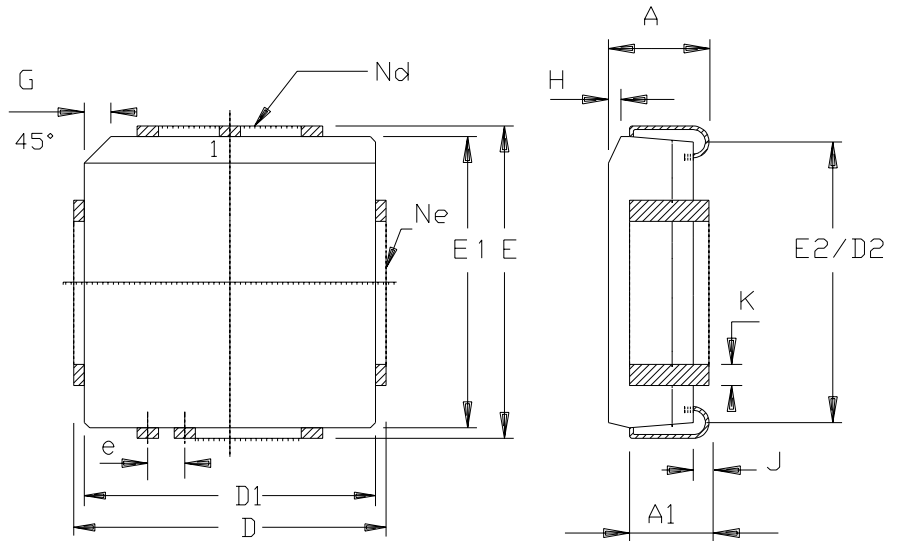
## BGA81



	mm		
	MIN	NOM	MAX
A	1.30	1.40	1.50
A1	0.31	0.36	0.41
A2	0.65	0.70	0.75
e	0.80 BSC		
b	0.46 TYP		
E/D	8.95	9.00	9.05
I/J	1.30 REF		

Compliant JEDEC MO-205

PLCC84



	MM		INCH	
A	4.20	5.08	.165	.200
A1	2.29	3.30	.090	.130
D	30.10	30.35	1.185	1.195
D1	29.21	29.41	1.150	1.158
D2	27.69	28.70	1.090	1.130
E	30.10	30.35	1.185	1.195
E1	29.21	29.41	1.150	1.158
E2	27.69	28.70	1.090	1.130
e	1.27	BSC	.050	BSC
G	1.07	1.22	.042	.048
H	1.07	1.42	.042	.056
J	0.51	-	.020	-
K	0.33	0.53	.013	.021
Nd	21		21	
Ne	21		21	
PKG STD	00			

**Datasheet Change Log for AT8xC51SND1C****Changes from 4109D-10/02 to 4109E-06/03**

1. Additional information on AT83C51SND1C product.
2. Added BGA81 package.
3. Updated AC/DC characteristics for AT89C51SND1C product.
4. Changed the endurance of Flash to 100, 000 Write/Erase cycles.
5. Added note on Flash retention formula for  $V_{IH1}$ , in Section "DC Characteristics", page 181.

<b>Features</b> .....	<b>1</b>
<b>Description</b> .....	<b>1</b>
<b>Typical Applications</b> .....	<b>2</b>
<b>Block Diagram</b> .....	<b>2</b>
<b>Pin Description</b> .....	<b>3</b>
Pinouts .....	3
Signals.....	6
Internal Pin Structure.....	11
<b>Clock Controller</b> .....	<b>12</b>
Oscillator .....	12
X2 Feature.....	12
PLL .....	13
Registers .....	15
<b>Program/Code Memory</b> .....	<b>17</b>
ROM Memory Architecture.....	17
Flash Memory Architecture .....	18
Hardware Security System.....	19
Boot Memory Execution .....	19
Preventing Flash Corruption.....	20
Registers .....	21
Hardware Bytes.....	22
<b>Data Memory</b> .....	<b>23</b>
Internal Space .....	23
External Space .....	25
Dual Data Pointer .....	27
Registers .....	28
<b>Special Function Registers</b> .....	<b>30</b>
<b>Interrupt System</b> .....	<b>36</b>
Interrupt System Priorities .....	36
External Interrupts .....	39
Registers .....	40
<b>Power Management</b> .....	<b>46</b>



Reset .....	46
Reset Recommendation to Prevent Flash Corruption .....	47
Idle Mode .....	47
Power-down Mode.....	48
Registers.....	50
<b>Timers/Counters .....</b>	<b>51</b>
Timer/Counter Operations .....	51
Timer Clock Controller .....	51
Timer 0.....	52
Timer 1.....	54
Interrupt .....	55
Registers.....	56
<b>Watchdog Timer .....</b>	<b>59</b>
Description.....	59
Watchdog Clock Controller .....	59
Watchdog Operation.....	60
Registers.....	61
<b>MP3 Decoder .....</b>	<b>62</b>
Decoder .....	62
Audio Controls .....	64
Decoding Errors.....	64
Frame Information .....	65
Ancillary Data.....	65
Interrupt .....	66
Registers.....	68
<b>Audio Output Interface .....</b>	<b>73</b>
Description.....	73
Clock Generator.....	74
Data Converter .....	74
Audio Buffer .....	75
MP3 Buffer.....	76
Interrupt Request.....	76
MP3 Song Playing .....	76
Voice or Sound Playing .....	77
Registers.....	78
<b>Universal Serial Bus .....</b>	<b>80</b>
Description.....	81
Configuration .....	84
Read/Write Data FIFO .....	86
Bulk/Interrupt Transactions.....	87
Control Transactions.....	91

