## Features

- 80C52X2 Core (6 Clocks per Instruction)
  - Maximum Core Frequency 40 MHz in X1 Mode
  - Dual Data Pointer
  - Full-duplex Enhanced UART (EUART)
  - Three 16-bit Timer/Counters: T0, T1 and T2
  - 256 Bytes of Scratchpad RAM
- 32-Kbyte On-chip Flash In-System Programming through USB or UART
- 4-Kbyte EEPROM for Boot (3-Kbyte) and Data (1-Kbyte)
- On-chip Expanded RAM (XRAM): 1024 Bytes
- USB Module with Interrupt on Transfer Completion
  - Endpoint 0 for Control Transfers: 32-byte FIFO
  - 6 Programmable Endpoints with In or Out Directions and with Bulk, Interrupt or Isochronous Transfers
    - Endpoint 1, 2, 3: 32-byte FIFO
    - Endpoint 4, 5: 2 x 64-byte FIFO with Double Buffering (Ping-pong Mode)
    - Endpoint 6: 2 x 512-byte FIFO with Double Buffering (Ping-pong Mode)
  - Suspend/Resume Interrupts
  - Power-on Reset and USB Bus Reset
  - 48 MHz DPLL for Full-speed Bus Operation
  - USB Bus Disconnection on Microcontroller Request
- 5 Channels Programmable Counter Array (PCA) with 16-bit Counter, High-speed Output, Compare/Capture, PWM and Watchdog Timer Capabilities
- Programmable Hardware Watchdog Timer (One-time Enabled with Reset-out): 50 ms to 6s at 4 MHz
- Keyboard Interrupt Interface on Port P1 (8 Bits)
- SPI Interface (Master/Slave Mode)
- 34 I/O Pins
- 4 Direct-drive LED Outputs with Programmable Current Sources: 2-6-10 mA Typical
- 4-level Priority Interrupt System (11 sources)
- Idle and Power-down Modes
- 0 to 32 MHz On-chip Oscillator with Analog PLL for 48 MHz Synthesis
- Voltage Regulator and Reference Output: 3.3V/4 mA
- Low Power Voltage Range
  - 3.0V to 3.6V
  - 30 mA Max Operating Current (at 40 MHz)
  - 100 µA Max Power-down Current
- Self-powered USB Voltage Range (Not Available on First Version)
  - 3.6V to 5.5V
  - 30 mA Max Operating Current (at 40 MHz)
  - 200 µA Max Power-down Current
- Commercial and Industrial Temperature Range
- Packages: PLCC52, VQFP64, MLF48, SO28

# 8-bit Flash Microcontroller with Full Speed USB Device

# AT89C5131

Rev. 4136A–USB–03/03

## Description

AT89C5131 is a high-performance Flash version of the 80C51 single-chip 8-bit microcontrollers with full speed USB functions.

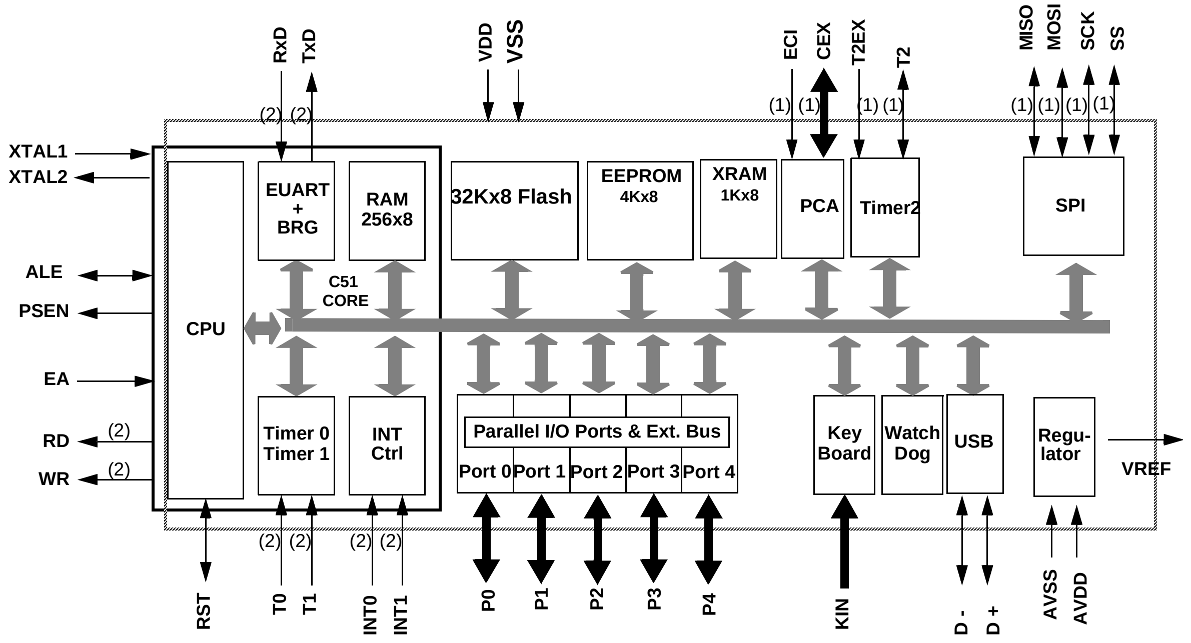
AT89C5131 features a full-speed USB module compatible with the USB specifications Version 1.1 and 2.0. This module integrates the USB transceivers with a 3.3V voltage regulator and the Serial Interface Engine (SIE) with Digital Phase Locked Loop and 48 MHz clock recovery. USB Event detection logic (Reset and Suspend/Resume) and FIFO buffers supporting the mandatory control Endpoint (EP0) and up to 6 versatile Endpoints (EP1/EP2/EP3/EP4/EP5/EP6) with minimum software overhead are also part of the USB module.

AT89C5131 retains the features of the Atmel 80C52 with extended Flash capacity (32-Kbyte), 256 bytes of internal RAM, a 4-level interrupt system, two 16-bit timer/counters (T0/T1), a full duplex enhanced UART (EUART) and an on-chip oscillator.

In addition, AT89C5131 has an on-chip expanded RAM of 1024 bytes (XRAM), a dual-data pointer, a 16-bit up/down Timer (T2), a Programmable Counter Array (PCA), up to 4 programmable LED current sources, a programmable hardware watchdog and a power-on reset.

AT89C5131 has two software-selectable modes of reduced activity for further reduction in power consumption. In the idle mode the CPU is frozen while the timers, the serial ports and the interrupt system are still operating. In the power-down mode the RAM is saved, the peripheral clock is frozen, but the device has full wake-up capability through USB events or external interrupts.

## Block Diagram



Notes:
1. Alternate function of Port 1
2. Alternate function of Port 3
3. Alternate function of Port 4 (Alternate function of Port 4 on PLCC52, under evaluation)

## Pinout Description

### Pinout

**Figure 1.** AT89C5131 52-pin PLCC Pinout

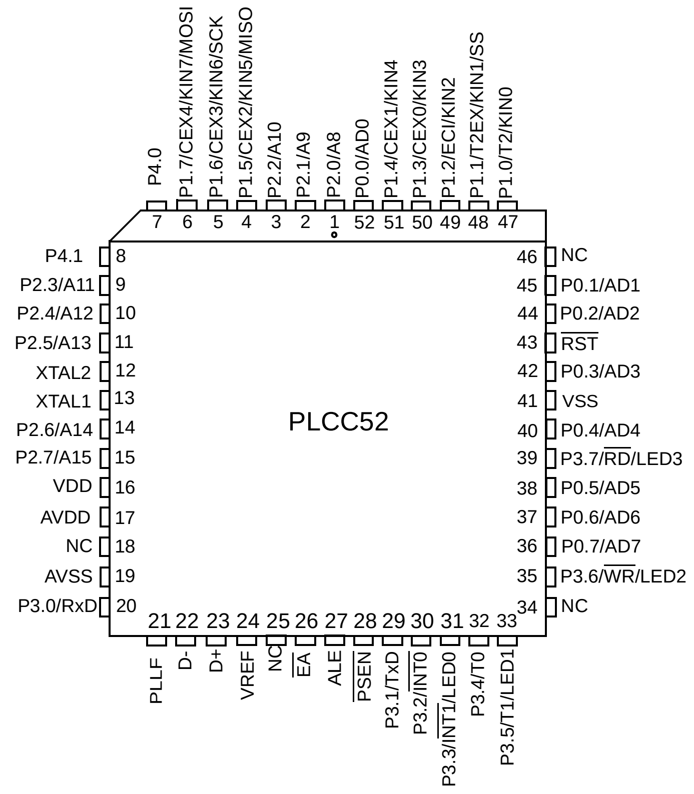


**Figure 2.** AT89C5131 64-pin VQFP Pinout

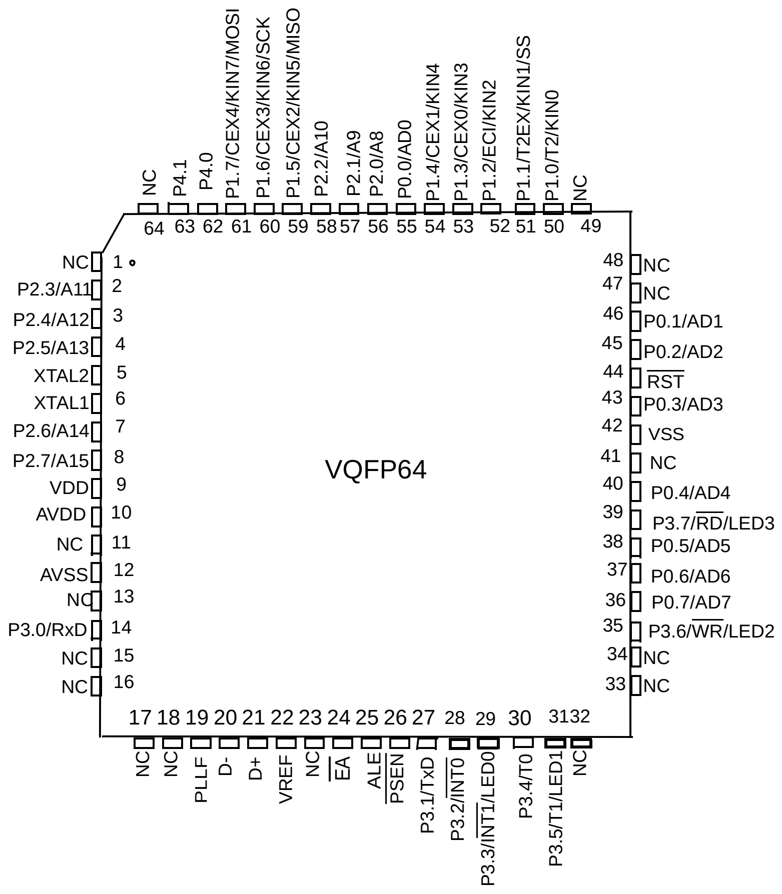


**Figure 3.** AT89C5131 48-pin MLF Pinout

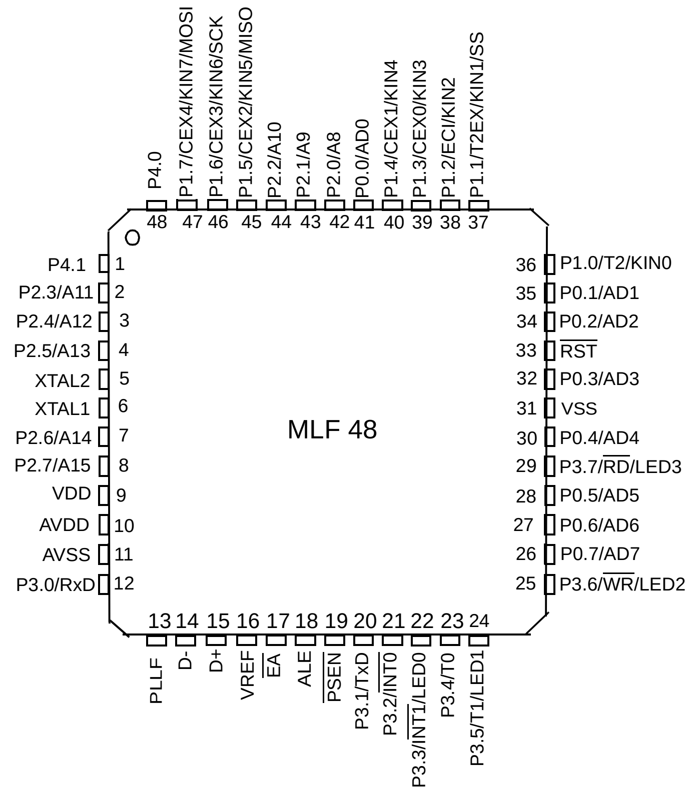
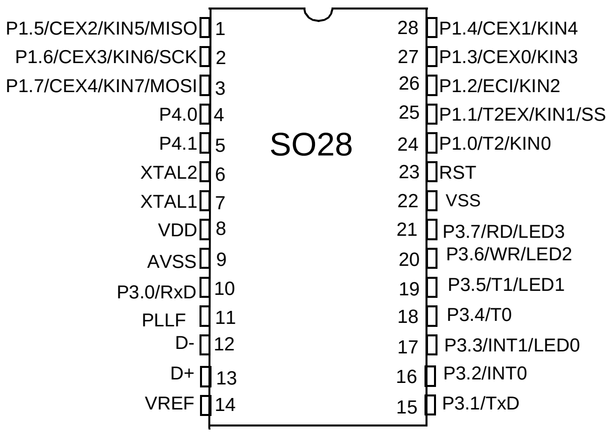


**Figure 4.** AT89C5131 28-pin SO Pinout



## Signals

All the AT89C5131 signals are detailed by functionality on Table 1 through Table 11.

**Table 1.** Keypad Interface Signal Description

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| KIN[7:0) | I | **Keypad Input Lines**<br>Holding one of these pins high or low for 24 oscillator periods triggers a keypad interrupt if enabled. Held line is reported in the KBCON register. | P1[7:0] |

**Table 2.** Programmable Counter Array Signal Description

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| ECI | I | **External Clock Input** | P1.2 |
| CEX[4:0] | I/O | Capture External Input<br>Compare External Output | P1.3<br>P1.4<br>P1.5<br>P1.6<br>P1.7 |

**Table 3.** Serial I/O Signal Description

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| RxD | I | **Serial Input**<br>The serial input is P3.0 after reset, but it can also be configured to P4.0 by software. | P3.0 |
| TxD | O | **Serial Output**<br>The serial output is P3.1 after reset, but it can also be configured to P4.1 by software. | P3.1 |

**Table 4.** Timer 0, Timer 1 and Timer 2 Signal Description

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| INT0 | I | **Timer 0 Gate Input**<br>$\overline{\text{INT0}}$ serves as external run control for timer 0, when selected by GATE0 bit in TCON register.<br>**External Interrupt 0**<br>$\overline{\text{INT0}}$ input set IE0 in the TCON register. If bit IT0 in this register is set, bits IE0 are set by a falling edge on $\overline{\text{INT0}}$. If bit IT0 is cleared, bits IE0 is set by a low level on $\overline{\text{INT0}}$. | P3.2 |
| INT1 | I | **Timer 1 Gate Input**<br>$\overline{\text{INT1}}$ serves as external run control for Timer 1, when selected by GATE1 bit in TCON register.<br>**External Interrupt 1**<br>$\overline{\text{INT1}}$ input set IE1 in the TCON register. If bit IT1 in this register is set, bits IE1 are set by a falling edge on $\overline{\text{INT1}}$. If bit IT1 is cleared, bits IE1 is set by a low level on $\overline{\text{INT1}}$. | P3.3 |

**Table 4.** Timer 0, Timer 1 and Timer 2 Signal Description (Continued)

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| T0 | I | **Timer Counter 0 External Clock Input**<br>When Timer 0 operates as a counter, a falling edge on the T0 pin increments the count. | P3.4 |
| T1 | I | **Timer/Counter 1 External Clock Input**<br>When Timer 1 operates as a counter, a falling edge on the T1 pin increments the count. | P3.5 |
| T2 | I<br>O | **Timer/Counter 2 External Clock Input**<br>Timer/Counter 2 Clock Output | P1.0 |
| T2EX | I | Timer/Counter 2 Reload/Capture/Direction Control Input | P1.1 |

**Table 5.** LED Signal Description

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| LED[3:0] | O | **Direct Drive LED Output**<br>These pins can be directly connected to the Cathode of standard LEDs without external current limiting resistors. The typical current of each output can be programmed by software to 2, 6 or 10 mA. Several outputs can be connected together to get higher drive capabilities. | P3.3<br>P3.5<br>P3.6<br>P3.7 |

**Table 6.** SPI Signal Description

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| SS | I/O | **$\overline{\text{SS}}$:** SPI Slave Select | P1.1 |
| MISO | I/O | **MISO:** SPI Master Input Slave Output line<br>When SPI is in master mode, MISO receives data from the slave peripheral. When SPI is in slave mode, MISO outputs data to the master controller. | P1.5 |
| SCK | I/O | **SCK:** SPI Serial Clock<br>SCK outputs clock to the slave peripheral or receive clock from the master | P1.6 |
| MOSI | I/O | **MOSI:** SPI Master Output Slave Input line<br>When SPI is in master mode, MOSI outputs data to the slave peripheral. When SPI is in slave mode, MOSI receives data from the master controller | P1.7 |

