

## Features

- 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
  - Flash: AT89C5132, ROM: AT83C5132<sup>(1)</sup>
- 4K Bytes of Boot Flash Memory (AT89C5132)
  - ISP: Download from USB or UART to any External Memory Cards
- USB Rev 1.1 Device Controller
  - “Full Speed” Data Transmission
- Built-in PLL
- 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 Bits)
  - 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
- Two Standard 16-bit Timers/Counters
- Hardware Watchdog Timer
- Standard Full Duplex UART with Baud Rate Generator
- 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, TQFP64, BGA81<sup>(1)</sup>, PLCC84 (Development Board Only)
  - Dice

Note: 1. Contact Atmel for availability.

## Description

The AT8xC5132 are mass storage devices controlling data exchange between various Flash modules, HDD and CD-ROM.

The AT89C5132 includes 64K Bytes of Flash memory and allows In-System Programming through an embedded 4K Bytes of Boot Flash Memory.

The AT83C5132 includes 64K Bytes of ROM memory.

The AT8xC5132 include 2304 Bytes of RAM memory.

The AT8xC5132 provide all the necessary features for man-machine interface including, timers, keyboard port, serial or parallel interface (USB, 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

- Flash Recorder/Writer
- PDA, Camera, Mobile Phone
- PC Add-on



**USB  
Microcontroller  
with 64K Bytes  
ROM or Flash**

**AT83C5132  
AT89C5132**

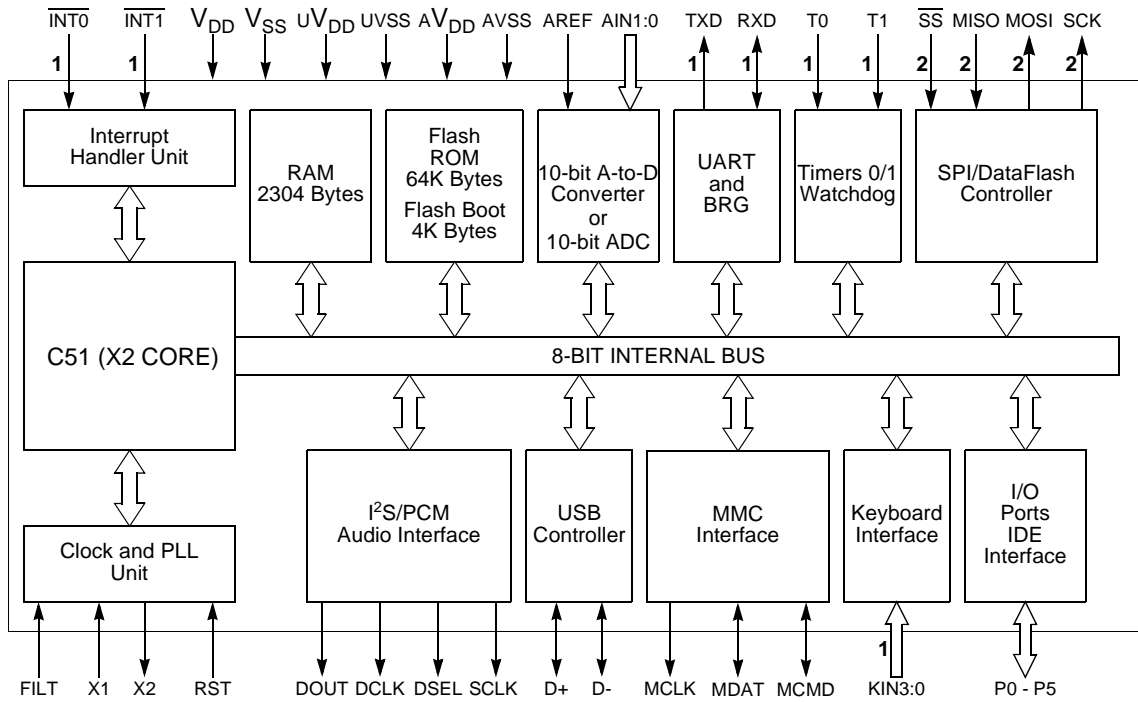
**Preliminary**

Rev. 4173A-8051-08/02



# Block Diagram

Figure 1. AT8xC5132 Block Diagram



- Notes:
1. Alternate function of Port 3
  2. Alternate function of Port 4

Pin Configuration

Figure 2. AT8xC5132 80-pin TQFP Package

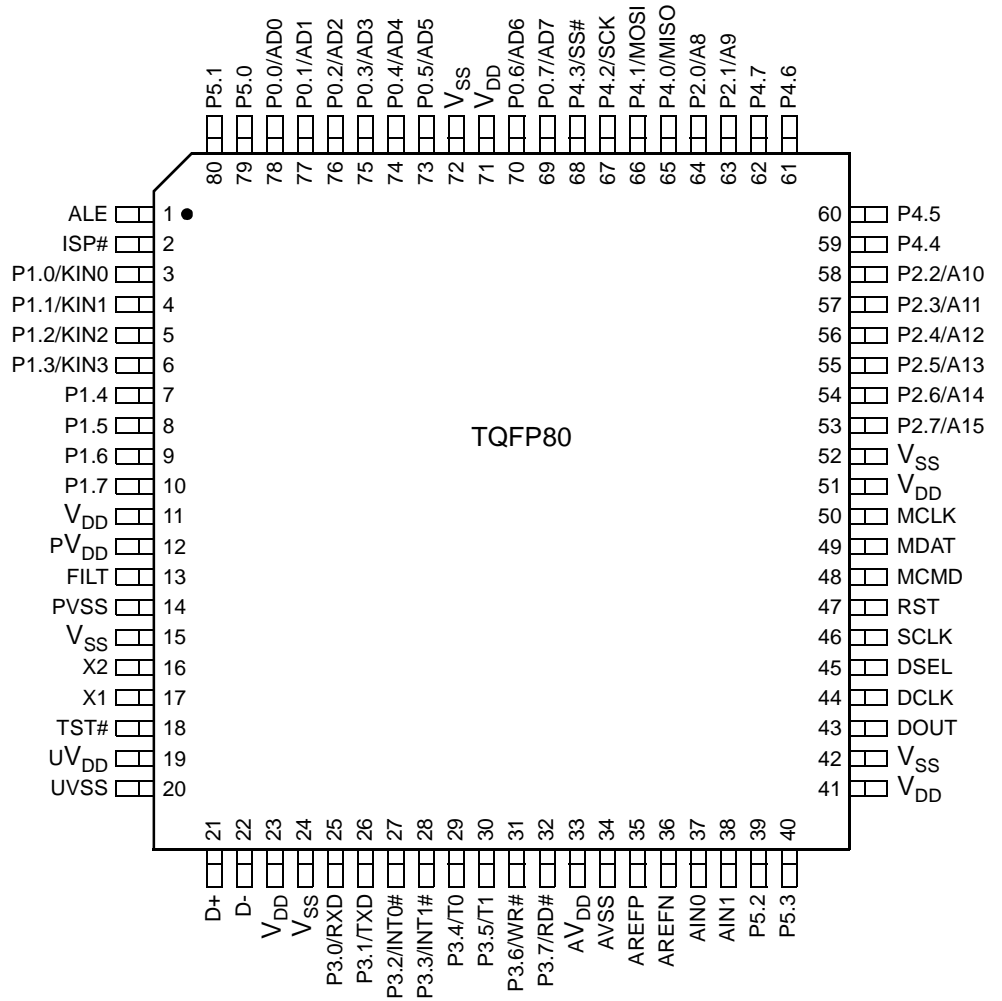


Figure 3. AT8xC5132 64-pin TQFP

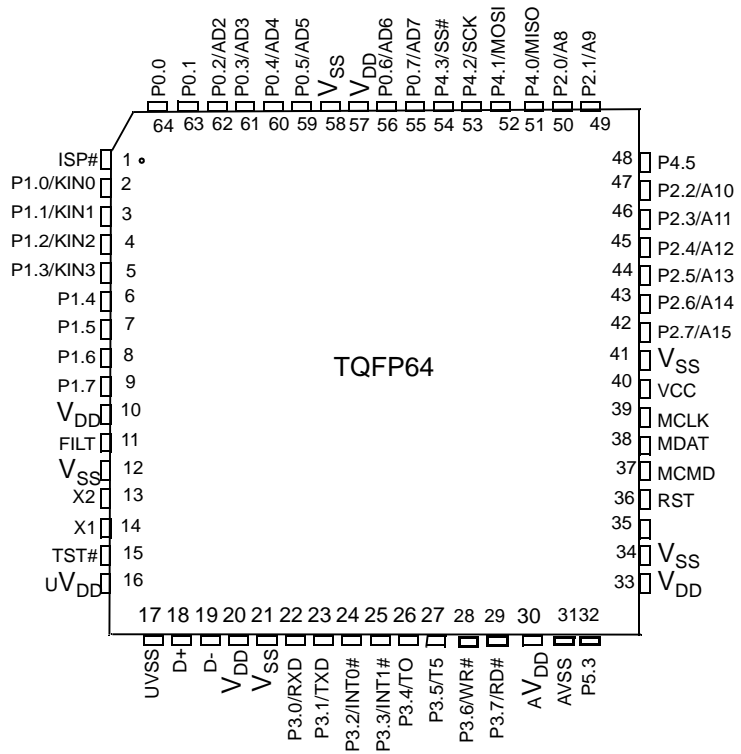
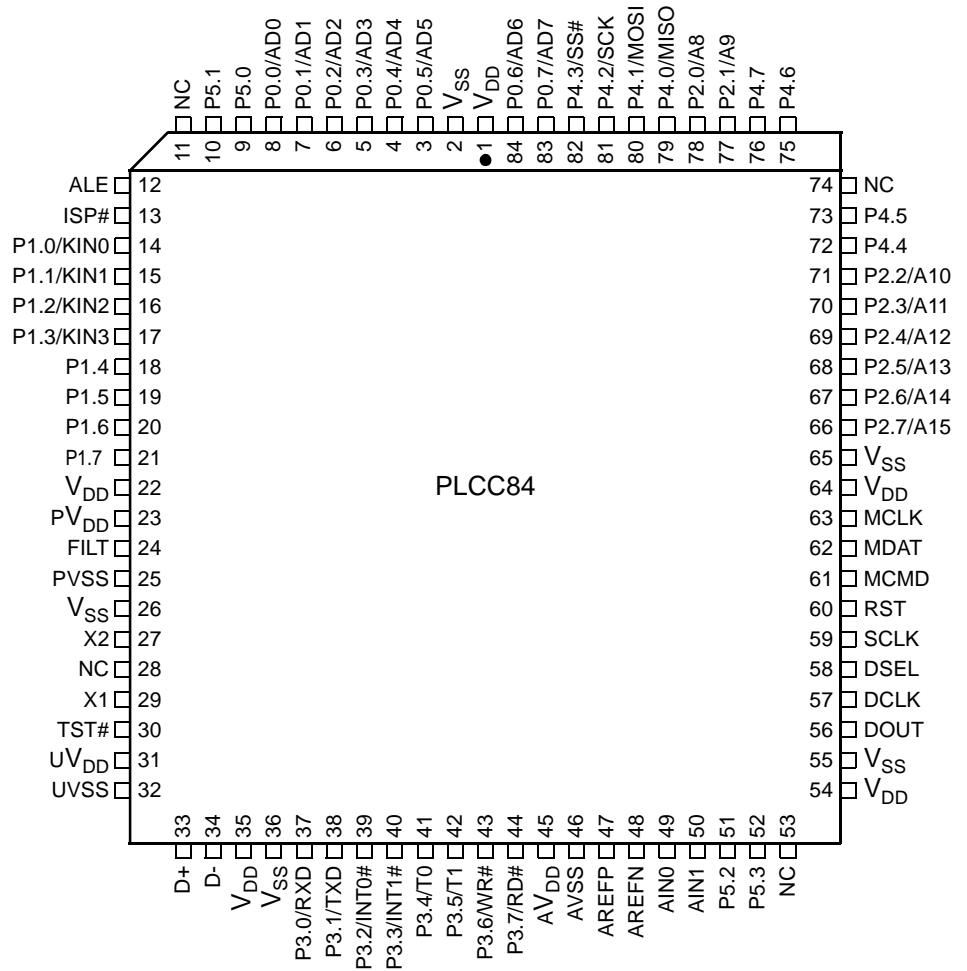


Figure 4. AT8xC5132 84-pin PLCC Package<sup>(1)</sup>



Note: 1. For development board only.

## Pin Description

All AT8xC5132 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 $\overline{\text{INT0}}$ 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 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.** 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
SS#	I	<b>SPI Slave Select Line</b> When in controlled slave mode, $\overline{SS}$ enables the slave mode.	P4.3

**Table 9.** Specific Controller

Signal Name	Type	Description	Alternate Function
SCL	I/O	Reserved Do not set this bit.	P1.6
SDA	I/O	Reserved Do not set this bit.	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 $AV_{SS}$ .	-

**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</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**

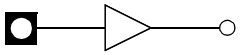
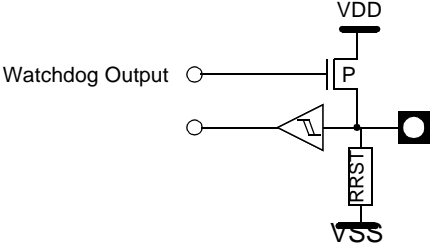
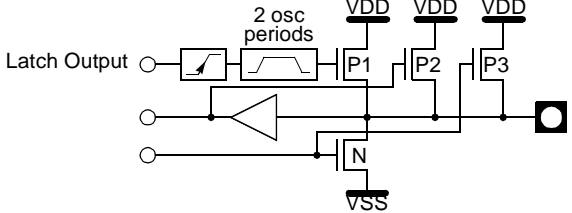
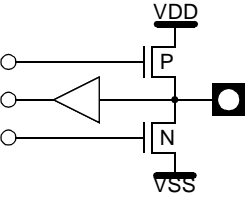
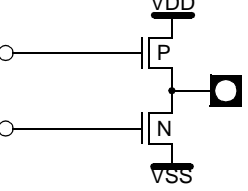
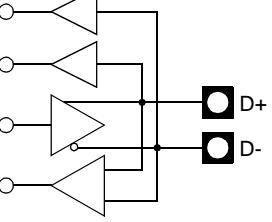
Signal Name	Type	Description	Alternate Function
RST	I	<p><b>Reset Input</b>            Holding this pin high for 64 oscillator periods while the oscillator is running resets the device. The Port pins are driven to their reset conditions when a voltage lower than <math>V_{IL}</math> is applied, whether 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 <math>V_{DD}</math>.            Asserting RST when the chip is in Idle mode or Power-down mode returns the chip to normal operation.</p>	-
$\overline{TST}$	I	<p><b>Test Input</b>            Test mode entry signal. This pin must be set to <math>V_{DD}</math>.</p>	-

**Table 14. Power Signal Description**

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

Internal Pin Structure

Table 15. Detailed Internal Pin Structure

Circuit <sup>(1)</sup>	Type	Pins
	Input	TST#
	Input/Output	RST
	Input/Output	P1 P2 <sup>(3)</sup> P3 P4 P5:0
	Input/Output	P0 MCMD MDAT ISP#
	Output	ALE SCLK DCLK DOUT DSEL MCLK
	Input/Output	D+ D-

Notes: 1. For information on resistors value, input/output levels, and drive capability, refer to the AT8xC5132 full Datasheet.  
 2. In Port 2, P<sub>1</sub> transistor is continuously driven when outputting a high level bit address (A15:8).

## Clock Controller

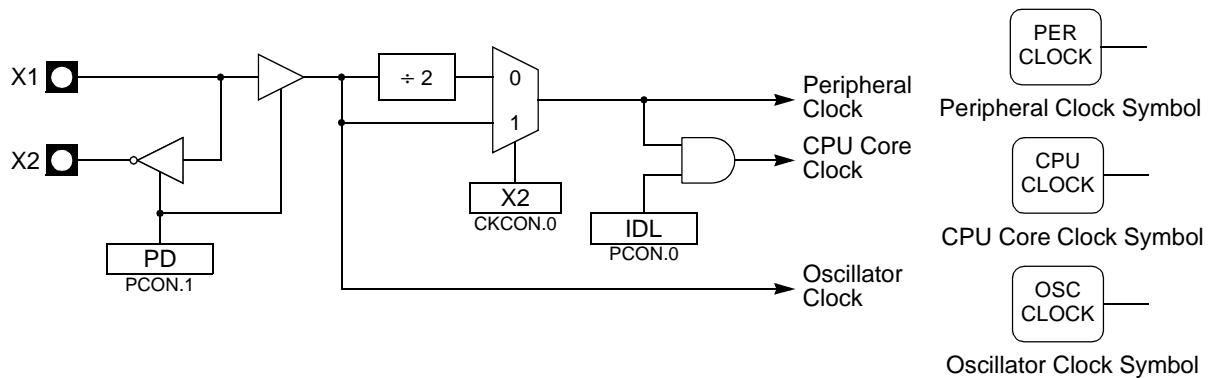
The AT8xC5132 clock controller is based on an on-chip oscillator feeding an on-chip Phase Lock Loop (PLL). All internal clocks to the peripherals and CPU core are generated by this controller.

## Oscillator

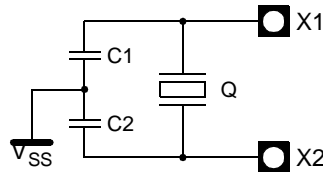
The AT8xC5132 X1 and X2 pins are the input and the output of a single-stage on-chip inverter (see Figure 5) that can be configured with off-chip components such as a Pierce oscillator (see Figure 6). Value of capacitors and crystal characteristics are detailed in the Section “DC Characteristics”.

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 “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

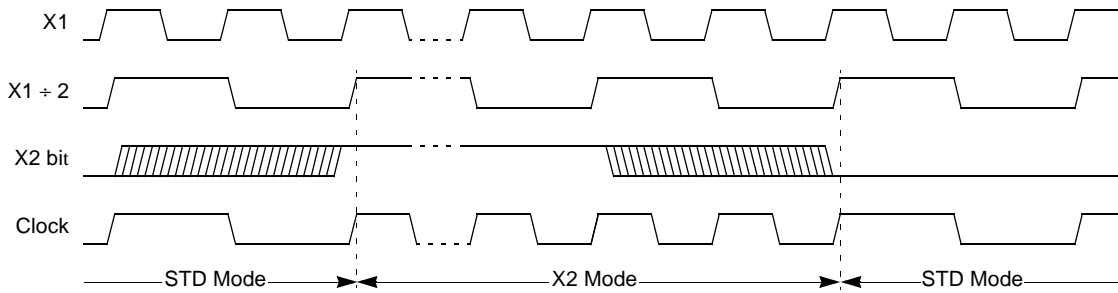


## X2 Feature

Unlike standard C51 products that require 12 oscillator clock periods per machine cycle, the AT8xC5132 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 AT8xC5132 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 Byte (see Table 21 on page 23). Using the AT89C5132 (Flash Version) the system can boot either in standard or X2 mode depending on the X2B value. Using AT83C5132 (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: In order to prevent any incorrect operation while operating in X2 mode, the user must be aware that all peripherals using clock frequency as time reference (timers...) will have their time reference divided by two. For example, a free running timer generating an interrupt every 20 ms will then generate an interrupt every 10 ms.

PLL

PLL Description

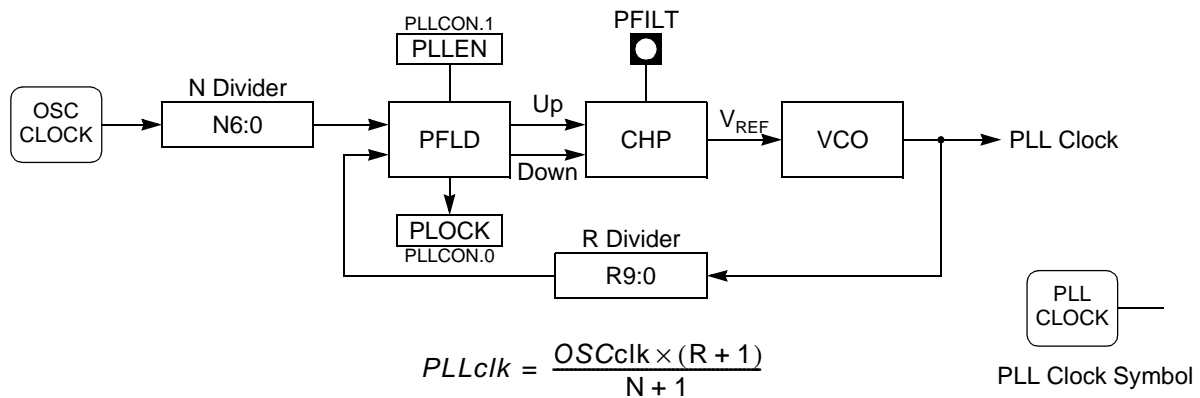
The AT8xC5132's PLL is used to generate internal high frequency clock (the PLL Clock) synchronized with an external low-frequency (the Oscillator Clock). The PLL clock provides the audio interface, and the USB interface clocks. Figure 8 shows the internal structure of the PLL.

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.

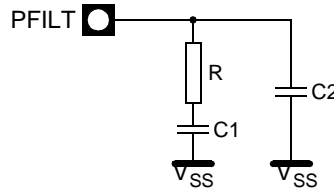
The CHP block is the Charge Pump that generates the voltage reference for the VCO by injecting or extracting charges from the external filter connected on PFILT pin (see Figure 9). Value of the filter components are detailed in the Section "DC Characteristics".

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



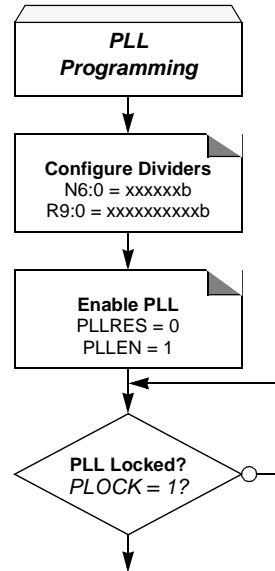
**Figure 9.** PLL Filter Connection



## 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 the 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 values 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 two 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 two 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 two 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

**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 values 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 values 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

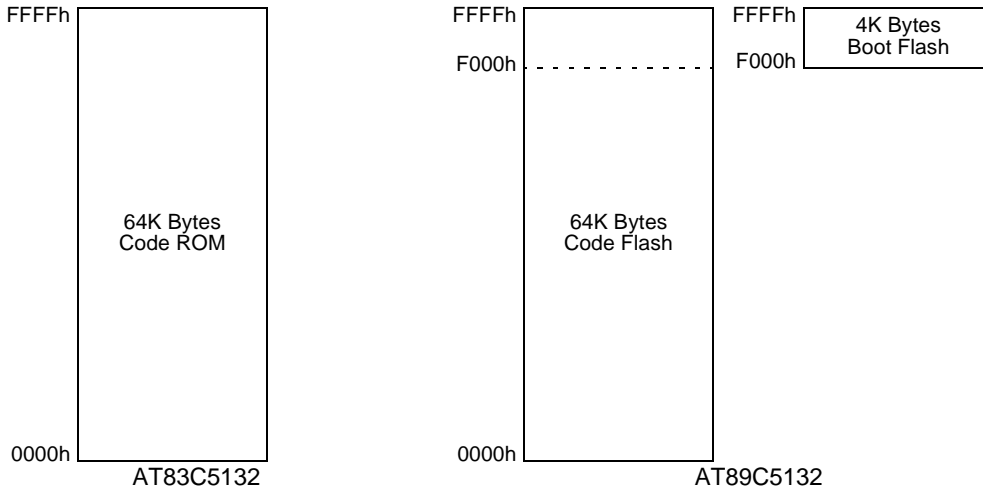
The AT89C5132 and AT83C5132 implement 64K Bytes of on-chip program/code memory. Figure 11 shows the split of internal and external program/code memory spaces depending on the product.

The AT83C5132 product provides the internal program/code memory in ROM memory while the AT89C5132 product provides it in Flash memory. These two 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 AT89C5132 can be programmed using only one voltage and allows in application software programming. Hardware programming mode is also available using common programming tools.

The AT89C5132 implements an additional 4K Bytes of on-chip boot Flash memory provided in Flash memory. This boot memory is delivered programmed with a standard bootloader software allowing In-System Programming (ISP). It also contains some Application Programming Interfaces (API), allowing In Application Programming (IAP) by using user's own bootloader.

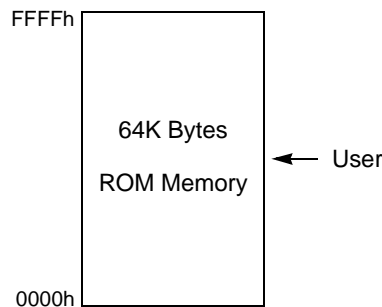
**Figure 11.** Program/Code Memory Organization



## ROM Memory Architecture

As shown in Figure 12, the AT83C5132 ROM memory is composed of two spaces detailed in the following section.

**Figure 12.** ROM Memory Architecture



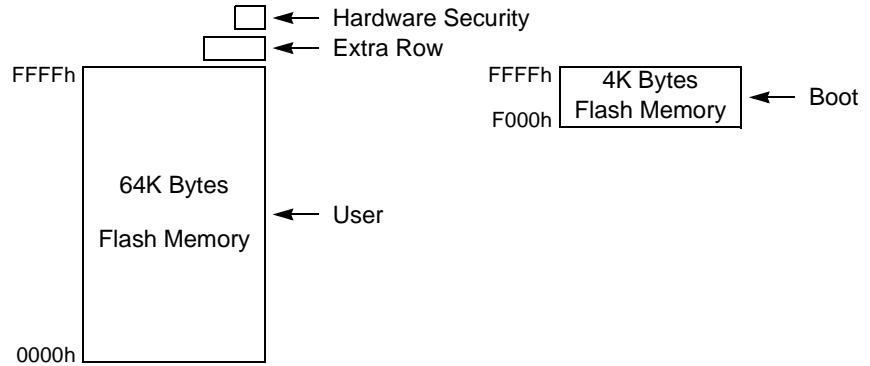
**User Space**

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 AT89C5132 Flash memory is composed of four spaces detailed in the following paragraphs.

**Figure 13.** Flash 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 bootloader for In-System Programming and the routines for In-System 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 21) divided in two separate nibbles see Table 21. The MSN contains the X2 mode configuration bit and the Boot Loader Jump Bit as detailed in section "Boot Memory Execution" 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" and can only be written by hardware.

## Extra Row Space

This space is composed of two Bytes:

- The Software Boot Vector (SBV see Table 22).  
This byte is used by the software bootloader to build the boot address.
- The Software Security Byte (SSB see Figure ).  
This byte is used to lock the execution of some bootloader commands.

## Hardware Security System

The AT89C5132 implements three lock Bits LB2:0 in the LSN of HSB (see Table 21) providing three levels of security for user's program as described in Table 21 while the AT83C5132 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 hardware verifying of both user and boot memories.
- Level 3 locks the external execution.

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 AT8xC5132 products.
  3. AT89C5132 products are delivered with third level programmed to ensure that the code programmed by software using ISP or user's bootloader being 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 Table 20). 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 bootloader or API routines execution.

### Hardware Condition Boot Mapping

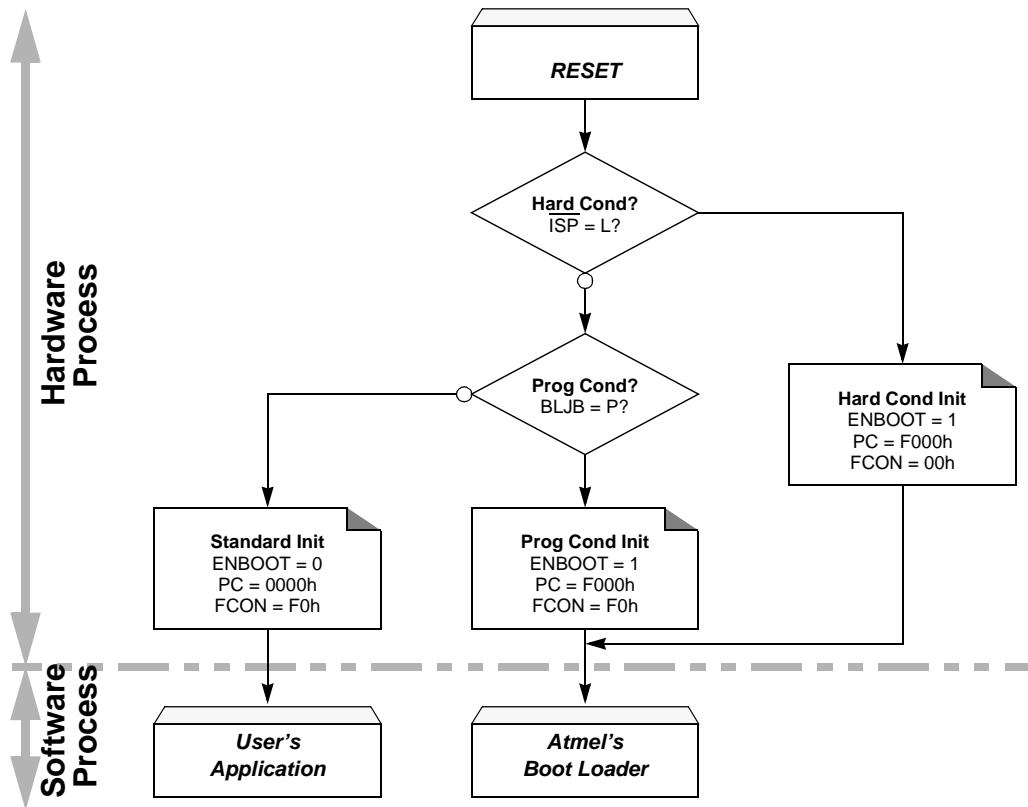
The hardware condition is based on the  $\overline{\text{ISP}}$  pin. When driving this pin to low level, the chip reset sets ENBOOT and forces the reset vector to F000h instead of 0000h in order to execute the bootloader software.

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 Bootloader Jump Bit (BLJB) in HSB. As shown in Figure 14, when this bit is programmed (by hardware or software programming mode), the chip resets ENBOOT and forces the reset vector to F000h instead of 0000h, in order to execute the bootloader software.

Figure 14. Hardware Boot Process Algorithm



The software process (bootloader) is detailed in the section “In-System and In-Application Programming”.

*Preventing Flash Corruption*

See “Reset Recommendation to Prevent Flash Corruption” on page 46 in the section “Power Management”.

## Registers

**Table 20.** 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 values read from these Bits are indeterminate. Do not set these Bits.					
5	ENBOOT	<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 values 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

## Hardware Bytes

**Table 21.** 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)(2)</sup>	<b>X2 Bit</b> Program this bit to start in X2 mode. Unprogram (erase) this bit to start in standard mode.					
6	BLJB	<b>Boot Loader Jump Bit</b> Program this bit to execute the bootloader 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 values read from these Bits are always unprogrammed. Do not program these Bits.					
3		<b>Reserved</b> The values read from this bit is always unprogrammed. Do not program this bit.					
2 - 0 <sup>(3)</sup>	LB2:0	<b>Hardware Lock Bits</b> Refer to Table 21 for Bits description.					

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

- Notes:
1. X2B initializes the X2 bit in CKCON during the reset phase.
  2. Using the AT89C5132 (Flash Version) the system can boot either in standard or X2 mode depending on the X2B value. Using AT83C5132 (ROM Version) the system always boots in standard mode. X2B bit can be changed to X2 mode later by software.
  3. Bits 0 to 3 (MSN) can only be programmed by hardware mode.

**Table 22.** 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 bootloader 16-bit address location</b> Refer to the AT8xC5132 datasheet for usage information (bootloader dependent).					

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

**Table 23.** 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 AT8xC5132 datasheet for usage information (bootloader dependent).					

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



## Data Memory

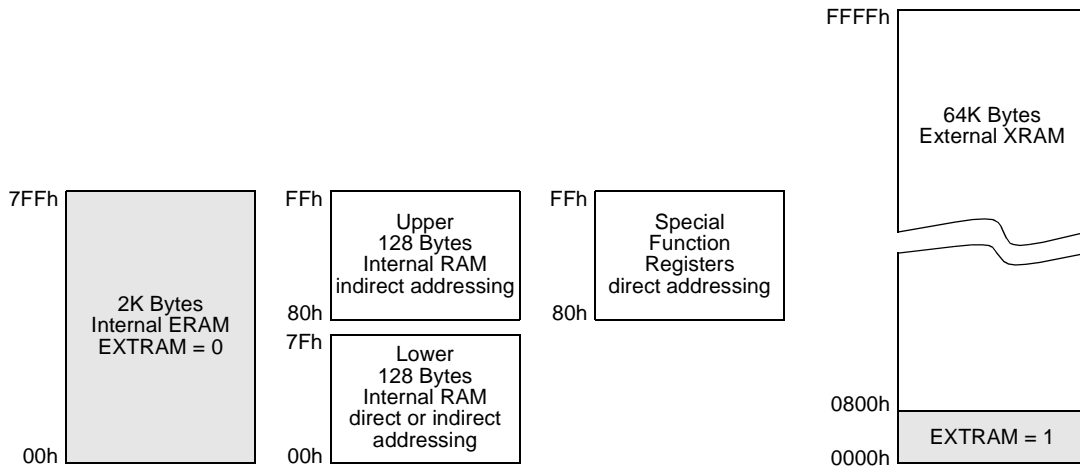
The AT8xC5132 provides data memory access in two different spaces:

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 31.