Isochronous Transactions.....	92
Miscellaneous .....	94
Suspend/Resume Management .....	95
USB Interrupt System .....	97
Registers.....	99
<b>MultiMedia Card Controller .....</b>	<b>109</b>
Card Concept.....	109
Bus Concept .....	109
Description.....	114
Clock Generator.....	114
Command Line Controller.....	116
Data Line Controller.....	118
Interrupt .....	124
Registers.....	125
<b>IDE/ATAPI Interface .....</b>	<b>131</b>
Description.....	131
Registers.....	133
<b>Serial I/O Port .....</b>	<b>134</b>
Mode Selection.....	134
Baud Rate Generator.....	134
Synchronous Mode (Mode 0) .....	135
Asynchronous Modes (Modes 1, 2 and 3).....	137
Multiprocessor Communication (Modes 2 and 3) .....	140
Automatic Address Recognition.....	140
Interrupt .....	142
Registers.....	143
<b>Synchronous Peripheral Interface .....</b>	<b>146</b>
Description.....	147
Interrupt .....	151
Configuration .....	152
Registers.....	156
<b>Two-wire Interface (TWI) Controller .....</b>	<b>158</b>
Description.....	158
Registers.....	172
<b>Analog to Digital Converter .....</b>	<b>174</b>
Description.....	174
Registers.....	177
<b>Keyboard Interface .....</b>	<b>179</b>
Description.....	179



Registers.....	180
<b>Electrical Characteristics .....</b>	<b>181</b>
Absolute Maximum Rating.....	181
DC Characteristics.....	181
AC Characteristics.....	186
SPI Interface.....	190
<b>Ordering Information .....</b>	<b>201</b>
<b>Package Information .....</b>	<b>202</b>
TQFP80 .....	202
BGA81 .....	203
PLCC84 .....	204
<b>Datasheet Change Log for AT8xC51SND1C .....</b>	<b>205</b>
Changes from 4109D-10/02 to 4109E-06/03.....	205



## Atmel Corporation

2325 Orchard Parkway  
San Jose, CA 95131  
Tel: 1(408) 441-0311  
Fax: 1(408) 487-2600

## Regional Headquarters

### Europe

Atmel Sarl  
Route des Arsenaux 41  
Case Postale 80  
CH-1705 Fribourg  
Switzerland  
Tel: (41) 26-426-5555  
Fax: (41) 26-426-5500

### Asia

Room 1219  
Chinachem Golden Plaza  
77 Mody Road Tsimshatsui  
East Kowloon  
Hong Kong  
Tel: (852) 2721-9778  
Fax: (852) 2722-1369

### Japan

9F, Tonetsu Shinkawa Bldg.  
1-24-8 Shinkawa  
Chuo-ku, Tokyo 104-0033  
Japan  
Tel: (81) 3-3523-3551  
Fax: (81) 3-3523-7581

## Atmel Operations

### Memory

2325 Orchard Parkway  
San Jose, CA 95131  
Tel: 1(408) 441-0311  
Fax: 1(408) 436-4314

### Microcontrollers

2325 Orchard Parkway  
San Jose, CA 95131  
Tel: 1(408) 441-0311  
Fax: 1(408) 436-4314

La Chantrerie  
BP 70602  
44306 Nantes Cedex 3, France  
Tel: (33) 2-40-18-18-18  
Fax: (33) 2-40-18-19-60

### ASIC/ASSP/Smart Cards

Zone Industrielle  
13106 Rousset Cedex, France  
Tel: (33) 4-42-53-60-00  
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.  
Colorado Springs, CO 80906  
Tel: 1(719) 576-3300  
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park  
Maxwell Building  
East Kilbride G75 0QR, Scotland  
Tel: (44) 1355-803-000  
Fax: (44) 1355-242-743

### RF/Automotive

Theresienstrasse 2  
Postfach 3535  
74025 Heilbronn, Germany  
Tel: (49) 71-31-67-0  
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.  
Colorado Springs, CO 80906  
Tel: 1(719) 576-3300  
Fax: 1(719) 540-1759

### Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Datacom

Avenue de Rochepleine  
BP 123  
38521 Saint-Egreve Cedex, France  
Tel: (33) 4-76-58-30-00  
Fax: (33) 4-76-58-34-80

---

### e-mail

[literature@atmel.com](mailto:literature@atmel.com)

### Web Site

<http://www.atmel.com>

**Disclaimer:** Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

© Atmel Corporation 2003. All rights reserved. Atmel® and combinations thereof are the registered trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be the trademarks of others.



Printed on recycled paper.