**Table 7.** Ports Signal Description

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| P0[7:0] | I/O | **Port 0**<br>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 pulled to $V_{DD}$ or $V_{SS}$. | AD[7:0] |
| P1[7:0] | I/O | **Port 1**<br>P1 is an 8-bit bidirectional I/O port with internal pull-ups, except for P1.6 and P1.7 that are true open drain outputs. | KIN[7:0]<br>T2<br>T2EX<br>ECI<br>CEX[4:0] |
| P2[7:0] | I/O | **Port 2**<br>P2 is an 8-bit bidirectional I/O port with internal pull-ups. | A[15:8] |
| P3[7:0] | I/O | **Port 3**<br>P3 is an 8-bit bidirectional I/O port with internal pull-ups. | LED[3:0]<br>RxD<br>TxD<br>$\overline{\text{INT0}}$<br>$\overline{\text{INT1}}$<br>T0<br>T1<br>$\overline{\text{WR}}$<br>$\overline{\text{RD}}$ |
| P4[1:0] | I/O | **Port 4**<br>P4 is an 2-bit bidirectional I/O port. | |

**Table 8.** Clock Signal Description

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| XTAL1 | I | **Input to the on-chip inverting oscillator amplifier**<br>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. | - |
| XTAL2 | O | **Output of the on-chip inverting oscillator amplifier**<br>To use the internal oscillator, a crystal/resonator circuit is connected to this pin. If an external oscillator is used, leave XTAL2 unconnected. | - |
| PLLF | I | **PLL Low Pass Filter input**<br>Receives the RC network of the PLL low pass filter. | - |

**Table 9.** USB Signal Description

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| D+ | I/O | USB Data + signal | - |
| D- | I/O | USB Data - signal | - |
| VREF | O | **USB Reference Voltage**<br>Connect this pin to D+ using a 1.5 kΩ resistor to use the Detach function. | - |

**Table 10.** System Signal Description

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| AD[7:0] | I/O | Multiplexed Address/Data LSB for external access<br>Data LSB for Slave port access (used for 8-bit and 16-bit modes) | P0[7:0] |
| A[15:8] | I/O | Address Bus MSB for external access<br>Data MSB for Slave port access (used for 16-bit mode only) | P2[7:0] |
| RD | I/O | **Read Signal**<br>Read signal asserted during external data memory read operation.<br>Control input for slave port read access cycles. | P3.7 |
| WR | I/O | **Write Signal**<br>Write signal asserted during external data memory write operation.<br>Control input for slave write access cycles. | P3.6 |
| RST | I | **Reset Input**<br>Holding this pin low for 64 oscillator periods while the oscillator is running resets the device. The Port pins are driven to their reset conditions when a voltage lower than $V_{IL}$ is applied, whether or not the oscillator is running.<br>This pin has an internal pull-up resistor which allows the device to be reset by connecting a capacitor between this pin and VSS.<br>Asserting $\overline{RST}$ when the chip is in Idle mode or Power-down mode returns the chip to normal operation.<br>This pin is set to 0 for at least 12 oscillator periods when an internal reset occurs. | - |
| ALE | O | **Address Latch Enable Output**<br>The falling edge of ALE strobes the address into external latch. This signal is active only when reading or writing external memory using MOVX instructions. | - |
| PSEN | O | **Program**<br>Test mode entry signal. This pin must be set to $V_{DD}$ for normal operation. | - |
| EA | I | **External Access Enable**<br>This pin must be held low to force the device to fetch code from external program memory starting at address 0000h. It is latched during reset and cannot be dynamically changed during operation. | - |

**Table 11.** Power Signal Description

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| AVSS | GND | **Alternate Ground**<br>AVSS is used to supply the on-chip PLL. | - |
| AVDD | PWR | **Alternate Ground**<br>AVDD is used to supply the on-chip PLL. | - |
| VSS | GND | **Digital Ground**<br>VSS is used to supply the buffer ring and the digital core. | - |
| VDD | PWR | **Digital Supply Voltage**<br>VDD is used to supply the buffer ring on all versions of the device.<br>It is also used to power the on-chip voltage regulator of the Standard versions or the digital core of the Low Power versions. | - |

**Table 11.** Power Signal Description (Continued)

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| VREF | PWR | **3V Voltage Reference**<br>VREF is used to supply the on-chip USB differential drivers.<br>It is internally connected to the on-chip voltage regulator output of the standard versions, which must be connected to an external decoupling capacitor and can be connected to D+ with a 15 kΩ resistor.<br>It must be provided from outside on the Low Power versions, which have no internal voltage regulator. | - |

## SFR Mapping

The Special Function Registers (SFRs) of the AT89C5131 fall into the following categories:

- C51 core registers: ACC, B, DPH, DPL, PSW, SP
- I/O port registers: P0, P1, P2, P3, P4
- Timer registers: T2CON, T2MOD, TCON, TH0, TH1, TH2, TMOD, TL0, TL1, TL2, RCAP2L, RCAP2H
- Serial I/O port registers: SADDR, SADEN, SBUF, SCON
- PCA (Programmable Counter Array) registers: CCON, CMOD, CCAPMx, CL, CH, CCAPxH, CCAPxL (x: 0 to 4)
- Power and clock control registers: PCON
- Hardware Watchdog Timer registers: WDTRST, WDTPRG
- Interrupt system registers: IE0, IPL0, IPH0, IE1, IPL1, IPH1
- Keyboard Interface registers: KBE, KBF, KBLS
- LED register: LEDCON
- Serial Port Interface (SPI) registers: SPCON, SPSTA, SPDAT
- USB registers: Uxxx (17 registers)
- PLL registers: PLLCON, PLLDIV0
- BRG (Baud Rate Generator) registers: BRL, BDRCON
- Flash register: FCON (FCON access is reserved for the Flash API and ISP software)
- EEPROM register: EECON
- Clock Prescaler register: CKRL
- 32 kHz Sub Clock Oscillator registers: CKSEL
- Others: AUXR, AUXR1, CKCON0, CKCON1

The table below shows all SFRs with their address and their reset value.

**Table 12.** SFR Descriptions

| | Bit Addressable | Non-Bit Addressable | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0/8 | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F | |
| F8h | UEPINT 0000 0000 | CH 0000 0000 | CCAP0H XXXX XXXX | CCAP1H XXXX XXXX | CCAP2H XXXX XXXX | CCAP3H XXXX XXXX | CCAP4H XXXX XXXX | | FFh |
| F0h | B 0000 0000 | LEDCON 0000 0000 | | | | | | | F7h |
| E8h | | CL 0000 0000 | CCAP0L XXXX XXXX | CCAP1L XXXX XXXX | CCAP2L XXXX XXXX | CCAP3L XXXX XXXX | CCAP4L XXXX XXXX | | EFh |
| E0h | ACC 0000 0000 | | UBYCTLX 0000 0000 | UBYCTHX 0000 0000 | | | | | E7h |
| D8h | CCON 00X0 0000 | CMOD 00XX X000 | CCAPM0 X000 0000 | CCAPM1 X000 0000 | CCAPM2 X000 0000 | CCAPM3 X000 0000 | CCAPM4 X000 0000 | | DFh |
| D0h | PSW 0000 0000 | FCON (1) XXXX 0000 | EECON 0000 0000 | | UEPCONX 1000 0000 | UEPRST 0000 0000 | UDPADDL 0000 0000 | UDPADDH 0000 0000 | D7h |
| C8h | T2CON 0000 0000 | T2MOD XXXX XX00 | RCAP2L 0000 0000 | RCAP2H 0000 0000 | TL2 0000 0000 | TH2 0000 0000 | UEPSTAX 0000 0000 | UEPDATX 0000 0000 | CFh |
| C0h | P4 XXXX 1111 | | UEPIEN 0000 0000 | SPCON 0001 0100 | SPSTA 0000 0000 | SPDAT XXXX XXXX | USBADDR 0000 0000 | UEPNUM 0000 0000 | C7h |
| B8h | IPL0 X000 000 | SADEN 0000 0000 | UFNUML 0000 0000 | UFNUMH 0000 0000 | USBCON 0000 0000 | USBINT 0000 0000 | USBIEN 0000 0000 | | BFh |
| B0h | P3 1111 1111 | IE1 XXXX X000 | IPL1 XXXX X000 | IPH1 XXXX X111 | | | | IPH0 X000 0000 | B7h |
| A8h | IE0 0000 0000 | SADDR 0000 0000 | | | | | | CKCON1 0000 0000 | AFh |
| A0h | P2 1111 1111 | | AUXR1 XXXX X0X0 | PLLCON XXXX XX00 | PLLDIV0 0000 0000 | | WDTRST XXXX XXXX | WDTPRG XXXX X000 | A7h |
| 98h | SCON 0000 0000 | SBUF XXXX XXXX | BRL 0000 0000 | BDRCON XXX0 0000 | KBLS 0000 0000 | KBE 0000 0000 | KBF 0000 0000 | | 9Fh |
| 90h | P1 1111 1111 | | | | | | | | 97h |
| 88h | TCON 0000 0000 | TMOD 0000 0000 | TL0 0000 0000 | TL1 0000 0000 | TH0 0000 0000 | TH1 0000 0000 | AUXR XX0X 0000 | CKCON0 0000 0000 | 8Fh |
| 80h | P0 1111 1111 | SP 0000 0111 | DPL 0000 0000 | DPH 0000 0000 | | | | PCON 00X1 0000 | 87h |
| | 0/8 | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F | |

Note: 1. FCON access is reserved for the Flash API and ISP software.

▨ Reserved

The Special Function Registers (SFRs) of the AT89C5131 fall into the following categories:

**Table 13.** 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 | | | | | | | | |
| SP | 81h | Stack Pointer<br>LSB of SPX | | | | | | | | |
| DPL | 82h | Data Pointer Low byte<br>LSB of DPTR | | | | | | | | |
| DPH | 83h | Data Pointer High byte<br>MSB of DPTR | | | | | | | | |

**Table 14.** I/O Port SFRs

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| P0 | 80h | Port 0 | | | | | | | | |
| P1 | 90h | Port 1 | | | | | | | | |
| P2 | A0h | Port 2 | | | | | | | | |
| P3 | B0h | Port 3 | | | | | | | | |
| P4 | C0h | Port 4 (x2) | | | | | | | | |

**Table 15.** Timer SFR's

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| TH0 | 8Ch | Timer/Counter 0 High byte | | | | | | | | |
| TL0 | 8Ah | Timer/Counter 0 Low byte | | | | | | | | |
| TH1 | 8Dh | Timer/Counter 1 High byte | | | | | | | | |
| TL1 | 8Bh | Timer/Counter 1 Low byte | | | | | | | | |
| TH2 | CDh | Timer/Counter 2 High byte | | | | | | | | |
| TL2 | CCh | Timer/Counter 2 Low byte | | | | | | | | |
| 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 |
| T2CON | C8h | Timer/Counter 2 control | TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/T2# | CP/RL2# |
| T2MOD | C9h | Timer/Counter 2 Mode | | | | | | | T2OE | DCEN |
| RCAP2H | CBh | Timer/Counter 2 Reload/Capture High byte | | | | | | | | |
| RCAP2L | CAh | Timer/Counter 2 Reload/Capture Low byte | | | | | | | | |
| WDTRST | A6h | WatchDog Timer Reset | | | | | | | | |
| WDTPRG | A7h | WatchDog Timer Program | | | | | | S2 | S1 | S0 |

**Table 16.** Serial I/O Port SFR's

| 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 | | | | | | | | |

**Table 17.** Baud Rate Generator SFR's

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| BRL | 9Ah | Baud Rate Reload | | | | | | | | |
| BDRCON | 9Bh | Baud Rate Control | | | | BRR | TBCK | RBCK | SPD | SRC |

**Table 18.** PCA SFR's

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| CCON | D8h | PCA Timer/Counter Control | CF | CR | | CCF4 | CCF3 | CCF2 | CCF1 | CCF0 |
| CMOD | D9h | PCA Timer/Counter Mode | CIDL | WDTE | | | | CPS1 | CPS0 | ECF |
| CL | E9h | PCA Timer/Counter Low byte | | | | | | | | |
| CH | F9h | PCA Timer/Counter High byte | | | | | | | | |
| CCAPM0<br>CCAPM1<br>CCAPM2<br>CCAPM3<br>CCAPM4 | DAh<br>DBh<br>DCh<br>DDh<br>DEh | PCA Timer/Counter Mode 0<br>PCA Timer/Counter Mode 1<br>PCA Timer/Counter Mode 2<br>PCA Timer/Counter Mode 3<br>PCA Timer/Counter Mode 4 | | ECOM0<br>ECOM1<br>ECOM2<br>ECOM3<br>ECOM4 | CAPP0<br>CAPP1<br>CAPP2<br>CAPP3<br>CAPP4 | CAP0<br>CAP1<br>CAP2<br>CAP3<br>CAP4 | MAT0<br>MAT1<br>MAT2<br>MAT3<br>MAT4 | TOG0<br>TOG1<br>TOG2<br>TOG3<br>TOG4 | PWM0<br>PWM1<br>PWM2<br>PWM3<br>PWM4 | ECCF0<br>ECCF1<br>ECCF2<br>ECCF3<br>ECCF4 |
| CCAP0H<br>CCAP1H<br>CCAP2H<br>CCAP3H<br>CCAP4H | FAh<br>FBh<br>FCh<br>FDh<br>FEh | PCA Compare Capture Module 0 H<br>PCA Compare Capture Module 1 H<br>PCA Compare Capture Module 2 H<br>PCA Compare Capture Module 3 H<br>PCA Compare Capture Module 4 H | CCAP0H7<br>CCAP1H7<br>CCAP2H7<br>CCAP3H7<br>CCAP4H7 | CCAP0H6<br>CCAP1H6<br>CCAP2H6<br>CCAP3H6<br>CCAP4H6 | CCAP0H5<br>CCAP1H5<br>CCAP2H5<br>CCAP3H5<br>CCAP4H5 | CCAP0H4<br>CCAP1H4<br>CCAP2H4<br>CCAP3H4<br>CCAP4H4 | CCAP0H3<br>CCAP1H3<br>CCAP2H3<br>CCAP3H3<br>CCAP4H3 | CCAP0H2<br>CCAP1H2<br>CCAP2H2<br>CCAP3H2<br>CCAP4H2 | CCAP0H1<br>CCAP1H1<br>CCAP2H1<br>CCAP3H1<br>CCAP4H1 | CCAP0H0<br>CCAP1H0<br>CCAP2H0<br>CCAP3H0<br>CCAP4H0 |
| CCAP0L<br>CCAP1L<br>CCAP2L<br>CCAP3L<br>CCAP4L | EAh<br>EBh<br>ECh<br>EDh<br>EEh | PCA Compare Capture Module 0 L<br>PCA Compare Capture Module 1 L<br>PCA Compare Capture Module 2 L<br>PCA Compare Capture Module 3 L<br>PCA Compare Capture Module 4 L | CCAP0L7<br>CCAP1L7<br>CCAP2L7<br>CCAP3L7<br>CCAP4L7 | CCAP0L6<br>CCAP1L6<br>CCAP2L6<br>CCAP3L6<br>CCAP4L6 | CCAP0L5<br>CCAP1L5<br>CCAP2L5<br>CCAP3L5<br>CCAP4L5 | CCAP0L4<br>CCAP1L4<br>CCAP2L4<br>CCAP3L4<br>CCAP4L4 | CCAP0L3<br>CCAP1L3<br>CCAP2L3<br>CCAP3L3<br>CCAP4L3 | CCAP0L2<br>CCAP1L2<br>CCAP2L2<br>CCAP3L2<br>CCAP4L2 | CCAP0L1<br>CCAP1L1<br>CCAP2L1<br>CCAP3L1<br>CCAP4L1 | CCAP0L0<br>CCAP1L0<br>CCAP2L0<br>CCAP3L0<br>CCAP4L0 |

**Table 19.** Interrupt SFR's

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| IE0 | A8h | Interrupt Enable Control 0 | EA | EC | ET2 | ES | ET1 | EX1 | ET0 | EX0 |
| IE1 | B1h | Interrupt Enable Control 1 | | EUSB | | | | ESPI | | EKB |
| IPL0 | B8h | Interrupt Priority Control Low 0 | | PPCL | PT2L | PSL | PT1L | PX1L | PT0L | PX0L |
| IPH0 | B7h | Interrupt Priority Control High 0 | | PPCH | PT2H | PSH | PT1H | PX1H | PT0H | PX0H |
| IPL1 | B2h | Interrupt Priority Control Low 1 | | PUSBL | | | | PSPIL | | PKBL |
| IPH1 | B3h | Interrupt Priority Control High 1 | | PUSBH | | | | PSPIH | | PKBH |

**Table 20.** PLL SFRs

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PLLCON | A3h | PLL Control | | | | | | | PLLEN | PLOCK |
| PLLDIV | A4h | PLL Divider | R3 | R2 | R1 | R0 | N3 | N2 | N1 | N0 |

**Table 21.** Keyboard SFRs

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| KBF | 9Eh | Keyboard Flag Register | KBF7 | KBF6 | KBF5 | KBF4 | KBF3 | KBF2 | KBF1 | KBF0 |
| KBE | 9Dh | Keyboard Input Enable Register | KBE7 | KBE6 | KBE5 | KBE4 | KBE3 | KBE2 | KBE1 | KBE0 |
| KBLS | 9Ch | Keyboard Level Selector Register | KBLS7 | KBLS6 | KBLS5 | KBLS4 | KBLS3 | KBLS2 | KBLS1 | KBLS0 |

**Table 22.** SPI SFRs

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| SPCON | C3h | Serial Peripheral Control | SPR2 | SPEN | SSDIS | MSTR | CPOL | CPHA | SPR1 | SPR0 |
| SPSTA | C4h | Serial Peripheral Status-Control | SPIF | WCOL | SSERR | MODF | - | - | - | - |
| SPDAT | C5h | Serial Peripheral Data | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |