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

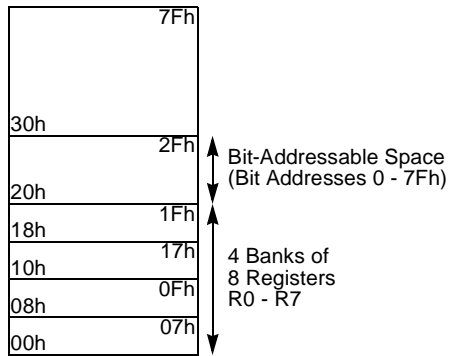
The lower 128 Bytes of RAM (see Figure 16) are accessible from address 00h to 7Fh using direct or indirect addressing modes. The lowest 32 Bytes are grouped into 4 banks of 8 registers (R0 to R7). Two Bits RS0 and RS1 in PSW register (see Table 27) select which bank is in use according to Table 24. This 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 24.** 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 28) 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.

The ERAM memory can be resized using XRS1:0 Bits in AUXR register to dynamically increase external access to the XRAM space. Table 25 details the selected ERAM size and address range.

**Table 25.** 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.

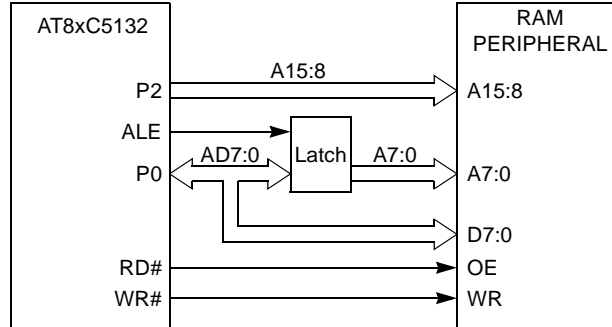
## External Space

### Memory Interface

The external memory interface comprises the external bus (port 0 and port 2) as well as the bus control signals (RD, 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 26 describes the external memory interface signals.

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



**Table 26.** 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.	-
RD#	O	<b>Read</b> Read signal output to external data memory.	P3.7
WR#	O	<b>Write</b> Write signal output to external memory.	P3.6

### Page Access Mode

The AT8xC5132 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 intensively modifies the EXTRAM bit to select access to ERAM or XRAM and must save it if it is used in the 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-purpose I/O usage.

## External Bus Cycles

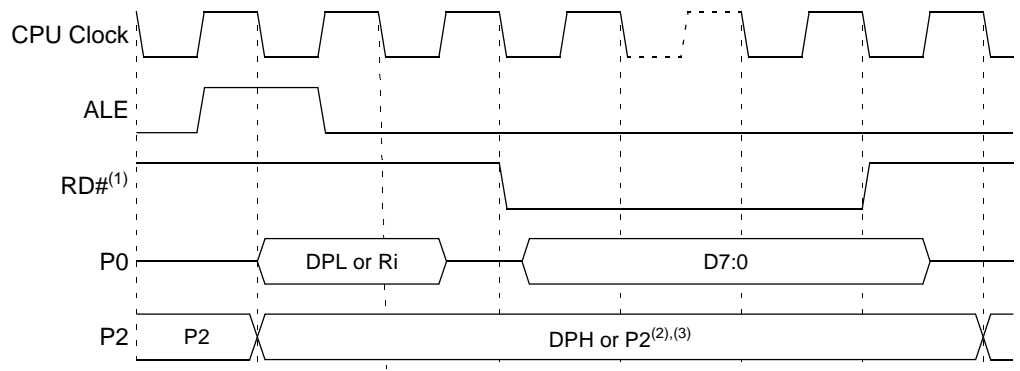
This section describes the bus cycles that AT8xC5132 execute to read (see Figure 18), and write data (see Figure 19) in the external data memory.

External memory cycle takes 6 CPU clock periods. This is equivalent to 12 oscillator clock 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 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 RD and WR signals from 3 to 15 CPU clock periods.

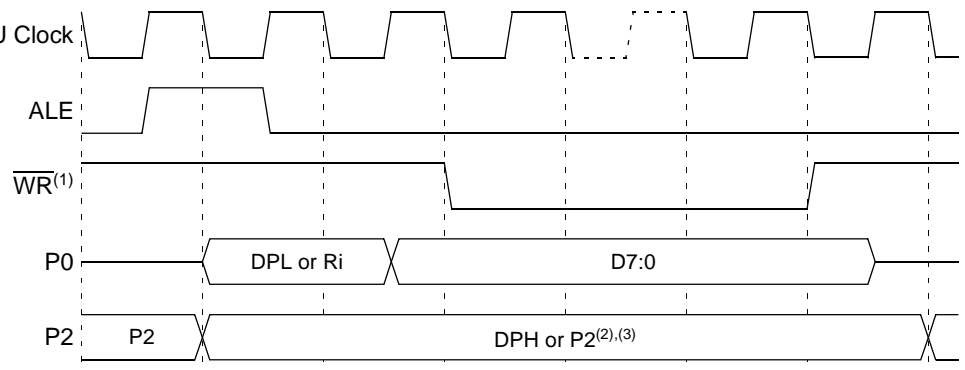
For simplicity, the accompanying figures depict the bus cycle waveforms in idealized form and do not provide precise timing information. For bus cycle timing parameters refer to the section “AC Characteristics”.

**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.

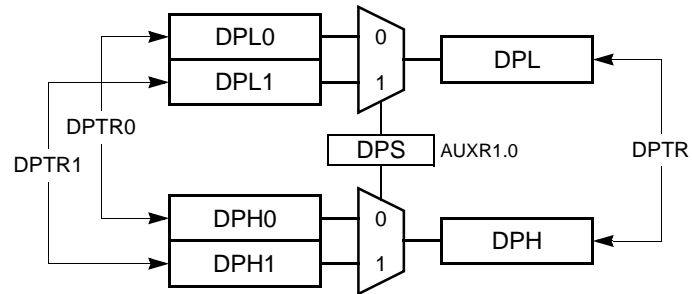
## Dual Data Pointer

### Description

The AT8xC5132 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 29) 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 two 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 forces 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
```

```
AUXR1EQU0A2h
```

```
move:movDPTR,#SOURCE ; address of SOURCE
      incAUXR1 ; switch data pointers
      movDPTR,#DEST ; address of DEST
mv_loop:incAUXR1; switch data pointers
      movx@DPTR; get a byte from SOURCE
      incDPTR; increment SOURCE address
      incAUXR1; switch data pointers
      movx@DPTR,A; write the byte to DEST
      incDPTR; increment DEST address
      jnzmv_loop; check for NULL terminator
end_move:
```

Registers

**Table 27.** 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 24 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 28.** 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 values 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 25 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

**Table 29.** AUXR1 Register  
*AUXR1 (S:A2h)* – Auxiliary Control Register 1

7	6	5	4	3	2	1	0
-	-	-	-	GF3	0	-	DPS

Bit Number	Bit Mnemonic	Description
7 - 4	-	<b>Reserved</b> The values read from these Bits are indeterminate. Do not set these Bits.
3	GF3	<b>General Purpose Flag 3.</b>
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

## Special Function Registers

The Special Function Registers (SFRs) of the AT8xC5132 derivatives fall into the categories detailed in Table 30 to Table 45. The relative addresses of these SFRs are provided together with their reset values in Table 46. 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	–	GF3	0	–	DPS
NVERS	FBh	Version Number	NV7	NV6	NV5	NV4	NV3	NV2	NV1	NV0

**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	v	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	–	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	–	IPHS	IPHT1	IPHX1	IPHT0	IPHX0
IPL0	B8h	Interrupt Priority Control Low 0	–	IPLAUD	–	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	D1h	Flash Control	FPL3	FPL2	FPL1	FPL0	FPS	FMOD1	FMOD0	FBUSY

**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. 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 38. USB Controller SFRs**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
USBCON	BC <sub>h</sub>	USB Global Control	USBE	SUSPCLK	SDRMWUP	–	UPRSM	RMWUPE	CONFIG	FADDEN
USBADDR	C6 <sub>h</sub>	USB Address	FEN	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0
USBINT	BD <sub>h</sub>	USB Global Interrupt	–	–	WUPCPU	EORINT	SOFINT	–	–	SPINT
USBIEN	BE <sub>h</sub>	USB Global Interrupt Enable	–	–	EWUPCPU	EEORINT	ESOFINT	–	–	ESPINT
UEPNUM	C7 <sub>h</sub>	USB Endpoint Number	–	–	–	–	–	–	EPNUM1	EPNUM0
UEPCONX	D4 <sub>h</sub>	USB Endpoint X Control	EPEN	–	–	–	DTGL	EPDIR	EPTYPE1	EPTYPE0
UEPSTAX	CE <sub>h</sub>	USB Endpoint X Status	DIR	–	STALLRQ	TXRDY	STLCRC	RXSETUP	RXOUT	TXCMP
UEPRST	D5 <sub>h</sub>	USB Endpoint Reset	–	–	–	–	EP3RST	EP2RST	EP1RST	EP0RST
UEPINT	F8 <sub>h</sub>	USB Endpoint Interrupt	–	–	–	–	EP3INT	EP2INT	EP1INT	EP0INT
UEPIEN	C2 <sub>h</sub>	USB Endpoint Interrupt Enable	–	–	–	–	EP3INTE	EP2INTE	EP1INTE	EP0INTE
UEPDATA	CF <sub>h</sub>	USB Endpoint X FIFO Data	FDAT7	FDAT6	FDAT5	FDAT4	FDAT3	FDAT2	FDAT1	FDAT0
UBYCTX	E2 <sub>h</sub>	USB Endpoint X Byte Counter	–	BYCT6	BYCT5	BYCT4	BYCT3	BYCT2	BYCT1	BYCT0
UFNUML	BA <sub>h</sub>	USB Frame Number Low	FNUM7	FNUM6	FNUM5	FNUM4	FNUM3	FNUM2	FNUM1	FNUM0
UFNUMH	BB <sub>h</sub>	USB Frame Number High	–	–	CRCOK	CRCERR	–	FNUM10	FNUM9	FNUM8
USBCLK	EA <sub>h</sub>	USB Clock Divider	–	–	–	–	–	–	USBCD1	USBCD0

**Table 39. MMC Controller SFRs**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
MMCON0	E4 <sub>h</sub>	MMC Control 0	DRPTR	DTPTR	CRPTR	CTPTR	MBLOCK	DFMT	RFMT	CRCDIS
MMCON1	E5 <sub>h</sub>	MMC Control 1	BLEN3	BLEN2	BLEN1	BLEN0	DATDIR	DATEN	RESPEN	CMDEN
MMCON2	E6 <sub>h</sub>	MMC Control 2	MMCEN	DCR	CCR	–	–	DATD1	DATD0	FLOWC
MMSTA	DE <sub>h</sub>	MMC Control and Status	–	–	CBUSY	CRC16S	DATFS	CRC7S	RESPFS	CFLCK
MMINT	E7 <sub>h</sub>	MMC Interrupt	MCBI	EORI	EOCI	EOF1	F2FI	F1FI	F2EI	F1EI
MMMSK	DF <sub>h</sub>	MMC Interrupt Mask	MCBM	EORM	EOCM	EOFM	F2FM	F1FM	F2EM	F1EM
MMCMD	DD <sub>h</sub>	MMC Command	MC7	MC6	MC5	MC4	MC3	MC2	MC1	MC0
MMDAT	DCh	MMC Data	MD7	MD6	MD5	MD4	MD3	MD2	MD1	MD0
MMCLK	ED <sub>h</sub>	MMC Clock Divider	MMCD7	MMCD6	MMCD5	MMCD4	MMCD3	MMCD2	MMCD1	MMCD0

**Table 40. IDE Interface SFR**

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
DAT16H	F9 <sub>h</sub>	High Order Data Byte	D15	D14	D13	D12	D11	D10	D9	D8

**Table 41. 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 42. 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 43. Special Register**

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

**Table 44. 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 45. 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 46. 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 <sup>(2)</sup> 1000 0010					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		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 0000 0000	UEPRST 0000 0000			D7h
C8h							UEPSTAX 0000 0000	UEPDATA 0000 0000	CFh
C0h	P4 <sup>(1)</sup> 1111 1111		UEPIEN 0000 0000	SPCON 0001 0100	SPSTA 0000 0000	SPDAT XXXX XXXX	USBADDR 1000 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				IPH0 X000 0000	B7h
A8h	IEN0 <sup>(1)</sup> 0000 0000	SADDR 0000 0000							AFh
A0h	P2 <sup>(1)</sup> 1111 1111		AUXR1 XXXX 00X0	KBCON 0000 1111	KBSTA 0000 0000		WDTRST 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			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:
1. SFR registers with least significant nibble address equal to 0 or 8 are bit-addressable.
  2. NVERS reset value depends on the silicon version.
  3. FCON register is only available in AT89C5132 product.
  4. FCON reset value is 00h in case of reset with hardware condition.
  5. CKCON reset value depends on the X2B bit (programmed or unprogrammed) in the Hardware Byte.

## Interrupt System

The AT8xC5132, 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 terminates, execution resumes at the point where the interrupt occurred. Interrupts may occur as a result of internal AT8xC5132 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:

1. An internal or external device initiates an interrupt-request signal. The AT8xC5132, latch this event into a flag buffer.
2. 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.
3. 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.
4. 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 47.** 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 134.	P1.3:0

Six interrupt registers are used to control the interrupt system. Two 8-bit registers are used to enable separately the interrupt sources: IEN0 and IEN1 registers (see Table 50 and Table 51).

Four 8-bit registers are used to establish the priority level of the thirteen sources: IPH0, IPL0, IPH1 and IPL1 registers (see Table 52 to Table 55).

## Interrupt System Priorities

Each of the eleven interrupt sources on the AT8xC5132 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 48.

**Table 48.** Priority Levels

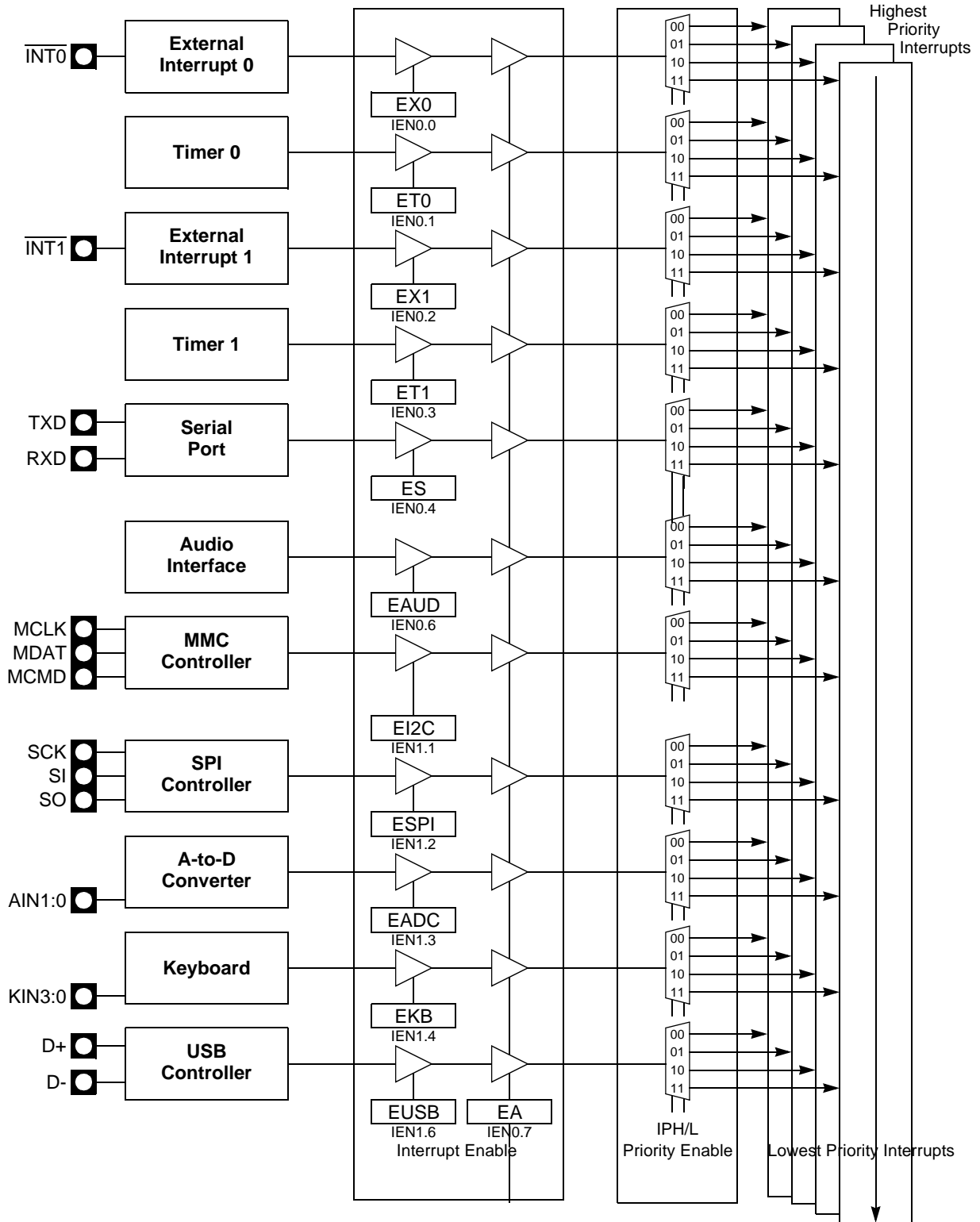
IPHxx	IPLxx	Priority Level
0	0	0 Lowest
0	1	1
1	0	2
1	1	3 Highest

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 49. 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 49.** 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
Audio Interface	7	C:0033h	S
MMC Interface	8	C:003Bh	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**



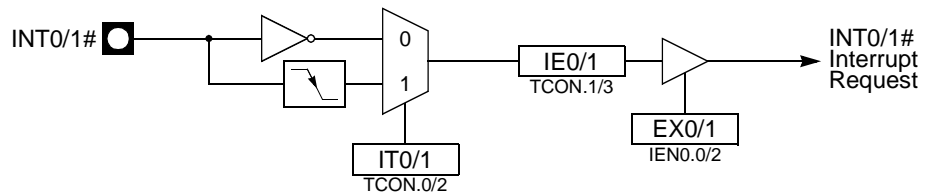
## External Interrupts

### 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, INTn is triggered by a low level at the pin. If ITn = 1, 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 IEN 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.** INT1:0# Input Circuitry



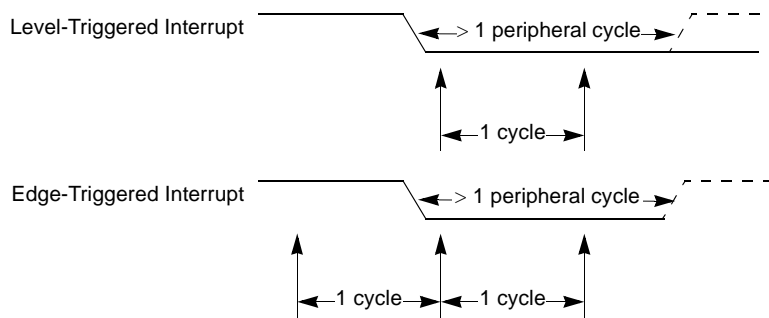
### KIN3:0 Inputs

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 134.

### 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 50.** IEN0 Register  
*IEN0 (S:A8h) – Interrupt Enable Register 0*

7	6	5	4	3	2	1	0
EA	EAUD	–	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	–	<b>Reserved</b> The values read from this bit is always 0. Do not set this bit.					
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 51.** 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 values 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 values 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>Reserved</b> The values read from this bit is always 0. Do not set this bit.					
0	EMMC	<b>MMC Interface Interrupt Enable Bit</b> Set to enable MMC interrupt. Clear to disable MMC interrupt.					

Reset Value = 0000 0000b

**Table 52.** IPH0 Register  
IPH0 (S:B7h) – Interrupt Priority High Register 0

7	6	5	4	3	2	1	0
-	IPHAUD	-	IPHS	IPHT1	IPHX1	IPHT0	IPHX0
Bit Number	Bit Mnemonic	Description					
7	-	<b>Reserved</b> The values read from this bit is indeterminate. Do not set this bit.					
6	IPHAUD	<b>Audio Interface Interrupt Priority Level MSB</b> Refer to Table 48 for priority level description.					
5	-	<b>Reserved</b> The values read from this bit is always 0. Do not set this bit.					
4	IPHS	<b>Serial Port Interrupt Priority Level MSB</b> Refer to Table 48 for priority level description.					
3	IPHT1	<b>Timer 1 Interrupt Priority Level MSB</b> Refer to Table 48 for priority level description.					
2	IPHX1	<b>External Interrupt 1 Priority Level MSB</b> Refer to Table 48 for priority level description.					
1	IPHT0	<b>Timer 0 Interrupt Priority Level MSB</b> Refer to Table 48 for priority level description.					
0	IPHX0	<b>External Interrupt 0 Priority Level MSB</b> Refer to Table 48 for priority level description.					

Reset Value = X000 0000b

**Table 53.** 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 values read from this bit is always 0. Do not set this bit.
6	IPHUSB	<b>USB Interrupt Priority Level MSB</b> Refer to Table 48 for priority level description.
5	-	<b>Reserved</b> The values read from this bit is always 0. Do not set this bit.
4	IPHKB	<b>Keyboard Interrupt Priority Level MSB</b> Refer to Table 48 for priority level description.
3	IPHADC	<b>A-to-D Converter Interrupt Priority Level MSB</b> Refer to Table 48 for priority level description.
2	IPHSPI	<b>SPI Interrupt Priority Level MSB</b> Refer to Table 48 for priority level description.
1	IPHI2C	<b>Reserved</b> The values read from this bit is always 0. Do not set this bit.
0	IPHMMC	<b>MMC Interrupt Priority Level MSB</b> Refer to Table 48 for priority level description.

Reset Value = 0000 0000b

**Table 54.** IPL0 Register  
IPL0 (S:B8h) – Interrupt Priority Low Register 0

7	6	5	4	3	2	1	0
-	IPLAUD	-	IPLS	IPLT1	IPLX1	IPLT0	IPLX0
Bit Number	Bit Mnemonic	Description					
7	-	<b>Reserved</b> The values read from this bit is indeterminate. Do not set this bit.					
6	IPLAUD	<b>Audio Interface Interrupt Priority Level LSB</b> Refer to Table 48 for priority level description.					
5	-	<b>Reserved</b> The values read from this bit is always 0. Do not set this bit.					
4	IPLS	<b>Serial Port Interrupt Priority Level LSB</b> Refer to Table 48 for priority level description.					
3	IPLT1	<b>Timer 1 Interrupt Priority Level LSB</b> Refer to Table 48 for priority level description.					
2	IPLX1	<b>External Interrupt 1 Priority Level LSB</b> Refer to Table 48 for priority level description.					
1	IPLT0	<b>Timer 0 Interrupt Priority Level LSB</b> Refer to Table 48 for priority level description.					
0	IPLX0	<b>External Interrupt 0 Priority Level LSB</b> Refer to Table 48 for priority level description.					

Reset Value = X000 0000b

**Table 55.** 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 values read from this bit is always 0. Do not set this bit.					
6	IPLUSB	<b>USB Interrupt Priority Level LSB</b> Refer to Table 48 for priority level description.					
5	-	<b>Reserved</b> The values read from this bit is always 0. Do not set this bit.					
4	IPLKB	<b>Keyboard Interrupt Priority Level LSB</b> Refer to Table 48 for priority level description.					
3	IPLADC	<b>A-to-D Converter Interrupt Priority Level LSB</b> Refer to Table 48 for priority level description.					
2	IPLSPI	<b>SPI Interrupt Priority Level LSB</b> Refer to Table 48 for priority level description.					
1	IPLI2C	<b>Reserved</b> The values read from this bit is always 0. Do not set this bit.					
0	IPLMMC	<b>MMC Interrupt Priority Level LSB</b> Refer to Table 48 for priority level description.					

Reset Value = 0000 0000b

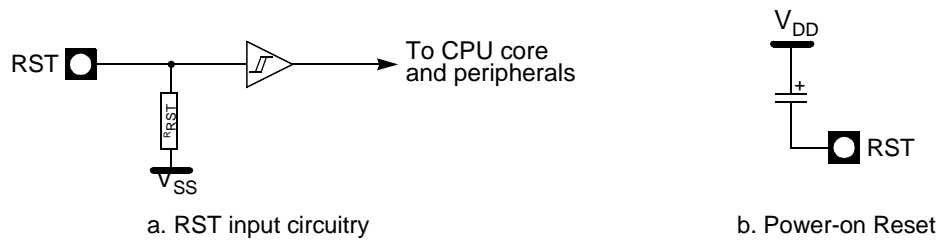
## Power Management

Two power reduction modes are implemented in the AT8xC5132: the Idle mode and the Power-down mode. 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

A reset is required after applying power at turn-on. 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. A device reset initializes the AT8xC5132 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. Resistor value and input characteristics are discussed in the Section “DC Characteristics”. The status of the Port pins during reset is detailed in Table 56.

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



**Table 56.** 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 61.

## Reset Recommendation to Prevent Flash Corruption

A bad reset sequence will lead to bad microcontroller initialization and system registers like SFR’s, Program Counter, etc. will not be correctly initialized. A bad initialization may lead to unpredictable behaviour of the C51 microcontroller.

An example of this 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 SFR’s 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 56.

## Entering Idle Mode

To enter Idle mode, the user must set the IDL bit in PCON register (see Table 57). The AT8xC5132 enter 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 AT8xC5132 enters Power-down mode. Then they do not go in Idle mode when exiting Power-down mode.

## Exiting Idle Mode

There are two 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 AT8xC5132 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 AT8xC5132 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 56.

Note:  $V_{DD}$  may be reduced to as low as  $V_{RET}$  during Power-down mode to further reduce power dissipation. Take care, 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 AT8xC5132 enter 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

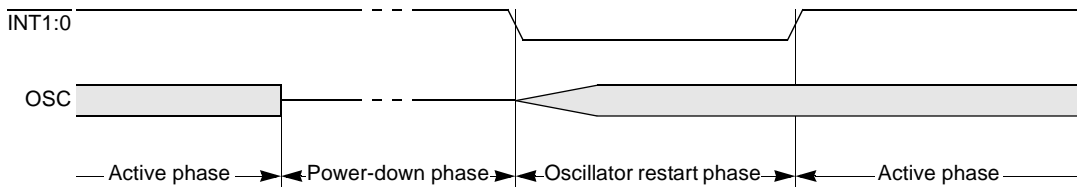
If  $V_{DD}$  was reduced during the Power-down mode, do not exit Power-down mode until  $V_{DD}$  is restored to the normal operating level.

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

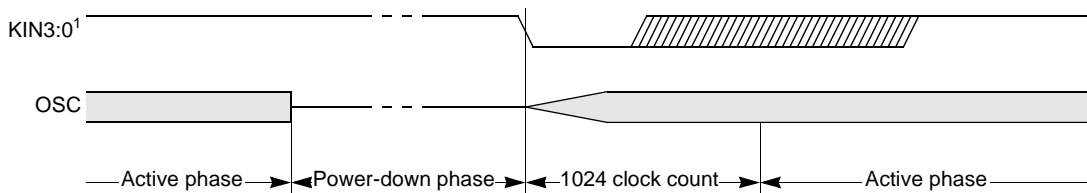
1. Generate an enabled external interrupt.
  - The AT8xC5132 provides capability to exit from Power-down using  $\overline{INT0}$ ,  $\overline{INT1}$ , and KIN3:0 inputs. In addition, using KIN input provides high or low level exit capability (see Section “Keyboard Interface”, page 134). Hardware clears PD bit in PCON register which starts the oscillator and restores the clocks to the CPU and peripherals. Using  $\overline{INTx}$  input, execution resumes when the input is released (see Figure 25) while using KINx input, execution resumes after counting 1024 clock ensuring the oscillator is restarted properly (see Figure 26). 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 ( $\overline{INT0}$  and  $\overline{INT1}$ ) 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 25.** Power-down Exit Waveform Using  $\overline{INT1:0}$



**Figure 26.** 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 the 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 AT8xC5132 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 57.** 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 values read from these Bits are 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 AT8xC5132 implement two 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 58) 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.

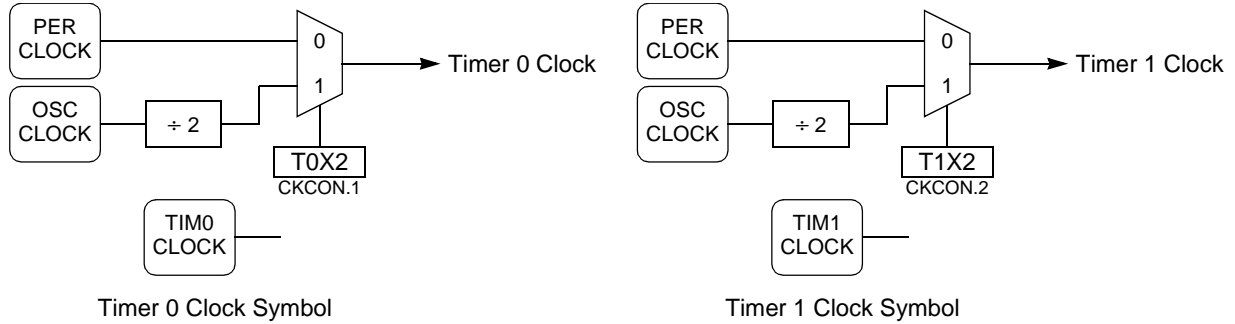