**Table 23.** USDB SFR's

| Mnemonic | Add | Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| USBCON | BCh | USB Global Control | USBE | SUSPCLK | SDRMWUP | DETACH | UPRSM | RMWUPE | CONFG | FADDEN |
| USBADDR | C6h | USB Address | FEN | UADD6 | UADD5 | UADD4 | UADD3 | UADD2 | UADD1 | UADD0 |
| USBINT | BDh | USB Global Interrupt | - | - | WUPCPU | EORINT | SOFINT | - | - | SPINT |
| USBIEN | BEh | USB Global Interrupt Enable | - | - | EWUPCPU | EEORINT | ESOFINT | - | - | ESPINT |
| UEPNUM | C7h | USB Endpoint Number | - | - | - | - | EPNUM3 | EPNUM2 | EPNUM1 | EPNUM0 |
| UEPCONX | D4h | USB Endpoint X Control | EPEN | - | - | - | DTGL | EPDIR | EPTYPE1 | EPTYPE0 |
| UEPSTAX | CEh | USB Endpoint X Status | DIR | RXOUTB1 | STALLRQ | TXRDY | STLCRC | RXSETUP | RXOUTB0 | TXCMP |
| UEPRST | D5h | USB Endpoint Reset | - | EP6RST | EP5RST | EP4RST | EP3RST | EP2RST | EP1RST | EP0RST |
| UEPINT | F8h | USB Endpoint Interrupt | - | EP6INT | EP5INT | EP4INT | EP3INT | EP2INT | EP1INT | EP0INT |
| UEPIEN | C2h | USB Endpoint Interrupt Enable | - | EP6INTE | EP5INTE | EP4INTE | EP3INTE | EP2INTE | EP1INTE | EP0INTE |
| UEPDATX | CFh | USB Endpoint X FIFO Data | FDAT7 | FDAT6 | FDAT5 | FDAT4 | FDAT3 | FDAT2 | FDAT1 | FDAT0 |
| UBYCTLX | E2h | USB Byte Counter Low (EP X) | BYCT7 | BYCT6 | BYCT5 | BYCT4 | BYCT3 | BYCT2 | BYCT1 | BYCT0 |
| UBYCTHX | E3h | USB Byte Counter High (EP X) | - | - | - | - | - | BYCT10 | BYCT9 | BYCT8 |
| UFNUML | BAh | USB Frame Number Low | FNUM7 | FNUM6 | FNUM5 | FNUM4 | FNUM3 | FNUM2 | FNUM1 | FNUM0 |
| UFNUMH | BBh | USB Frame Number High | - | - | CRCOK | CRCERR | - | FNUM10 | FNUM9 | FNUM8 |

**Table 24.** Other SFR's

| **Mnemonic** | **Add** | **Name** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|---|---|
| PCON | 87h | Power Control | SMOD1 | SMOD0 | - | POF | GF1 | GF0 | PD | IDL |
| AUXR | 8Eh | Auxiliary Register 0 | DPU | - | M0 | - | XRS1 | XRS2 | EXTRAM | A0 |
| AUXR1 | A2h | Auxiliary Register 1 | - | - | ENBOOT | - | GF3 | - | - | DPS |
| CKCON0 | 8Fh | Clock Control 0 | - | WDX2 | PCAX2 | SIX2 | T2X2 | T1X2 | T0X2 | X2 |
| CKCON1 | AFh | Clock Control 1 | - | - | - | - | - | - | - | SPIX2 |
| LEDCON | F1h | LED Control | LED3 | | LED2 | | LED1 | | LED0 | |
| FCON | D1h | Flash Control | FPL3 | FPL2 | FPL1 | FPL0 | FPS | FMOD1 | FMOD0 | FBUSY |
| EECON | D2h | EEPROM Contol | EEPL3 | EEPL2 | EEPL1 | EEPL0 | - | - | EEE | EEBUSY |

## Clock Controller

### Introduction

The AT89C5131 clock controller is based on an on-chip oscillator feeding an on-chip Phase Lock Loop (PLL). All the internal clocks to the peripherals and CPU core are generated by this controller.

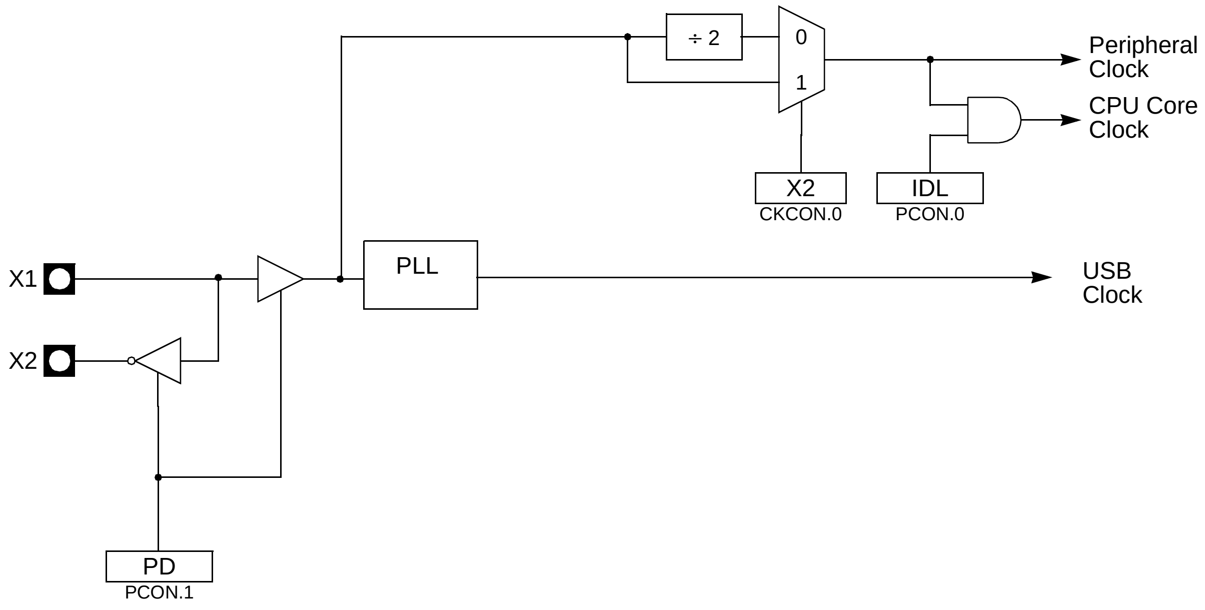
The AT89C5131 X1 and X2 pins are the input and the output of a single-stage on-chip inverter (see Figure 5) that can be configured with off-chip components as a Pierce oscillator (see Figure 6). Value of capacitors and crystal characteristics are detailed in the section "DC Characteristics".

The clock controller outputs three different clocks as shown in Figure 5:

- a clock for the CPU core
- a clock for the peripherals which is used to generate the Timers, PCA, WD, and Port sampling clocks
- a clock for the USB controller

These clocks are enabled or disabled depending on the power reduction mode as detailed in Section "Power Management", page 145.

**Figure 5.** Oscillator Block Diagram



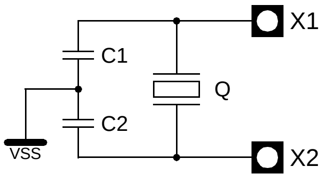
### Oscillator

Two clock sources are available for CPU:

- Crystal oscillator on X1 and X2 pins: Up to 32 MHz

In order to optimize the power consumption, the oscillator inverter is inactive when the PLL output is not selected for the USB device.

**Figure 6.** Crystal Connection



## PLL

### PLL Description

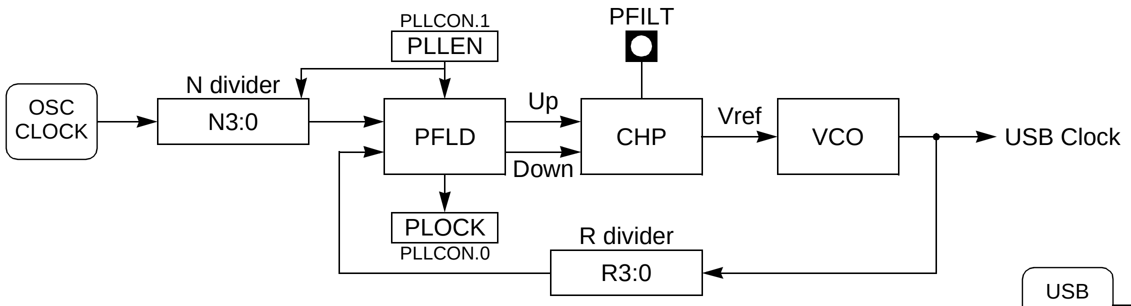
The AT89C5131 PLL is used to generate internal high frequency clock (the USB Clock) synchronized with an external low-frequency (the Peripheral Clock). The PLL clock is used to generate the USB interface clock. Figure 7 shows the internal structure of the PLL.

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 PLLEN bit in PLLCON register is used to enable the clock generation. When the PLL is locked, the bit PLOCK in PLLCON register (see Figure 7) is set.

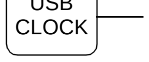
The CHP block is the Charge Pump that generates the voltage reference for the VCO by injecting or extracting charges from the external filter connected on PLLF pin (see Figure 8). Value of the filter components are detailed in the Section “DC Characteristics”.

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 7.** PLL Block Diagram and Symbol

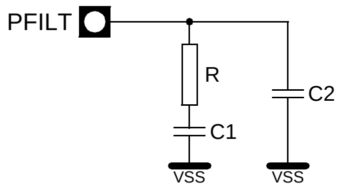


$$USBclk = \frac{OSC\text{clk} \times (\text{R} + 1)}{\text{N} + 1}$$



USB Clock Symbol

**Figure 8.** PLL Filter Connection

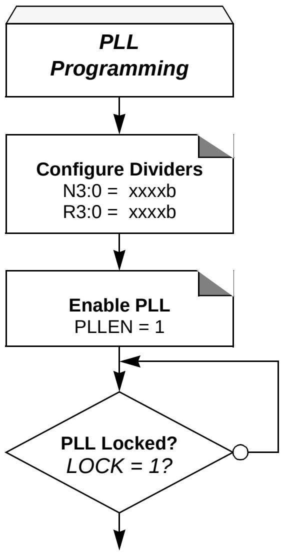


The typical values are: R = 100 Ω, C1 = 10 nf, C2 = 2.2 nF.

## PLL Programming

The PLL is programmed using the flow shown in Figure 9. As soon as clock generation is enabled user must wait until the lock indicator is set to ensure the clock output is stable.

**Figure 9.** PLL Programming Flow



## Divider Values

To generate a 48 MHz clock using the PLL, the divider values have to be configured following the oscillator frequency. The typical divider values are shown in Table 25.

**Table 25.** Typical Divider Values

| Oscillator Frequency | R+1 | N+1 | PLLDIV |
|---|---|---|---|
| 3 MHz | 16 | 1 | F0h |
| 6 MHz | 8 | 1 | 70h |
| 8 MHz | 6 | 1 | 50h |
| 12 MHz | 4 | 1 | 30h |
| 16 MHz | 3 | 1 | 20h |
| 18 MHz | 8 | 3 | 72h |
| 20 MHz | 12 | 5 | B4h |
| 24 MHz | 2 | 1 | 10h |
| 32 MHz | 3 | 2 | 21h |
| 40 MHz | 12 | 10 | B9h |

## Registers

**Table 26.** CKCON0 (S:8Fh)
Clock Control Register 0

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | WDX2 | PCAX2 | SIX2 | T2X2 | T1X2 | T0X2 | X2 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is always 0. Do not set this bit. |
| 6 | WDX2 | **Watchdog Clock**<br>**This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect.**<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 5 | PCAX2 | **Programmable Counter Array Clock**<br>**This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect.**<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 4 | SIX2 | **Enhanced UART Clock (Mode 0 and 2)**<br>**This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect.**<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 3 | T2X2 | **Timer2 Clock**<br>**This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect.**<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 2 | T1X2 | **Timer1 Clock**<br>**This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect.**<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 1 | T0X2 | **Timer0 Clock**<br>**This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect.**<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |
| 0 | X2 | **System Clock Control bit**<br>Clear to select 12 clock periods per machine cycle (STD mode, $F_{CPU} = F_{PER} = F_{OSC}/2$).<br>Set to select 6 clock periods per machine cycle (X2 mode, $F_{CPU} = F_{PER} = F_{OSC}$). |

Reset Value = 0000 0000b

**Table 27.** CKCON1 (S:AFh)
Clock Control Register 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | SPIX2 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-1 | - | **Reserved**<br>The value read from this bit is always 0. Do not set this bit. |
| 0 | SPIX2 | **SPI Clock**<br>**This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect.**<br>Clear to select 6 clock periods per peripheral clock cycle.<br>Set to select 12 clock periods per peripheral clock cycle. |

Reset Value = 0000 0000b

**Table 28.** PLLCON (S:A3h)
PLL Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | PLLEN | PLOCK |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-3 | - | **Reserved**<br>The value read from this bit is always 0. Do not set this bit. |
| 2 | - | **Reserved**<br>The value read from this bit is always 0. Do not set this bit. |
| 1 | PLLEN | **PLL Enable Bit**<br>Set to enable the PLL.<br>Clear to disable the PLL. |
| 0 | PLOCK | **PLL Lock Indicator**<br>Set by hardware when PLL is locked.<br>Clear by hardware when PLL is unlocked. |

Reset Value = 0000 0000b

**Table 29.** PLLDIV (S:A4h)
PLL Divider Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| R3 | R2 | R1 | R0 | N3 | N2 | N1 | N0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-4 | R3:0 | **PLL R Divider Bits** |
| 3-0 | N3:0 | **PLL N Divider Bits** |

Reset Value = 0000 0000

## Dual Data Pointer Register

The additional data pointer can be used to speed up code execution and reduce code size.

The dual DPTR structure is a way by which the chip will specify the address of an external data memory location. There are two 16-bit DPTR registers that address the external memory, and a single bit called DPS = AUXR1.0 (see Table 30) that allows the program code to switch between them (see Figure 10).

**Figure 10.** Use of Dual Pointer

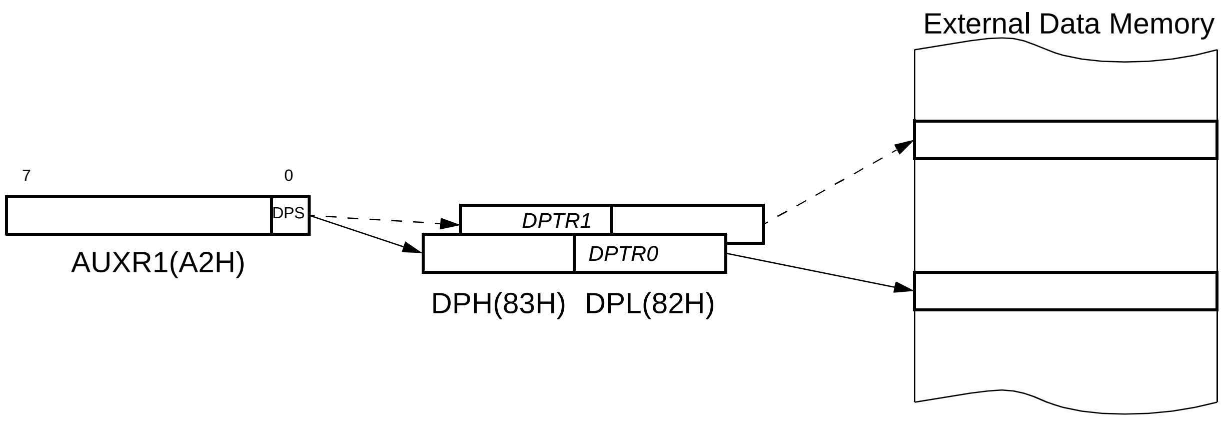


**Table 30.** AUXR1 Register
AUXR1- Auxiliary Register 1(0A2h)

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | ENBOOT | **Enable Boot Flash**<br>Cleared to disable boot ROM.<br>Set to map the boot ROM between F800h - 0FFFFh. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | GF3 | This bit is a general-purpose user flag. |
| 2 | 0 | Always cleared. |
| 1 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 0 | DPS | **Data Pointer Selection**<br>Cleared to select DPTR0.<br>Set to select DPTR1. |

Reset Value = XXXX XX0X0b
Not bit addressable

a. Bit 2 stuck at 0; this allows to use INC AUXR1 to toggle DPS without changing GF3.

ASSEMBLY LANGUAGE

```
; Block move using dual data pointers
; Modifies DPTR0, DPTR1, A and PSW
; note: DPS exits opposite of entry state
; unless an extra INC AUXR1 is added
;
00A2     AUXR1 EQU 0A2H
;
0000 909000MOV DPTR,#SOURCE ; address of SOURCE
0003 05A2 INC AUXR1 ; switch data pointers
0005 90A000 MOV DPTR,#DEST ; address of DEST
0008    LOOP:
0008 05A2 INC AUXR1 ; switch data pointers
000A E0 MOVX A,@DPTR ; get a byte from SOURCE
000B A3 INC DPTR ; increment SOURCE address
000C 05A2 INC AUXR1 ; switch data pointers
000E F0 MOVX @DPTR,A ; write the byte to DEST
000F A3 INC DPTR ; increment DEST address
0010 70F6JNZ LOOP ; check for 0 terminator
0012 05A2 INC AUXR1 ; (optional) restore DPS
```

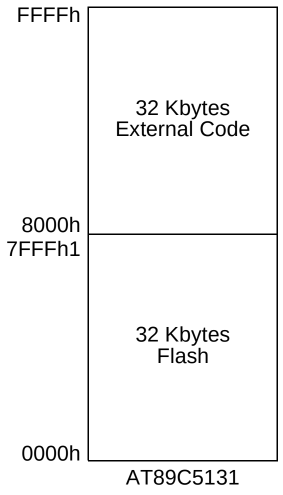
INC is a short (2 bytes) and fast (12 clocks) way to manipulate the DPS bit in the AUXR1 SFR. However, note that the INC instruction does not directly force the DPS bit to a particular state, but simply toggles it. In simple routines, such as the block move example, only the fact that DPS is toggled in the proper sequence matters, not its actual value. In other words, the block move routine works the same whether DPS is '0' or '1' on entry. Observe that without the last instruction (INC AUXR1), the routine will exit with DPS in the opposite state.

## Program/Code Memory

The AT89C5131 implement 32 Kbytes of on-chip program/code memory. Figure 11 shows the split of internal and external program/code memory spaces depending on the product.

The Flash memory increases EPROM and ROM functionality by in-circuit electrical erasure and programming. Thanks to the internal charge pump, the high voltage needed for programming or erasing Flash cells is generated on-chip using the standard $V_{DD}$ voltage. Thus, the Flash Memory can be programmed using only one voltage and allows In-application Software Programming commonly known as IAP. Hardware programming mode is also available using specific programming tool.

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



Note: If the program executes exclusively from on-chip code memory (not from external memory), beware of executing code from the upper byte of on-chip memory (7FFFh) and thereby disrupting I/O Ports 0 and 2 due to external prefetch. Fetching code constant from this location does not affect Ports 0 and 2.

## External Code Memory Access

### Memory Interface

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

Figure 12 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 31 describes the external memory interface signals.

**Figure 12.** External Code Memory Interface Structure

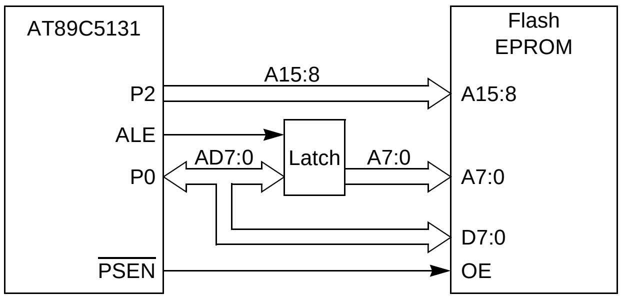


**Table 31.** External Data Memory Interface Signals

| Signal Name | Type | Description | Alternate Function |
|---|---|---|---|
| A15:8 | O | **Address Lines**<br>Upper address lines for the external bus. | P2.7:0 |
| AD7:0 | I/O | **Address/Data Lines**<br>Multiplexed lower address lines and data for the external memory. | P0.7:0 |
| ALE | O | **Address Latch Enable**<br>ALE signals indicates that valid address information are available on lines AD7:0. | - |
| $\overline{\text{PSEN}}$ | O | **Program Store Enable Output**<br>This signal is active low during external code fetch or external code read (MOVC instruction). | - |

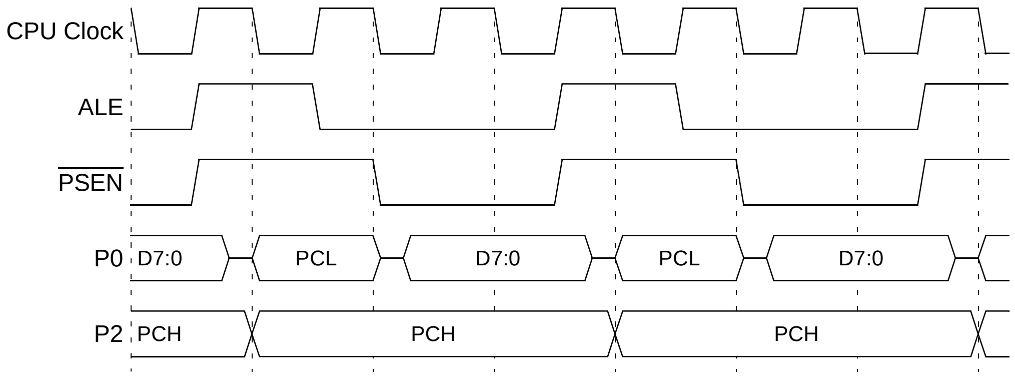
## External Bus Cycles

This section describes the bus cycles the AT89C5131 executes to fetch code (see Figure 13) in the external program/code 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 (see the clock Section).

For simplicity, the accompanying figure depicts the bus cycle waveforms in idealized form and do not provide precise timing information.

**Figure 13.** External Code Fetch Waveforms



## Flash Memory Architecture

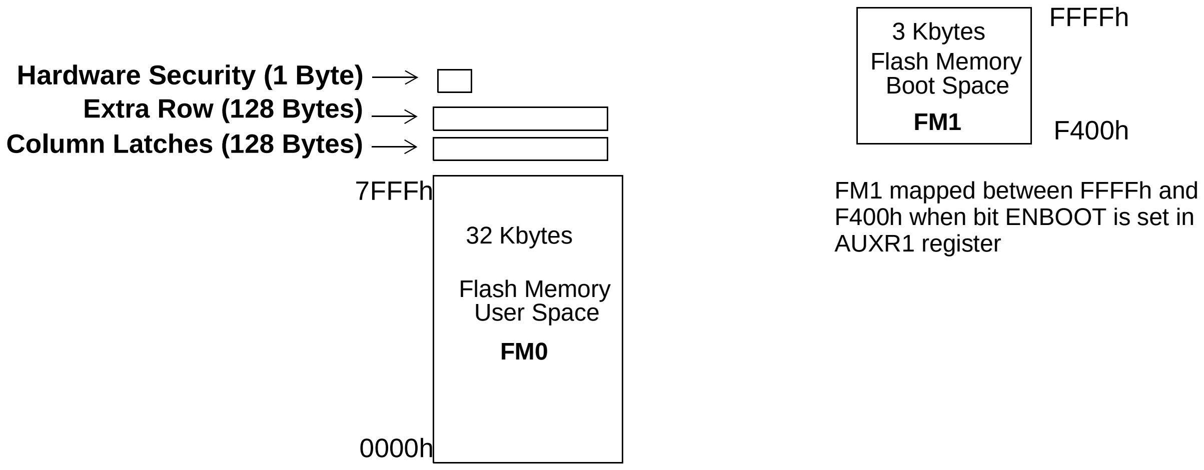
AT89C5131 features two on-chip Flash memories:

- Flash memory FM0:
  containing 32 Kbytes of program memory (user space) organized into 128-byte pages,
- Flash memory FM1:
  3 Kbytes for bootloader and Application Programming Interfaces (API).

The FM0 supports both parallel programming and Serial In-System Programming (ISP) whereas FM1 supports only parallel programming by programmers. The ISP mode is detailed in the "In-System Programming" section.

All Read/Write access operations on Flash memory by user application are managed by a set of API described in the "In-System Programming" section.

**Figure 14.** Flash Memory Architecture


Hardware Security (1 Byte)
Extra Row (128 Bytes)
Column Latches (128 Bytes)
7FFFh
32 Kbytes
Flash Memory User Space
FM0
0000h
3 Kbytes
Flash Memory Boot Space
FM1
FFFFh
F400h
FM1 mapped between FFFFh and F400h when bit ENBOOT is set in AUXR1 register

**FM0 Memory Architecture**

The Flash memory is made up of 4 blocks (see Figure 14):

1. The memory array (user space) 32 Kbytes
2. The Extra Row
3. The Hardware security bits
4. The column latch registers

*User Space*

This space is composed of a 32 Kbytes Flash memory organized in 256 pages of 128 bytes. It contains the user's application code.

*Extra Row (XRow)*

This row is a part of FM0 and has a size of 128 bytes. The extra row may contain information for bootloader usage.

*Hardware Security Space*

The hardware security space is a part of FM0 and has a size of 1 byte.
The 4 MSB can be read/written by software. The 4 LSB can only be read by software and written by hardware in parallel mode.

*Column Latches*

The column latches, also part of FM0, have a size of full page (128 bytes).
The column latches are the entrance buffers of the three previous memory locations (user array, XRow and Hardware security byte).

## Overview of FM0 Operations

The CPU interfaces to the Flash memory through the FCON register and AUXR1 register.

These registers are used to:

- Map the memory spaces in the adressable space
- Launch the programming of the memory spaces
- Get the status of the Flash memory (busy/not busy)
- Select the Flash memory FM0/FM1.

**Mapping of the Memory Space**

By default, the user space is accessed by MOVC instruction for read only. The column latches space is made accessible by setting the FPS bit in FCON register. Writing is possible from 0000h to 7FFFh, address bits 6 to 0 are used to select an address within a page while bits 14 to 7 are used to select the programming address of the page.

Setting this bit takes precedence on the EXTRAM bit in AUXR register.

The other memory spaces (user, extra row, hardware security) are made accessible in the code segment by programming bits FMOD0 and FMOD1 in FCON register in accordance with Table 32. A MOVC instruction is then used for reading these spaces.

**Table 32.** FM0 Blocks Select Bits

| FMOD1 | FMOD0 | FM0 Adressable Space |
|---|---|---|
| 0 | 0 | User (0000h-FFFFh) |
| 0 | 1 | Extra Row(FF80h-FFFFh) |
| 1 | 0 | Hardware Security (0000h) |
| 1 | 1 | reserved |

## Launching Programming

FPL3:0 bits in FCON register are used to secure the launch of programming. A specific sequence must be written in these bits to unlock the write protection and to launch the programming. This sequence is 5 followed by A. Table 33 summarizes the memory spaces to program according to FMOD1:0 bits.

**Table 33.** Programming Spaces

| | Write to FCON | | | | |
|---|---|---|---|---|---|
| | FPL3:0 | FPS | FMOD1 | FMOD0 | Operation |
| User | 5 | X | 0 | 0 | No action |
| | A | X | 0 | 0 | Write the column latches in user space |
| Extra Row | 5 | X | 0 | 1 | No action |
| | A | X | 0 | 1 | Write the column latches in extra row space |
| Security Space | 5 | X | 1 | 0 | No action |
| | A | X | 1 | 0 | Write the fuse bits space |
| Reserved | 5 | X | 1 | 1 | No action |
| | A | X | 1 | 1 | No action |

The Flash memory enters a busy state as soon as programming is launched. In this state, the memory is not available for fetching code. Thus to avoid any erratic execution during programming, the CPU enters Idle mode. Exit is automatically performed at the end of programming.

Note: Interrupts that may occur during programming time must be disabled to avoid any spurious exit of the idle mode.

## Status of the Flash Memory

The bit FBUSY in FCON register is used to indicate the status of programming.

FBUSY is set when programming is in progress.

## Selecting FM0/FM1

The bit ENBOOT in AUXR1 register is used to choose between FM0 and FM1 mapped up to F800h.

## Loading the Column Latches

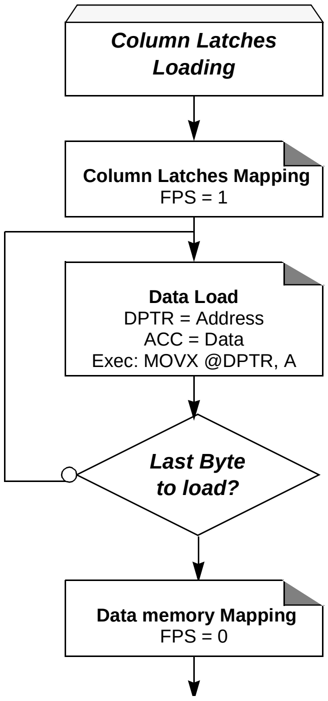
Any number of data from 1 byte to 128 bytes can be loaded in the column latches. This provides the capability to program the whole memory by byte, by page or by any number of bytes in a page.

When programming is launched, an automatic erase of the locations loaded in the column latches is first performed, then programming is effectively done. Thus, no page or block erase is needed and only the loaded data are programmed in the corresponding page.

The following procedure is used to load the column latches and is summarized in Figure 15:

- Map the column latch space by setting FPS bit.
- Load the DPTR with the address to load.
- Load Accumulator register with the data to load.
- Execute the MOVX @DPTR, A instruction.
- If needed loop the three last instructions until the page is completely loaded.

**Figure 15.** Column Latches Loading Procedure



## Programming the Flash Spaces

### *User*

The following procedure is used to program the User space and is summarized in Figure 16:

- Load data in the column latches from address 0000h to 7FFFh[1].
- Disable the interrupts.
- Launch the programming by writing the data sequence 50h followed by A0h in FCON register.
  The end of the programming indicated by the FBUSY flag cleared.
- Enable the interrupts.

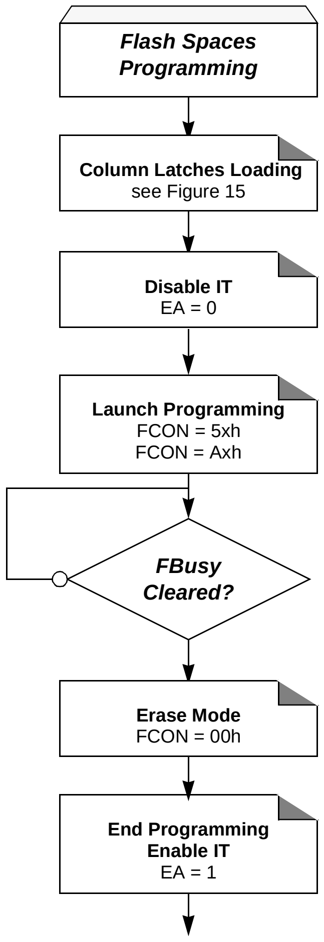
Note: 1. The last page address used when loading the column latch is the one used to select the page programming address.

*Extra Row*

The following procedure is used to program the Extra Row space and is summarized in Figure 16:

- Load data in the column latches from address FF80h to FFFFh.
- Disable the interrupts.
- Launch the programming by writing the data sequence 52h followed by A2h in FCON register.
  The end of the programming indicated by the FBUSY flag cleared.
- Enable the interrupts.

**Figure 16.** Flash and Extra Row Programming Procedure

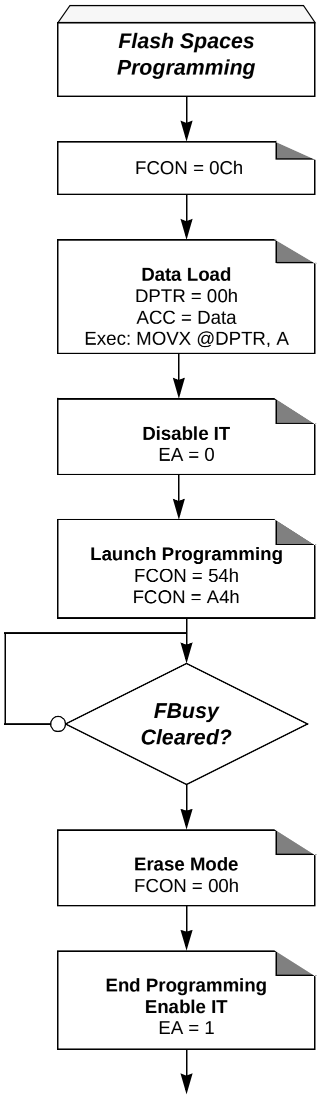


*Hardware Security*

The following procedure is used to program the Hardware Security space and is summarized in Figure 17:

- Set FPS and map Hardware byte (FCON = 0x0C)
- Disable the interrupts.
- Load DPTR at address 0000h.
- Load Accumulator register with the data to load.
- Execute the MOVX @DPTR, A instruction.
- Launch the programming by writing the data sequence 54h followed by A4h in FCON register.
  The end of the programming indicated by the FBusy flag cleared.
- Enable the interrupts.

**Figure 17.** Hardware Programming Procedure



## Reading the Flash Spaces

*User*

The following procedure is used to read the User space and is summarized in Figure 18:

- Map the User space by writing 00h in FCON register.
- Read one byte in Accumulator by executing MOVC A, @A+DPTR with A = 0 & DPTR = 0000h to FFFFh.

*Extra Row*

The following procedure is used to read the Extra Row space and is summarized in Figure 18:

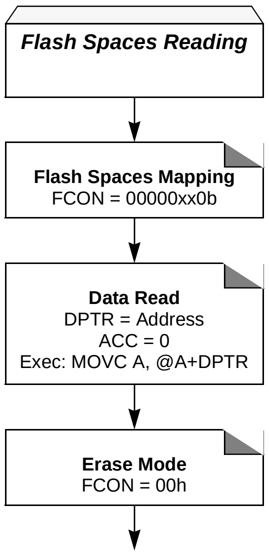
- Map the Extra Row space by writing 02h in FCON register.
- Read one byte in Accumulator by executing MOVC A, @A+DPTR with A = 0 & DPTR = FF80h to FFFFh.

*Hardware Security*

The following procedure is used to read the Hardware Security space and is summarized in Figure 18:

- Map the Hardware Security space by writing 04h in FCON register.
- Read the byte in Accumulator by executing MOVC A, @A+DPTR with A = 0 & DPTR = 0000h.

**Figure 18.** Reading Procedure



## Registers

**Table 34.** FCON (S:D1h)
Flash Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **FPL3** | **FPL2** | **FPL1** | **FPL0** | **FPS** | **FMOD1** | **FMOD0** | **FBUSY** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-4 | FPL3:0 | **Programming Launch Command Bits**<br>Write 5Xh followed by AXh to launch the programming according to FMOD1:0. (see Table 33.) |
| 3 | FPS | **Flash Map Program Space**<br>Set to map the column latch space in the data memory space.<br>Clear to re-map the data memory space. |
| 2-1 | FMOD1:0 | **Flash Mode**<br>See Table 32 or Table 33. |
| 0 | FBUSY | **Flash Busy**<br>Set by hardware when programming is in progress.<br>Clear by hardware when programming is done.<br>Can not be cleared by software. |

Reset Value = 0000 0000b

## Flash EEPROM Memory

### General Description

The Flash memory increases EPROM functionality with in-circuit electrical erasure and programming. It contains 32 Kbytes of program memory organized in 256 pages of 128 bytes, respectively. This memory is both parallel and serial In-System Programmable (ISP). ISP allows devices to alter their own program memory in the actual end product under software control. A default serial loader (bootloader) program allows ISP of the Flash.