For Timer operation ( $C/Tx\# = 0$ ), the Timer register counts the divided-down peripheral clock. The Timer register is incremented once every peripheral cycle (6 peripheral clock periods). The Timer clock rate is  $F_{PER}/6$ , i.e.,  $F_{OSC}/12$  in standard mode or  $F_{OSC}/6$  in X2 mode.

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 27, 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 'CKCON Register', page 15. 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 27.** 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 28 through Figure 34 show the logical configuration of each mode.

Timer 0 is controlled by the four lower Bits of TMOD register (see Table 59) and Bits 0, 1, 4 and 5 of TCON register (see Table 58). TMOD register selects the method of Timer gating (GATE0), Timer or Counter operation (T/C0#) 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.

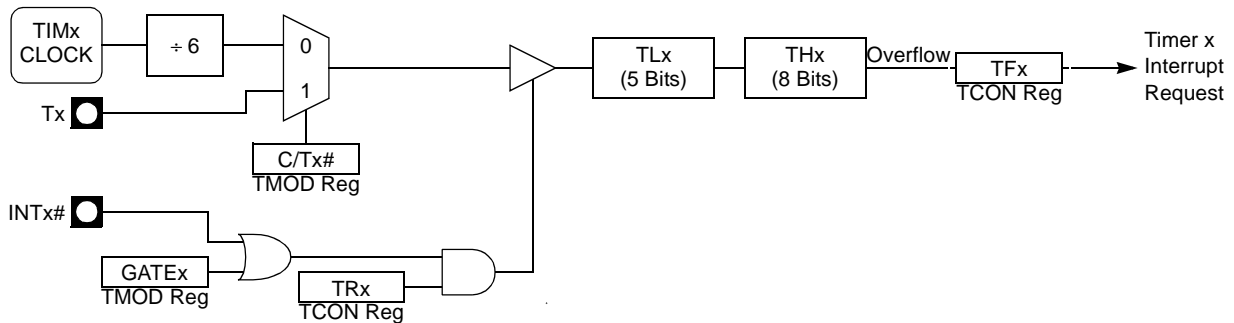
Timer 0 overflow (count rolls over from all 1s to all 0s) sets TF0 flag generating an interrupt request.

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 28). The upper three Bits of TL0 register are indeterminate and should be ignored. Prescaler overflow increments TH0 register. Figure 29 gives the overflow period calculation formula.

**Figure 28.** Timer/Counter x (x = 0 or 1) in Mode 0



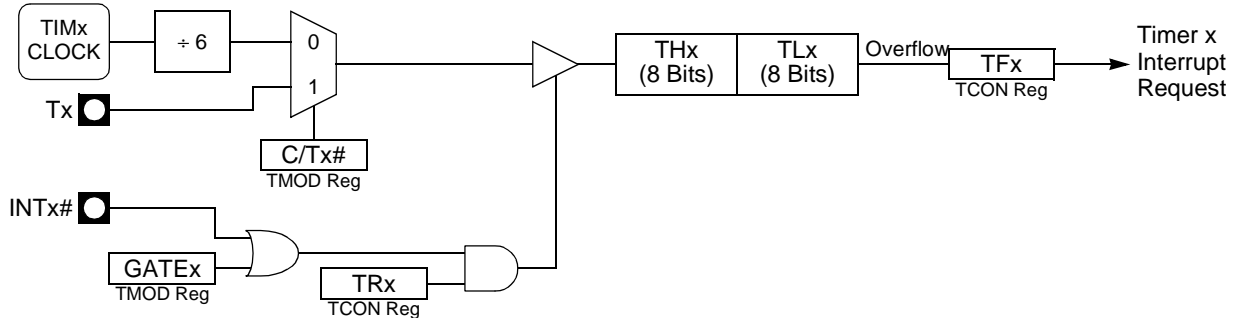
**Figure 29.** Mode 0 Overflow Period Formula

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

### 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 30). The selected input increments TL0 register. Figure 31 gives the overflow period calculation formula when in timer mode.

**Figure 30.** Timer/Counter x (x = 0 or 1) in Mode 1



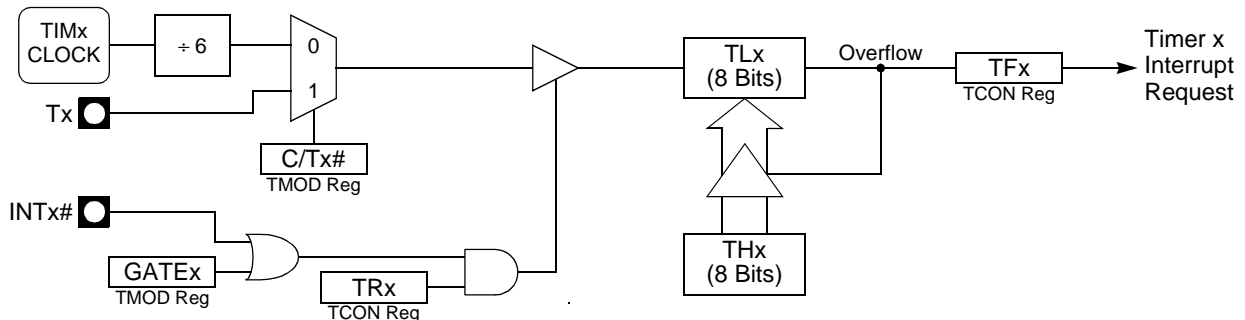
**Figure 31.** Mode 1 Overflow Period Formula

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

### 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 60). 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 33 gives the autoreload period calculation formula when in timer mode.

**Figure 32.** Timer/Counter x (x = 0 or 1) in Mode 2



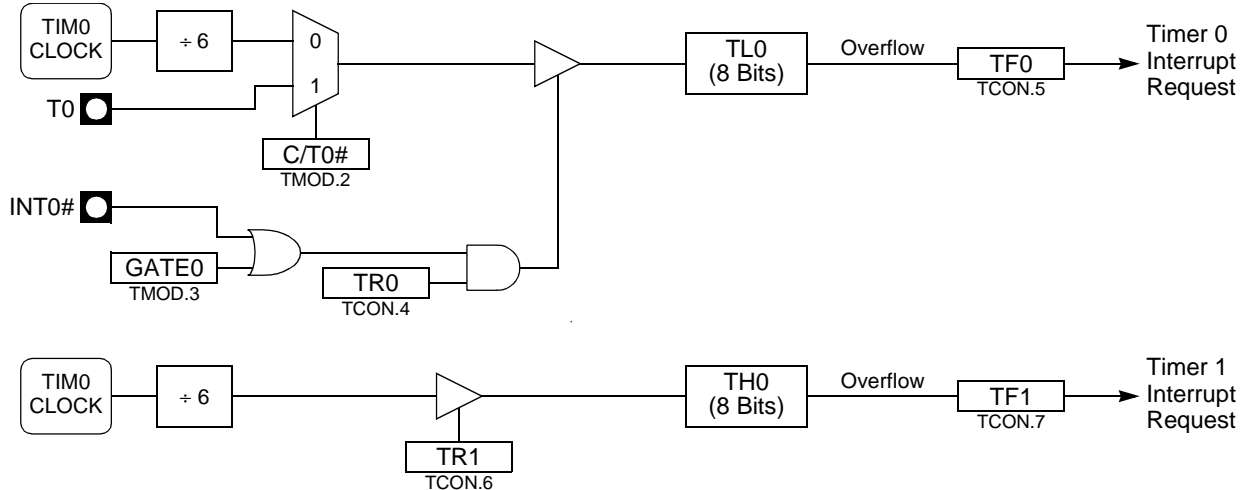
**Figure 33.** Mode 2 Autoreload Period Formula

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

### Mode 3 (Two 8-bit Timers)

Mode 3 configures Timer 0 such that registers TL0 and TH0 operate as separate 8-bit Timers (see Figure 34). 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  $F_{T1}/6$ ) 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 33 gives the autoreload period calculation formulas for both TF0 and TF1 flags.

**Figure 34.** Timer/Counter 0 in Mode 3: Two 8-bit Counters



**Figure 35.** Mode 3 Overflow Period Formula

$$TF0_{PER} = \frac{6 \cdot (256 - TL0)}{F_{TIM0}} \qquad TF1_{PER} = \frac{6 \cdot (256 - TH0)}{F_{TIM0}}$$

**Timer 1**

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

- Timer 1 functions as either a Timer or event Counter in three modes of operation. Figure 28 through Figure 32 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 Table 59) and Bits 2, 3, 6 and 7 of TCON register (see Figure 58). 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 28). 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 30). 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 32). 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.

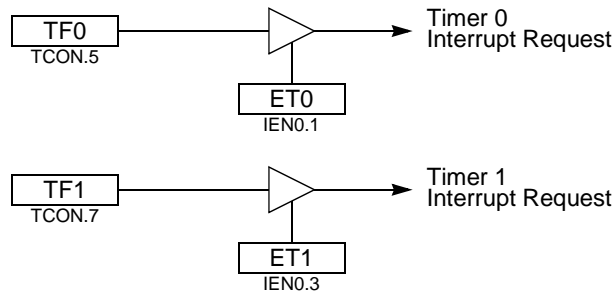
**Mode 3 (Halt)**

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 36.** Timer Interrupt System



Registers

**Table 58.** 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 59.** TMOD Register  
TMOD (89:h) - Timer/Counter 0 and 1 Modes

	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> <table style="width: 100%; border-collapse: collapse;"> <tr> <th style="width: 10%;"><u>M11</u></th> <th style="width: 10%;"><u>M01</u></th> <th style="width: 80%;"><u>Operating mode</u></th> </tr> <tr> <td>0</td> <td>0</td> <td>Mode 0: 8-bit Timer/Counter (TH1) with 5-bit prescaler (TL1).</td> </tr> <tr> <td>0</td> <td>1</td> <td>Mode 1: 16-bit Timer/Counter.</td> </tr> <tr> <td>1</td> <td>0</td> <td>Mode 2: 8-bit auto-reload Timer/Counter (TL1).<sup>(1)</sup></td> </tr> <tr> <td>1</td> <td>1</td> <td>Mode 3: Timer 1 halted. Retains count.</td> </tr> </table>	<u>M11</u>	<u>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.
<u>M11</u>	<u>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> <table style="width: 100%; border-collapse: collapse;"> <tr> <th style="width: 10%;"><u>M10</u></th> <th style="width: 10%;"><u>M00</u></th> <th style="width: 80%;"><u>Operating mode</u></th> </tr> <tr> <td>0</td> <td>0</td> <td>Mode 0: 8-bit Timer/Counter (TH0) with 5-bit prescaler (TL0).</td> </tr> <tr> <td>0</td> <td>1</td> <td>Mode 1: 16-bit Timer/Counter.</td> </tr> <tr> <td>1</td> <td>0</td> <td>Mode 2: 8-bit auto-reload Timer/Counter (TL0).<sup>(2)</sup></td> </tr> <tr> <td>1</td> <td>1</td> <td>Mode 3: TL0 is an 8-bit Timer/Counter. TH0 is an 8-bit Timer using Timer 1's TR0 and TF0 Bits.</td> </tr> </table>	<u>M10</u>	<u>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.
<u>M10</u>	<u>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																

Reset Value = 0000 0000b

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

**Table 60.** 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 61.** 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 62.** 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 63.** 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 AT8xC5132 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

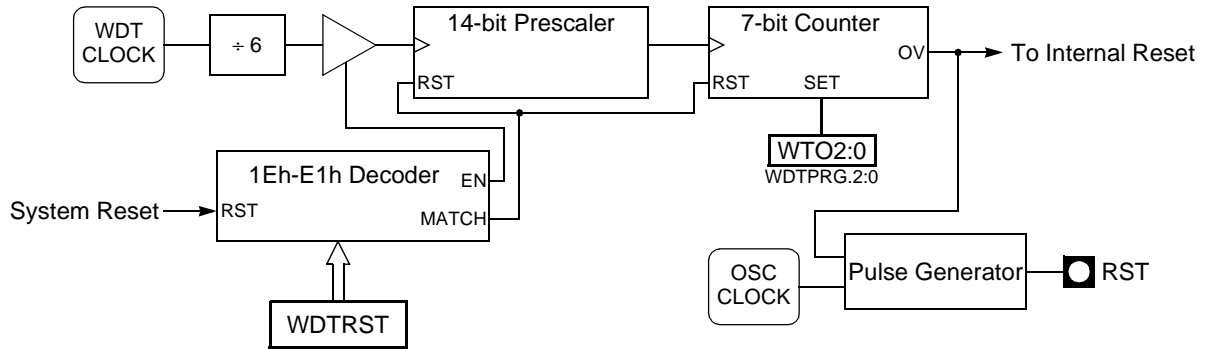
The WDT consists of a 14-bit prescaler followed by a 7-bit programmable counter. As shown in Figure 37, the 14-bit prescaler is fed by the WDT clock detailed in section "Watchdog Clock Controller", page 58.

The Watchdog Timer Reset register (WDTRST, see Table 64) provides control access to the WDT, while the Watchdog Timer Program register (WDTPRG, see Figure 65) 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 two-byte sequence to the WDTRST register clears and enables the WDT.

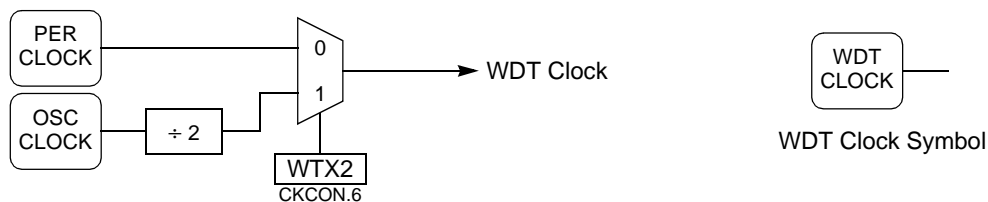
**Figure 37.** WDT Block Diagram



### Watchdog Clock Controller

As shown in Figure 38 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 38.** 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.

The WDT time-out period can be adjusted using WTO2:0 Bits located in the WDTPRG register accordingly to the formula shown in Figure 39. In this formula, WTOval represents the decimal value of WTO2:0 Bits. Table 65 reports the time-out period depending on the WDT frequency.

**Figure 39.** WDT Time-Out Formula

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

WTO2	WTO1	WTO0	F <sub>WDT</sub>					
			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 ms	12.28 ms	9.83 ms	8.19 ms	6.14 ms	4.92 ms
0	0	1	32.77 ms	24.57 ms	19.66 ms	16.38 ms	12.28 ms	9.83 ms
0	1	0	65.54 ms	49.14 ms	39.32 ms	32.77 ms	24.57 ms	19.66 ms
0	1	1	131.07 ms	98.28 ms	78.64 ms	65.54 ms	49.14 ms	39.32 ms
1	0	0	262.14 ms	196.56 ms	157.29 ms	131.07 ms	98.28 ms	78.64 ms
1	0	1	524.29 ms	393.12 ms	314.57 ms	262.14 ms	196.56 ms	157.29 ms
1	1	0	1.05 s	786.24 ms	629.15 ms	524.29 ms	393.12 ms	314.57 ms
1	1	1	2.10 s	1.57 s	1.26 s	1.05 s	786.24 ms	629.15 ms

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

## 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 AT8xC5132 are in Idle mode. This means that the user 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 AT8xC5132 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 64.** 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

**Table 65.** 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 values read from these Bits are indeterminate. Do not set these Bits.					
2-0	WTO2:0	<b>Watchdog Timer Time-Out Selection Bits</b> Refer to Table 64 for time-out periods.					

Reset Value = XXXX X000b

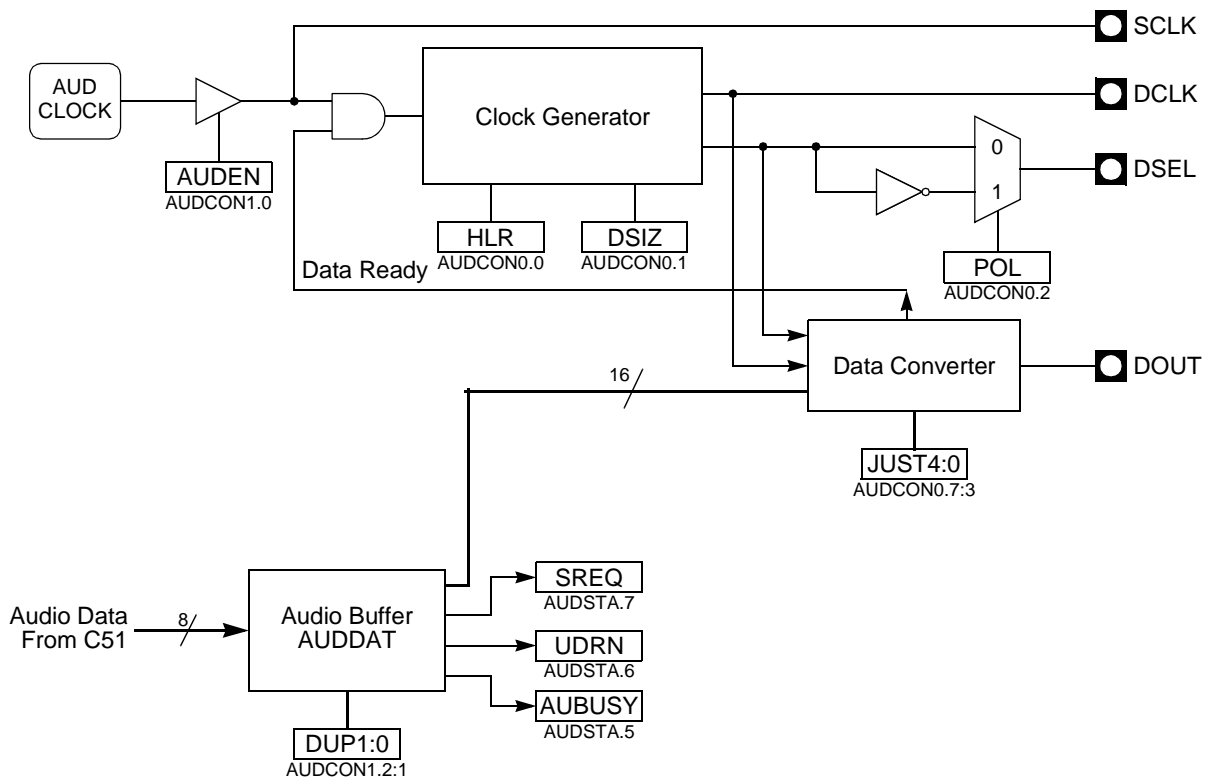
## Audio Output Interface

The AT8xC5132 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.

### Description

The C51 core interfaces to the audio interface through five special function registers: AUDCON0 and AUDCON1, the Audio Control registers (see Table 67 and Table 68); AUDSTA, the Audio Status register (see Table 69); AUDDAT, the Audio Data register (see Table 70); and AUDCLK, the Audio Clock Divider register (see Table 71). Figure 40 shows the audio interface block diagram, blocks are detailed in the following sections.

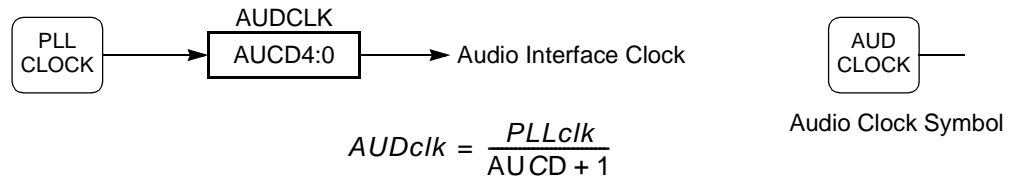
Figure 40. Audio Interface Block Diagram



## Clock Generator

The audio interface clock is generated by division of the PLL clock. The division factor is given by AUCD4:0 Bits in AUDCLK register. Figure 41 shows the audio interface clock generator and its calculation formula. The audio interface clock frequency depends on the audio DAC used.

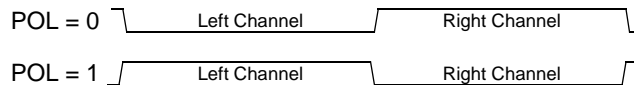
**Figure 41.** Audio Clock Generator and Symbol



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 62), 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 42.

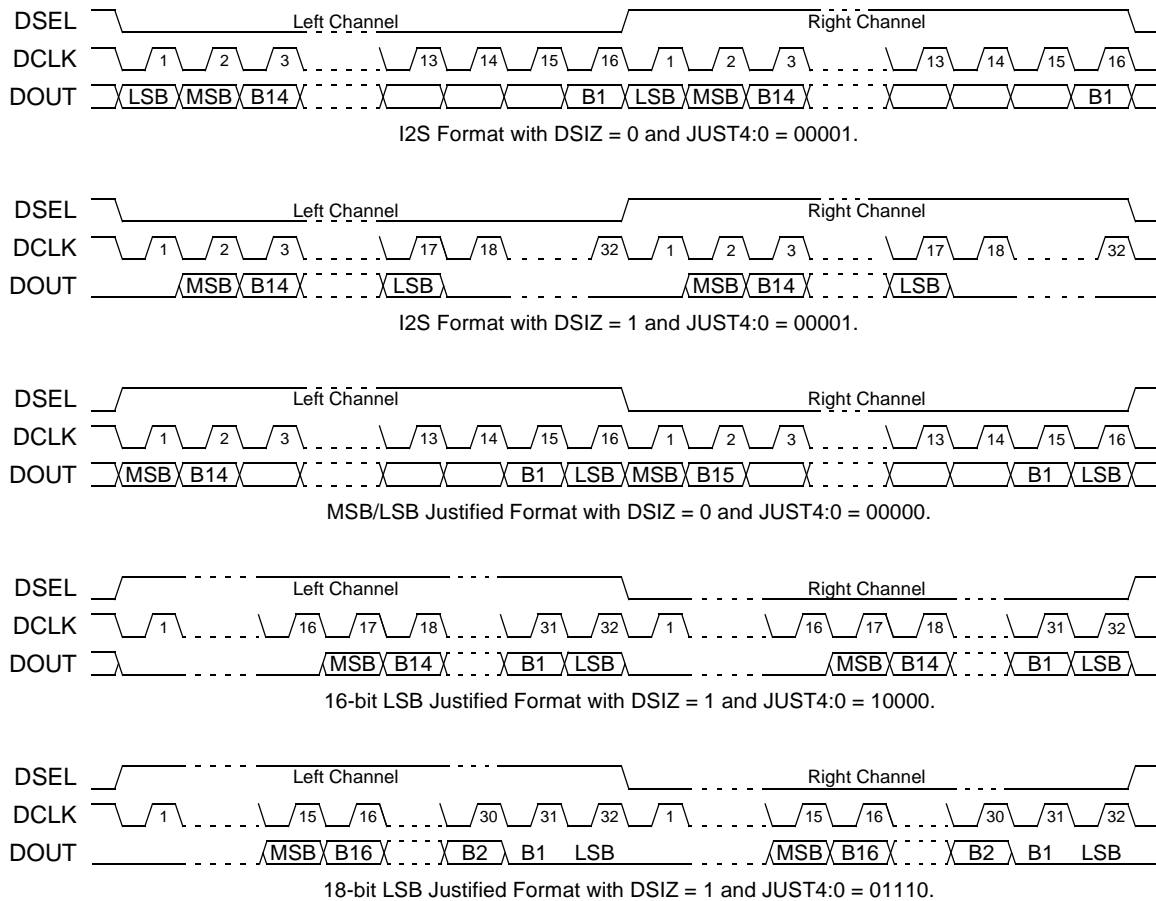
**Figure 42.** DSEL Output Polarity



## 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 43, 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 43. Audio Output Format



The data converter receives its audio stream from two sources selected by the SRC bit in AUDCON1 register.

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 66.

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 64. 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 66.** Sample Duplication Factor

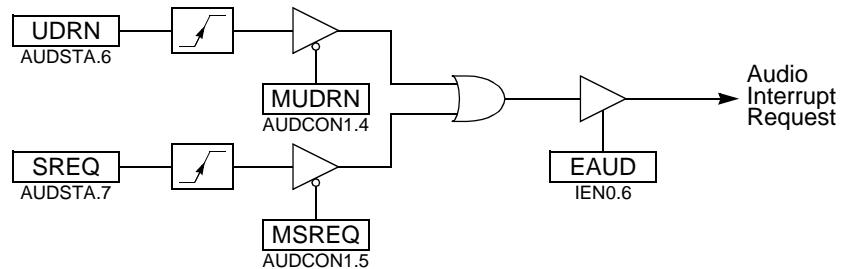
DUP1	DUP0	Factor
0	0	No sample duplication, DAC rate = 8 kHz (C51 rate).
0	1	One sample duplication, DAC rate = 16 kHz (2 x C51 rate).
1	0	Two samples duplication, DAC rate = 32 kHz (4 x C51 rate).
1	1	Three samples duplication, DAC rate = 48 kHz (6 x C51 rate).

## Interrupt Request

The audio interrupt request can be generated by two 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 two 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.

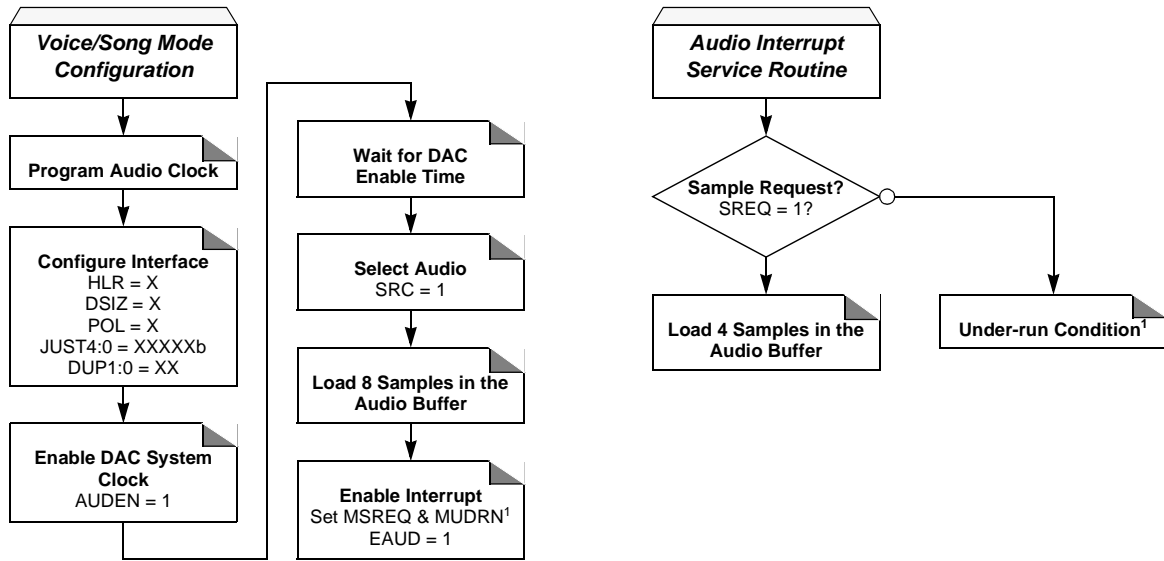
**Figure 44.** Audio Interface Interrupt System



## 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. The data flow sent by the C51 is then regulated by interrupt and data is loaded 4 Bytes by 4 Bytes. Figure 45 shows the configuration flow of the audio interface when in voice or sound mode.

Figure 45. Voice or Sound Mode Audio Flows



Note: 1. An under-run occurrence signifies that the 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 67.** 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 62 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 68.** AUDCON1 Register

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

7	6	5	4	3	2	1	0
–	–	MSREQ	MUDRN	–	DUP1	DUP0	AUDEN
Bit Number	Bit Mnemonic	Description					
7	–	<b>Reserved</b> The values read from this bit is always 0. Do not set this bit.					
6	–	<b>Reserved</b> The values read from this bit is always 0. Do not set this bit.					
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 values read from this bit is always 0. Do not set this bit.					
2 - 1	DUP1:0	<b>Audio Duplication Factor</b> Refer to Table 66 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 69.** 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 cannot accept more sample (buffer full). Cleared by hardware when buffer is no more full.
4-0	-	<b>Reserved</b> The values read from these Bits are always 0. Do not set these Bits.

Reset Value = 1100 0000b

**Table 70.** 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 71.** 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 values read from these Bits are 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 AT8xC5132 implement a USB device controller supporting Full-speed data transfer. In addition to the default control endpoint 0, it provides 3 other endpoints, which can be configured in Control, Bulk, Interrupt or Isochronous types.

This allows to develop firmware conforming to most USB device classes, for example the AT8xC5132 support:

- USB Mass Storage Class Control/Bulk/Interrupt (CBI) Transport, Revision 1.0 – December 14, 1998
- USB Mass Storage Class Bulk-Only Transport, Revision 1.0 – September 31, 1999
- USB Device Firmware Upgrade Class, Revision 1.0 – May 13, 1999

### USB Mass Storage Class CBI Transport

Within the CBI framework, the Control endpoint is used to transport command blocks as well as to transport standard USB requests. One Bulk Out endpoint is used to transport data from the host to the device. One Bulk In endpoint is used to transport data from the device to the host. And one interrupt endpoint may also be used to signal command completion (protocol 0) but it is optional and may not be used (protocol 1).

The following AT8xC5132 configuration adheres to that requirements:

- Endpoint 0: 32 Bytes, Control In-Out
- Endpoint 1: 64 Bytes, Bulk Out
- Endpoint 2: 64 Bytes, Bulk In
- Endpoint 3: 8 Bytes, Interrupt In

### 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. No interrupt endpoint is needed.

The following AT8xC5132 configuration adheres to that requirements:

- Endpoint 0: 32 Bytes, Control In-Out
- Endpoint 1: 64 Bytes, Bulk Out
- Endpoint 2: 64 Bytes, Bulk In
- Endpoint 3: not used

### USB Device Firmware Upgrade (DFU)

The USB Device Firmware Update (DFU) protocol can be used to upgrade the on-chip Flash memory of the AT89C5132. This allows installing product enhancements and patches to devices that are already in the field. Two 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 shall be USB Mass Storage for AT89C5132. 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**

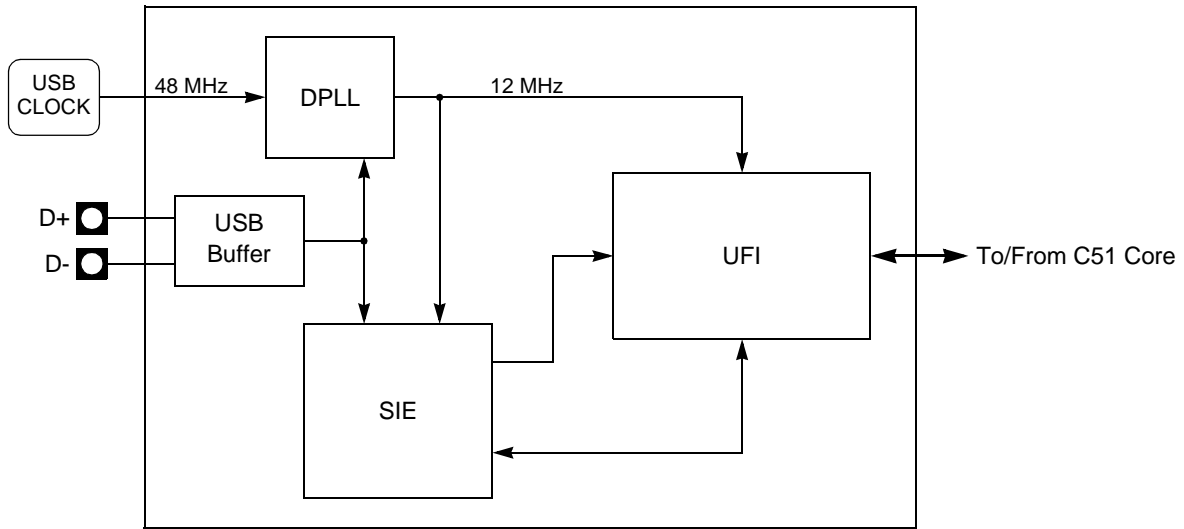
The USB device controller provides the hardware that the AT8xC5132 need to interface a USB link to data flow stored in a double port memory.

It requires a 48 MHz reference clock provided by the clock controller as detailed in Section "Clock Controller", page 69. 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.

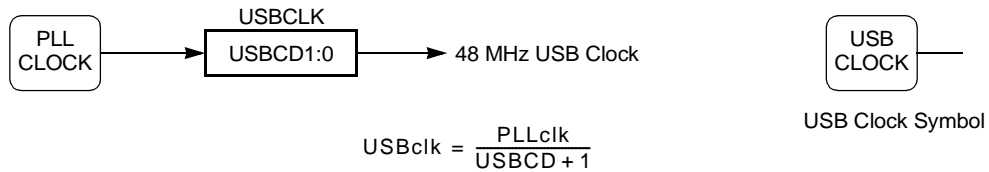
**Figure 46.** USB Device Controller Block Diagram



**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 86). Figure 47 shows the USB controller clock generator and its calculation formula. The USB controller clock frequency must always be 48 MHz.

**Figure 47.** USB Clock Generator and Symbol

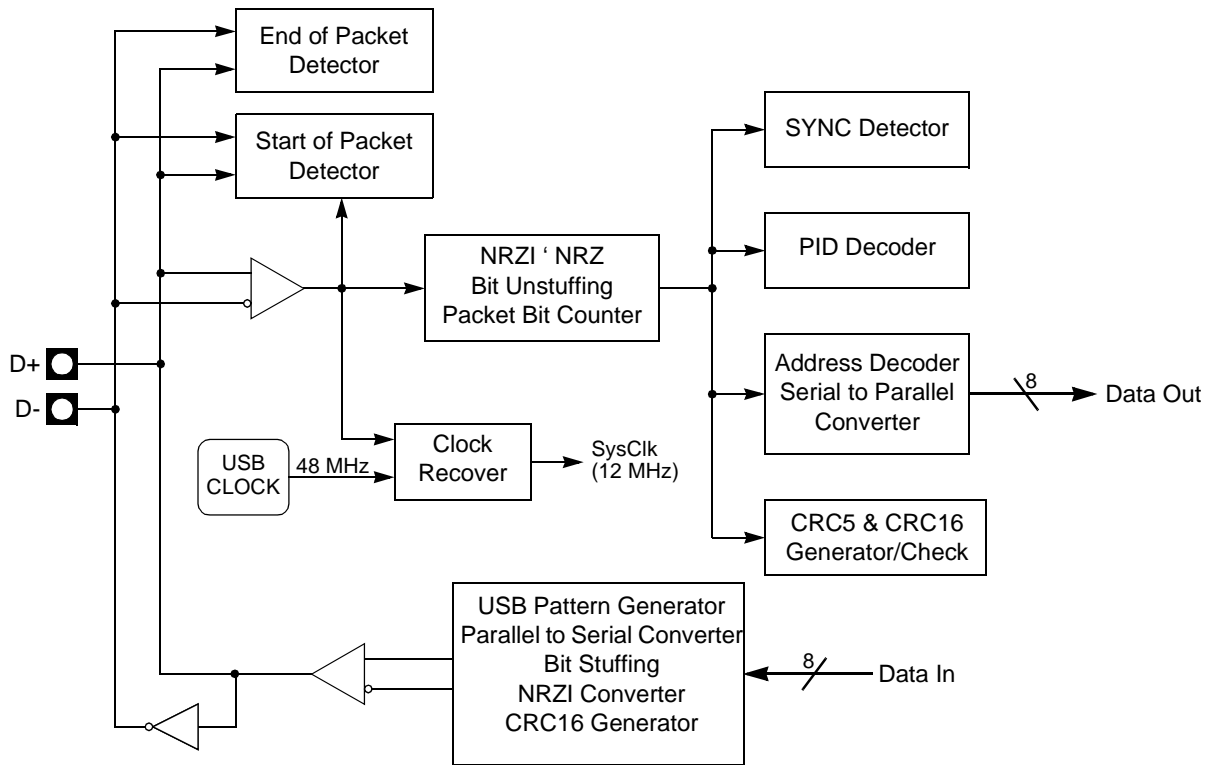


### Serial Interface Engine (SIE)

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 48. SIE Block Diagram



### Function Interface Unit (UFI)

The Function Interface Unit provides the interface between the AT8xC5132 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 50 shows typical USB IN and OUT transactions reporting the split in the hardware (UFI) and software (C51) load.

Figure 49. UFI Block Diagram

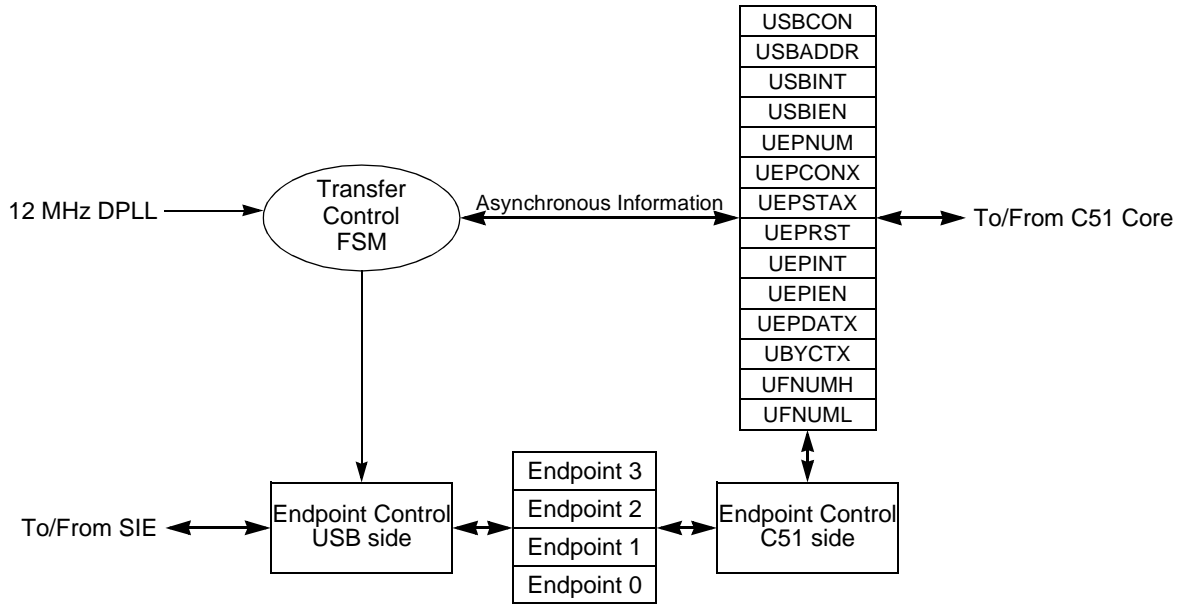
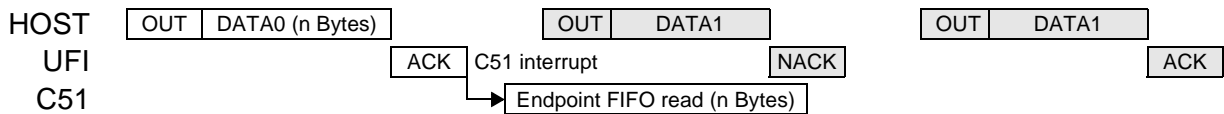


Figure 50. USB Typical Transaction Load

**OUT Transactions:**



**IN Transactions:**



**USB Interrupt System**

As shown in Figure 51, the USB controller of the AT8xC5132 handle sixteen interrupt sources. These sources are separated in two groups: the endpoints interrupts and the controller interrupts, combined together to appear as single interrupt source for the C51 core. The USB interrupt is enabled by setting the EUSB bit in IEN1.

**Controller Interrupt Sources**

There are four controller interrupt sources which can be enabled separately in USBIEN:

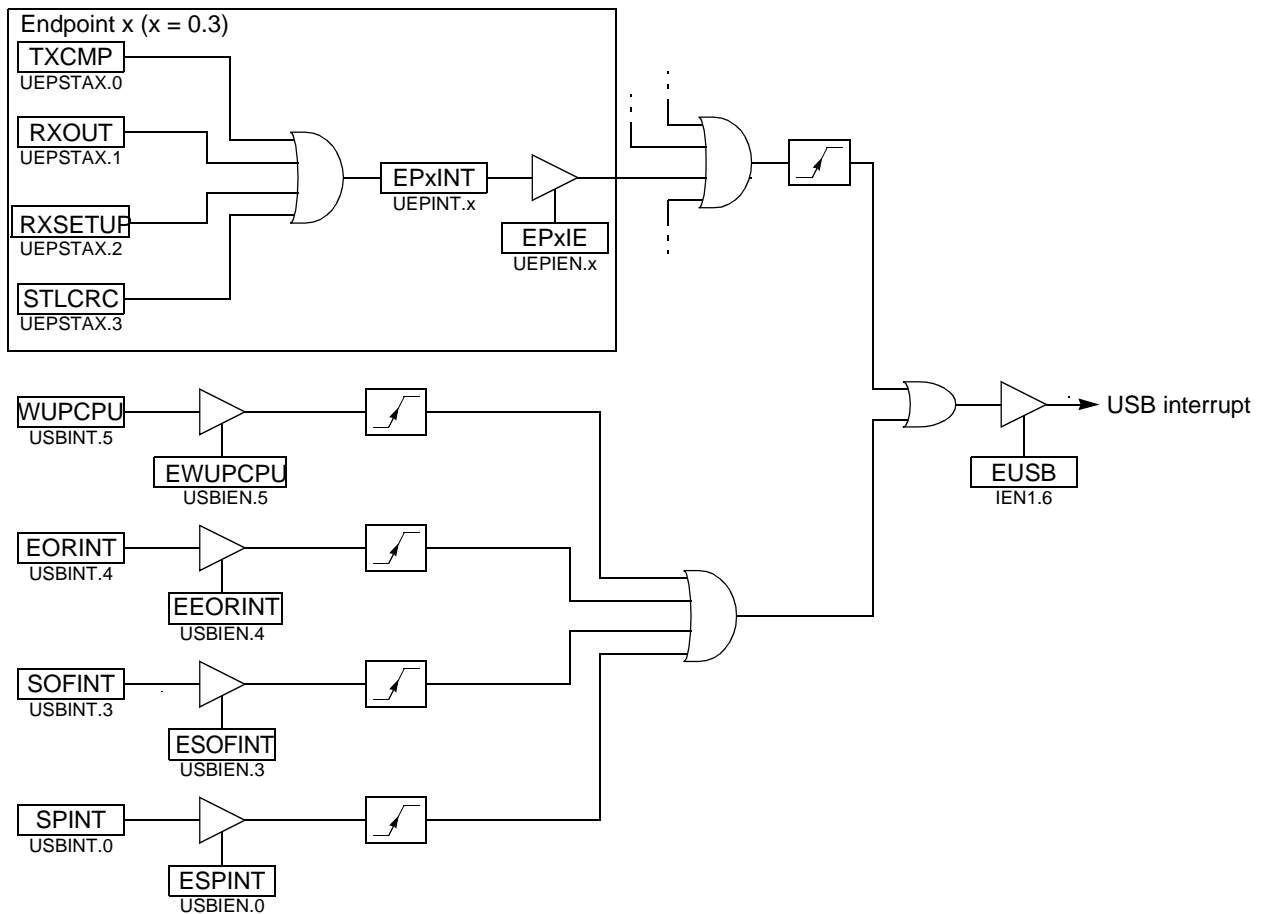
- **SPINT: Suspend Interrupt Flag.**  
This flag triggers an interrupt when a USB Suspend (Idle bus for three frame periods: a J state for 3 ms) is detected.
- **SOFINT: Start Of Frame Interrupt Flag.**  
This flag triggers an interrupt when a USB start of frame packet has been received.
- **EORINT: End Of Reset Interrupt Flag.**  
This flag triggers an interrupt when a End Of Reset has been detected by the USB controller.
- **WUPCPU: Wake Up CPU Interrupt Flag.**  
This flag triggers an interrupt when the USB controller is in SUSPEND state and is re-activated by a non-idle signal from USB line.

## Endpoint Interrupt Sources

Each endpoint supports four interrupt sources reported in UEPSTAX and combined together to appear as a single endpoint interrupt source in UEPINT. Each endpoint interrupt can be enabled separately in UEPIEN.

- **TXCMP:** Transmitted In Data Interrupt Flag.  
This flag triggers an interrupt after an IN packet has been transmitted for Isochronous endpoints or after it has been accepted (ACK'ed) by the host for Control, Bulk and Interrupt endpoints.
- **RXOUT:** Received Out Data Interrupt Flag.  
This flag triggers an interrupt after a new packet has been received.
- **RXSETUP:** Receive Setup Interrupt Flag.  
This flag triggers an interrupt when a valid SETUP packet has been received from the host.
- **STLCRC:** Stall Sent Interrupt Flag/CRC Error Interrupt Flag.  
This flag triggers an interrupt after a STALL handshake has been sent on the bus, for Control, Bulk and Interrupt endpoints.  
This flag triggers an interrupt when the last data received is corrupted for Isochronous endpoints.

**Figure 51.** USB Interrupt Control Block Diagram



Registers

Table 72. USBCON Register

USBCON (S:BCh) – USB Global Control Register

7	6	5	4	3	2	1	0
USBE	SUSPCLK	SDRMWUP	-	UPRSM	RMWUPE	CONFIG	FADDEN
Bit Number	Bit Mnemonic	Description					
7	USBE	<b>USB Enable Bit</b> Set to enable the USB controller. Clear to disable and reset the USB controller.					
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 values 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 enable request an upstream resume signalling 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	CONFIG	<b>Configuration Bit</b> Set after a SET_CONFIGURATION request with a non-zero value has been correctly processed. Cleared by software when a SET_CONFIGURATION request with a zero value is received. Cleared by hardware on hardware reset or when an USB reset is detected on the bus.					
0	FADDEN	<b>Function Address Enable Bit</b> Set by the device firmware after a successful status phase of a SET_ADDRESS transaction. It shall not be cleared afterwards by the device firmware. Cleared by hardware on hardware reset or when an USB reset is received. When this bit is cleared, the default function address is used (0).					

Reset Value = 0000 0000b

**Table 73.** 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	<p><b>Function Enable Bit</b> Set to enable the function. The device firmware shall 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.</p>
6-0	UADD6:0	<p><b>USB Address Bits</b> This field contains the default address (0) after power-up or USB bus reset. It shall be written with the value set by a SET_ADDRESS request received by the device firmware.</p>

Reset Value = 0000 0000b

**Table 74.** 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	-	<p><b>Reserved</b> The values read from these Bits are always 0. Do not set these Bits.</p>
5	WUPCPU	<p><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.</p>
4	EORINT	<p><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.</p>
3	SOFINT	<p><b>Start of Frame Interrupt Flag</b> Set by hardware when a USB Start of Frame packet (SOF) has been properly received. This triggers a USB interrupt when ESOFINT is set in USBIEN. Cleared by software.</p>
2 - 1	-	<p><b>Reserved</b> The values read from these Bits are always 0. Do not set these Bits.</p>
0	SPINT	<p><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.</p>

Reset Value = 0000 0000b

**Table 75.** USBIEN Register

USBIEN (S:BEh) – USB Global Interrupt Enable Register

	7	6	5	4	3	2	1	0
	-	-	EWUPCPU	EERINT	ESOFINT	-	-	ESPINT

Bit Number	Bit Mnemonic	Description
7-6	-	<b>Reserved</b> The values read from these Bits are 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	EEOFINT	<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 values read from these Bits are 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 76.** 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 values read from these Bits are 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 shall be accessed when reading or writing to registers UEPSTAX, UEPDATX, UBYCTLX or UEPCONX.

Reset Value = 0000 0000b

**Table 77.** UEPCONX Register

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

	7	6	5	4	3	2	1	0
	EPEN	-	-	-	DTGL	EPDIR	EPTYPE1	EPTYPE0

Bit Number	Bit Mnemonic	Description												
7	EPEN	<p><b>Endpoint Enable Bit</b>            Set to enable the endpoint according to the device configuration. Endpoint 0 shall 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.</p>												
6 - 4	-	<p><b>Reserved</b>            The values read from this bit is always 0. Do not set this bit.</p>												
3	DTGL	<p><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.            Note: When a new data packet is received without DTGL toggling from 1 to 0 or 0 to 1, a packet may have been lost. When this occurs for a Bulk endpoint, the device firmware shall consider the host has retried transmitting a properly received packet because the host has not received a valid ACK, then the firmware shall discard the new packet (N.B. The endpoint resets to DATA0 only upon configuration).            For interrupt endpoints, data toggling is managed as for Bulk endpoints when used.            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 shall retrieve every new data packet and may ignore this bit.</p>												
2	EPDIR	<p><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.</p>												
1 - 0	EPTYPE1: 0	<p><b>Endpoint Type Bits</b>            Set this field according to the endpoint configuration (Endpoint 0 shall always be configured as Control):</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 5%;">0</td> <td style="width: 5%;">0</td> <td style="width: 90%;">Control endpoint</td> </tr> <tr> <td>0</td> <td>1</td> <td>Isochronous endpoint</td> </tr> <tr> <td>1</td> <td>0</td> <td>Bulk endpoint</td> </tr> <tr> <td>1</td> <td>1</td> <td>Interrupt endpoint</td> </tr> </table>	0	0	Control endpoint	0	1	Isochronous endpoint	1	0	Bulk endpoint	1	1	Interrupt endpoint
0	0	Control endpoint												
0	1	Isochronous endpoint												
1	0	Bulk endpoint												
1	1	Interrupt endpoint												

Reset Value = 0000 0000b

**Table 78.** UEPSTAX Register

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

7	6	5	4	3	2	1	0
DIR	-	STALLRQ	TXRDY	STLCRC	RXSETUP	RXOUT	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	-	<p><b>Reserved</b>            The values read from this Bits are always 0. Do not set this bit.</p>					
5	STALLRQ	<p><b>Stall Handshake Request Bit</b>            Set to send a STALL answer to the host for the next handshake. Clear otherwise.</p>					
4	TXRDY	<p><b>TX Packet Ready Control Bit</b>            Set after a packet has been written into the endpoint FIFO for IN data transfers. Data shall 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.</p>					
3	STLCRC	<p><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.</p>					
2	RXSETUP	<p><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.</p>					
1	RXOUT	<p><b>Received OUT Data Interrupt Flag</b>            Set by hardware after an OUT packet has been received. Then, the endpoint interrupt is triggered if enabled in UEPIEN and all the following OUT packets to the endpoint are rejected (NACK'ed) until this bit is cleared. 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. Clear by software after reading the OUT data from the endpoint FIFO.</p>					
0	TXCMP	<p><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.</p>					

Reset Value = 0000 0000b



**Table 79.** UEPRST Register

UEPRST (S:D5h) – USB Endpoint FIFO Reset Register

7	6	5	4	3	2	1	0
-	-	-	-	EP3RST	EP2RST	EP1RST	EP0RST

Bit Number	Bit Mnemonic	Description
7 - 4	-	<b>Reserved</b> The values read from these Bits are always 0. Do not set these Bits.
3	EP3RST	<b>Endpoint 3 FIFO Reset</b> Set and clear to reset the endpoint 3 FIFO prior to any other operation, upon hardware reset or when an USB bus reset has been received.
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 80.** UEPINT Register

UEPINT (S:F8h Read-only) – USB Endpoint Interrupt Register

7	6	5	4	3	2	1	0
-	-	-	-	EP3INT	EP2INT	EP1INT	EP0INT

Bit Number	Bit Mnemonic	Description
7 - 4	-	<b>Reserved</b> The values read from these Bits are always 0. Do not set these Bits.
3	EP3INT	<b>Endpoint 3 Interrupt Flag</b> Set by hardware when an interrupt is triggered in UEPSTAX and the endpoint 3 interrupt is enabled in UEPIEN. Must be cleared by software.
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 81.** UEPIEN Register  
UEPIEN (S:C2h) – USB Endpoint Interrupt Enable Register

7	6	5	4	3	2	1	0
-	-	-	-	EP3INTE	EP2INTE	EP1INTE	EP0INTE

Bit Number	Bit Mnemonic	Description
7 - 4	-	<b>Reserved</b> The values read from these Bits are always 0. Do not set these Bits.
3	EP3INTE	<b>Endpoint 3 Interrupt Enable Bit</b> Set to enable the interrupts for endpoint 3. Clear to disable the interrupts for endpoint 3.
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 82.** 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 83.** UBYCTLX 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 values read from this Bits are 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 84.** 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 85.** 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 values read from these Bits are 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 values read from this Bits are 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 86.** 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 values read from these Bits are 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 AT8xC5132 implements a MultiMedia Card (MMC) controller. The MMC is used to store 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 minimal number of signals.

### Card Signals

The communication signals are:

- CLK: with each cycle of this signal an 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 bidirectional command channel used for card initialization and data transfer commands. The CMD signal has two 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 bidirectional 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 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 bidirectional.

## 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 bidirectional 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 an 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.

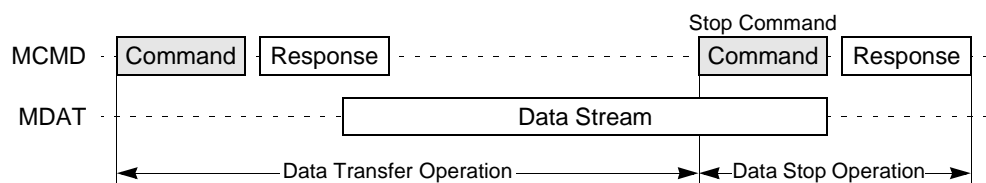
MultiMedia Card bus data transfers are composed of these tokens. One data transfer is a bus operation. There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have data token, the others transfer their information directly 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.

Two types of data transfer commands are defined:

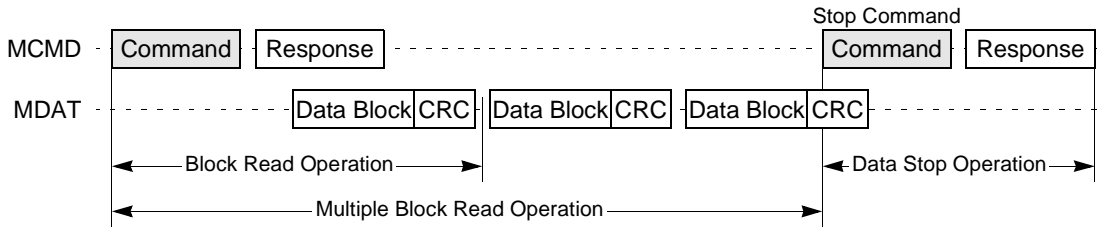
- Sequential commands: These commands initiate a continuous data stream, they are terminated only when a stop command follows on the MCMD line. This mode reduces the command overhead to an absolute minimum.
- Block-oriented commands: These commands send data block succeeded by CRC Bits. Both read and 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 52 to Figure 56 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 52.** Sequential Read Operation

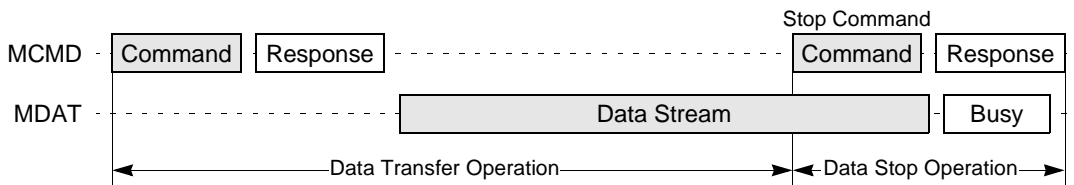


**Figure 53.** (Multiple) Block Read Operation

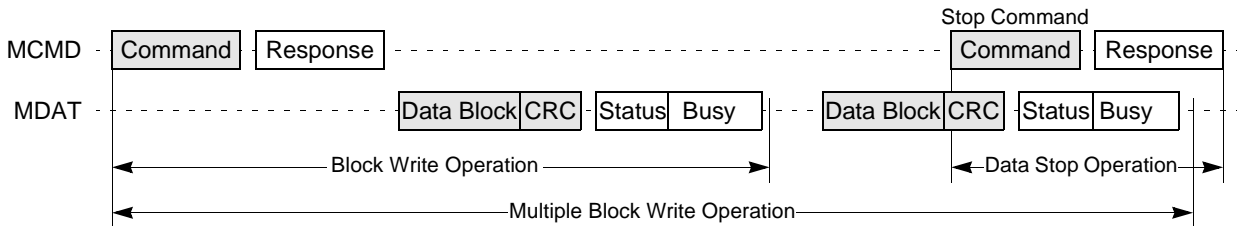


As shown in Figure 54 and Figure 55 the data write operation uses a simple busy signaling of the write operation duration on the data line (MDAT).

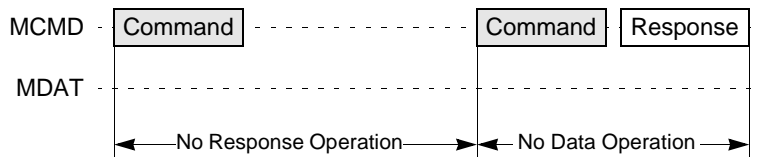
**Figure 54.** Sequential Write Operation



**Figure 55.** (Multiple) Block Write Operation



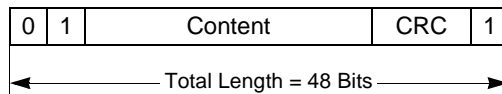
**Figure 56.** No Response and No Data Operation



**Command Token Format**

As shown in Figure 57, 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 57.** Command Token Format



**Table 87.** Command Token Format

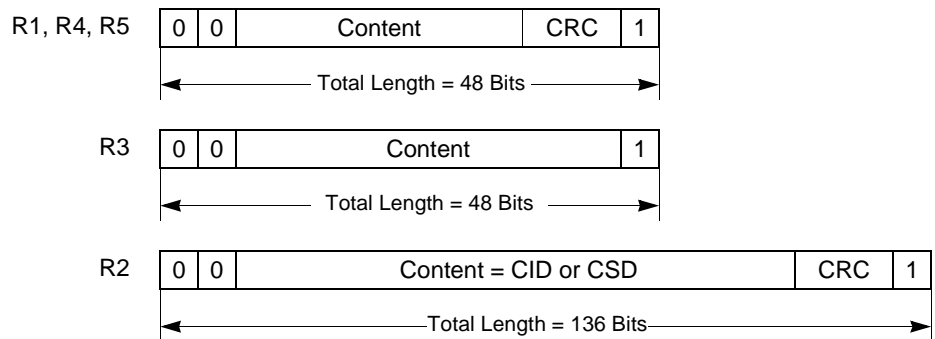
<b>Bit Position</b>	<b>47</b>	<b>46</b>	<b>45:40</b>	<b>39:8</b>	<b>7:1</b>	<b>0</b>
<b>Width (Bits)</b>	1	1	6	32	7	1
<b>Value</b>	'0'	'1'	-	-	-	'1'
<b>Description</b>	Start bit	Transmission bit	Command Index	Argument	CRC7	End bit

**Response Token Format**

There are five types of response tokens (R1 to R5). As shown in Figure 58, 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 58.** Response Token Format



**Table 88.** 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 89.** 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 90.** 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 91.** 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

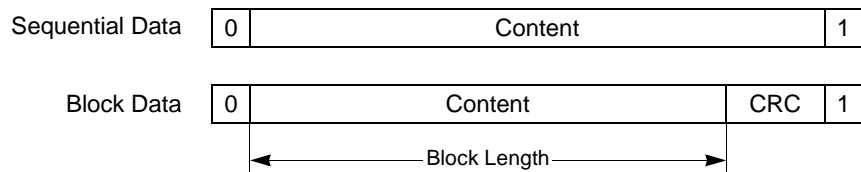
**Table 92.** R5 Response Format

<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 two types of data packets: stream and block. As shown in Figure 59, 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 59.** 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.

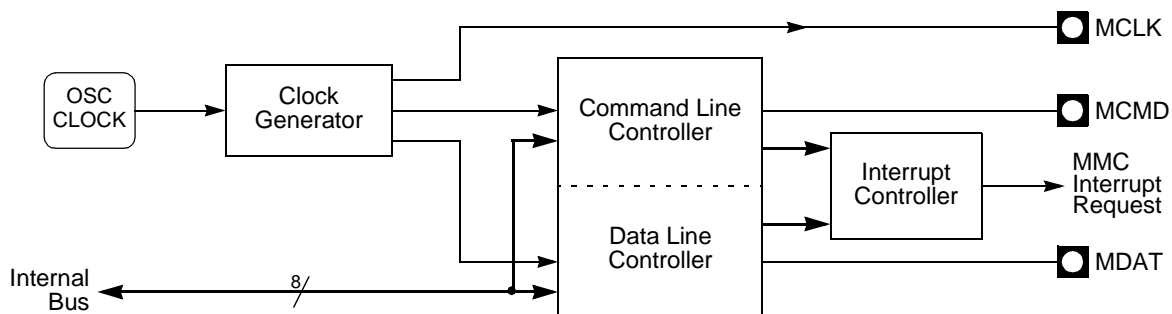
**Description**

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

MMCON0, MMCON1, MMCON2, the three MMC control registers (see Figure 94 to Figure ); MMSTA, the MMC status register (see Figure 97); MMINT, the MMC interrupt register (see Figure ); MMMSK, the MMC interrupt mask register (see Figure 99); MMCMD, the MMC command register (see Figure 100); MMDAT, the MMC data register (see Figure ); and MMCLK, the MMC clock register (see Figure 102).

As shown in Figure 60, 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.

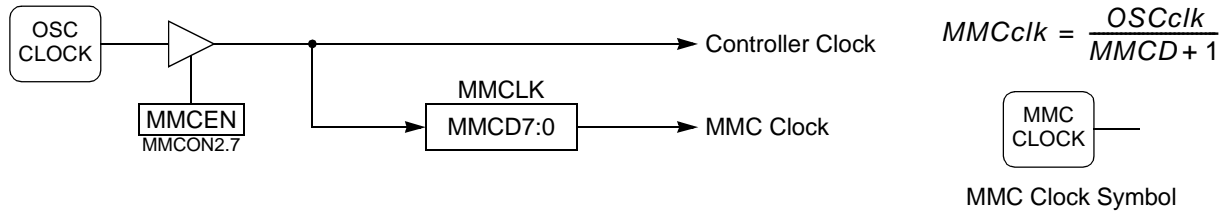
**Figure 60.** MMC Controller Block Diagram



## 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. Figure 61 shows the MMC clock generator and its output clock calculation formula.

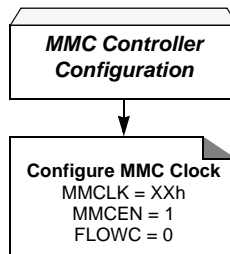
**Figure 61.** MMC Clock Generator and Symbol



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 62 shows the MMC controller configuration flow.

As exposed in Section "Clock Control", MMCD7:0 Bits can be used to dynamically increase or reduce the MMC clock.

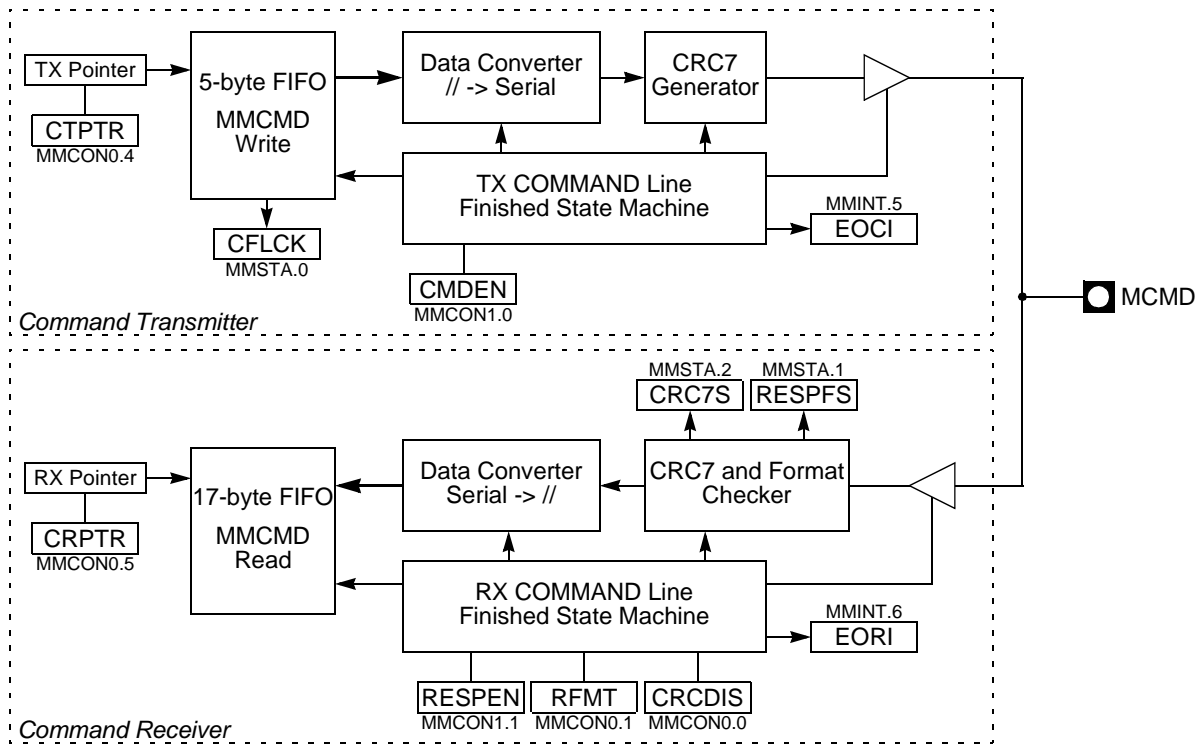
**Figure 62.** Configuration Flow



**Command Line Controller**

As shown in Figure 63, the command line controller is divided in two 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 63.** Command Line Controller Block Diagram



**Command Transmitter**

To send a command to the card, the 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, the 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 responses that do not include CRC7.

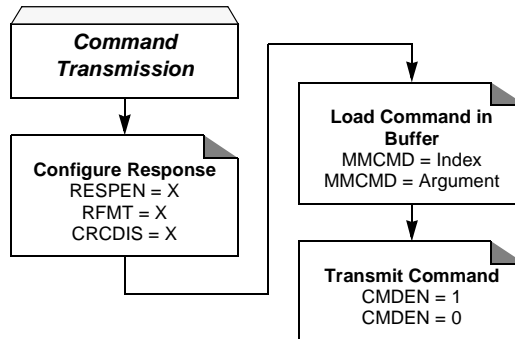
Figure 64 summarizes the command transmission flow.

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 over-run.

The end of the command transmission is signalled by the EOCI flag in MMINT register becoming set. This flag may generate an MMC interrupt request as detailed in Section "Interrupt", page 97. The end of the command transmission also resets the CFLCK flag.

The 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 64.** Command Transmission Flow