The programming does not require 12V external programming voltage. The necessary high programming voltage is generated on-chip using the standard $V_{CC}$ pins of the microcontroller.

### Features

- Flash EEPROM internal program memory.
- Boot vector allows user-provided Flash loader code to reside anywhere in the Flash memory space. This configuration provides flexibility to the user.
- Default loader in Boot EEPROM allows programming via the serial port without the need of a user provided loader.
- Up to 64K bytes external program memory if the internal program memory is disabled (EA = 0).
- Programming and erase voltage with standard 5V or 3.3V $V_{CC}$ supply.
- Read/Program/Erase:
- Byte-wise read (without wait state).
- Byte or page erase and programming (10 ms).
- Typical programming time (32 Kbytes) in 10 sec.
- Parallel programming with 87C51 compatible hardware interface to programmer.
- Programmable security for the code in the Flash.
- 100K write cycles
- 10 years data retention

### Flash Programming and Erasure

The 32 Kbytes Flash is programmed by bytes or by pages of 128 bytes. It is not necessary to erase a byte or a page before programming. The programming of a byte or a page includes a self erase before programming.

There are three methods of programming the Flash memory:

1. The on-chip ISP bootloader may be invoked which will use low level routines to program the pages. The interface used for serial downloading of Flash is the UART.
2. The Flash may be programmed or erased in the end-user application by calling low-level routines through a common entry point in the Boot ROM.
3. The Flash may be programmed using the parallel method by using a conventional EPROM programmer. The parallel programming method used by these devices is similar to that used by EPROM 87C51 but it is not identical and the commercially available programmers need to have support for the AT89C5131.

The bootloader and the Application Programming Interface (API) routines are located in the Boot ROM.

## Flash Registers and Memory Map

The AT89C5131 Flash memory uses several registers:

- Hardware registers can only be accessed through the parallel programming modes which are handled by the parallel programmer.
- Software registers are in a special page of the Flash memory which can be accessed through the API or with the parallel programming modes. This page, called "Extra Flash Memory", is not in the internal Flash program memory addressing space.

### Hardware Registers

The only hardware registers of the AT89C5131 is called Hardware Security Byte (HSB).

**Table 35.** Hardware Security Byte (HSB)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| X2 | BLJB | OSCON1 | OSCON0 | - | LB2 | LB1 | LB0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | X2 | **X2 Mode**<br>Cleared to force X2 mode (6 clocks per instruction)<br>Set to force X1 mode, Standard Mode (Default). |
| 6 | BLJB | **Bootloader Jump Bit**<br>Set this bit to start the user's application on next reset at address 0000h.<br>Cleared this bit to start the bootloader at address F400h (default). |
| 5-4 | OSCON1-0 | **Oscillator Control Bits**<br>These two bits are used to control the oscillator in order to reduce consummation.<br>OSCON / OSCON0 / Description<br>1 / 1 / The oscillator is configured to run from 0 to 32 MHz<br>1 / 0 / The oscillator is configured to run from 0 to 16 MHz<br>0 / 1 / The oscillator is configured to run from 0 to 8 MHz<br>0 / 0 / This configuration shouldn't be set |
| 3 | - | **Reserved** |
| 2-0 | LB2-0 | **User Memory Lock Bits**<br>See Table 36 |

*Bootloader Jump Bit (BLJB)*

One bit of the HSB, the BLJB bit, is used to force the boot address:

- When this bit is set the boot address is 0000h.
- When this bit is reset the boot address is F400h. By default, this bit is cleared and the ISP is enabled.

*Flash Memory Lock Bits*

The three lock bits provide different levels of protection for the on-chip code and data, when programmed as shown in Table 36.

**Table 36.** Program Lock bits

| Program Lock Bits | | | | |
|---|---|---|---|---|
| **Security level** | **LB0** | **LB1** | **LB2** | **Protection Description** |
| 1 | U | U | U | No program lock features enabled. |
| 2 | P | U | U | MOVC instruction executed from external program memory is disabled from fetching code bytes from any internal memory, $\overline{EA}$ is sampled and latched on reset, and further parallel programming of the Flash and of the EEPROM (boot and Xdata) is disabled. ISP and software programming with API are still allowed. |
| 3 | X | P | U | Same as 2, also verify through parallel programming interface is disabled and serial programming ISP is disabled. |
| 4 | X | X | P | Same as 3, also external execution is disabled. |

Notes: 1. U: unprogrammed or “one” level.
2. P: programmed or “zero” level.
3. X: don’t care
4. WARNING: Security level 2 and 3 should only be programmed after Flash and code verification.

These security bits protect the code access through the parallel programming interface. They are set by default to level 4. The code access through the ISP is still possible and is controlled by the “software security bits” which are stored in the extra Flash memory accessed by the ISP firmware.

To load a new application with the parallel programmer, a chip erase must be done first. This will set the HSB in its inactive state and will erase the Flash memory. The part reference can always be read using Flash parallel programming modes.

### *Default Values*

The default value of the HSB provides parts ready to be programmed with ISP:

- BLJB: Cleared to force ISP operation.
- X2: Set to force X1 mode (Standard Mode)
- OSCON1-0: Set to start with 32 MHz oscillator configuration value.
- XRAM: Unprogrammed to valid XRAM
- LB2-0: Security level four to protect the code from a parallel access with maximum security.

## Software Registers

Several registers are used, in factory and by parallel programmers, to make copies of hardware registers contents. These values are used by Atmel ISP (see Section “In-System Programming (ISP)”).

These registers are in the “Extra Flash Memory” part of the Flash memory. This block is also called ”XAF” or eXtra Array Flash. They are accessed in the following ways:

- Commands issued by the parallel memory programmer.
- Commands issued by the ISP software.
- Calls of API issued by the application software.

Several software registers are described in Table 37.

**Table 37.** Software Registers

| Mnemonic | Description | Default value | |
|---|---|---|---|
| SBV | Software Boot Vector | FCh | – |
| HSB | Copy of the Hardware Security Byte | 1011 1000b | – |
| BSB | Boot Status Byte | 0FFh | – |
| SSB | Software Security Byte | FFh | – |
| – | Copy of the Manufacturer Code | 58h | Atmel |
| – | Copy of the Device ID #1: Family Code | D7h | C51 X2, Electrically Erasable |
| – | Copy of the Device ID #2: Memories | F7h | AT89C5131 32 Kbyte |
| – | Size and Type | FBh | AT89C5131 16 Kbyte |
| – | Copy of the Device ID #3: Name | EFh | AT89C5131 32 Kbyte, revision 0 |
| – | Revision | FFh | AT89C5131 16 Kbyte, revision 0 |

After programming the part by ISP, the BSB must be cleared (00h) in order to allow the application to boot at 0000h.

The content of the Software Security Byte (SSB) is described in Table 38 and Table 39.

To assure code protection from a parallel access, the HSB must also be at the required level.

**Table 38.** Software Security Byte (SSB)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | LB1 | LB0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | Reserved<br>Do not clear this bit. |
| 6 | - | Reserved<br>Do not clear this bit. |
| 5 | - | Reserved<br>Do not clear this bit. |
| 4 | - | Reserved<br>Do not clear this bit. |
| 3 | - | Reserved<br>Do not clear this bit. |
| 2 | - | Reserved<br>Do not clear this bit. |
| 1-0 | LB1-0 | User Memory Lock Bits<br>See Table 39 |

The two lock bits provide different levels of protection for the on-chip code and data, when programmed as shown to Table 39.

**Table 39.** Program Lock Bits of the SSB

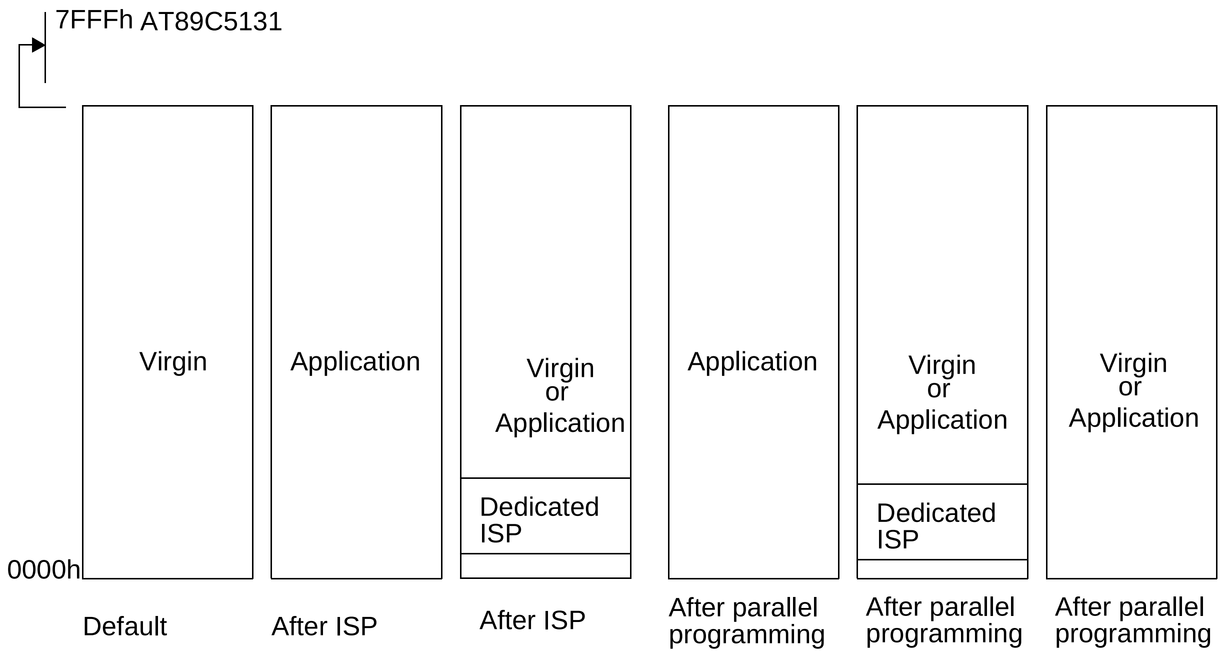
| Program Lock Bits | | | |
|---|---|---|---|
| **Security Level** | **LB0** | **LB1** | **Protection Description** |
| 1 | U | U | No program lock features enabled. |
| 2 | P | U | ISP programming of the Flash is disabled. |
| 3 | X | P | Same as 2, also verify through ISP programming interface is disabled. |

Notes: 1. U: unprogrammed or "one" level.
2. P: programmed or “zero” level.
3. X: don’t care
4. WARNING: Security level 2 and 3 should only be programmed after Flash and code verification.

## Flash Memory Status

AT89C5131 parts are delivered with the ISP boot in the Flash memory. After ISP or parallel programming, the possible contents of the Flash memory are summarized in Figure 19:

**Figure 19.** Flash Memory Possible Contents



## Memory Organization

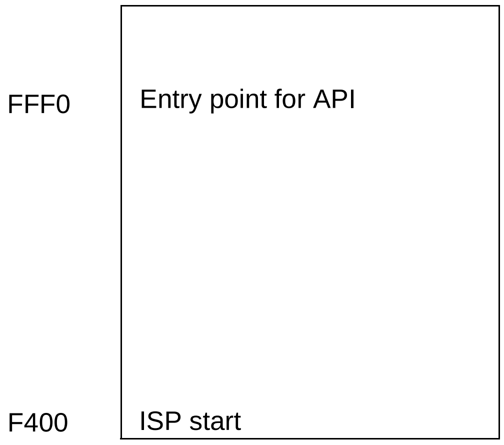
In the AT89C5131, the lowest 16K or 32K of the 64 Kbyte program memory address space is filled by internal Flash.
When the $\overline{EA}$ is pin high, the processor fetches instructions from internal program Flash. Bus expansion for accessing program memory from 16K or 32K upward is automatic since external instruction fetches occur automatically when the program counter exceeds 3FFFh (16K) or 7FFFh (32K). If the $\overline{EA}$ pin is tied low, all program memory fetches are from external memory. If all storage is on chip, then byte location 3FFFh (16K) or 7FFFh (32K) should be left vacant to prevent and undesired pre-fetch from external program memory address 4000h (16K) or 8000h (32K).

## Boot Process

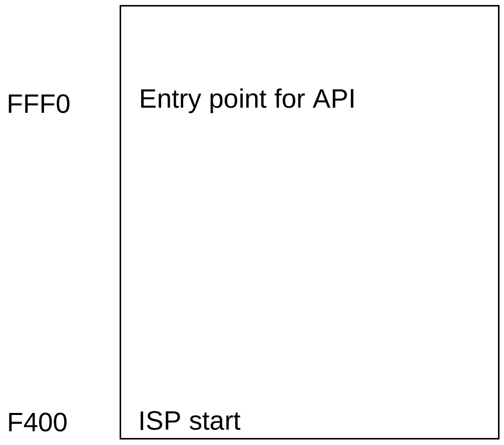
### Boot Flash

When the user application programs its own Flash memory, all of the low level details are handled by a code that is permanently contained in a 3 Kbyte "Boot ROM". A user program simply calls the common entry point in the Boot ROM with appropriate parameters to accomplish the desired operation. Boot ROM operations include: erase block, program byte or page, verify byte or page, program security lock bit, etc. The Boot ROM is placed in the program memory space at the top of the address space from F800h to FFFFh (Figure 20).

**Figure 20.** Boot ROM Loader Memory Map

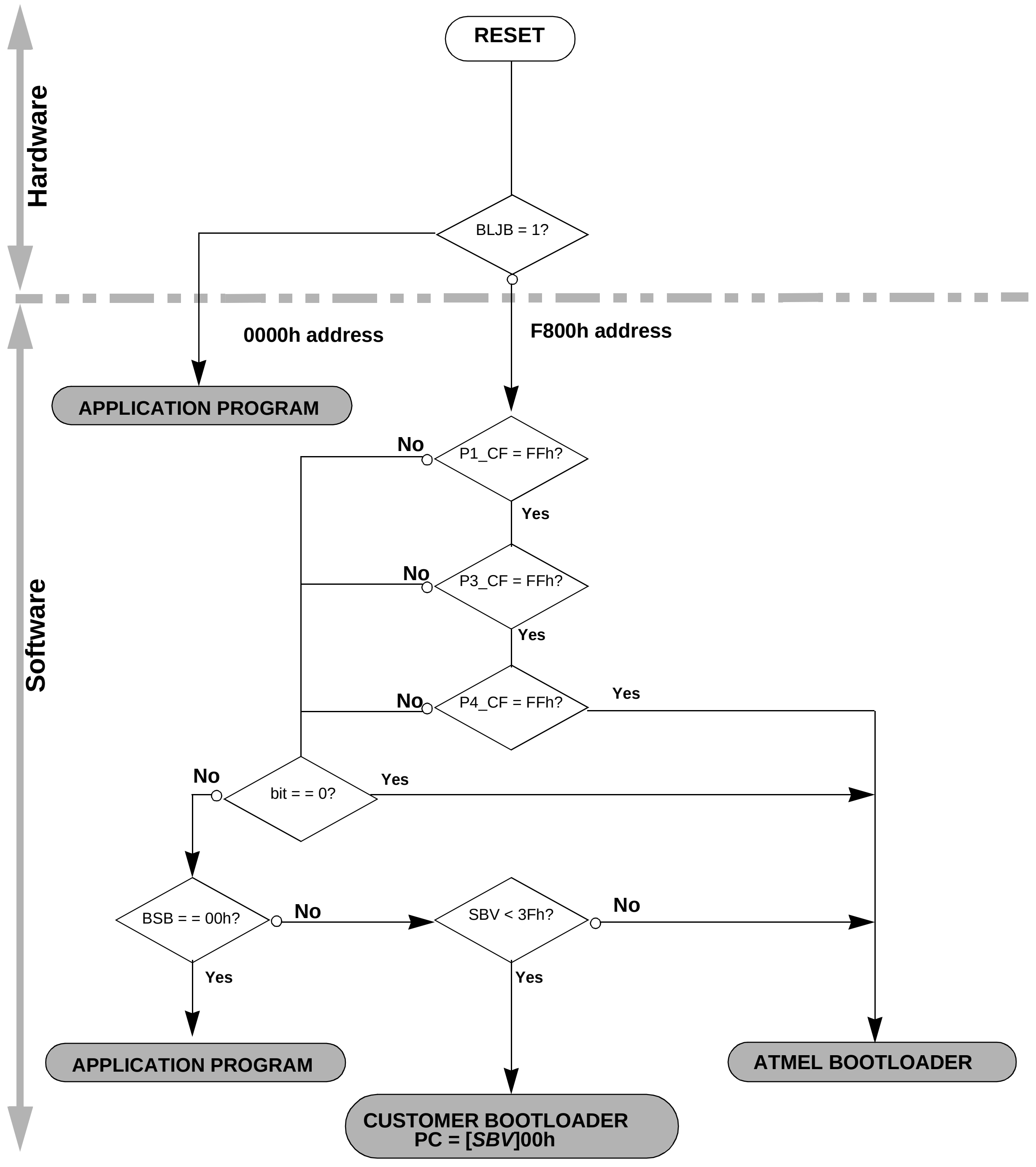


### Boot Process Secondary



The boot process is summarized in Figure 21.

**Figure 21.** Boot Process Flowchart



## In-System Programming (ISP)

The In-System Programming (ISP) is performed without removing the microcontroller from the system. The ISP facility consists of a series of internal hardware resources coupled with internal firmware to facilitate remote programming of the AT89C5131 through the serial port.