### Command Receiver

The end of the response reception is signalled by the EORI flag in MMINT register. This flag may generate an MMC interrupt request as detailed in Section "Interrupt", page 97. When this flag is set, two 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.

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

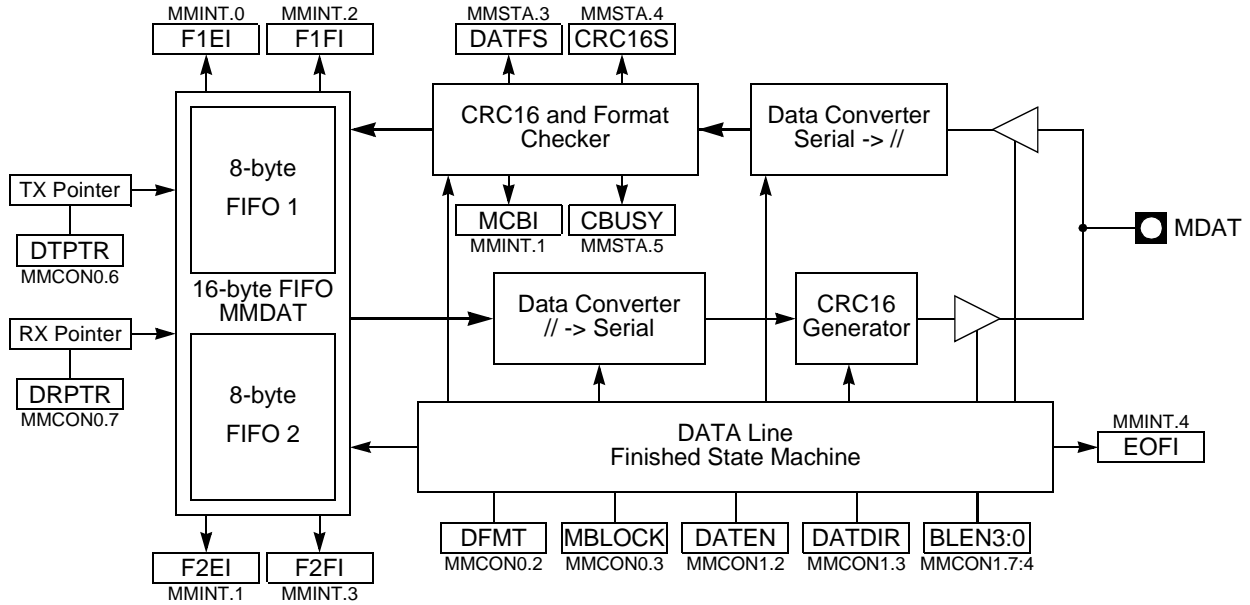
According to the MMC specification delay between a command and a response (formally  $N_{CR}$  parameter) cannot exceed 64 MMC clock periods. To avoid any locking of the MMC controller when card does not send its response (e.g. physically removed from the bus), user must launch a timeout period to exit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

This timeout 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 65.** Data Line Controller Block Diagram



**FIFO Implementation**

The 16-byte FIFO is based on a dual 8-byte FIFO managed using two 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”.

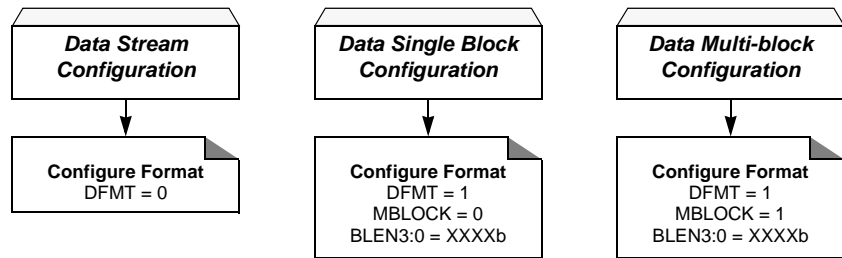
**Data Configuration**

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 93. Figure 66 summarizes the data modes configuration flows.

**Table 93.** Block Length Programming

BLEN3:0	Block Length (Byte)
BLEN = 0000 to 1011	Length = 2 <sup>BLEN</sup> : 1 to 2048
> 1011	Reserved: do not program BLEN3:0 > 1011

**Figure 66. 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 67 summarizes the data stream transmission flows in both polling and interrupt modes while Figure 68 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 2 MMC clock periods when DATD1:0 Bits are cleared to 8 MMC clock periods when DATD2:0 Bits are set, by step of 2 MMC clock periods.

*End of Transmission*

The end of data frame (block or stream) transmission is signalled by the EOFI flag in MMINT register. This flag may generate an MMC interrupt request as detailed in Section "Interrupt", page 97.

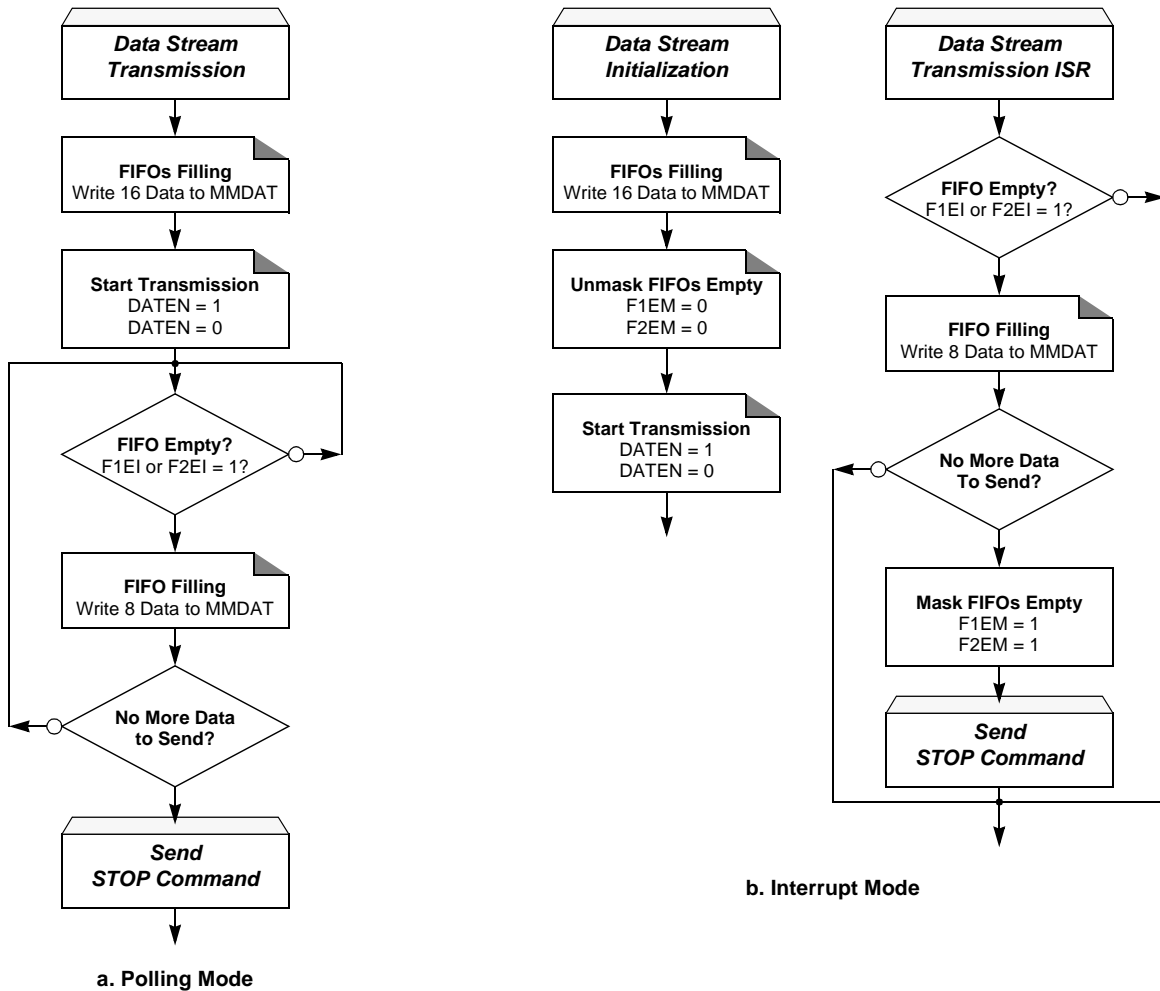
In data stream mode, EOFI flag is set, after reception of the End bit. This assumes user has previously sent the STOP command to the card, which is the only way to stop stream transfer.

In data block mode, EOFI flag is set, after reception of the CRC status token (see Figure 55). Two 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.

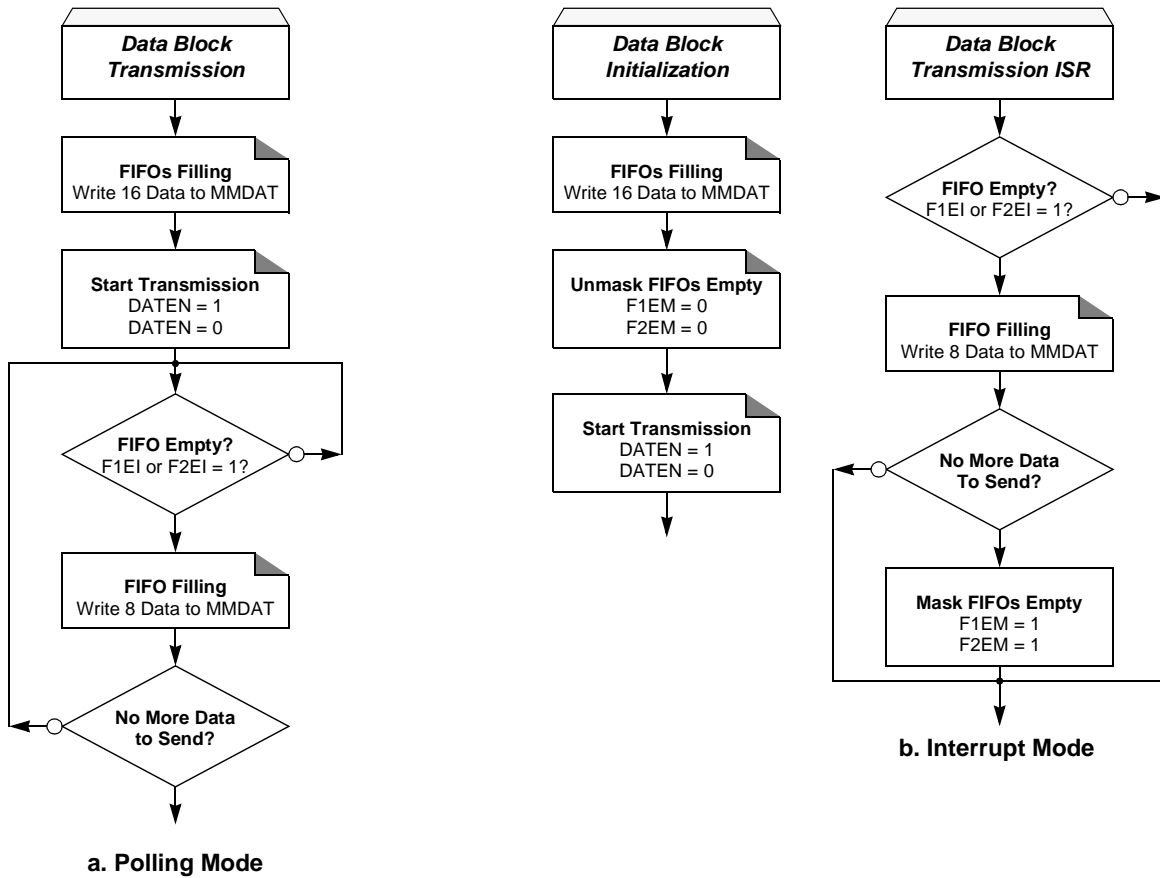
*Busy Status*

As shown in Figure 55 the card uses a busy token during a block write operation. This busy status is reported 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 97.

Figure 67. Data Stream Transmission Flows



**Figure 68. Data Block Transmission Flows**



**Data Receiver**

*Configuration*

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

Figure 69 summarizes the data stream reception flows in both polling and interrupt modes while Figure 70 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 data frame (block or stream) reception is signalled by the EOFI flag in MMINT register. This flag may generate an MMC interrupt request as detailed in Section "Interrupt", page 97. When this flag is set, two 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.

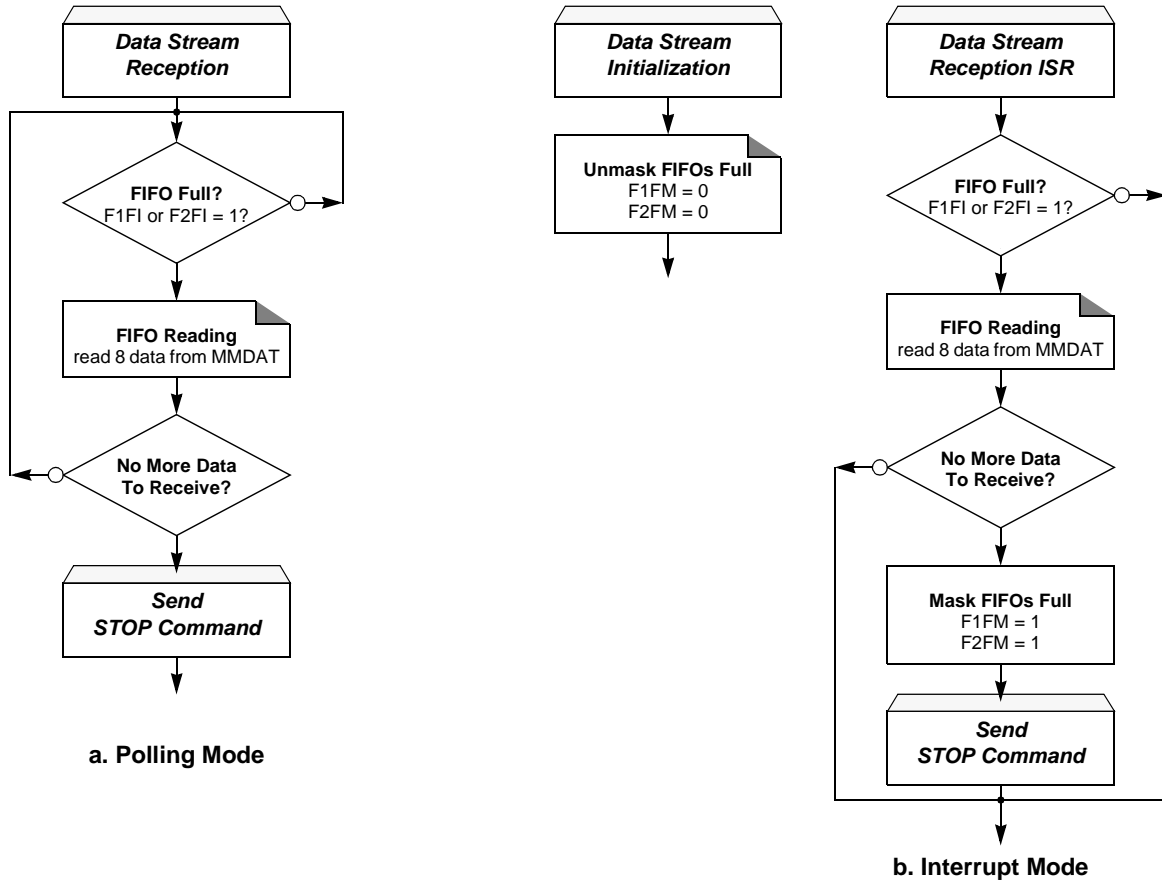
According to the MMC specification data transmission, the card starts after the access time delay (formally  $N_{AC}$  parameter) beginning from the End bit of the read command. To avoid any locking of the MMC controller when card does not send its data (e.g. physically removed from the bus), the user must launch a time-out period to exit from such situation. In case of time-out the user 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).

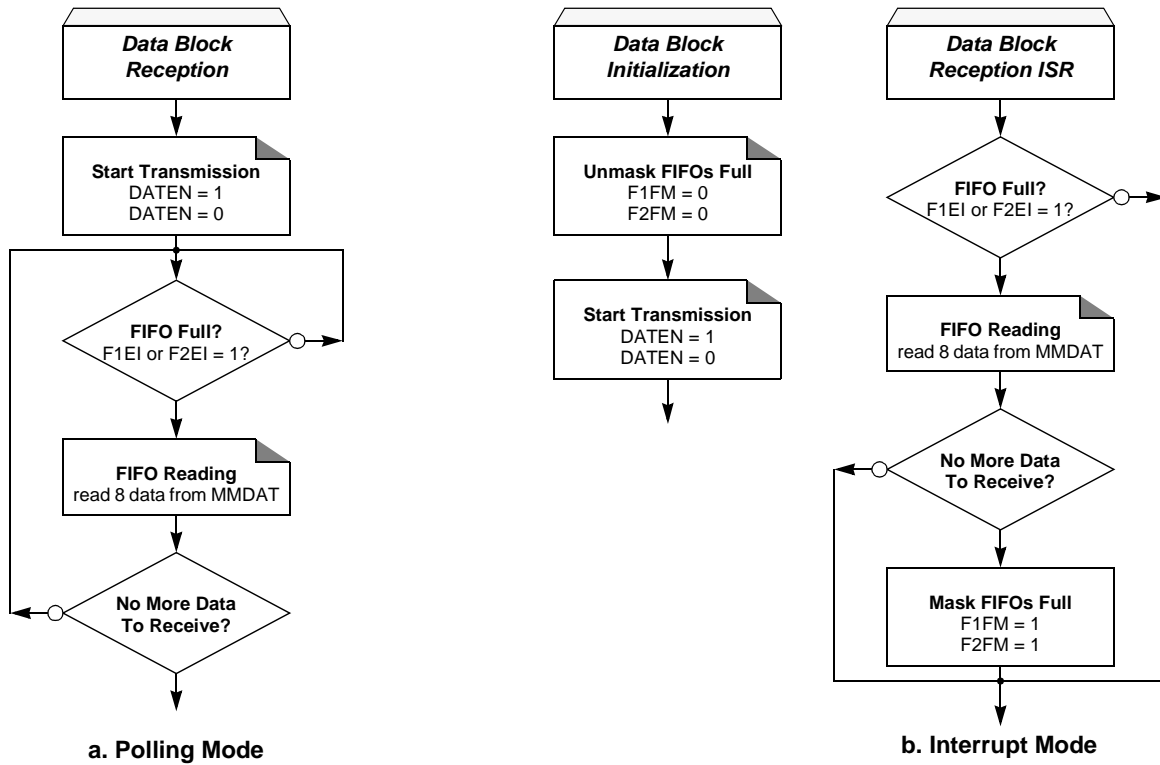
Data Reading

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 69. Data Stream Reception Flows



**Figure 70. 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**

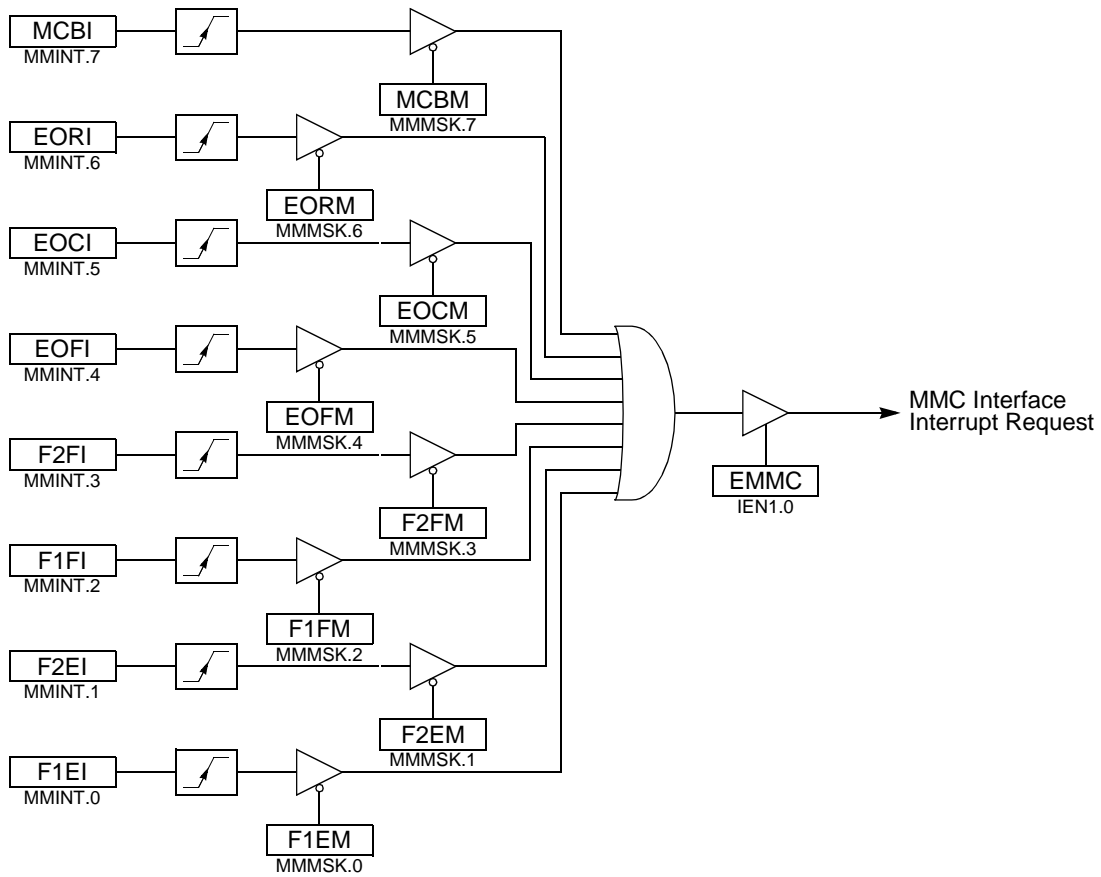
As shown in Figure 71, the MMC controller implements eight interrupt sources reported in MCBI, EORI, EOCl, EOFI, F2FI, F1FI, and F2EI flags in MMCINT register. These flags were detailed in the previous sections.

All of 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 overlook any interrupts.

**Figure 71.** MMC Controller Interrupt System



## Registers

**Table 94.** 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 95.** MMCON1 Register

MMCON1 (S:E5h) – MMC Control Register 1

	7	6	5	4	3	2	1	0
	<b>BLEN3</b>	<b>BLEN2</b>	<b>BLEN1</b>	<b>BLEN0</b>	<b>DATDIR</b>	<b>DATEN</b>	<b>RESPEN</b>	<b>CMDEN</b>

Bit Number	Bit Mnemonic	Description
7 - 4	BLEN3:0	<b>Block Length Bits</b> Refer to Table 93 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 96.** MMCON2 Register

MMCON2 (S:E6h) – MMC Control Register 2

	7	6	5	4	3	2	1	0
	<b>MMCEN</b>	<b>DCR</b>	<b>CCR</b>	-	-	<b>DATD1</b>	<b>DATD0</b>	<b>FLOWC</b>

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 values read from these Bits are 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 2 MMC clock periods (all Bits cleared) to 8 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 97.** 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 values read from these Bits are 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 98.** 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 99.** MMMSK Register

MMMSK (S:DFh) – MMC Interrupt Mask Register

7	6	5	4	3	2	1	0
MCBM	EORM	EOCM	EOFM	F2FM	F1FM	F2EM	F1EM
Bit Number	Bit Mnemonic	Description					
7	MCBM	<b>MMC Card Busy Interrupt Mask Bit</b> Set to prevent MCBI flag from generating an MMC interrupt. Clear to allow MCBI flag to generate an MMC interrupt.					
6	EORM	<b>End Of Response Interrupt Mask Bit</b> Set to prevent EORI flag from generating an MMC interrupt. Clear to allow EORI flag to generate an MMC interrupt.					
5	EOCM	<b>End Of Command Interrupt Mask Bit</b> Set to prevent EOCI flag from generating an MMC interrupt. Clear to allow EOCI flag to generate an MMC interrupt.					
4	EOFM	<b>End Of Frame Interrupt Mask Bit</b> Set to prevent EOFI flag from generating an MMC interrupt. Clear to allow EOFI flag to generate an MMC interrupt.					
3	F2FM	<b>FIFO 2 Full Interrupt Mask Bit</b> Set to prevent F2FI flag from generating an MMC interrupt. Clear to allow F2FI flag to generate an MMC interrupt.					
2	F1FM	<b>FIFO 1 Full Interrupt Mask Bit</b> Set to prevent F1FI flag from generating an MMC interrupt. Clear to allow F1FI flag to generate an MMC interrupt.					
1	F2EM	<b>FIFO 2 Empty Interrupt Mask Bit</b> Set to prevent F2EI flag from generating an MMC interrupt. Clear to allow F2EI flag to generate an MMC interrupt.					
0	F1EM	<b>FIFO 1 Empty Interrupt Mask Bit</b> Set to prevent F1EI flag from generating an MMC interrupt. Clear to allow F1EI flag to generate an MMC interrupt.					

Reset Value = 1111 1111b

**Table 100.** 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 101.** MMDAT Register

MMDAT (S:DCh) – MMC Data Register

7	6	5	4	3	2	1	0
<b>MD7</b>	<b>MD6</b>	<b>MD5</b>	<b>MD4</b>	<b>MD3</b>	<b>MD2</b>	<b>MD1</b>	<b>MD0</b>
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 102.** MMCLK Register

MMCLK (S:EDh) – MMC Clock Divider Register

7	6	5	4	3	2	1	0
<b>MMCD7</b>	<b>MMCD6</b>	<b>MMCD5</b>	<b>MMCD4</b>	<b>MMCD3</b>	<b>MMCD2</b>	<b>MMCD1</b>	<b>MMCD0</b>
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 AT8xC5132 provide 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 AT8xC5132 and the IDE devices.

### Description

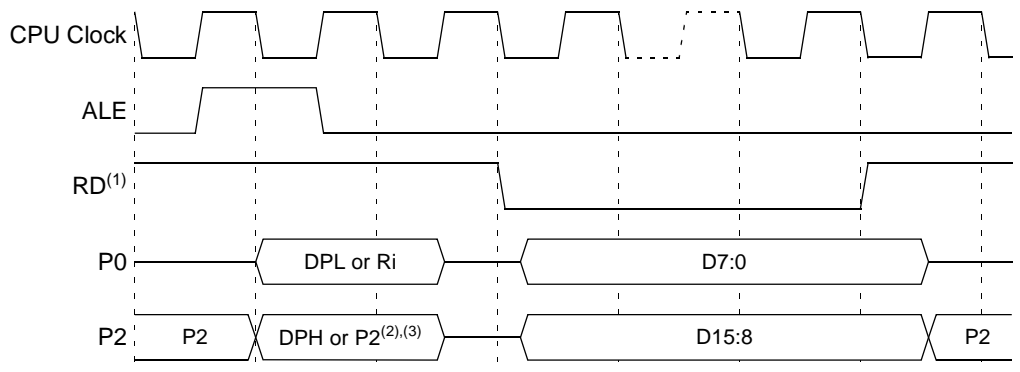
The IDE interface mode is enabled by setting the EXT16 bit in AUXR (see Table 28 on page 30). As soon as this bit is set, all MOVX instructions read or write are done in a 16-bit mode compare to the standard 8-bit mode. P0 carries the low order multiplexed address and data bus (A7:0, D7:0) while P2 carries 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 104) 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 72 shows the IDE read bus cycle while Figure 73 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.

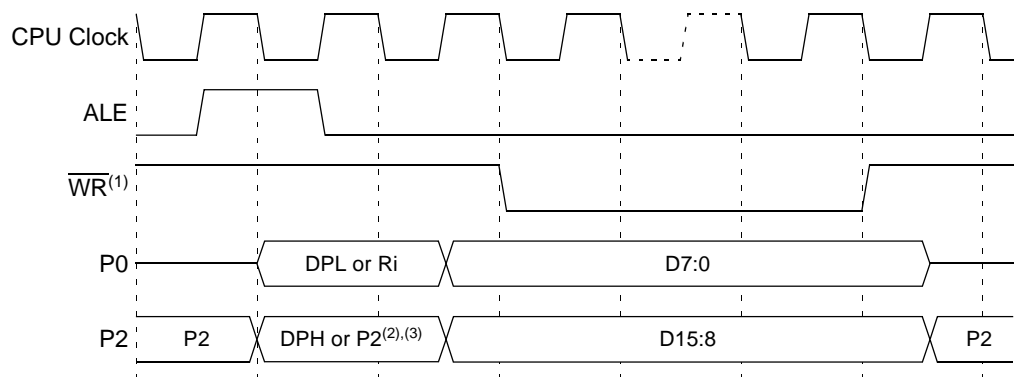
Slow IDE devices can be accessed by stretching the read and write cycles. This is done using the M0 bit in AUXR. Setting this bit changes the width of the RD and WR signals from 3 to 15 CPU clock periods.

**Figure 72. IDE 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 73. IDE Write Waveforms**

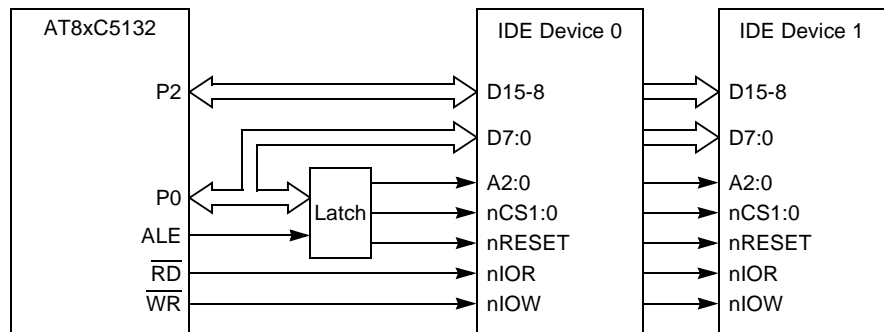


- Notes:
1.  $\overline{WR}$  signal may be stretched using M0 bit in AUXR register.
  2. When executing MOVX @Ri instruction, P2 outputs SFR content.
  3. When executing MOVX @DPTR instruction, if DPHDIS is set (Page Access Mode), P2 outputs SFR content instead of DPH.

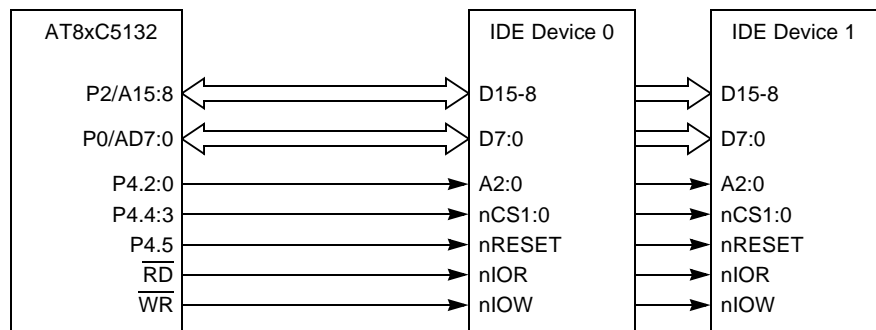
**IDE Device Connection**

Figure 74 and Figure 75 show two examples on how to interface up to two IDE devices to the AT8xC5132. In both examples P0 carries IDE low order data Bits D7:0, P2 carries IDE high order data Bits D15:8, while RD# and WR# signals are respectively connected to the IDE nIOR and nIOW signals. Other IDE control signals are generated by the address latch outputs in the first example – they are generated by port I/Os in the second example.

**Figure 74. IDE Device Connection Example 1**



**Figure 75. IDE Device Connection Example 2**



**Table 103.** 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 104.** 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 = 0000 0000b

## Serial I/O Port

The serial I/O port in the AT8xC5132 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 107) are used to select a mode among the single synchronous and the three asynchronous modes according to Table 105.

**Table 105.** 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.

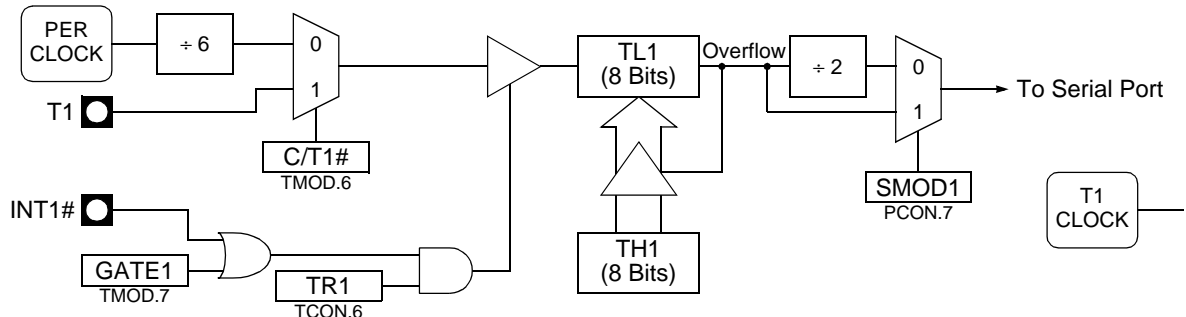
The addition of the Internal Baud Rate Generator allows freeing of the Timer 1 for other purposes in the application. It is highly recommended to use the Internal Baud Rate Generator as it allows higher and more accurate baud rates than Timer 1.

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 76 Timer 1 is used in its 8-bit auto-reload mode (detailed in Section "Mode 2 (8-bit Timer with Auto-Reload)", page 52). SMOD1 bit in PCON register allows doubling of the generated baud rate.

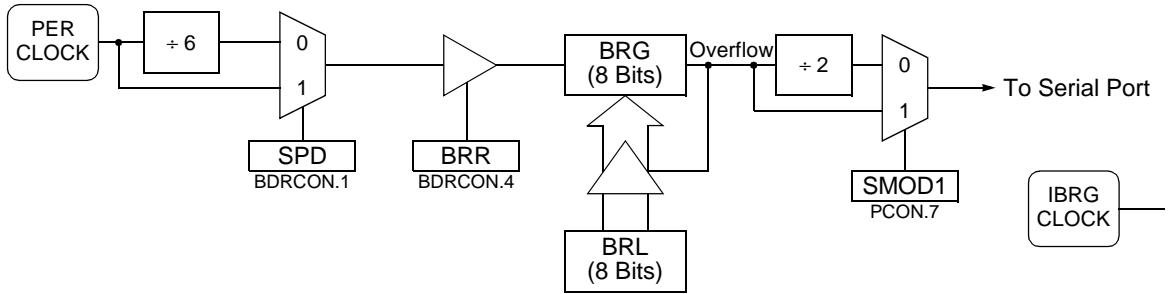
**Figure 76.** Timer 1 Baud Rate Generator Block Diagram



### 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 77, the Internal Baud Rate Generator is an 8-bit auto-reload timer feed by the peripheral clock or by the peripheral clock divided by 6 depending on the SPD bit in BDRCON register (see Table 111). 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.

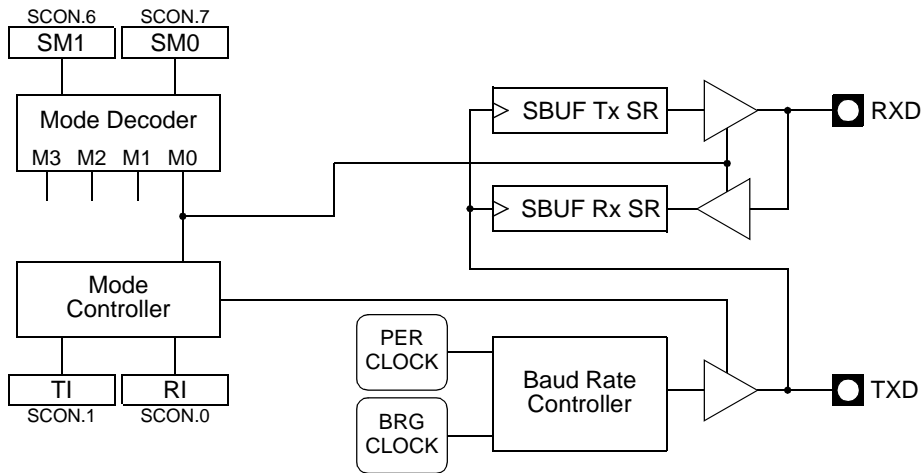
**Figure 77.** Internal Baud Rate Generator Block Diagram



### 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 109). Figure 78 shows the serial port block diagram in Mode 0.

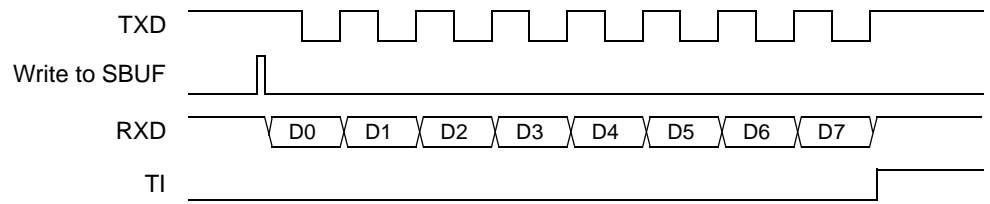
**Figure 78.** Serial I/O Port Block Diagram (Mode 0)



### Transmission (Mode 0)

To start a transmission mode 0, write to SCON register clearing Bits SM0, SM1. As shown in Figure 79, 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 79. Transmission Waveforms (Mode 0)**

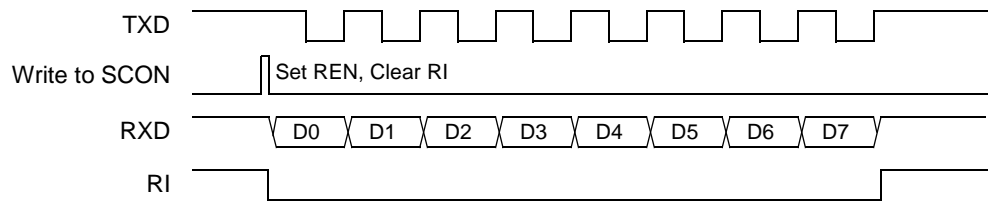


**Reception (Mode 0)**

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 80, 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 sampling, 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.

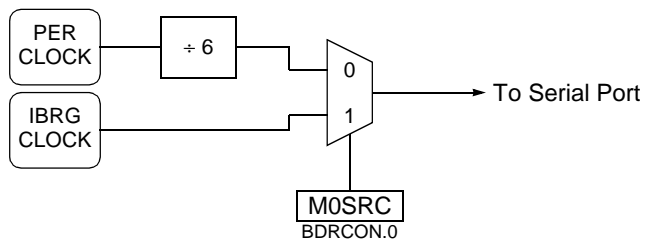
**Figure 80. Reception Waveforms (Mode 0)**



**Baud Rate Selection (Mode 0)**

In mode 0, the baud rate can be either fixed or variable. As shown in Figure 81, the selection is done using MOSRC bit in BDRCON register. Figure 82 gives the baud rate calculation formulas for each baud rate source.

**Figure 81. Baud Rate Source Selection (mode 0)**



**Figure 82. Baud Rate Formulas (Mode 0)**

$$\text{Baud\_Rate} = \frac{F_{\text{PER}}}{6}$$

**a. Fixed Formula**

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

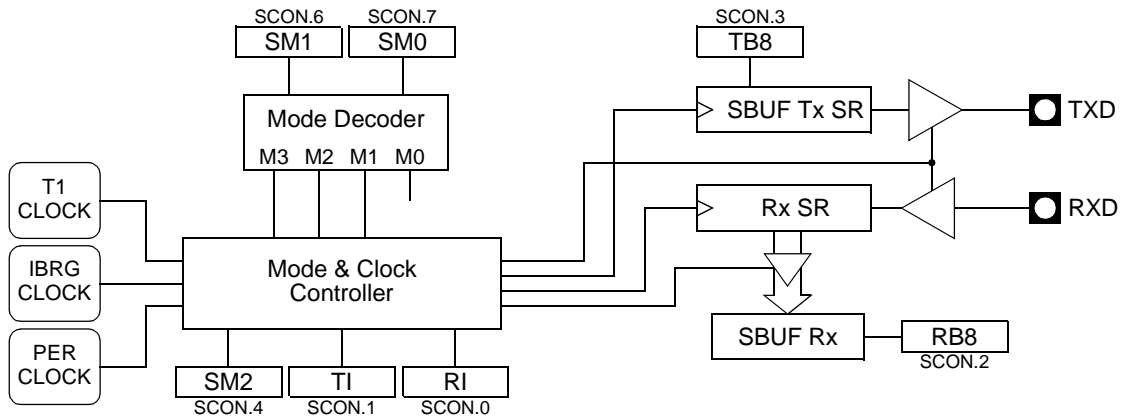
$$\text{BRL} = 256 - \frac{2^{\text{SMOD}1} \cdot F_{\text{PER}}}{6^{(1-\text{SPD})} \cdot 32 \cdot \text{Baud\_Rate}}$$

**b. Variable Formula**

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

The Serial Port has one 8-bit and two 9-bit asynchronous modes of operation. Figure 83 shows the Serial Port block diagram in asynchronous modes.

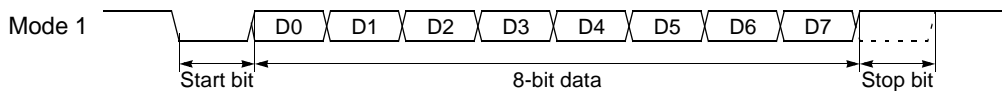
**Figure 83.** Serial I/O Port Block Diagram (Modes 1, 2 and 3)



### Mode 1

Mode 1 is a full-duplex, asynchronous mode. The data frame (see Figure 84) 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 data is received, the stop bit is read in the RB8 bit in SCON register.

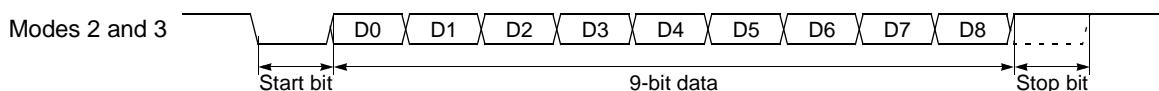
**Figure 84.** Data Frame Format (Mode 1)



### Modes 2 and 3

Modes 2 and 3 are full-duplex, asynchronous modes. The data frame (see Figure 85) 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, the ninth bit can be used as a command/data flag.

**Figure 85.** Data Frame Format (Modes 2 and 3)



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

To initiate a transmission, write to SCON register, setting SM0 and SM1 Bits according to Table 105, and setting 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, setting SM0 and SM1 Bits according to Table 105, and set the REN bit. The actual reception is then initiated by a detected high-to-low transition on the RXD pin.

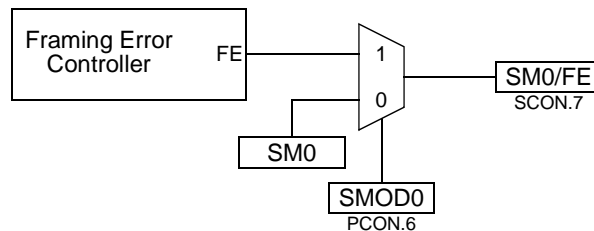
## Framing Error Detection (Modes 1, 2 and 3)

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 86.

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 two 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 clears 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 92.

**Figure 86.** Framing Error Block Diagram



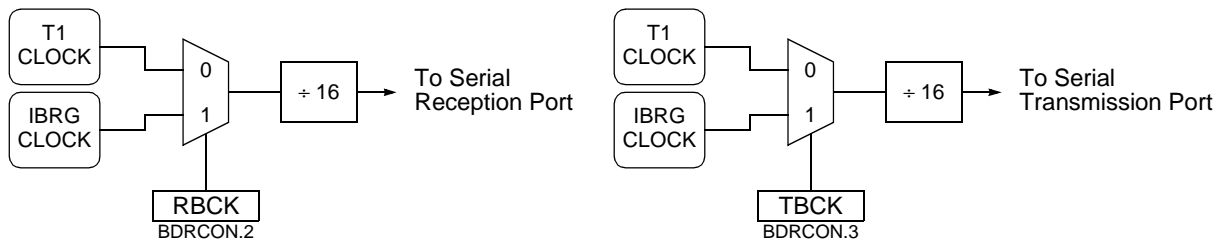
## Baud Rate Selection (Modes 1 and 3)

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 87, the selection is done using RBCK and TBCK Bits in BDRCON register.

Figure 88 gives the baud rate calculation formulas for each baud rate source. Table 106 details Internal Baud Rate Generator configuration for different peripheral clock frequencies and gives baud rates closer to the standard baud rates.

**Figure 87.** Baud Rate Source Selection (Modes 1 and 3)



**Figure 88.** Baud Rate Formulas (Modes 1 and 3)

$$\text{Baud\_Rate} = \frac{2^{\text{SMOD1}} \cdot F_{\text{PER}}}{6^{(1-\text{SPD})} \cdot 32 \cdot (256 - \text{BRL})}$$

$$\text{BRL} = 256 - \frac{2^{\text{SMOD1}} \cdot F_{\text{PER}}}{6^{(1-\text{SPD})} \cdot 32 \cdot \text{Baud\_Rate}}$$

**A. IBRG Formula**

$$\text{Baud\_Rate} = \frac{2^{\text{SMOD1}} \cdot F_{\text{PER}}}{6 \cdot 32 \cdot (256 - \text{TH1})}$$

$$\text{TH1} = 256 - \frac{2^{\text{SMOD1}} \cdot F_{\text{PER}}}{192 \cdot \text{Baud\_Rate}}$$

**B. T1 Formula**

**Table 106.** Baud Rate Generator Configuration

Baud Rate	F <sub>PER</sub> = 6 MHz <sup>(1)</sup>				F <sub>PER</sub> = 8 MHz <sup>(1)</sup>				F <sub>PER</sub> = 10 MHz <sup>(1)</sup>			
	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

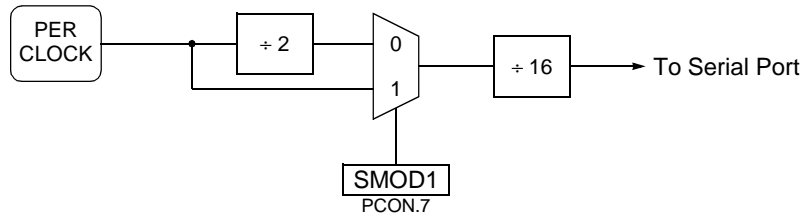
Baud Rate	F <sub>PER</sub> = 12 MHz <sup>(2)</sup>				F <sub>PER</sub> = 16 MHz <sup>(2)</sup>				F <sub>PER</sub> = 20 MHz <sup>(2)</sup>			
	SPD	SMOD1	BRL	Error%	SPD	SMOD1	BRL	Error%	SPD	SMOD1	BRL	Error%
115200	-	-	-	-	1	1	247	3.55	1	1	245	1.36
57600	1	1	243	0.16	1	1	239	2.12	1	1	234	1.36
38400	1	1	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<sub>PER</sub> = F<sub>OSC</sub> ÷ 2.  
 2. These frequencies are achieved in X2 mode, F<sub>PER</sub> = F<sub>OSC</sub>.

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

As shown in Figure 89 the selection is done using SMOD1 bit in PCON register. Figure 90 gives the baud rate calculation formula depending on the selection.

**Figure 89.** Baud Rate Generator Selection (mode 2)



**Figure 90.** Baud Rate Formula (Mode 2)

$$\text{Baud\_Rate} = \frac{2^{\text{SMOD1}} \cdot F_{\text{PER}}}{32}$$

## 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 AT8xC5132 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 user may enable the automatic address recognition feature in mode 1. 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 xxxx<sub>b</sub> (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.

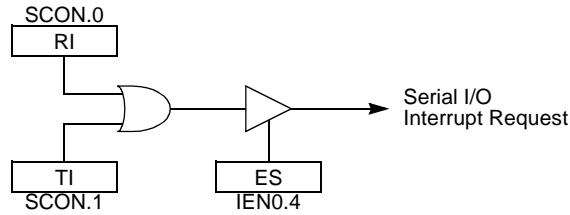
**Interrupt**

The Serial I/O Port handles two interrupt sources that are the “end of reception” (RI in SCON) and “end of transmission” (TI in SCON) flags. As shown in Figure 91 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.

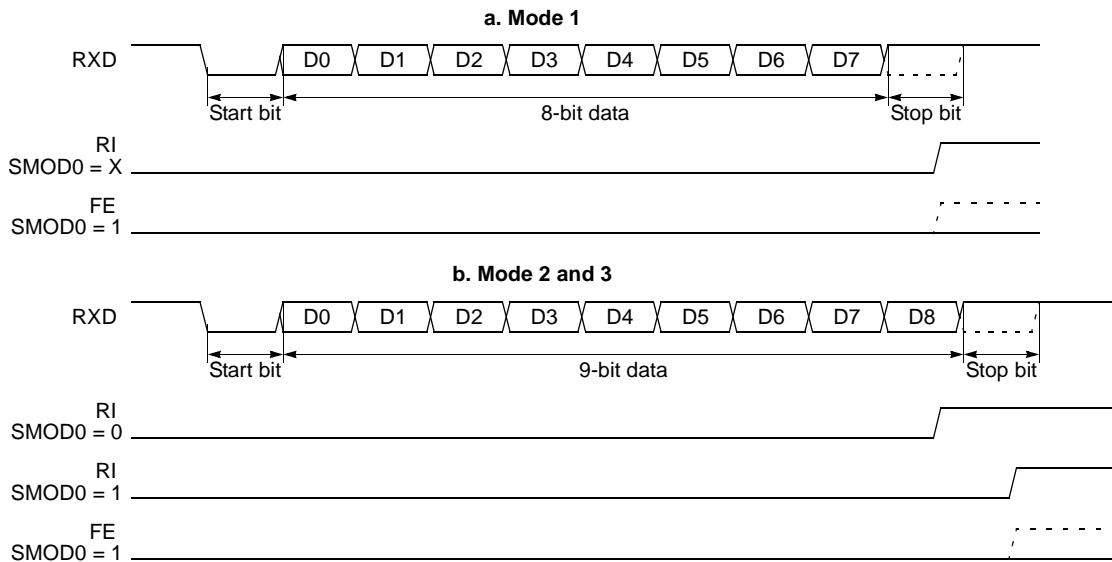
The serial interrupt is enabled by setting ES bit in IEN0 register. This assumes interrupts are globally enabled by setting EA bit in IEN0 register.

Depending on the selected mode and whether the framing error detection is enabled or not, RI flag is set during the stop bit or during the ninth bit as detailed in Figure 92.

**Figure 91.** Serial I/O Interrupt System



**Figure 92.** Interrupt Waveforms



## Registers

**Table 107.** 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 105 for mode selection.						
6	SM1	<b>Serial Port Mode Bit 1</b> Refer to Table 105 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 108.** SBUF Register

SBUF (S:99h) – Serial Buffer Register

7	6	5	4	3	2	1	0
<b>SD7</b>	<b>SD6</b>	<b>SD5</b>	<b>SD4</b>	<b>SD3</b>	<b>SD2</b>	<b>SD1</b>	<b>SD0</b>
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 109.** SADDR Register

SADDR (S:A9h) – Slave Individual Address Register

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

Reset Value = 0000 0000b

**Table 110.** SADEN Register

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

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

Reset Value = 0000 0000b

**Table 111.** 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 values 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 112.** 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 AT8xC5132 implement a Synchronous Peripheral Interface with master and slave modes capability.

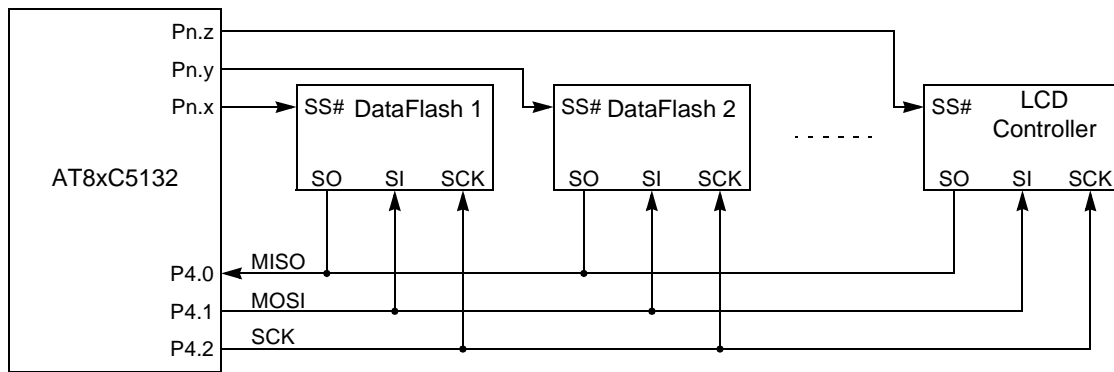
Figure 93 shows an SPI bus configuration using the AT8xC5132 as master connected to slave peripherals. Figure 94 shows an SPI bus configuration using the AT8xC5132 as slave of an other master.

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

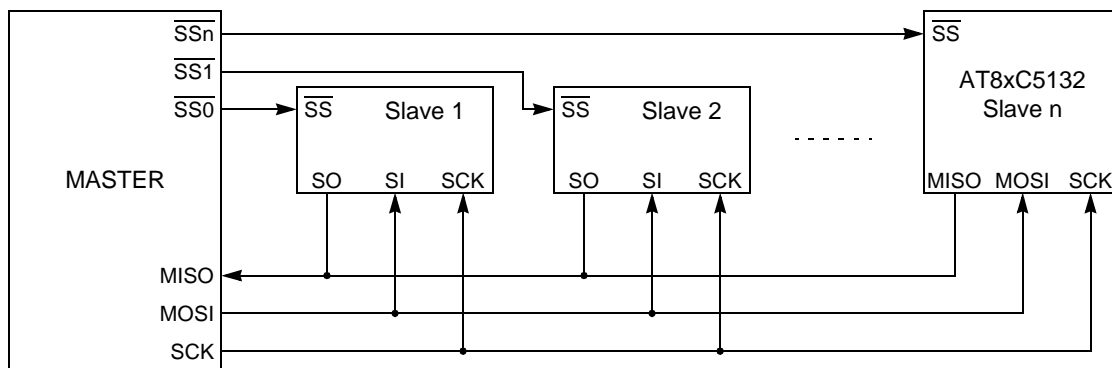
- Master Output Slave Input (MOSI): it is used to transfer data in series from the master to a slave. It is driven by the master.
- Master Input Slave Output (MISO): it is used to transfer data in series from a slave to the master. It is driven by the selected slave.
- Serial Clock (SCK): it is used to synchronize the data transmission both in and out of 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 AT8xC5132 may select each device by software through port pins (Pn.x). Special care should be taken not to select two slaves at the same time to avoid bus conflicts.

**Figure 93.** Typical Master SPI Bus Configuration



**Figure 94.** 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 114); SPSTA, the SPI status register (see Table 115); and SPDAT, the SPI data register (see Table 116).

## Master Mode

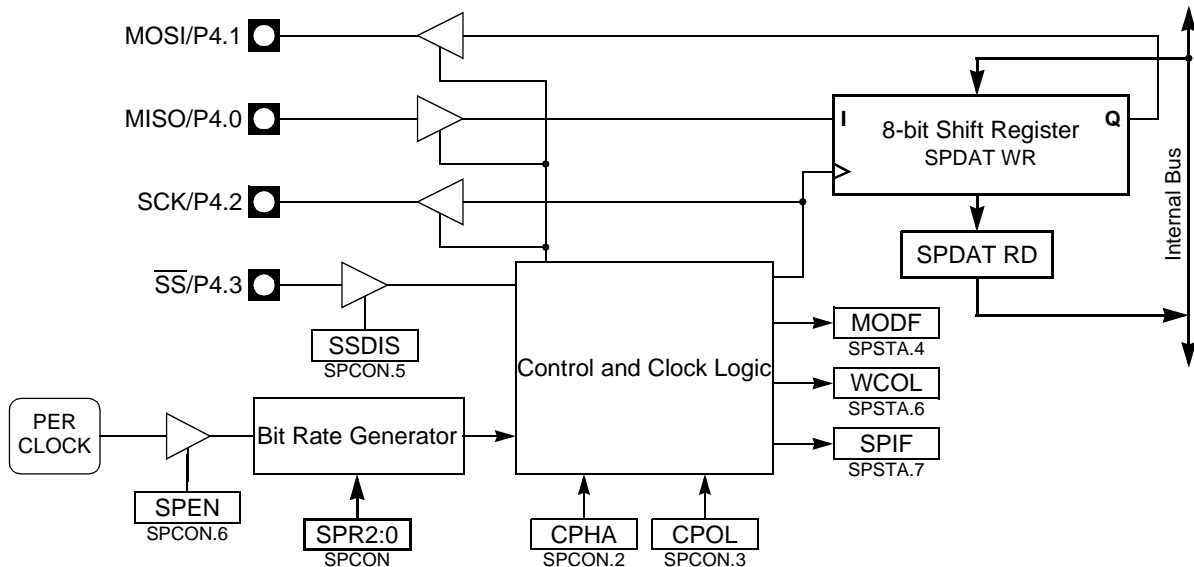
The SPI operates in master mode when the MSTR bit in SPCON is set.

Figure 95 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 signalled by SPIF being set.

In case of the AT8xC5132 is the only master on the bus, it can be useful not to use  $\overline{SS}$  pin and get it back to I/O functionality. This is achieved by setting SSDIS bit in SPCON.

**Figure 95.** SPI Master Mode Block Diagram



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

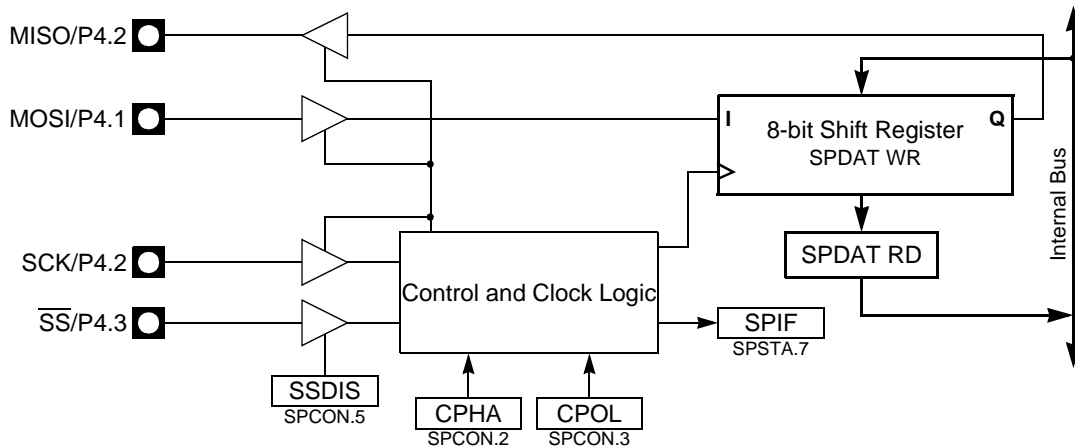
## Slave Mode

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

Figure 96 shows the SPI block diagram in slave mode. In slave mode, before 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.

In case of the AT8xC5132 is the only slave on the bus, it can be useful not to use  $\overline{SS}$  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 122).

**Figure 96. SPI Slave Mode Block Diagram**



Note: 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 113. These bit rates are derived from the peripheral clock ( $F_{PER}$ ) issued from the Clock Controller block as detailed in Section “Clock Controller”, page 12.

**Table 113. Serial Bit Rates**

SPR2	SPR1	SPR0	Bit Rate (kHz) Vs $F_{PER}$						$F_{PER}$ 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,  $F_{PER} = F_{OSC}$ .

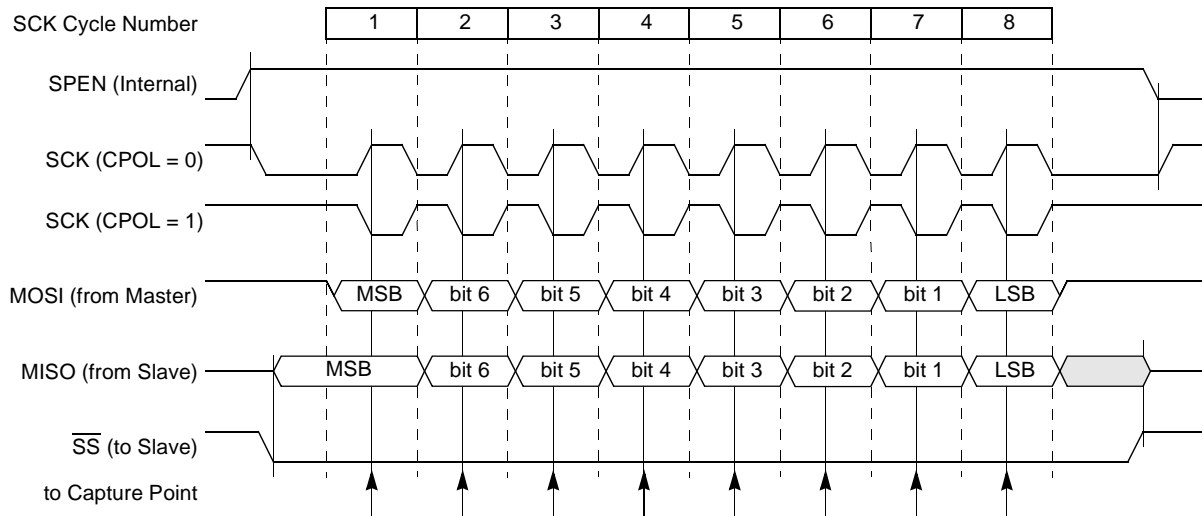
### Data Transfer

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 97 and Figure 98). The SI signal is output from the selected slave and the SO signal is the output from the master. The AT8xC5132 captures data from the SI line while the selected slave captures data from the SO line.

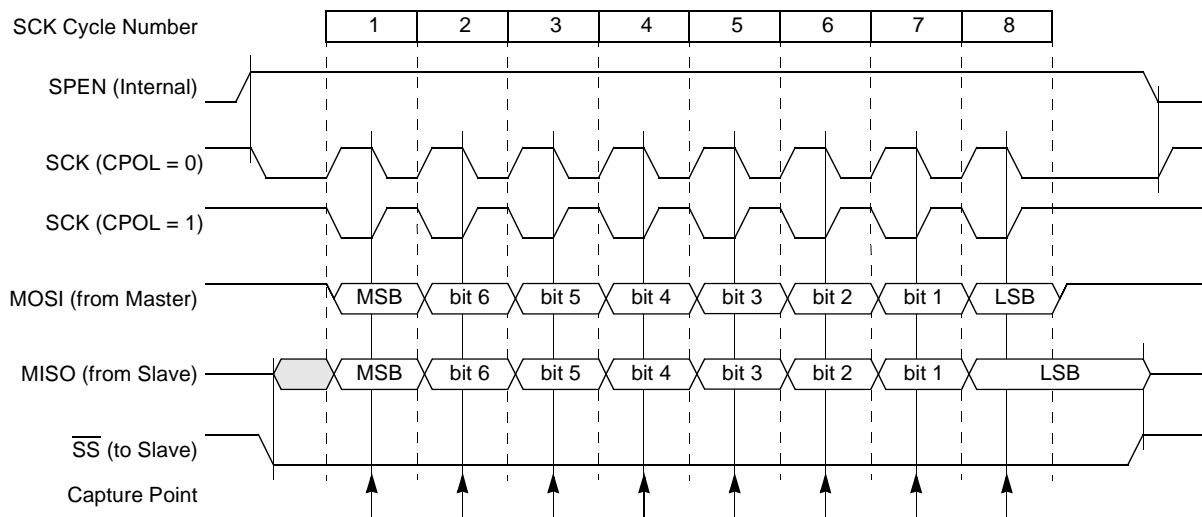
For simplicity, the following figures depict the SPI waveforms in idealized form and do not provide precise timing information. For timing parameters refer to the Section “AC Characteristics”.

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

**Figure 97. Data Transmission Format (CPHA = 0)**



**Figure 98. Data Transmission Format (CPHA = 1)**

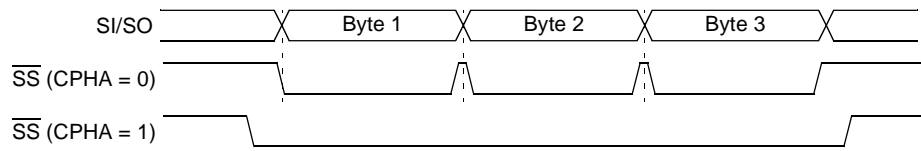


### $\overline{SS}$ Management

Figure 97 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 99). SPDAT must be loaded with data before  $\overline{SS}$  is asserted again.

Figure 98 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 99).

**Figure 99. 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 another master on the bus has asserted  $\overline{SS}$  pin and so, may create a conflict on the bus with two masters 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 123.  
 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 continues 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 ongoing.

**Interrupt**

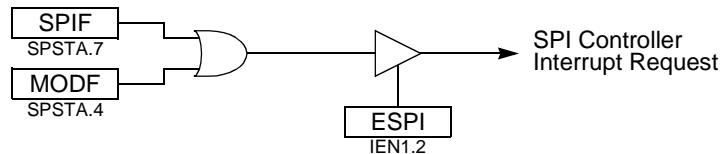
The SPI handles two interrupt sources; the “end of transfer” and the “mode fault” flags.

As shown in Figure 100 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 100. 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 two possible Policies to exchange data in master and slave modes:

- polling
- interrupts

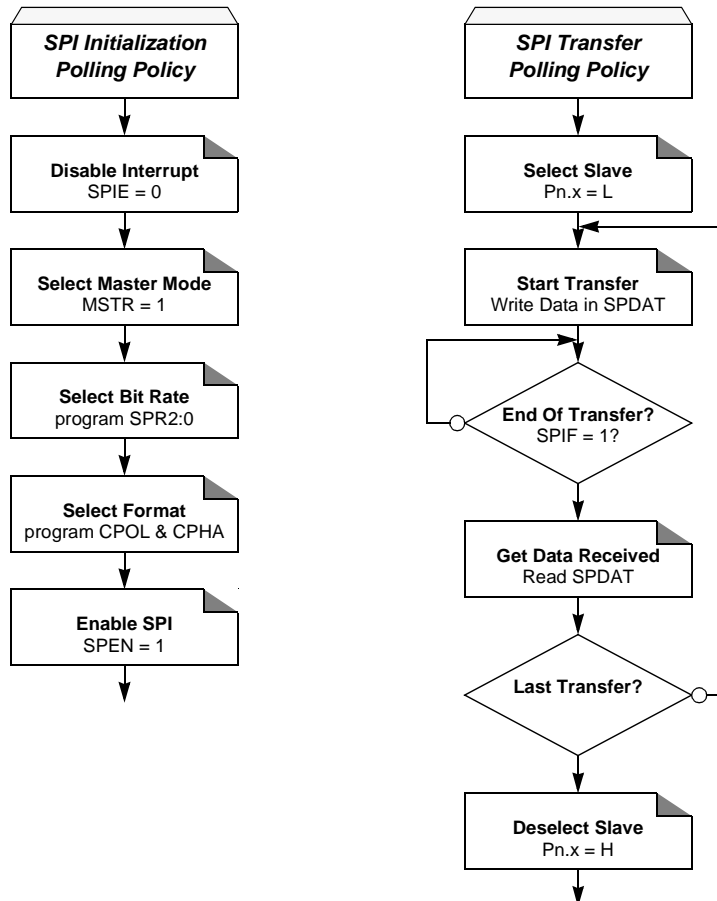
## Master Mode with Polling Policy

Figure 101 shows the initialization phase and the transfer phase flows using the polling policy. Using this flow prevents any overrun error occurrence.

- The bit rate is selected according to Table 113.
- 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 policy 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 101.** Master SPI Polling Policy Flows



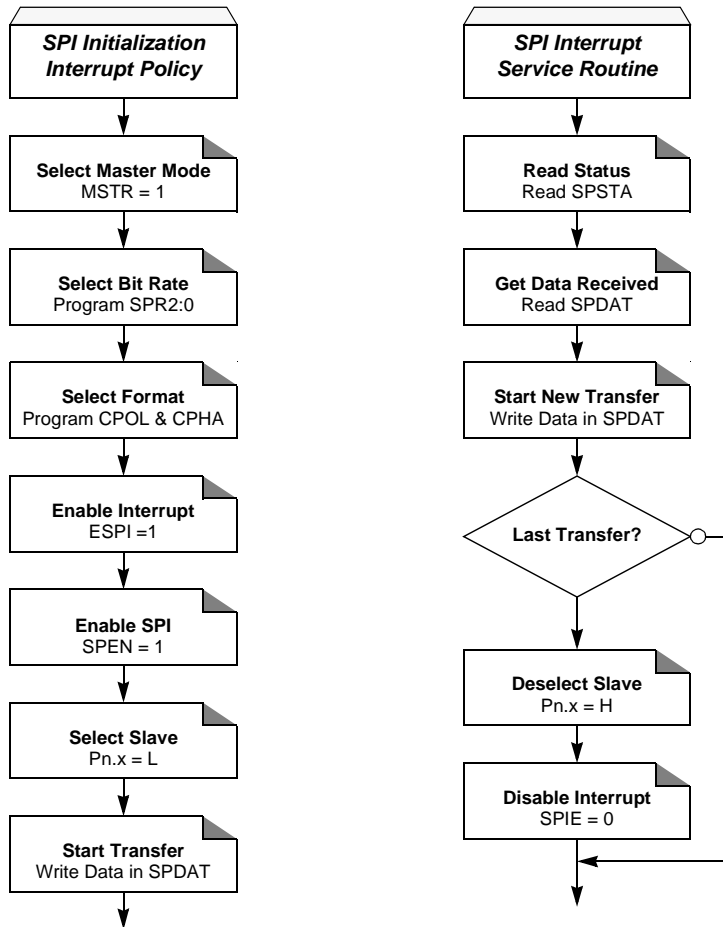
**Master Mode with Interrupt Policy**

Figure 102 shows the initialization phase and the transfer phase flows using the interrupt policy. Using this flow prevents any overrun error occurrence.

- The bit rate is selected according to Table 113.
- 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 102.** Master SPI Interrupt Policy Flows



## Slave Mode with Polling Policy

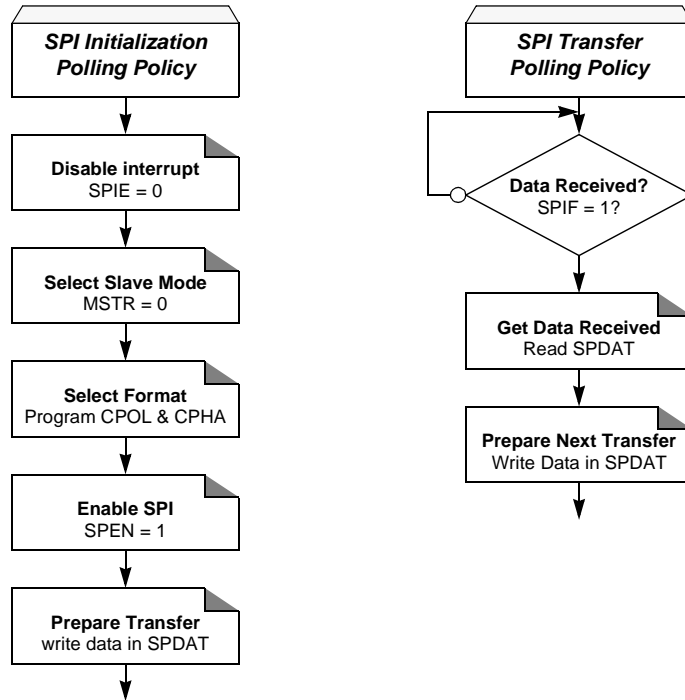
Figure 103 shows the initialization phase and the transfer phase flows using the polling policy.

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

**Figure 103.** Slave SPI Polling Policy Flows



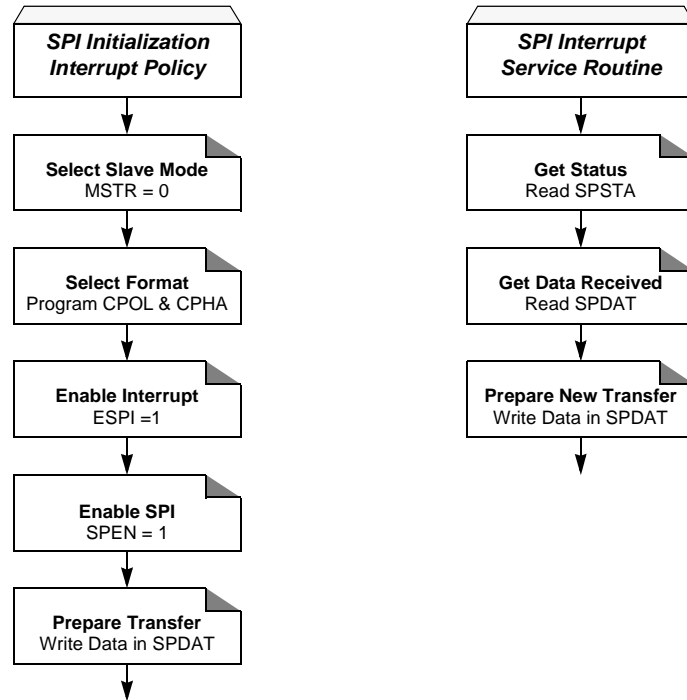
**Slave Mode with Interrupt Policy**

Figure 102 shows the initialization phase and the transfer phase flows using the interrupt policy.

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 104.** Slave SPI Interrupt Policy Flows



**Registers**

**Table 114.** 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 113 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.						

Bit Number	Bit Mnemonic	Description
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 113 for bit rate description.

Reset Value = 0001 0100b

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

**Table 115.** 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 values 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 values read from these Bits are indeterminate. Do not set these Bits.

Reset Value = 00000 0000b

**Table 116.** 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

## Analog-to-Digital Converter

The AT8xC5132 implement a 2-channel 10-bit (8 true Bits) analog-to-digital converter (ADC). The first channel of this ADC can be used for battery monitoring while the second channel can be used for voice sampling at 8 kHz.

### Description

The A/D converter interfaces with the C51 core through four special function registers: ADCON, the ADC control register (see Table 118); ADDH and ADDL, the ADC data registers (see Table 120 and Table 121); and ADCLK, the ADC clock register (see Table 119).

As shown in Figure 105, 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 117).

Figure 105. ADC Structure

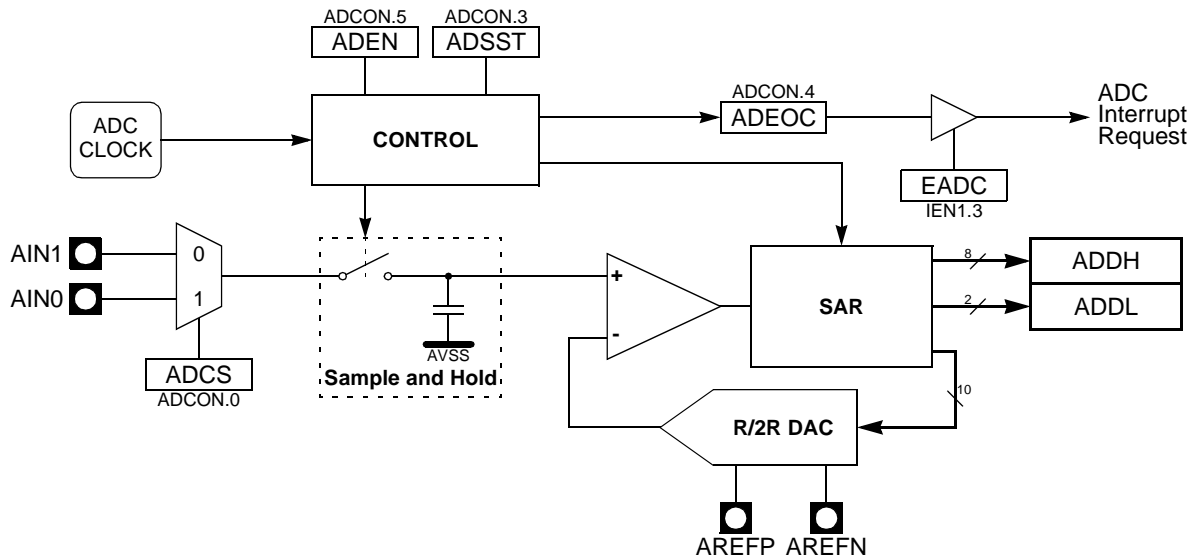
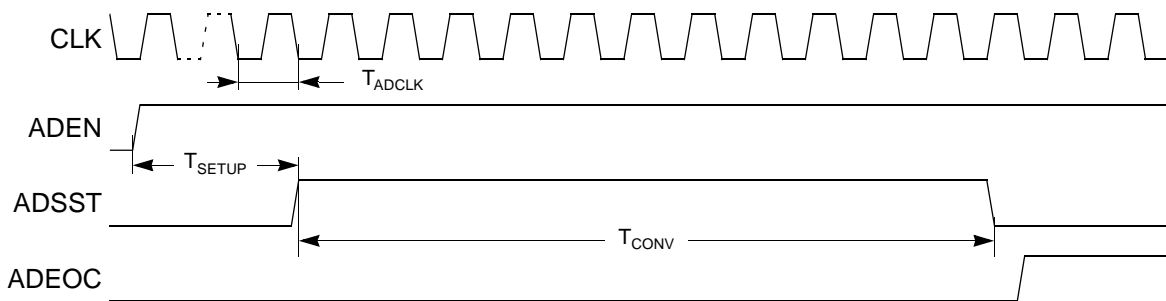


Figure 106 shows the timing diagram of a complete conversion. For simplicity, the figure depicts the waveforms in idealized form and does not provide precise timing information. For ADC characteristics and timing parameters refer to the Section “AC Characteristics”.

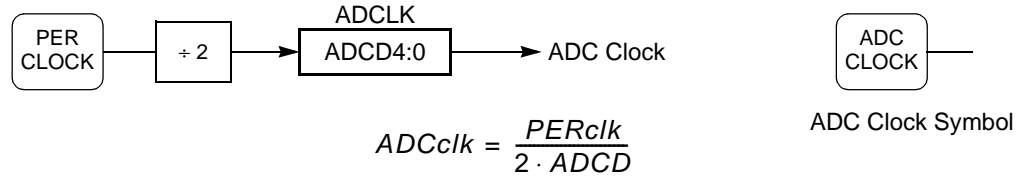
Figure 106. Timing Diagram



## Clock Generation

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 AD<sub>CP</sub>4:0 Bits in ADCLK register. Figure 107 shows the ADC clock generator and its calculation formula<sup>(1)</sup>.

**Figure 107.** ADC Clock Generator and Symbol Caution:



Note: In all cases, the ADC clock frequency may be higher than the maximum  $F_{ADCLK}$  parameter reported in the Section “AC Characteristics”.

## Channel Selection

The channel on which conversion is performed is selected by the AD<sub>CS</sub> bit in AD<sub>CON</sub> register according to Table 117.

**Table 117.** ADC Channel Selection

AD <sub>CS</sub>	Channel
0	AIN1
1	AIN0

## Conversion Precision

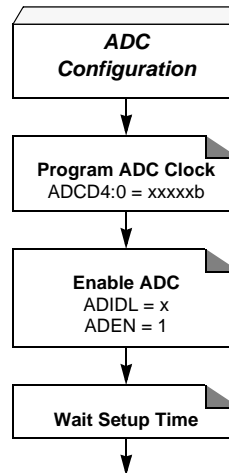
The 10-bit precision conversion is achieved by stopping the CPU core activity during conversion for limiting the digital noise induced by the core. This mode called the Pseudo-Idle mode<sup>(1), (2)</sup> is enabled by setting the AD<sub>IDL</sub> bit in AD<sub>CON</sub> register. Thus, when conversion is launched (see Section “Conversion Launching”, page 131), the CPU core is stopped until the end of the conversion (see Section “End of Conversion”, page 131). 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.

## Configuration

The ADC configuration consists in programming the ADC clock as detailed in the Section “Clock Generation”, page 130. The ADC is enabled using the AD<sub>EN</sub> bit in AD<sub>CON</sub> register. As shown in Figure 93, user must wait for the setup time ( $T_{SETUP}$ ) before launching any conversion.

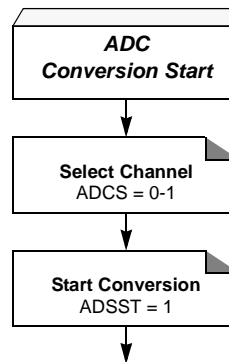
**Figure 108.** ADC Configuration Flow



**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 109.** 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.

The 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 118.** 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 values 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 standby 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 values read from these Bits are 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 119.** 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 values read from these Bits are always 0. Do not set these Bits.					
4 - 0	ADCD4:0	<b>ADC Clock Divider</b> 5-bit divider for ADC clock generation.					

**Table 120.** ADDH Register

ADDH (S:F5h Read Only) – ADC Data High Byte Register

7	6	5	4	3	2	1	0
ADAT9	ADAT8	ADAT7	ADAT6	ADAT5	ADAT4	ADAT3	ADAT2
Bit Number	Bit Mnemonic	Description					
7 - 0	ADAT9:2	<b>ADC Data</b> Eight Most Significant Bits of the 10-bit ADC data.					

Reset Value = 0000 0000b

**Table 121.** ADDL Register

ADDL (S:F4h Read Only) – ADC Data Low Byte Register

7	6	5	4	3	2	1	0
-	-	-	-	-	-	ADAT1	ADAT0
Bit Number	Bit Mnemonic	Description					
7 - 2	-	<b>Reserved</b> The values read from these Bits are always 0. Do not set these Bits.					
1 - 0	ADAT1:0	<b>ADC Data</b> Two Least Significant Bits of the 10-bit ADC data.					

Reset Value = 0000 0000b

## Keyboard Interface

The AT8xC5132 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.

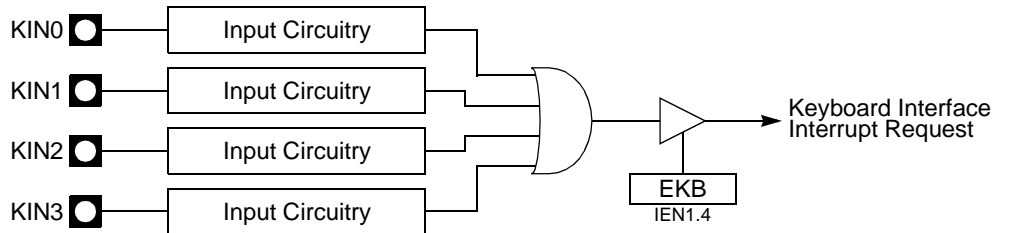
### Description

The keyboard interfaces with the C51 core through two special function registers: KBCON, the keyboard control register (see Table 122); and KBSTA, the keyboard control and status register (see Table 123).

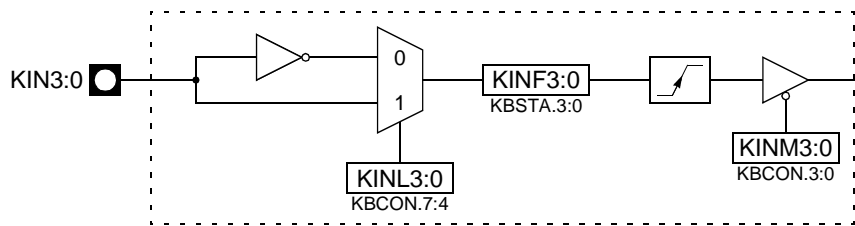
The keyboard inputs are considered as 4 independent interrupt sources sharing the same interrupt vector. An interrupt enable bit (EKB in IEN1 register) allows global enable or disable of the keyboard interrupt (see Figure 110). As detailed in Figure 111 each keyboard input has the capability to detect 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 110.** Keyboard Interface Block Diagram



**Figure 111.** 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 122.** 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 123.** 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 values read from these Bits are 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 Ratings

Storage Temperature .....	-65°C to +150°C
Voltage on any other Pin to $V_{SS}$ .....	-0.3 to +4.0V
$I_{OL}$ per I/O Pin .....	5 mA
Power Dissipation .....	1 W
Ambient Temperature Under Bias.....	-40°C to +85°C
$V_{DD}$ .....	2.7V to 3.3V

NOTE: 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.

### DC Characteristics

#### Digital Logic

**Table 124.** Digital DC Characteristics  $V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = -40^\circ$  to  $+85^\circ C$

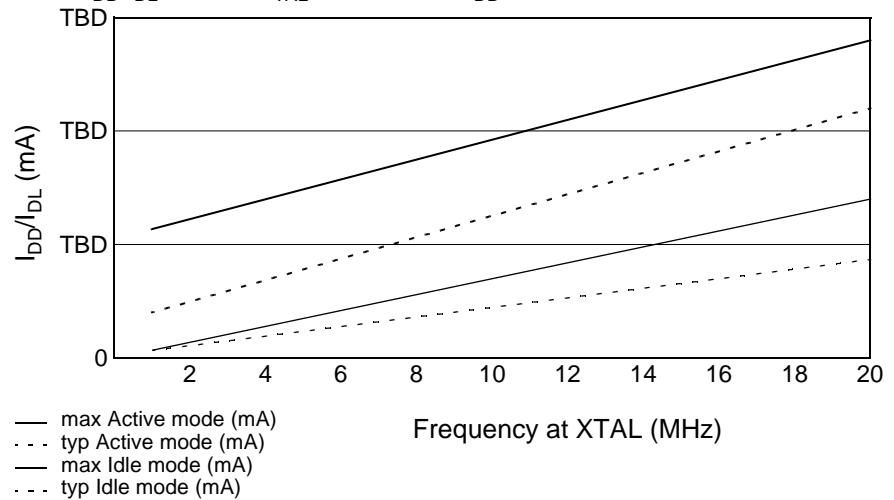
Symbol	Parameter	Min	Typ <sup>(1)</sup>	Max	Units	Test Conditions
$V_{IL}$	Input Low Voltage	-0.5		$0.2 \cdot V_{DD} - 0.1$	V	
$V_{IH1}$	Input High Voltage (except RST)	$0.2 \cdot V_{DD} + 0.9$		$V_{DD}$	V	
$V_{IH2}$	Input High Voltage (RST)	$0.7 \cdot V_{DD}$		$V_{DD} + 0.5$	V	
$V_{OL1}$	Output Low 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	$I_{OH} = -30$ $\mu$ 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	$\mu$ A	$V_{in} = 0.45V$

**Table 124.** Digital DC Characteristics  $V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = -40^\circ$  to  $+85^\circ C$

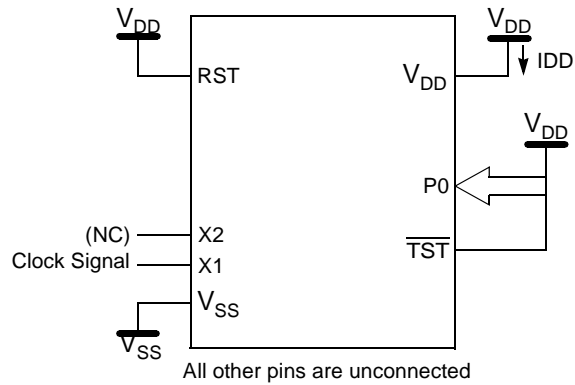
Symbol	Parameter	Min	Typ <sup>(1)</sup>	Max	Units	Test Conditions
$I_{LI}$	Input Leakage Current (P0, ALE, MCMD, MDAT, MCLK, SCLK, DCLK, DSEL, DOUT)			10	$\mu A$	$0.45 < V_{IN} < V_{DD}$
$I_{TL}$	Logical 1 to 0 Transition Current (P1, P2, P3, P4 and P5)			-650	$\mu A$	$V_{in} = 2.0V$
$R_{RST}$	Pull-down Resistor	50	90	200	$k\Omega$	
$C_{IO}$	Pin Capacitance		10		$pF$	$T_A = 25^\circ C$
$V_{RET}$	$V_{DD}$ Data Retention Limit			1.8	V	
$I_{DD}$	Operating Current		TBD	TBD	mA	12 MHz, $V_{DD} < 3.3V$ 16 MHz, $V_{DD} < 3.3V$ 20 MHz, $V_{DD} < 3.3V$
$I_{DL}$	Idle Mode Current		TBD	TBD	mA	12 MHz, $V_{DD} < 3.3V$ 16 MHz, $V_{DD} < 3.3V$ 20 MHz, $V_{DD} < 3.3V$
$I_{PD}$	Power-down Current		TBD	TBD	$\mu A$	$V_{RET} < V_{DD} < 3.3V$

Note: 1. Typical values are obtained using  $V_{DD} = 3V$  and  $T_A = 25^\circ C$ . They are not tested and there is no guarantee on these values.