The Atmel ISP feature has made in-circuit programming in an embedded application possible with a minimum of additional expenses in components and circuit board area.

The ISP function through UART uses four pins: TxD, RxD, $V_{SS}$, $V_{CC}$. Only a small connector needs to be available to interface the application to an external circuit in order to use this feature.

### Using In-System Programming

The ISP feature allows a wide range of baud rates in the user application. It is also adaptable to a wide range of oscillator frequencies. This is accomplished by measuring the bit-time of a single bit in a received character. This information is then used to program the baud rate in terms of timer counts based on the oscillator frequency. The ISP feature requires that an initial character (an uppercase U) be sent to the AT89C5131 to establish the baud rate. The ISP firmware provides auto-echo of received characters.

Once baud rate initialization has been performed, the ISP firmware will only accept Intel Hex-type records. Intel Hex records consist of ASCII characters used to represent hexadecimal values and are summarized below:

**:NNAAAARRDD..DDCC<crlf>**

AT89C5131 will accept up to 16 (10h) data bytes. The “AAAA” string represents the address of the first byte in the record. If there are zero bytes in the record, this field is often set to ‘‘0000’’. The “RR” string indicates the record type. A record type of “00” is a data record. A record type of “01” indicates the end-of-file mark. In this application, additional record types will be added to indicate either commands or data for the ISP feature. The “DD” string represents the data bytes. The maximum number of data bytes in a record is limited to 16 (decimal). The “CC” string represents the checksum byte. ISP commands are summarized in Table 40.

As a record is received by the AT89C5131, the information in the record is stored internally and a checksum calculation is performed and compared to ‘‘CC’’.

The operation indicated by the record type is not performed until the entire record has been received. Should an error occur in the checksum, the AT89C5131 will send an “X” out the serial port indicating a checksum error. If the checksum calculation is found to match the checksum in the record, then the command will be executed. In most cases, successful reception of the record will be indicated by transmitting a “.” character out the serial port (displaying the contents of the internal program memory is an exception). In the case of a Data Record (record type ‘‘00’’), an additional check is made. A “.” character will NOT be sent unless the record checksum matched the calculated checksum and all of the bytes in the record were successfully programmed. For a data record, an “X” indicates that the checksum failed to match, and an “R” character indicates that one of the bytes did not properly program.

Atmel_ISP, a software utility to implement ISP programming with a PC, is available from Atmel. Please visit our web site www.atmel.com.

**Table 40.** Intel-Hex Records Used by In-System Programming

| Record Type | Command/Data Function |
|---|---|
| 00 | Data Record<br>:nnaaaa00dd....ddcc<br>Where:<br>Nn = number of bytes (hex) in record<br>aaaa = memory address of first byte in record<br>dd....dd = data bytes<br>cc = checksum<br>Example:<br>:05008000AF5F67F060B6 |
| 01 | End of File (EOF), no operation<br>:xxxxxx01cc<br>Where:<br>xxxxxx = required field, but value is a “don’t care”<br>cc = checksum<br>Example:<br>:00000001FF |
| 02 | Specify Oscillator Frequency (Not required, left for Philips compatibility)<br>:01xxxx02ddcc<br>Where:<br>xxxx = required field, but value is a “don’t care”<br>dd = required field, but value is a “don’t care”<br>cc = checksum<br>Example:<br>:0100000210ED |

**Table 40.** Intel-Hex Records Used by In-System Programming (Continued)

| Record Type | Command/Data Function |
|---|---|
| 03 | Miscellaneous Write Functions<br>:nnxxxx03ffssddcc<br>Where:<br>nn = number of bytes (hex) in record<br>xxxx = required field, but value is a "don't care"<br>03 = Write Function<br>ff = subfunction code<br>ss = selection code<br>dd = data input (as needed)<br>cc = checksum<br>Subfunction Code = 01 (Erase Block)<br>ff = 01<br>ss = block number in bits 7:5, Bits 4:0 = zeros<br>Example:<br>:0200000301A05A erase block 5<br>Subfunction Code = 04 (Reset Boot Vector and Status Byte)<br>ff = 04<br>ss = don't care<br>dd = don't care<br>Example:<br>:020000034500F8 Reset boot vector (FCh) and status byte (FFh)<br>Subfunction Code = 05 (Program Software Security Bits)<br>ff = 05<br>ss = 00 program software security bit 1 (Level 2 inhibit writing to Flash)<br>ss = 01 program software security bit 2 (Level 3 inhibit Flash verify)<br>ss = 02 program security bit 3 (No effect, left for Philips compatibility; disable external memory is already set in the default hardware security byte)<br>Example:<br>:020000030501F6 program security bit 2<br>Subfunction Code = 06 (Program Boot Status Byte, Boot Vector, X2 bit, Osc bit or BLJB fuse bit)<br>ff = 06<br>ss = 00 program Boot Status byte<br>ss = 01 program Software Boot vector<br>ss = 02 program X2 bit<br>ss = 03 program Osc bit<br>ss = 04 program BLJB<br>Example:<br>:03000003060100F5 program boot vector with 00<br>Subfunction Code = 07 (Full chip erase)<br>ff = 07<br>ss = don't care<br>dd = don't care<br>Example:<br>:03000007F5 program boot vector with 00 |

**Table 40.** Intel-Hex Records Used by In-System Programming (Continued)

| Record Type | Command/Data Function |
|---|---|
| 04 | Display Device Data or Blank Check<br>Record type 04 causes the contents of the entire Flash array to be sent out the serial port in a formatted display. This display consists of an address and the contents of 16 bytes starting with that address. No display of the device contents will occur if security bit 2 has been programmed. The dumping of the device data to the serial port is terminated by the reception of any character.<br>General Format of Function 04<br>:05xxxx04sssseeeeffcc<br>Where:<br>05 = number of bytes (hex) in record<br>xxxx = required field, but value is a “don’t care”<br>04 = “Display Device Data or Blank Check” function code<br>ssss = starting address<br>eeee = ending address<br>ff = subfunction<br>00 = display data<br>01 = blank check<br>cc = checksum<br>Example:<br>:0500000440004FFF0069 (display 4000–4FFF) |
| 05 | Miscellaneous Read Functions<br>General Format of Function 05<br>:02xxxx05ffsscc<br>Where:<br>02 = number of bytes (hex) in record<br>xxxx = required field, but value is a “don’t care”<br><br>05 = “Miscellaneous Read” function code<br>ffss = subfunction and selection code<br>0000 = read copy of the signature byte – manufacturer id (58H)<br>0001 = read copy of the signature byte – device ID# 1 (Family code)<br>0002 = read copy of the signature byte – device ID # 2 (Memories size and type)<br>0003 = read copy of the signature byte – device ID # 3 (Product name and revision)<br>0700 = read the software security bits<br>0701 = read BSB<br>0702 = read SBV<br>0704 = read HSB<br>cc = checksum<br>Example:<br>:020000050001F0 read copy of the signature byte – device id # 1 |

## In-application Programming Method

Several Application Program Interface (API) calls are available for use by an application program to permit selective erasing and programming of Flash pages. All calls are made through a common interface, PGM_MTP. The programming functions are selected by setting up the microcontroller’s registers before making a call to PGM_MTP at FFF0h. Results are returned in the registers. The API calls are shown in Table 41.

A set of Philips compatible API calls is provided.

When several bytes have to be programmed, it is highly recommended to use the Atmel API “PROGRAM DATA PAGE” call. Indeed, this API call writes up to 128 bytes in a single command.

**Table 41.** API Calls

| API Call | Parameter |
|---|---|
| PROGRAM DATA BYTE | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 02h<br>DPTR = address of byte to program<br>ACC = byte to program<br>Return Parameter<br>ACC = 00 if pass,!00 if fail |
| PROGRAM DATA PAGE | Input Parameters:<br>R0 = osc freq (integer Not required)<br>R1 = 09h<br>DPTR0 = address of the first byte to program in the Flash memory<br>DPTR1 = address in XRAM of the first data to program (second data pointer)<br>ACC = number of bytes to program<br>Return Parameter<br>ACC = 00 if pass,!00 if fail<br>Remark: number of bytes to program is limited such as the Flash write remains in a single 128bytes page. Hence, when ACC is 128, valid values of DPL are 00h, or, 80h. |
| ERASE BLOCK | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 01h<br>DPH = block number in bits 7:5, bits 4:0 ='0'<br>DPL = 00h<br>Return Parameter<br>None<br>Remark: Command for Philips compatibility, as no erase is needed; the ISP firmware write FFh in the corresponding block. |
| ERASE BOOT VECTOR | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 04h<br>DPH = 00h<br>DPL = don't care<br>Return Parameter<br>none |

**Table 41.** API Calls (Continued)

| API Call | Parameter |
|---|---|
| PROGRAM SOFTWARE SECURITY BIT | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 05h<br>DPH = 00h<br>DPL = 00h – security bit # 1 (inhibit writing to Flash)<br>01h – security bit # 2 (inhibit Flash verify)<br>10h - allows ISP writing to Flash (see Note 1)<br>11h - allows ISP Flash verify (see Note 1)<br>Return Parameter<br>none |
| PROGRAM BOOT STATUS BYTE | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 06h<br>DPH = 00h<br>DPL = 00h – program status byte<br>ACC = status byte<br>Return Parameter<br>ACC = status byte |
| PROGRAM BOOT VECTOR | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 06h<br>DPH = 00h<br>DPL = 01h – program boot vector<br>ACC = boot vector<br>Return Parameter<br>ACC = boot vector |
| PROGRAM X2 MODE | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 06h<br>DPH = 00h<br>DPL = 02h – program X2 mode at reset<br>ACC = value (00 or 01h)<br>Return Parameter<br>ACC = boot vector |

Table 41. API Calls (Continued)

| API Call | Parameter |
|---|---|
| PROGRAM OSC MODE | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 06h<br>DPH = 00h<br>DPL = 03h – program OscA/OscB at reset<br>ACC = value (00 or 01h)<br>Return Parameter<br>ACC = boot vector |
| PROGRAM BLJB | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 06h<br>DPH = 00h<br>DPL = 04h – program FSBt<br>ACC = value (00 or 01h)<br>Return Parameter<br>ACC = boot vector |
| LOCK MEMORY AREA | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 0Ch<br>DPTR0 = address of the first byte to lock in the Flash memory<br>DPTR1 =<br>Return Parameter<br>none |
| UNLOCK MEMORY AREA | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 0Ch<br>DPTR0 = address of the first byte to lock in the Flash memory<br>DPTR1 =<br>Return Parameter<br>none |
| READ DEVICE DATA | Input Parameters:<br>R1 = 03h<br>DPTR = address of byte to read<br>Return Parameter<br>ACC = value of byte read |

**Table 41.** API Calls (Continued)

| API Call | Parameter |
|---|---|
| READ copy of the MANUFACTURER ID | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 00h<br>DPH = 00h<br>DPL = 00h (manufacturer ID)<br>Return Parameter<br>ACC = value of byte read |
| READ copy of the device ID # 1 | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 00h<br>DPH = 00h<br>DPL = 01h (device ID #1)<br>Return Parameter<br>ACC = value of byte read |
| READ copy of the device ID # 2 | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 00h<br>DPH = 00h<br>DPL = 02h (device ID #2)<br>Return Parameter<br>ACC = value of byte read |
| READ copy of the device ID # 3 | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 00h<br>DPH = 00h<br>DPL = 03h (device ID #2)<br>Return Parameter<br>ACC = value of byte read |
| READ SOFTWARE SECURITY BITS | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 07h<br>DPH = 00h<br>DPL = 00h (Software security bits)<br>Return Parameter<br>ACC = value of byte read |
| READ HARDWARE SECURITY BITS | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 07h<br>DPH = 00h<br>DPL = 04h (Hardware security bits)<br>Return Parameter<br>ACC = value of byte read |

**Table 41.** API Calls (Continued)

| API Call | Parameter |
|---|---|
| READ BOOT STATUS BYTE | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 07h<br>DPH = 00h<br>DPL = 01h (status byte)<br>Return Parameter<br>ACC = value of byte read |
| READ BOOT VECTOR | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 07h<br>DPH = 00h<br>DPL = 02h (boot vector)<br>Return Parameter<br>ACC = value of byte read |
| FULL CHIP ERASE | Input Parameters:<br>R0 = osc freq (integer Not required, left for Philips compatibility)<br>R1 = 08h<br>DPH = don't care<br>DPL = don't care<br>Return Parameter<br>none |

Note: 1. These functions can only be called by user's code. The standard bootloader cannot decrease the security level.

## EEPROM Data Memory

### Description

The 1-Kbyte on-chip EEPROM memory block is located at addresses 0000h to 03FFh of the XRAM memory space and is selected by setting control bits in the EECON register.

A read in the EEPROM memory is done with a MOVX instruction.

A physical write in the EEPROM memory is done in two steps: write data in the column latches and transfer of all data latches into an EEPROM memory row (programming).

The number of data written on the page may vary from 1 to 128 bytes (the page size). When programming, only the data written in the column latch is programmed and a ninth bit is used to obtain this feature. This provides the capability to program the whole memory by bytes, by page or by a number of bytes in a page. Indeed, each ninth bit is set when the writing the corresponding byte in a row and all these ninth bits are reset after the writing of the complete EEPROM row.

### Write Data in the Column Latches

Data is written by byte to the column latches as for an external RAM memory. Out of the 11 address bits of the data pointer, the 4 MSBs are used for page selection (row) and 7 are used for byte selection. Between two EEPROM programming sessions, all the addresses in the column latches must stay on the same page, meaning that the 4 MSB must not be changed.

The following procedure is used to write to the column latches:

- Set bit EEE of EECON register
- Stretch the MOVX to accommodate the slow access time of the column latch (Set bit M0 of AUXR register)
- Load DPTR with the address to write
- Store A register with the data to be written
- Execute a MOVX @DPTR, A
- If needed, loop the three last instructions until the end of a 128 bytes page

### Programming

The EEPROM programming consists on the following actions:

- Writing one or more bytes of one page in the column latches. Normally, all bytes must belong to the same page; if not, the first page address will be latched and the others discarded.
- Launching programming by writing the control sequence (54h followed by A4h) to the EECON register.
- EEBUSY flag in EECON is then set by hardware to indicate that programming is in progress and that the EEPROM segment is not available for reading.
- The end of programming is indicated by a hardware clear of the EEBUSY flag.

### Read Data

The following procedure is used to read the data stored in the EEPROM memory:

- Set bit EEE of EECON register
- Stretch the MOVX to accommodate the slow access time of the column latch (Set bit M0 of AUXR register)
- Load DPTR with the address to read
- Execute a MOVX A, @DPTR

## Registers

**Table 42.** EECON (S:0D2h)
EECON Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **EEPL3** | **EEPL2** | **EEPL1** | **EEPL0** | **-** | **-** | **EEE** | **EEBUSY** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-4 | EEPL3-0 | **Programming Launch command bits**<br>Write 5Xh followed by AXh to EEPL to launch the programming. |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 2 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 1 | EEE | **Enable EEPROM Space bit**<br>Set to map the EEPROM space during MOVX instructions (Write in the column latches)<br>Clear to map the XRAM space during MOVX. |
| 0 | EEBUSY | **Programming Busy flag**<br>Set by hardware when programming is in progress.<br>Cleared by hardware when programming is done.<br>Cannot be set or cleared by software. |

Reset Value = XXXX XX00b
Not bit addressable

## In-System Programming (ISP)

With the implementation of the User EEPROM and the Boot EEPROM in Flash technology the AT89C5131 allows the system engineer to develop applications with a very high level of flexibility. This flexibility is based on the possibility to alter the customer programming on all stages of a product's life:

- During the final production phase, the 1st personalization of the product by parallel or serial charging of the code in the User EEPROM and if wanted also a customized Bootloader in the Boot memory (Atmel will provide also a standard Bootloader by default).
- After assembling of the product in its final stage, embedded position by serial mode via the USB bus.

This In-System Programming (ISP) allows code modification over the total lifetime of the product.

Besides the default Bootloader, Atmel will provide to the customer all the needed Application Programming Interfaces (API) which are needed for ISP. The API will be located in the Boot memory.

This will allow the customer to have a full use of the 32-Kbyte user memory.

Two blocks Flash memories are implemented (see Figure 22):

- Flash memory FM0:
  32-Kbytes of program memory organized in a page of 128 bytes,
- Flash memory FM1:
  3-Kbytes for default bootloader and Application Programming Interfaces (API).

The FM0 supports both, hardware (parallel) and software programming whereas FM1 supports only hardware programming.

The ISP functions are assumed by:

- FCON register and bit ENBOOT in AUXR1 register
- Bootloader Jump Bit (BLJB), which forces the application execution
- Software Boot Vector (SBV), which can be read and modified by using an API or the parallel programming mode. The SBV is stored in XROW
- The Extra Byte (EB) and Boot Status Byte (BSB) can be modified only by using API. EB is stored in XROW

The bit ENBOOT in AUXR1 register allows to map FM1 between address F400h and FFFFh of FM0.

The FM0 can be programmed by:

- The Atmel bootloader, located by default in FM1.
- The user bootloader located in FM0
- The user bootloader located in FM1 in place of Atmel bootloader.

API contained in FM1 can be called by the user bootloader located in FM0 at the address [SBV]00h.

The user program simply calls the common entry point with appropriate parameters in FM1 to accomplish the desired operation (all these methods will be described in Application Notes on API-description).