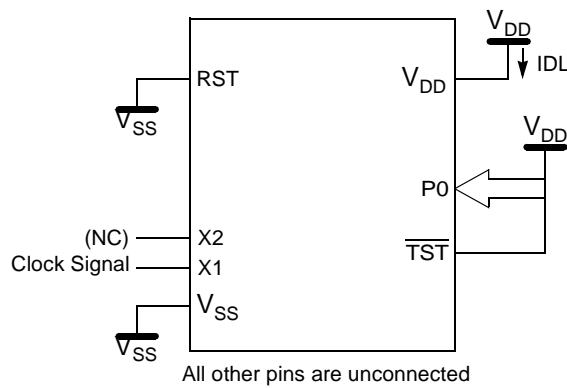
**Figure 112.**  $I_{DD}/I_{DL}$  Versus  $X_{TAL}$  Frequency;  $V_{DD} = 2.7$  to  $3.3V$



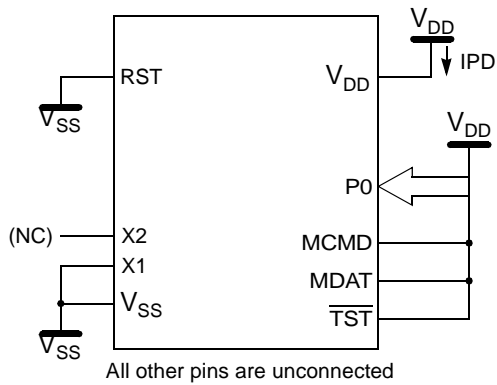
**$I_{DD}$ ,  $I_{DL}$  and  $I_{PD}$  Test Conditions** **Figure 113.**  $I_{DD}$  Test Condition, Active Mode



**Figure 114.**  $I_{DL}$  Test Condition, Idle Mode



**Figure 115.**  $I_{PD}$  Test Condition, Power-Down Mode



## A-to-D Converter

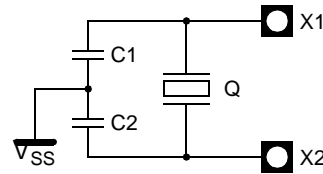
**Table 125.** A-to-D Converter DC Characteristics  $V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = -40^{\circ}C$  to  $+85^{\circ}C$

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	$\mu A$	$AV_{DD} = 3.3V$ $AIN1:0 = 0$ to $AV_{DD}$
$AI_{PD}$	Analog Standby Current			2	$\mu A$	$AV_{DD} = 3.3V$ $A_{DEN} = 0$ 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 V	
$R_{REF}$	AREF Input Resistance	10		30	k $\Omega$	$T_A = 25^{\circ}C$
$C_{IA}$	Analog Input capacitance			10	pF	$T_A = 25^{\circ}C$

## Oscillator and Crystal

### Schematic

**Figure 116.** 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

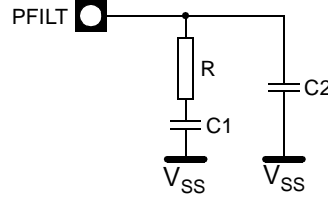
**Table 126.** Oscillator and Crystal Characteristics  $V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = -40^{\circ}$  to  $+85^{\circ}C$

Symbol	Parameter	Min	Typ	Max	Unit
$C_{X1}$	Internal Capacitance (X1 - $V_{SS}$ )		10		pF
$C_{X2}$	Internal Capacitance (X2 - $V_{SS}$ )		10		pF
$C_L$	Equivalent Load Capacitance (X1 - X2)		5		pF
DL	Drive Level			50	$\mu W$
F	Crystal Frequency			20	MHz
RS	Crystal Series Resistance			40	$\Omega$
CS	Crystal Shunt Capacitance			6	pF

## Phase Lock Loop

### Schematic

**Figure 117.** PLL Filter Connection



### Parameters

**Table 127.** PLL Filter Characteristics

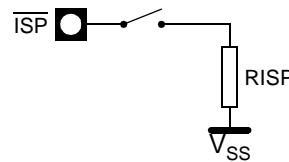
$V_{DD} = 2.7$  to  $3.3V$  ,  $T_A = -40^\circ$  to  $+85^\circ C$

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 118.** ISP Pull-down Connection



### Parameters

**Table 128.** ISP Pull-Down Characteristics  $V_{DD} = 2.7$  to  $3.3V$  ,  $T_A = -40^\circ$  to  $+85^\circ C$

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 129.** External 8-bit Bus Cycles Timing Symbol Definitions

Signals		Conditions	
A	Address	H	High
D	Data In	L	Low
L	ALE	V	Valid
Q	Data Out	X	No Longer Valid
R	$\overline{RD}$	Z	Floating
W	$\overline{WR}$		

#### Timings

Test conditions: capacitive load on all pins = 50 pF.

**Table 130.** External 8-bit Bus Cycle – Data Read AC Timings

$V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = -40^\circ$  to  $+85^\circ C$

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 131.** External 8-bit Bus Cycle – Data Write AC Timings

$V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = -40^\circ$  to  $+85^\circ C$

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 119.** External 8-bit Bus Cycle – Data Read Waveforms

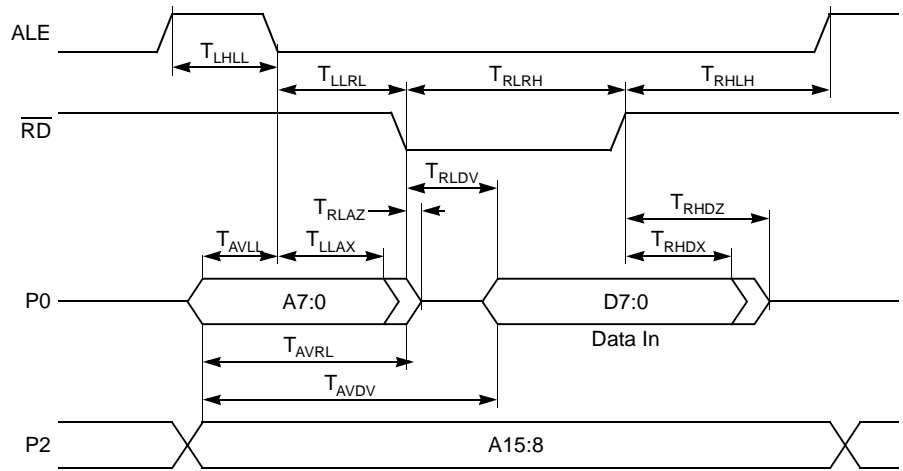
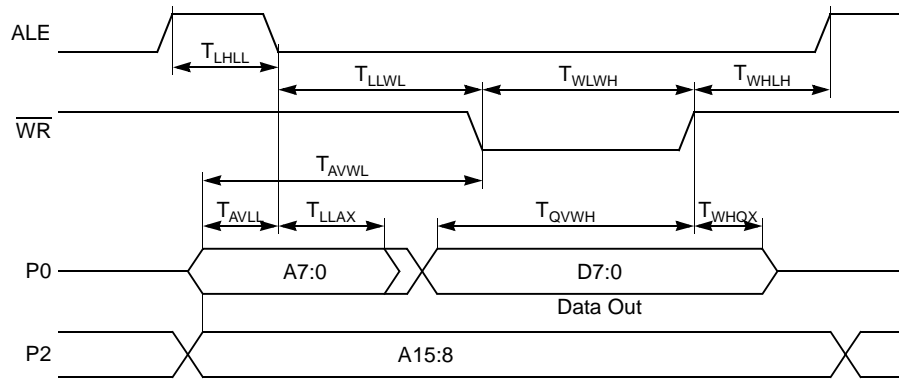


Figure 120. External 8-bit Bus Cycle – Data Write Waveforms



External IDE 16-bit Bus Cycles

Definition of Symbols

Table 132. 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 133.** External IDE 16-bit Bus Cycle – Data Read AC Timings

$V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = -40^\circ$  to  $+85^\circ C$

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

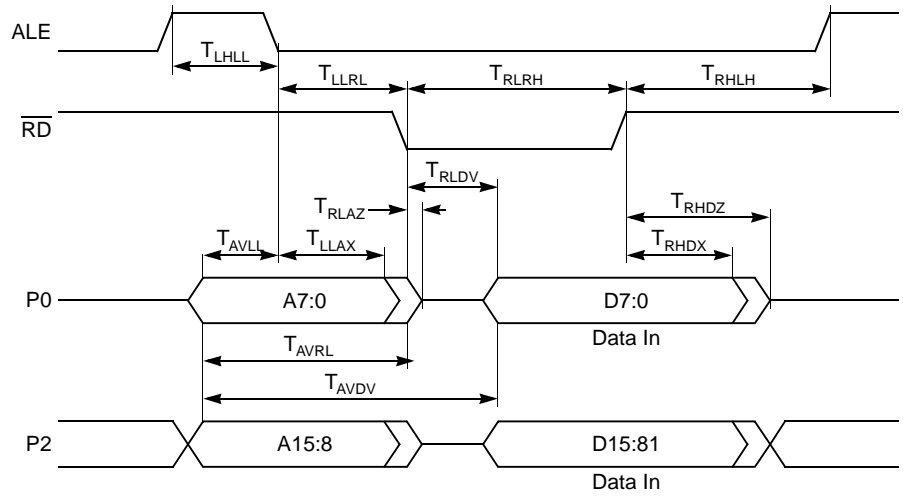
**Table 134.** External IDE 16-bit Bus Cycle – Data Write AC Timings

$V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = -40^\circ$  to  $+85^\circ C$

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

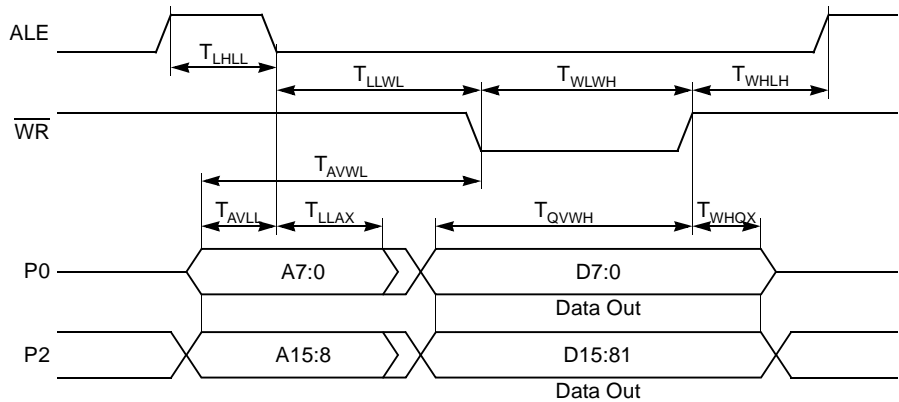
Waveforms

**Figure 121.** External IDE 16-bit Bus Cycle – Data Read Waveforms



Note: D15:8 is written in DAT16H SFR.

**Figure 122.** External IDE 16-bit Bus Cycle – Data Write Waveforms



Note: D15:8 is the content of DAT16H SFR.

SPI Interface

Definition of Symbols

**Table 135.** 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

**Table 136.** SPI Interface Master AC Timing

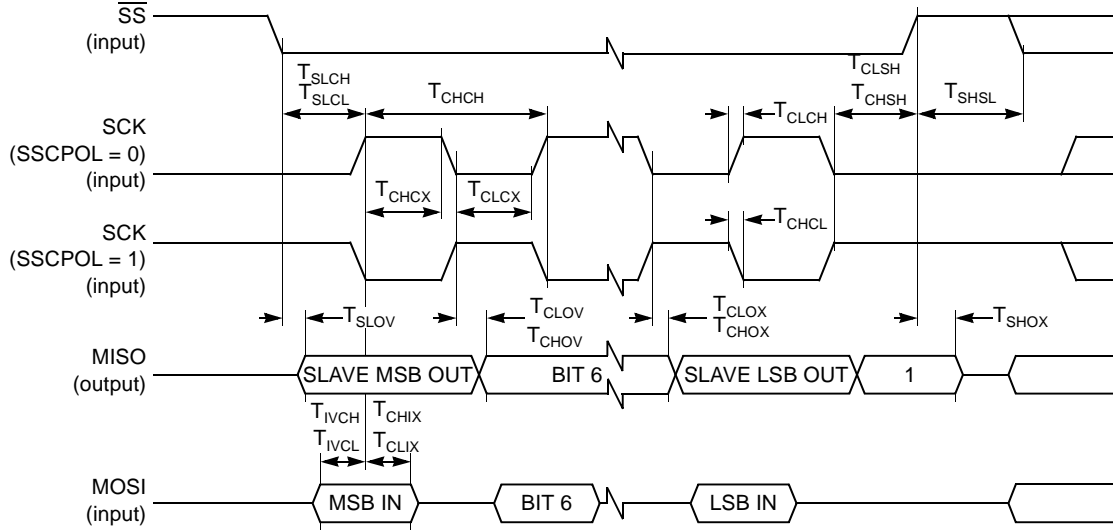
$V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = -40^\circ$  to  $+85^\circ C$

Symbol	Parameter	Min	Max	Unit
<b>Slave Mode</b>				
$T_{CHCH}$	Clock Period	8		$T_{OSC}$
$T_{CHCX}$	Clock High Time	3.2		$T_{OSC}$
$T_{CLCX}$	Clock Low Time	3.2		$T_{OSC}$
$T_{SLCH}, T_{SLCL}$	$\overline{SS}$ Low to Clock edge	200		ns
$T_{IVCL}, T_{IVCH}$	Input Data Valid to Clock Edge	100		ns
$T_{CLIX}, T_{CHIX}$	Input Data Hold after Clock Edge	100		ns
$T_{CLOV}, T_{CHOV}$	Output Data Valid after Clock Edge		100	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
$T_{IVCL}, T_{IVCH}$	Input Data Valid to Clock Edge	100		ns
$T_{CLIX}, T_{CHIX}$	Input Data Hold after Clock Edge	100		ns
$T_{SLOV}$	$\overline{SS}$ Low to Output Data Valid		130	ns
$T_{SHOX}$	Output Data Hold after $\overline{SS}$ High		130	ns
$T_{SHSL}$	$\overline{SS}$ High to $\overline{SS}$ Low	(1)		
$T_{ILIH}$	Input Rise Time		2	$\mu s$
$T_{IHIL}$	Input Fall Time		2	$\mu s$
$T_{OLOH}$	Output Rise Time		100	ns
$T_{OHOL}$	Output Fall Time		100	ns
<b>Master Mode</b>				
$T_{CHCH}$	Clock Period	4		$T_{OSC}$
$T_{CHCX}$	Clock High Time	1.6		$T_{OSC}$
$T_{CLCX}$	Clock Low Time	1.6		$T_{OSC}$
$T_{IVCL}, T_{IVCH}$	Input Data Valid to Clock Edge	50		ns
$T_{CLIX}, T_{CHIX}$	Input Data Hold after Clock Edge	50		ns
$T_{CLOV}, T_{CHOV}$	Output Data Valid after Clock Edge		65	ns
$T_{CLOX}, T_{CHOX}$	Output Data Hold Time after Clock Edge	0		ns
$T_{ILIH}$	Input Data Rise Time		2	$\mu s$
$T_{IHIL}$	Input Data Fall Time		2	$\mu s$
$T_{OLOH}$	Output Data Rise Time		50	ns
$T_{OHOL}$	Output Data Fall Time		50	ns

Notes: 1. Value of this parameter depends on software.  
 2. Test conditions: capacitive load on all pins = 100 pF

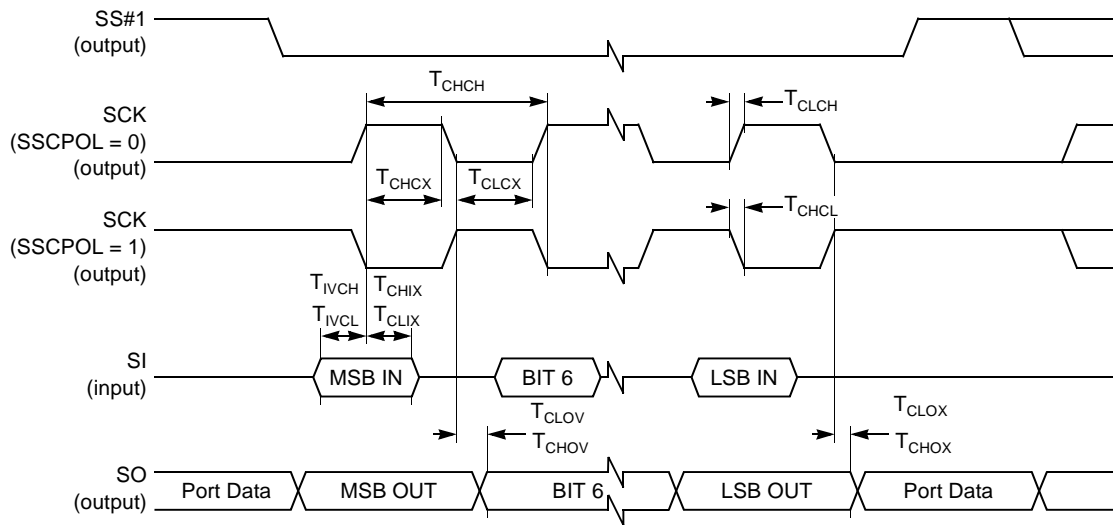
Waveforms

**Figure 123.** SPI Slave Waveforms (SSCPHA = 0)



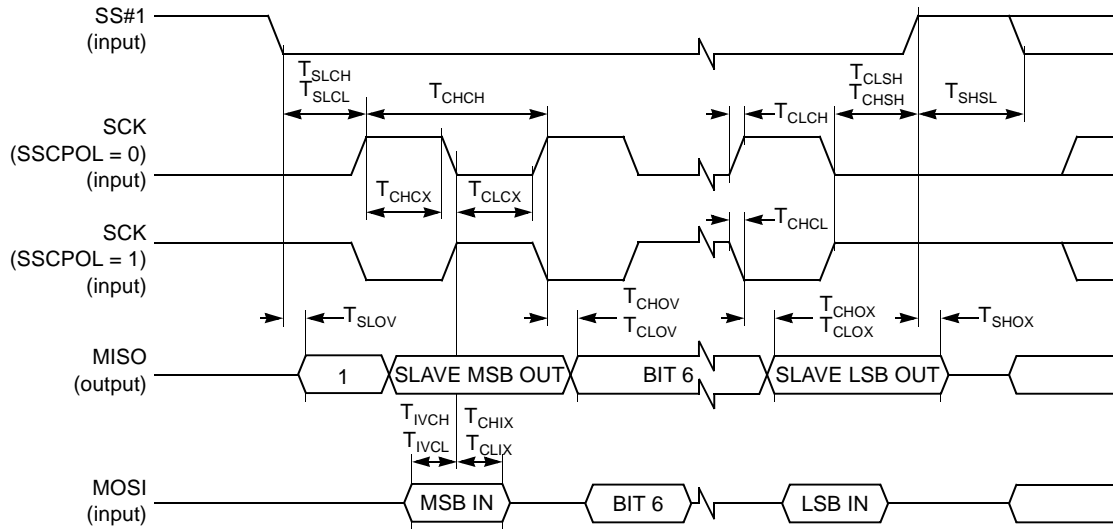
Note: Not Defined but generally the MSB of the character that has just been received.

**Figure 124.** SPI Slave Waveforms (SSCPHA = 1)



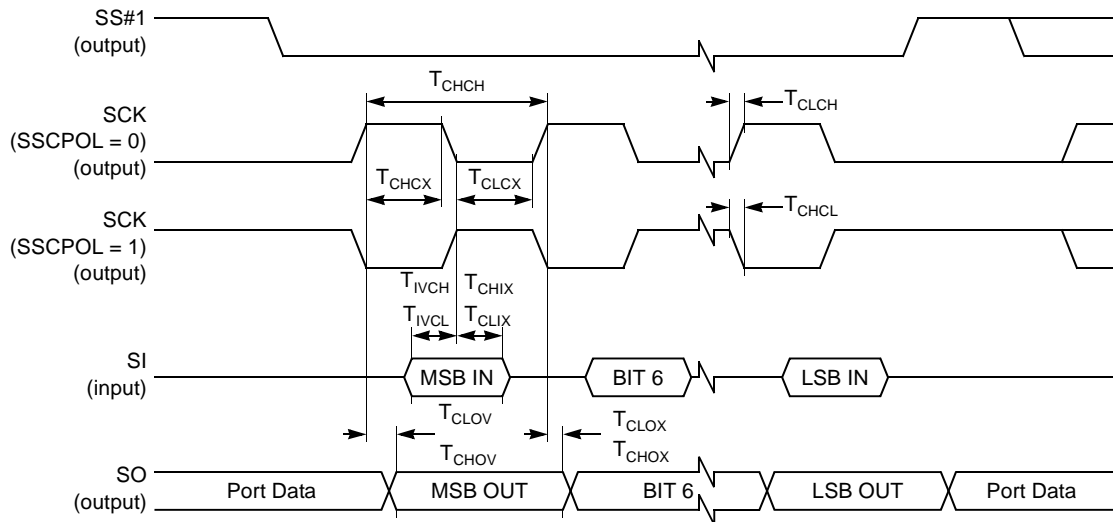
Note: Not Defined but generally the LSB of the character that has just been received.

**Figure 125. SPI Master Waveforms (SSCPHA = 0)**



Note:  $\overline{SS}$  handled by software using general purpose port pin.

**Figure 126. SPI Master Waveforms (SSCPHA = 1)**



Note:  $\overline{SS}$  handled by software using general purpose port pin.

Specific Controller

To be defined.

MMC Interface

Definition of Symbols

Table 137. MMC Interface Timing Symbol Definitions

Signals		Conditions	
C	Clock	H	High
D	Data In	L	Low
O	Data Out	V	Valid
		X	No Longer Valid

Timings

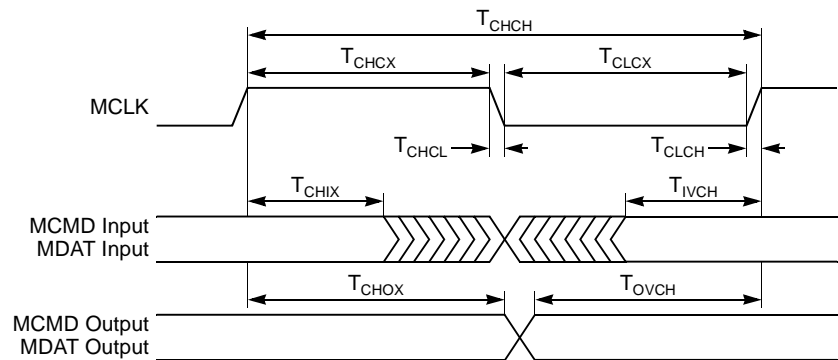
Table 138. MMC Interface AC timings

$V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = 0$  to  $70^\circ C$ ,  $CL \leq 100pF$  (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 127. MMC Input Output Waveforms



## Audio Interface

### Definition of Symbols

**Table 139.** Audio Interface Timing Symbol Definitions

Signals		Conditions	
C	Clock	H	High
O	Data Out	L	Low
S	Data Select	V	Valid
		X	No Longer Valid

### Timings

**Table 140.** Audio Interface AC timings

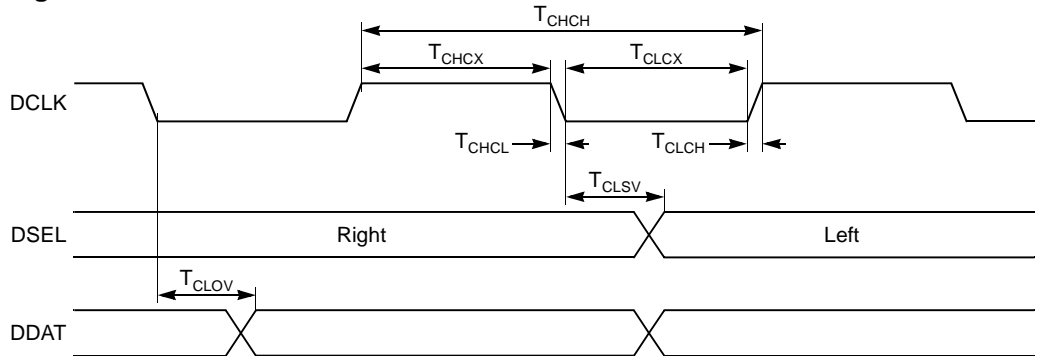
$V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = 0$  to  $70^\circ C$ ,  $CL \leq 30pF$

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: 32-bit format with  $F_s = 48$  kHz.

### Waveforms

**Figure 128.** Audio Interface Waveforms



Analog to Digital Converter

Definition of Symbols

**Table 141.** 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 142.** Analog to Digital Converter AC Characteristics

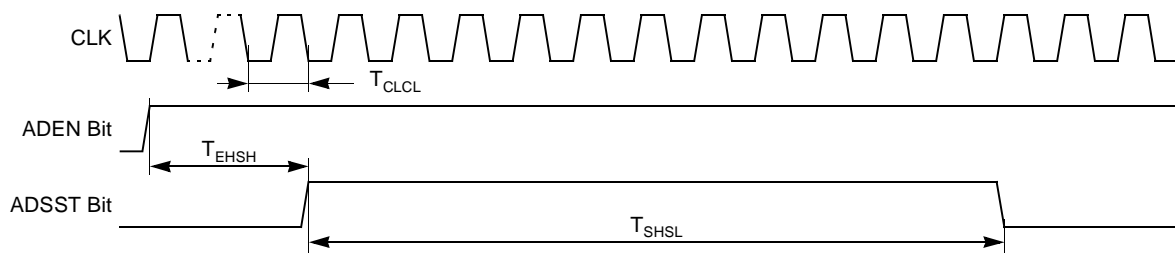
$V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = 0$  to  $70^{\circ}C$

Symbol	Parameter	Min	Max	Unit
$T_{CLCL}$	Clock Period	1.43		$\mu s$
$T_{EHS}$	Start-up Time		4	$\mu s$
$T_{SHSL}$	Conversion Time		$11 \cdot T_{CLCL}$	$\mu s$
DLe	Differential non-linearity error <sup>(1),(2)</sup>		TBD	LSB
ILe	Integral non-linearity error <sup>(1),(3)</sup>		TBD	LSB
OSe	Offset error <sup>(1),(4)</sup>		TBD	LSB
Ge	Gain error <sup>(1),(5)</sup>		TBD	%

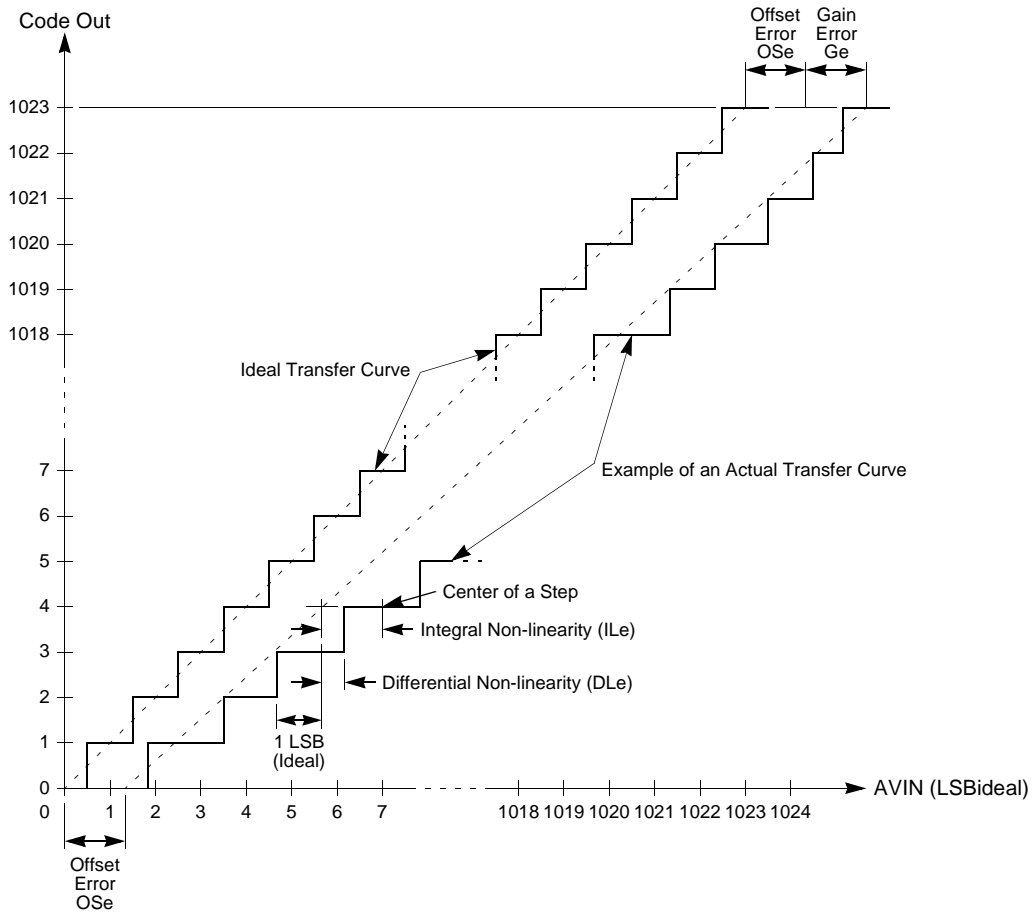
- Notes:
1.  $AV_{DD} = AV_{REFP} = 3.0V$ ,  $AV_{SS} = AV_{REFN} = 0V$ . ADC is monotonic with no missing code.
  2. The differential non-linearity is the difference between the actual step width and the ideal step width (see Figure 130).
  3. The integral non-linearity is the peak difference between the center of the actual step and the ideal transfer curve after appropriate adjustment of gain and offset errors (see Figure 130).
  4. 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 130).
  5. 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 130).

Waveforms

**Figure 129.** Analog-to-Digital Converter Internal Waveforms



**Figure 130.** Analog-to-Digital Converter Characteristics



## Flash Memory

### Definition of Symbols

**Table 143.** Flash Memory Timing Symbol Definitions

Signals	
S	ISP#
R	RST
B	FBUSY flag

Conditions	
L	Low
V	Valid
X	No Longer Valid

### Timings

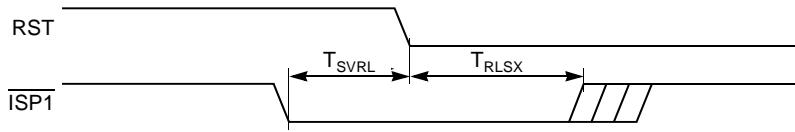
**Table 144.** Flash Memory AC Timing

$V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = -40^\circ$  to  $+85^\circ C$

Symbol	Parameter	Min	Typ	Max	Unit
$T_{SVRL}$	Input $\overline{ISP}$ Valid to RST Edge	50			ns
$T_{RLSX}$	Input $\overline{ISP}$ Hold after RST Edge	50			ns
$T_{BHBL}$	Flash Internal Busy (Programming) Time		10		ms

## Waveforms

**Figure 131.** Flash Memory – ISP Waveforms



Note:  $\overline{\text{ISP}}$  must be driven through a pull-down resistor (see Section “In-system Programming”, page 140).

**Figure 132.** Flash Memory – Internal Busy Waveforms



## External Clock Drive and Logic Level References

### Definition of Symbols

**Table 145.** External Clock Timing Symbol Definitions

Signals	
C	Clock

Conditions	
H	High
L	Low
X	No Longer Valid

### Timings

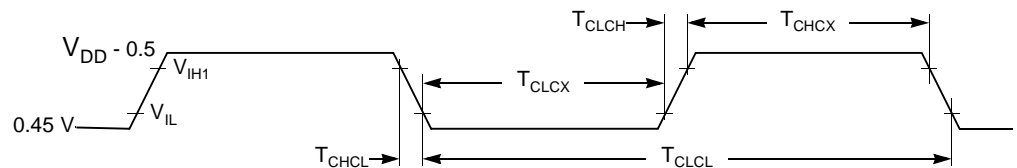
**Table 146.** External Clock AC Timings

$V_{DD} = 2.7$  to  $3.3V$ ,  $T_A = 0$  to  $70^\circ C$

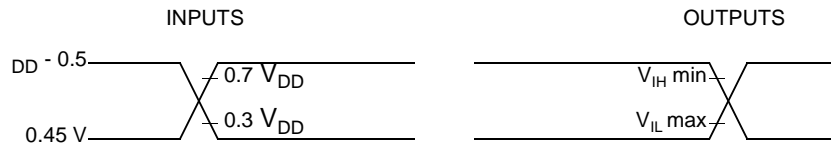
Symbol	Parameter	Min	Max	Unit
$T_{CLCL}$	Clock Period	50		ns
$T_{CHCX}$	High Time	10		ns
$T_{CLCX}$	Low Time	10		ns
$T_{CLCH}$	Rise Time	3		ns
$T_{CHCL}$	Fall Time	3		ns
$T_{CR}$	Cyclic Ratio in X2 Mode	40	60	%

## Waveforms

**Figure 133.** External Clock Waveform

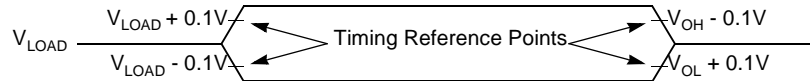


**Figure 134.** AC Testing Input/Output Waveforms



- Notes:
1. During AC testing, all inputs are driven at  $V_{DD} - 0.5V$  for a logic 1 and  $0.45V$  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 135.** 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

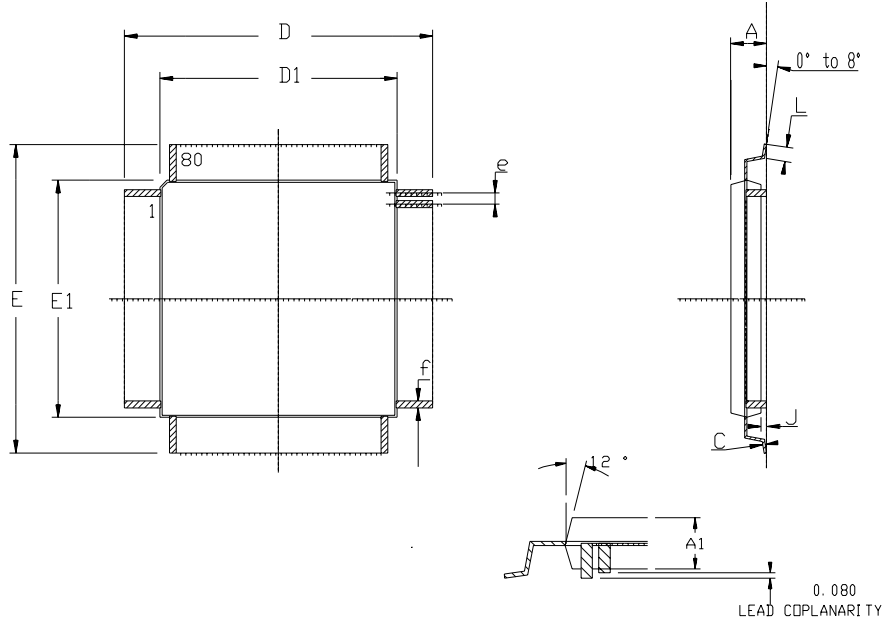
Possible Order Entries<sup>(2)</sup>

Part Number	Memory Size (Bytes)	Supply Voltage	Temperature Range	Max Frequency (MHz)	Package	Packing
AT89C5132-ROTIL	64K Flash	3V	Industrial	40	TQFP80	Tray
AT83C5132xxx <sup>(1)</sup> -ROTIL	64K ROM	3V	Industrial	40	TQFP80	Tray
AT89C5132-RDTIL	64K ROM	3V	Industrial	40	TQFP64	Tray
AT83C5132xxx <sup>(1)</sup> -RDTIL	64K ROM	3V	Industrial	40	TQFP64	Tray

- Notes:
1. Refers to ROM code. Check for availability.
  2. PLCC84 package only available for development board.

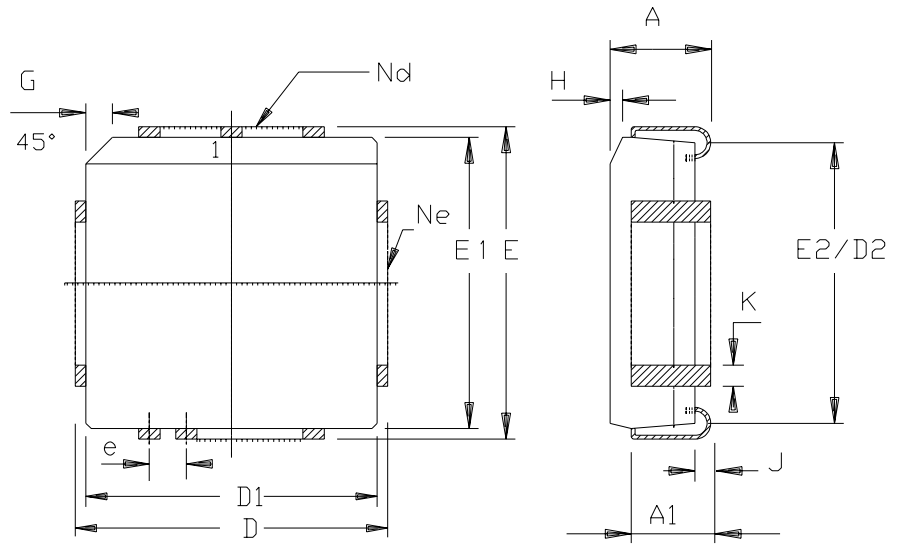
# Package Information

## TQFP80



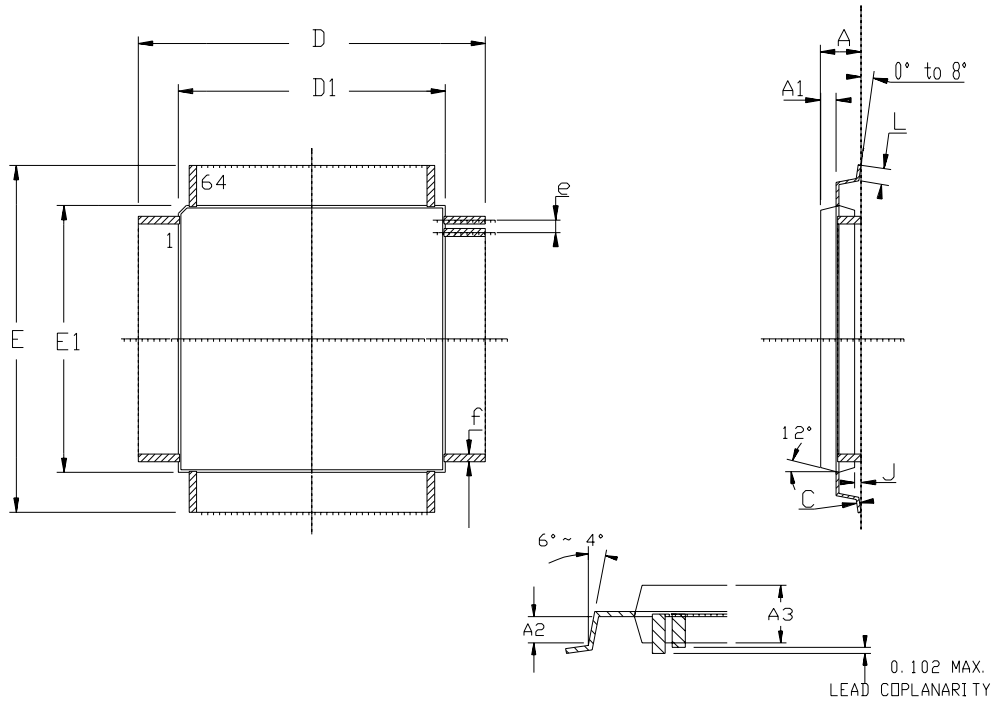
	MM		INCH	
	Min	Max	Min	Max
A	1.40	1.60	.055	.063
A1	1.35	1.45	.053	.057
C	0.17 BSC		.007 BSC	
D	15.80	16.20	.622	.638
D1	13.90	14.10	.547	.555
E	15.80	16.20	.622	.638
E1	13.90	14.10	.547	.555
J	0.05	0.15	.002	.006
L	0.45	0.75	.018	.030
e	0.65 BSC		.0256 BSC	
f	0.30 BSC		.012 BSC	

PLCC84



	MM		INCH	
	A	4.20	5.08	.165
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			

TQFP64



	MM		INCH	
	Min	Max	Min	Max
A	-	1.60	-	.063
A1	0.64 REF		.025 REF	
A2	0.64 REF		.025 REF	
A3	1.35	1.45	.053	.057
D	11.75	12.25	.463	.483
D1	9.90	10.10	.390	.398
E	11.75	12.25	.463	.483
E1	9.90	10.10	.390	.398
J	0.05	-	.002	-
L	0.45	0.75	.018	.030
e	0.50 BSC		.0197 BSC	
f	0.25 BSC		.010 BSC	

Table of Contents

**Features ..... 1**

**Description ..... 1**

**Typical Applications ..... 1**

**Block Diagram ..... 2**

**Pin Configuration ..... 3**

**Pin Description ..... 6**

    Internal Pin Structure ..... 11

**Clock Controller ..... 12**

    Oscillator ..... 12

    X2 Feature ..... 12

    PLL ..... 13

    Registers ..... 15

**Program/Code Memory ..... 18**

    ROM Memory Architecture ..... 18

    Flash Memory Architecture ..... 19

    Hardware Security System ..... 20

    Boot Memory Execution ..... 20

    Registers ..... 22

    Hardware Bytes ..... 23

**Data Memory ..... 24**

    Internal Space ..... 24

    External Space ..... 26

    Dual Data Pointer ..... 28

    Registers ..... 29

**Special Function Registers ..... 31**

**Interrupt System ..... 36**

    Interrupt System Priorities ..... 36

    External Interrupts ..... 39

    Registers ..... 40

**Power Management ..... 46**

    Reset ..... 46

    Reset Recommendation to Prevent Flash Corruption ..... 46

    Idle Mode ..... 47

Power-down Mode.....	47
Registers.....	49
<b>Timers/Counters .....</b>	<b>50</b>
Timer/Counter Operations .....	50
Timer Clock Controller .....	50
Timer 0.....	51
Timer 1.....	53
Interrupt .....	54
Registers.....	55
<b>Watchdog Timer .....</b>	<b>58</b>
Description.....	58
Watchdog Clock Controller .....	58
Watchdog Operation.....	59
Registers.....	60
<b>Audio Output Interface .....</b>	<b>61</b>
Description.....	61
Clock Generator.....	62
Data Converter .....	62
Audio Buffer .....	63
Interrupt Request.....	64
Voice or Sound Playing .....	64
Registers.....	66
<b>Universal Serial Bus .....</b>	<b>68</b>
Description.....	69
USB Interrupt System .....	71
Registers.....	73
<b>MultiMedia Card Controller .....</b>	<b>82</b>
Card Concept.....	82
Bus Concept .....	82
Description.....	87
Clock Generator.....	88
Command Line Controller.....	89
Data Line Controller.....	91
Interrupt .....	97
Registers.....	98
<b>IDE/ATAPI Interface .....</b>	<b>104</b>
Description.....	104
Registers.....	106

<b>Serial I/O Port .....</b>	<b>107</b>
Mode Selection .....	107
Baud Rate Generator.....	107
Synchronous Mode (Mode 0) .....	108
Asynchronous Modes (Modes 1, 2 and 3) .....	110
Multiprocessor Communication (Modes 2 and 3) .....	113
Automatic Address Recognition.....	113
Interrupt .....	115
Registers.....	116
<b>Synchronous Peripheral Interface .....</b>	<b>119</b>
Description.....	120
Interrupt .....	123
Configuration .....	124
Registers.....	127
<b>Analog-to-Digital Converter .....</b>	<b>129</b>
Description.....	129
Registers.....	132
<b>Keyboard Interface .....</b>	<b>134</b>
Description.....	134
Registers.....	135
<b>Electrical Characteristics .....</b>	<b>136</b>
Absolute Maximum Ratings .....	136
DC Characteristics.....	136
AC Characteristics .....	141
<b>Ordering Information .....</b>	<b>155</b>
<b>Package Information .....</b>	<b>156</b>
TQFP80 .....	156
PLCC84 .....	157
TQFP64 .....	158



## Atmel Headquarters

### Corporate Headquarters

2325 Orchard Parkway  
San Jose, CA 95131  
TEL 1(408) 441-0311  
FAX 1(408) 487-2600

### 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 Tsimhatsui  
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 Data- com

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>

### © Atmel Corporation 2002.

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<sup>®</sup>, is a registered trademark of Atmel. MultiMedia Card<sup>®</sup> is a registered trademark of MultiMedia Corporation. SmartMedia<sup>®</sup> is a registered trademark of Toshiba Corporation. CompactFlash<sup>™</sup> is a trademark of CompactFlash Corporation.

Other terms and product names may be the trademarks of others.



Printed on recycled paper.