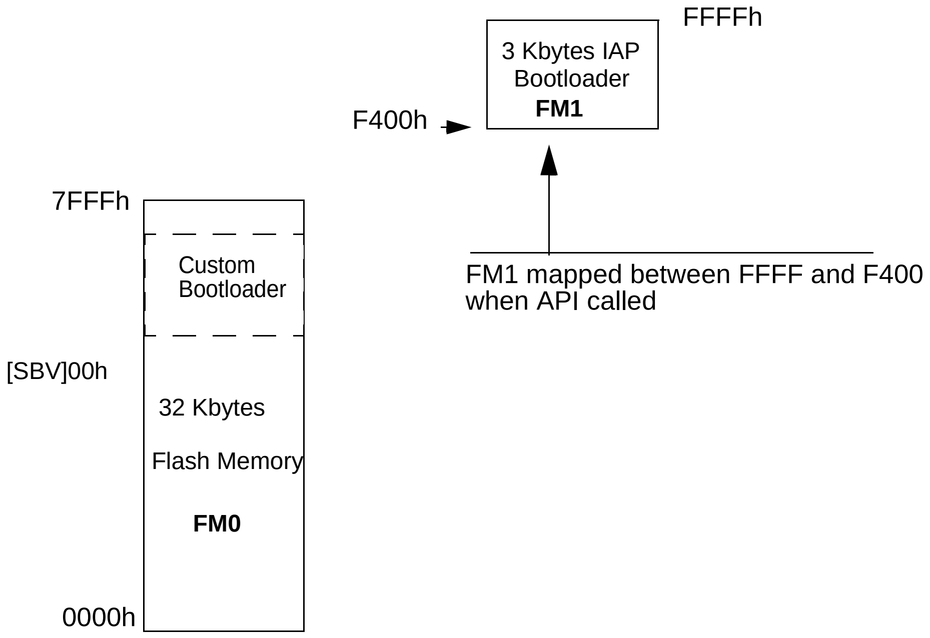
Boot Flash operations include: erase block, program byte or page, verify byte or page, program security lock bit, etc. Indeed, Atmel provides the binary code of the default Flash bootloader.

## Flash Programming and Erasure

There are three methods of programming the Flash memory:

- The Atmel bootloader located in FM1 is activated by the application. Low level API routines (located in FM1) to program FM0 will be used. The interface used for serial downloading to FM0 is the UART or the USB. API can be called also by user's bootloader located in FM0 at [SBV]00h.
- A further method exist for activating the Atmel bootloader by using hardware activation.
- The FM0 can be also programmed by the parallel mode using a programmer.

**Figure 22.** Flash Memory Mapping



### Flash Parallel Programming

The three lock bits in Hardware byte are programmed according to Table 43. They will provide different level of protection for the on-chip code and data located in FM0 and FM1.

The only way to write these bits are in parallel mode.

**Table 43.** Program Lock bit

| Program Lock Bits | | | | |
|---|---|---|---|---|
| Security level | LB0 | LB1 | LB2 | **Protection Description** |
| 1 | U | U | U | No program lock features enabled. MOVC instruction executed from external program memory returns non encrypted data. |
| 2 | P | U | U | MOVC instruction executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further parallel programming of the Flash is disabled. |
| 3 | U | P | U | Same as 2, also verify that parallel programming interface is disabled. |
| 4 | U | U | P | Same as 3, also external execution is disabled. |

Program Lock bits
U: unprogrammed
P: programmed

WARNING: Security level 2 and 3 should only be programmed after Flash and Core verification.

*Program Lock Bits*

These security bits protect the code access through the parallel programming interface. They are set by default to level 4.

## Low Pin-count Boot Process

The bootloader can be activated by two means: Regular boot process or Hardware condition.

The hardware condition must be configured by the user using the P1_CF, P3_CF and P4_CF bytes.

The hardware condition is detected by a low level on the corresponding input.

Example: Configure the pin 2 of the port 3 as Hardware condition.

The corresponding value for these bytes are:

P1_CF = FFh

P3_CF = FBh

P4_CF = FFh

Note: If more than 1 nit of P1_CF, P3_CF and P4_CF are to zero, the higher priority is on P1_CF.0, the lowest priority is on P4_CF.1

The bootloader USB must be activated if the BLJB = 0.

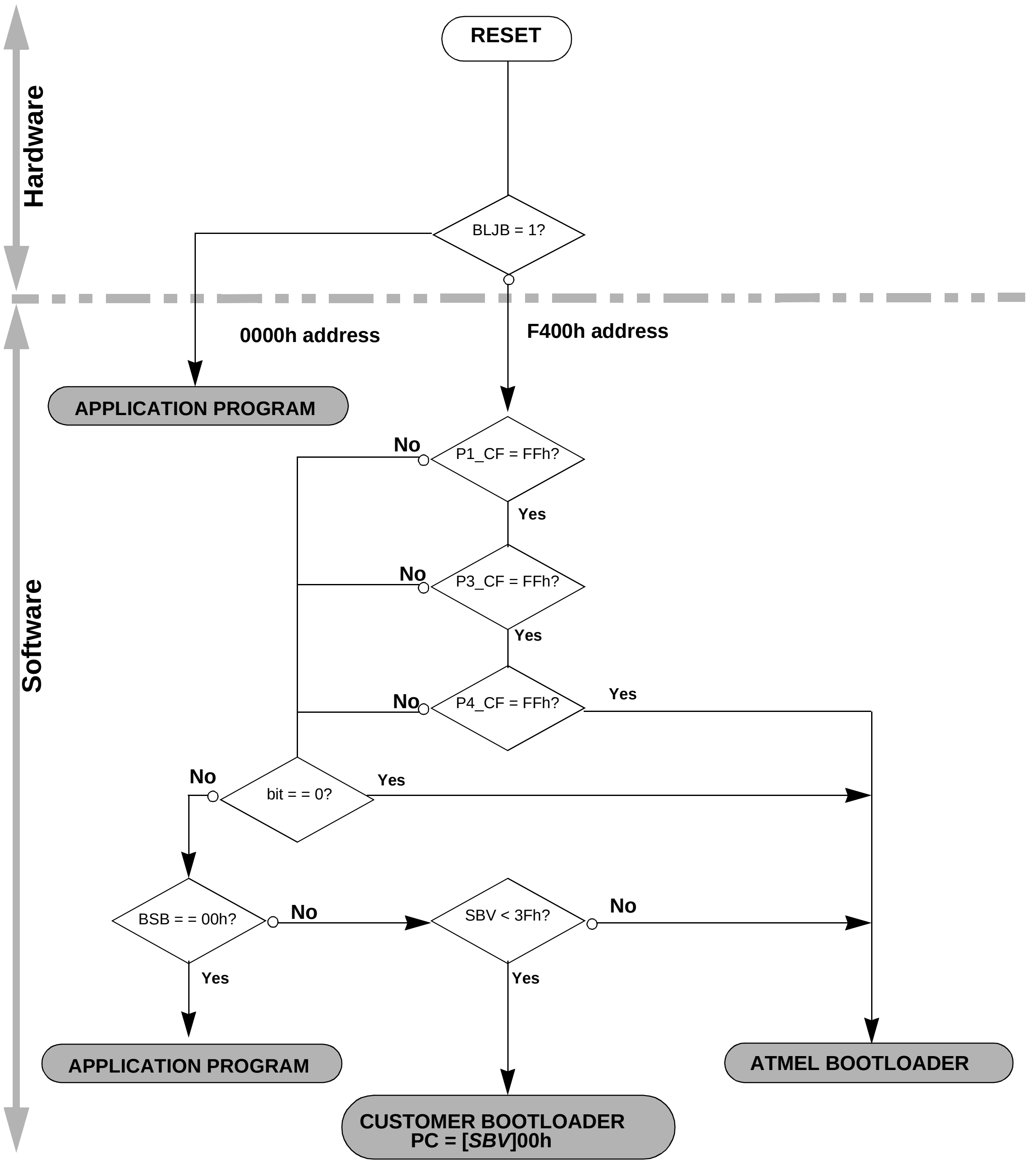
The on-chip bootloader boot process is shown Figure 23.

| | **Purpose** |
|---|---|
| BLJB | The Bootloader Jump Bit forces the application execution.<br>BLJB = 0 = > Bootloader execution.<br>BLJB = 1 = > Application execution<br>The BLJB is a fuse bit in the Hardware Byte.<br>That can be modified by hardware (programmer) or by software (API).<br>Note: 1. The BLJB test is perform by hardware to prevent any program execution. |
| SBV | The Software Boot Vector contains the high address of customer bootloader stored in the application.<br>SBV = FCh (default value) if no customer bootloader in user Flash.<br>Note: The customer bootloader is called by JMP [SBV]00h instruction. |
| P1_CF | The P1_CF can contain the user condition for Hardware condition on Port 1<br>P1_CF is a byte of the XROW area |
| P3_CF | The P3_CF can contain the user condition for Hardware condition on Port 3<br>P3_CF is a byte of the XROW area |
| P4_CF | The P4_CF can contain the user condition for Hardware condition on Port 4<br>P4_CF is a byte of the XROW area |

All routines for software access are provided in the C Flash driver (see reference section).

Example of boot process in FM1 (see Figure 23)

**Figure 23.** Low Pin-count Boot Process Algorithm



## High Pin-Count Boot Process

At the falling edge of RESET, the bit ENBOOT in AUXR1 register is initialized with the value of Bootloader Jump Bit (BLJB).

Further at the falling edge of RESET if the following conditions (called Hardware condition) are detected:

- PSEN low
- EA high

- ALE high (or not connected)
  - After Hardware Condition the FCON register is initialized with the value 00h and the PC is initialized with F800h (FM1).

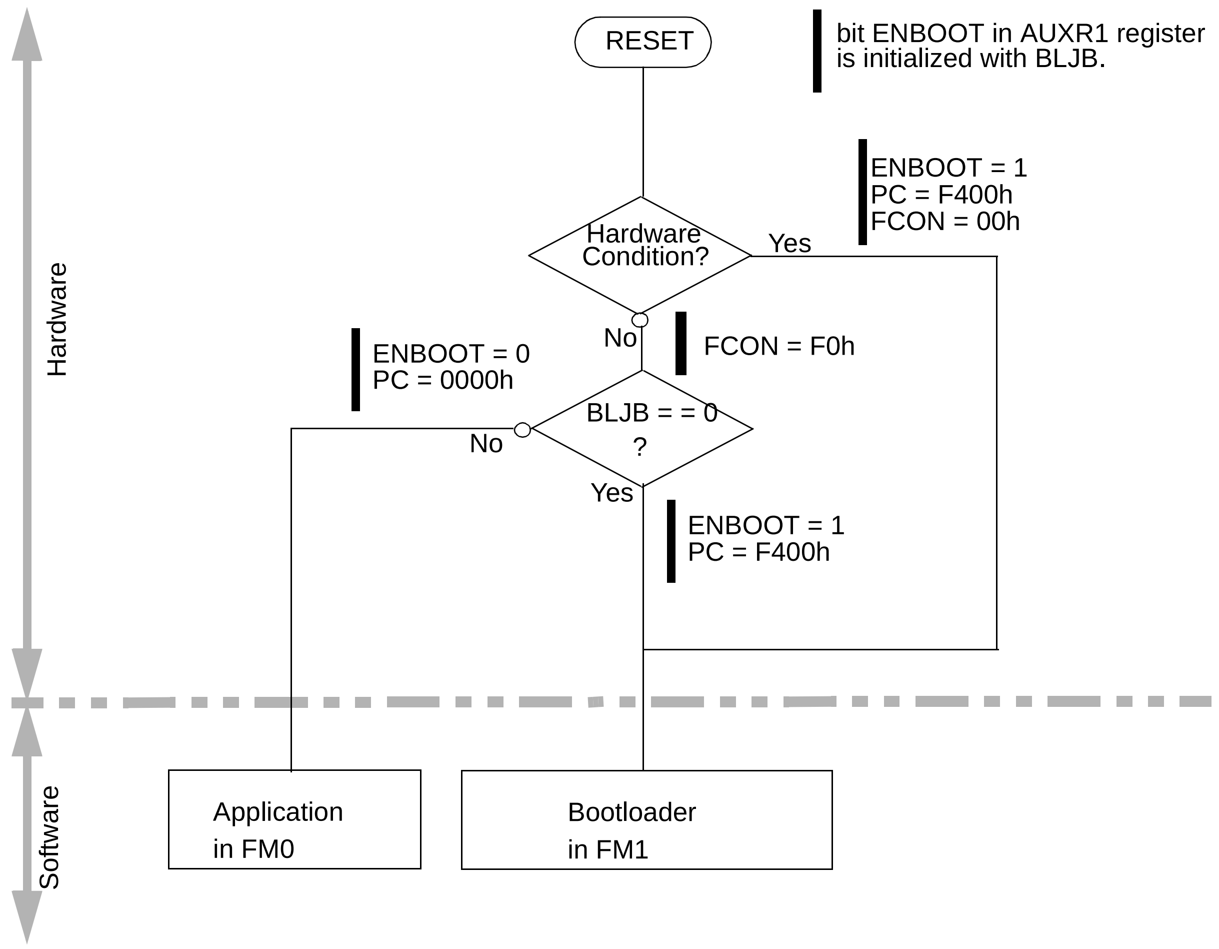
The Hardware condition makes the bootloader to be executed, whatever BLJB value is.

If no hardware condition is detected, the FCON register is initialized with the value F0h.

Check of the BLJB value.

- If bit BLJB = 1:
  User application in FM0 will be started at 0000h (standard reset).
- If bit BLJB = 0:
  Bootloader will be started at F800h in FM1.

**Figure 24.** Hardware Boot Process Algorithm



## Application Programming Interface

Several Application Program Interface (API) calls are available for use by an application program to permit selective erasing and programming of Flash pages. All calls are made by functions.

All these APIs will be described in an application note.

| API Call | Description |
|---|---|
| PROGRAM DATA BYTE | Write a byte in Flash memory |
| PROGRAM DATA PAGE | Write a page (128 bytes) in Flash memory |
| PROGRAM EEPROM BYTE | Write a byte in EEPROM memory |
| ERASE BLOCK | Erase all Flash memory |
| ERASE SW BOOT VECTOR (SBV) | Erase the software boot vector |
| PROGRAM SW BOOT VECTOR (SBV) | Write the software boot vector |
| ERASE HW BOOT VECTOR (HBV) | Erase the hardware boot vector |
| PROGRAM HW BOOT VECTOR (HBV) | Write the hardware boot vector |
| PROGRAM EXTRA BYTE (EB) | Write the extra byte |
| READ DATA BYTE | |
| READ EEPROM BYTE | |
| READ FAMILY CODE | |
| READ MANUFACTURER CODE | |
| READ PRODUCT NAME | |
| READ REVISION NUMBER | |
| READ STATUS BIT (BSB) | Read the status bit |
| READ BOOT VECTOR (SBV) | Read the boot vector |
| READ EXTRA BYTE (EB) | Read the extra byte |
| PROGRAM X2 | Write the hardware flag for X2 mode |
| READ X2 | Read the hardware flag for X2 mode |
| PROGRAM BLJB | Write the hardware flag BLJB |
| READ BLJB | Read the hardware flag BLJB |

## Application Remarks

- A user bootloader can be mapped at address [SBV]00h. The byte SBV contains the high byte of the boot address, and can be read and written by API.
- The API can be called during user application, without disabling interrupt.
  The interrupts are disabled by some APIs, for complex operations.

## XROW Bytes

**Table 44.** XRow Mapping

| Mnemonic | Description | Default Value | Address |
|---|---|---|---|
| BSB | Boot Status Byte | | 00h |
| SBV | Software Boot Vector | F4h | 01h |
| P1_CF | | FFh | 02h |
| P3_CF | | FFh | 03h |
| P4_CF | | FFh | 04h |
| SSB | Software Security Byte | FFh | 05h |
| EB | Extra Byte | FFh | 06h |
| | reserved | | 07h-2Fh |
| | Manufacturer Code | 58h | 30h |
| ID1 | Device ID#1: Family code | D7h | 31h |
| | reserved | | 32h-5Fh |
| ID2 | Device ID#2: Memories size and type | F7h | 60h |
| ID3 | Device ID#3: Name and Revision | DFh | 61h |
| | reserved | | 62h-7Fh |

**Table 45.** SBV Register
Software Boot Vector

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **ADD 7** | **ADD 6** | **ADD 5** | **ADD 4** | **ADD 3** | **ADD 2** | **ADD 1** | **ADD 0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 - 0 | ADD7:0 | **MSB of user bootloader address location** |

Notes: 1. Default value after erasing chip: FFh
2. Only accessed by the **API or in the parallel programming mode**.

**Table 46.** EB Register
EXTRA BYTE

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 - 0 | - | **User definition** |

Notes: 1. Default value after erasing chip: FFh
2. Only accessed by the **API or in the parallel programming mode**.

## Hardware Byte

**Table 47.** Hardware Byte

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| X2B | BLJB | OSCON1 | OSCON0 | - | LB2 | LB1 | LB0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | X2B | **X2 Bit**<br>Set this bit to start in standard mode<br>Clear this bit to start in X2 mode. |
| 6 | BLJB | **Bootloader Jump Bit**<br>Set this bit to start the user's application on next reset at address 0000h.<br>Cleared this bit to start the bootloader at address F400h (default). |
| 5-4 | OSCON1-0 | **Oscillator Control Bits**<br>These two bits are used to control the oscillator in order to reduce consumption.<br>OSCON1 OSCON0 Description<br>1 1 oscillator is configured to run from 0 to 32 MHz<br>1 0 oscillator is configured to run from 0 to 16 MHz<br>0 1 oscillator is configured to run from 0 to 8 MHz<br>0 0 this configuration shouldn't be set |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. |
| 2-0 | LB2:0 | **Lock Bits** |

Default value after erasing chip: FFh

Notes: 1. Only the 4 MSB bits can be access by software.
2. The 4 LSB bits can only be access by parallel mode.

## On-chip Expanded RAM (XRAM)

The AT89C5131 provides additional Bytes of random access memory (RAM) space for increased data parameter handling and high level language usage.

AT89C5131 devices have expanded RAM in external data space; maximum size and location are described in Table 48.

**Table 48.** Description of Expanded RAM

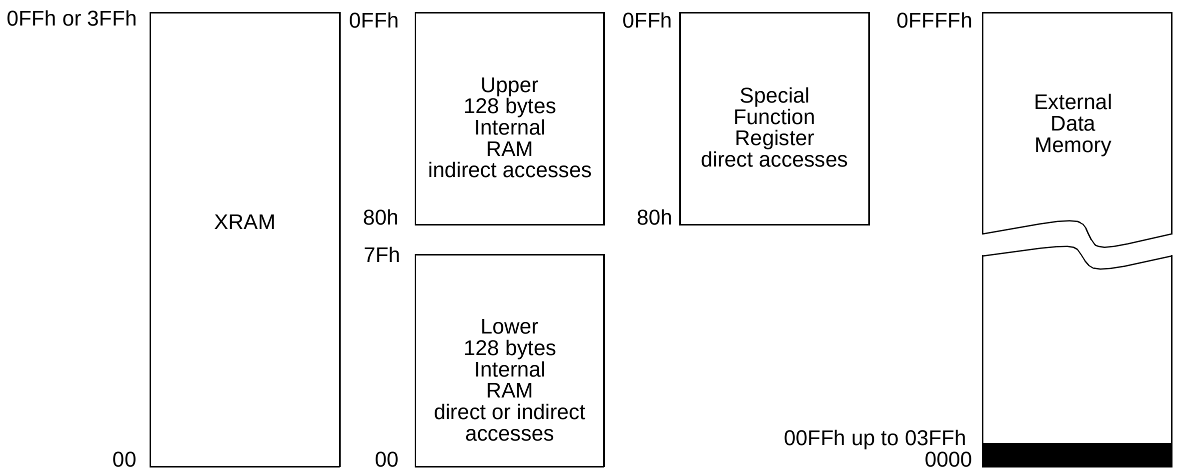
| Part Number | XRAM Size | Address | |
|---|---|---|---|
| | | Start | End |
| AT89C5131 | 1024 | 00h | 3FFh |

The AT89C5131 has on-chip data memory that is mapped into the following four separate segments.

1. The Lower 128 bytes of RAM (addresses 00h to 7Fh) are directly and indirectly addressable.
2. The Upper 128 bytes of RAM (addresses 80h to FFh) are indirectly addressable only.
3. The Special Function Registers, SFRs, (addresses 80h to FFh) are directly addressable only.
4. The expanded RAM bytes are indirectly accessed by MOVX instructions, and with the EXTRAM bit cleared in the AUXR register (see Table 48)

The lower 128 bytes can be accessed by either direct or indirect addressing. The Upper 128 bytes can be accessed by indirect addressing only. The Upper 128 bytes occupy the same address space as the SFR. That means they have the same address, but are physically separate from SFR space.

**Figure 25.** Internal and External Data Memory Address


0FFh or 3FFh
XRAM
00
0FFh
Upper 128 bytes Internal RAM indirect accesses
80h
7Fh
Lower 128 bytes Internal RAM direct or indirect accesses
00
0FFh
Special Function Register direct accesses
80h
0FFFFh
External Data Memory
00FFh up to 03FFh
0000

When an instruction accesses an internal location above address 7Fh, the CPU knows whether the access is to the upper 128 bytes of data RAM or to SFR space by the addressing mode used in the instruction.

- Instructions that use direct addressing access SFR space. For example: MOV 0A0H, # data, accesses the SFR at location 0A0h (which is P2).
- Instructions that use indirect addressing access the Upper 128 bytes of data RAM. For example: MOV atR0, # data where R0 contains 0A0h, accesses the data byte at address 0A0h, rather than P2 (whose address is 0A0h).
- The XRAM bytes can be accessed by indirect addressing, with EXTRAM bit cleared and MOVX instructions. This part of memory which is physically located on-chip, logically occupies the first bytes of external data memory. The bits XRS0 and XRS1 are used to hide a part of the available XRAM as explained in Table 48. This can be useful if external peripherals are mapped at addresses already used by the internal XRAM.
- With EXTRAM = 0, the XRAM is indirectly addressed, using the MOVX instruction in combination with any of the registers R0, R1 of the selected bank or DPTR. An access to XRAM will not affect ports P0, P2, P3.6 (WR) and P3.7 (RD). For example, with EXTRAM = 0, MOVX atR0, # data where R0 contains 0A0H, accesses the XRAM at address 0A0H rather than external memory. An access to external data memory locations higher than the accessible size of the XRAM will be performed with the MOVX DPTR instructions in the same way as in the standard 80C51, with P0 and P2 as data/address busses, and P3.6 and P3.7 as write and read timing signals. Accesses to XRAM above 0FFH can only be done by the use of DPTR.
- With EXTRAM = 1, MOVX @Ri and MOVX @DPTR will be similar to the standard 80C51. MOVX at Ri will provide an eight-bit address multiplexed with data on Port0 and any output port pins can be used to output higher order address bits. This is to provide the external paging capability. MOVX @DPTR will generate a sixteen-bit address. Port2 outputs the high-order eight address bits (the contents of DPH) while Port0 multiplexes the low-order eight address bits (DPL) with data. MOVX at Ri and MOVX @DPTR will generate either read or write signals on P3.6 ($\overline{\text{WR}}$) and P3.7 ($\overline{\text{RD}}$).

The stack pointer (SP) may be located anywhere in the 256 bytes RAM (lower and upper RAM) internal data memory. The stack may not be located in the XRAM.

The M0 bit allows to stretch the XRAM timings; if M0 is set, the read and write pulses are extended from 6 to 30 clock periods. This is useful to access external slow peripherals.

**Table 49.** AUXR Register
AUXR - Auxiliary Register (8Eh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DPU | - | M0 | - | XRS1 | XRS0 | EXTRAM | AO |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | DPU | **Disable Weak Pull Up**<br>Cleared to enabled weak pull up on standard Ports.<br>Set to disable weak pull up on standard Ports. |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit |
| 5 | M0 | **Pulse length**<br>Cleared to stretch MOVX control: the $\overline{RD}$ and the $\overline{WR}$ pulse length is 6 clock periods (default).<br>Set to stretch MOVX control: the $\overline{RD}$ and the $\overline{WR}$ pulse length is 30 clock periods. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit |
| 3 | XRS1 | **XRAM Size**<br>XRS1 / XRS0 / XRAM size<br>0 0 256 bytes<br>0 1 512 bytes<br>1 0 768 bytes<br>1 1 1024 bytes (default) |
| 2 | XRS0 | |
| 1 | EXTRAM | **EXTRAM bit**<br>Cleared to access internal XRAM using MOVX at $\overline{Ri}$ at DPTR.<br>Set to access external memory. |
| 0 | AO | **ALE Output bit**<br>Cleared, ALE is emitted at a constant rate of 1/6 the oscillator frequency (or 1/3 if X2 mode is used) (default).<br>Set, ALE is active only when a MOVX or MOVC instruction is used. |

Reset Value = 0X0X 1100b
Not bit addressable

## Timer 2

The Timer 2 in the AT89C5131 is the standard C52 Timer 2. It is a 16-bit timer/counter: the count is maintained by two cascaded eight-bit timer registers, TH2 and TL2. It is controlled by T2CON (Table 50) and T2MOD (Table 51) registers. Timer 2 operation is similar to Timer 0 and Timer 1. $C/\overline{T2}$ selects $F_{OSC}/12$ (timer operation) or external pin T2 (counter operation) as the timer clock input. Setting TR2 allows TL2 to be incremented by the selected input.

Timer 2 has 3 operating modes: capture, auto reload and Baud Rate Generator. These modes are selected by the combination of RCLK, TCLK and $CP/\overline{RL2}$ (T2CON).

Refer to the Atmel 8-bit microcontroller hardware documentation for the description of Capture and Baud Rate Generator Modes.

Timer 2 includes the following enhancements:

- Auto-reload mode with up or down counter
- Programmable Clock-output

### Auto-reload Mode

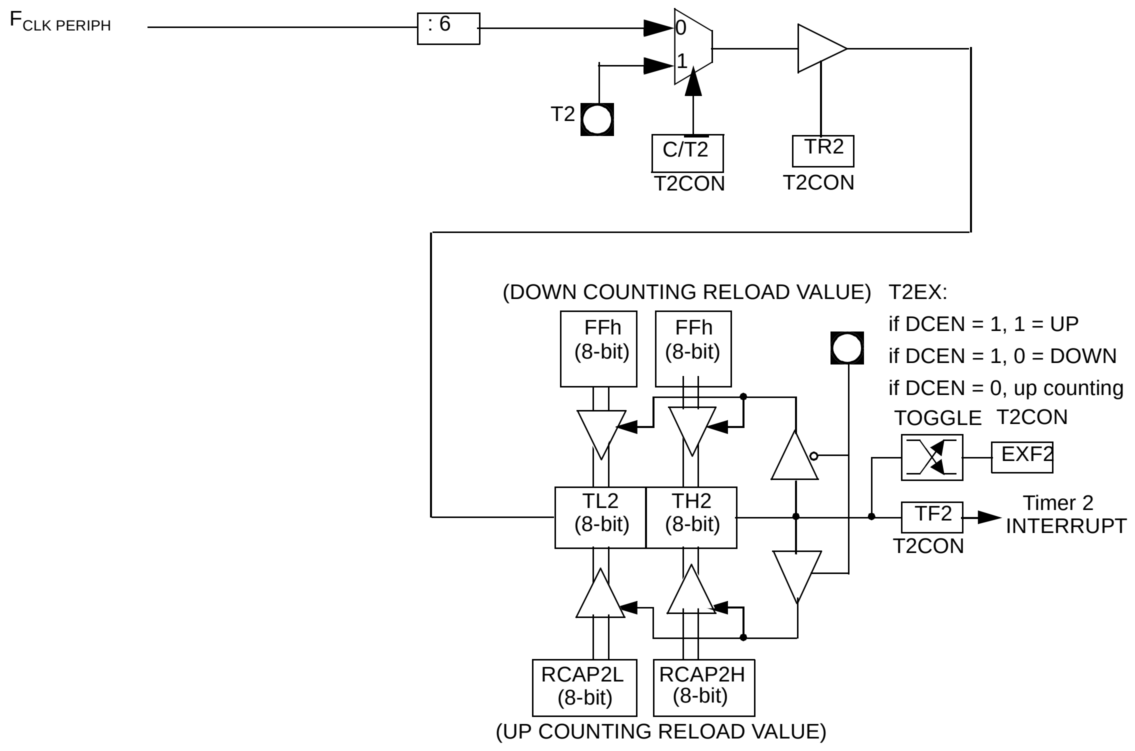
The Auto-reload mode configures Timer 2 as a 16-bit timer or event counter with automatic reload. If DCEN bit in T2MOD is cleared, Timer 2 behaves as in 80C52 (refer to the Atmel 8-bit microcontroller hardware description). If DCEN bit is set, Timer 2 acts as an Up/down timer/counter as shown in Figure 26. In this mode the T2EX pin controls the direction of count.

When T2EX is high, Timer 2 counts up. Timer overflow occurs at FFFFh which sets the TF2 flag and generates an interrupt request. The overflow also causes the 16-bit value in RCAP2H and RCAP2L registers to be loaded into the timer registers TH2 and TL2.

When T2EX is low, Timer 2 counts down. Timer underflow occurs when the count in the timer registers TH2 and TL2 equals the value stored in RCAP2H and RCAP2L registers. The underflow sets TF2 flag and reloads FFFFh into the timer registers.

The EXF2 bit toggles when Timer 2 overflows or underflows according to the direction of the count. EXF2 does not generate any interrupt. This bit can be used to provide 17-bit resolution.

**Figure 26.** Auto-reload Mode Up/Down Counter (DCEN = 1)



## Programmable Clock Output

In the Clock-out mode, Timer 2 operates as a 50%-duty-cycle, programmable clock generator (See Figure 27). The input clock increments TL2 at frequency $F_{CLK\ PERIPH}$/2. The timer repeatedly counts to overflow from a loaded value. At overflow, the contents of RCAP2H and RCAP2L registers are loaded into TH2 and TL2. In this mode, Timer 2 overflows do not generate interrupts. The following formula gives the Clock-out frequency as a function of the system oscillator frequency and the value in the RCAP2H and RCAP2L registers

$$Clock - OutFrequency = \frac{F_{CLKPERIPH}}{4 \times (65536 - RCAP2H/RCAP2L)}$$

For a 16 MHz system clock, Timer 2 has a programmable frequency range of 61 Hz ($F_{CLK\ PERIPH}/2^{16}$) to 4 MHz ($F_{CLK\ PERIPH}$/4). The generated clock signal is brought out to T2 pin (P1.0).

Timer 2 is programmed for the Clock-out mode as follows:

- Set T2OE bit in T2MOD register.
- Clear C/$\overline{T2}$ bit in T2CON register.
- Determine the 16-bit reload value from the formula and enter it in RCAP2H/RCAP2L registers.
- Enter a 16-bit initial value in timer registers TH2/TL2. It can be the same as the reload value or a different one depending on the application.
- To start the timer, set TR2 run control bit in T2CON register.

It is possible to use Timer 2 as a baud rate generator and a clock generator simultaneously. For this configuration, the baud rates and clock frequencies are not independent since both functions use the values in the RCAP2H and RCAP2L registers.

**Figure 27.** Clock-out Mode C/$\overline{\text{T2}}$ = 0

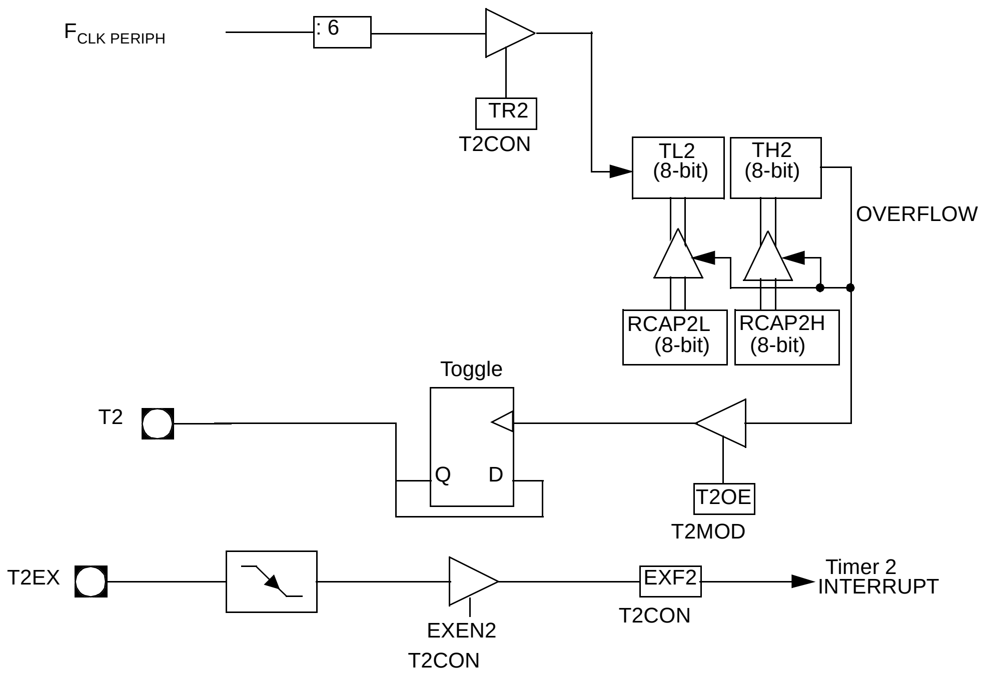


**Table 50.** T2CON Register
T2CON - Timer 2 Control Register (C8h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/T2# | CP/RL2# |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | TF2 | **Timer 2 overflow Flag**<br>Must be cleared by software.<br>Set by hardware on Timer 2 overflow, if RCLK = 0 and TCLK = 0. |
| 6 | EXF2 | **Timer 2 External Flag**<br>Set when a capture or a reload is caused by a negative transition on T2EX pin if EXEN2 = 1.<br>When set, causes the CPU to vector to Timer 2 interrupt routine when Timer 2 interrupt is enabled.<br>Must be cleared by software. EXF2 doesn't cause an interrupt in Up/down counter mode (DCEN = 1). |
| 5 | RCLK | **Receive Clock bit**<br>Cleared to use Timer 1 overflow as receive clock for serial port in mode 1 or 3.<br>Set to use Timer 2 overflow as receive clock for serial port in mode 1 or 3. |
| 4 | TCLK | **Transmit Clock bit**<br>Cleared to use Timer 1 overflow as transmit clock for serial port in mode 1 or 3.<br>Set to use Timer 2 overflow as transmit clock for serial port in mode 1 or 3. |
| 3 | EXEN2 | **Timer 2 External Enable bit**<br>Cleared to ignore events on T2EX pin for Timer 2 operation.<br>Set to cause a capture or reload when a negative transition on T2EX pin is detected, if Timer 2 is not used to clock the serial port. |
| 2 | TR2 | **Timer 2 Run control bit**<br>Cleared to turn off Timer 2.<br>Set to turn on Timer 2. |
| 1 | C/T2# | **Timer/Counter 2 select bit**<br>Cleared for timer operation (input from internal clock system: $F_{CLK\ PERIPH}$).<br>Set for counter operation (input from T2 input pin, falling edge trigger). Must be 0 for clock out mode. |
| 0 | CP/RL2# | **Timer 2 Capture/Reload bit**<br>If RCLK = 1 or TCLK = 1, CP/RL2# is ignored and timer is forced to Auto-reload on Timer 2 overflow.<br>Cleared to Auto-reload on Timer 2 overflows or negative transitions on T2EX pin if EXEN2 = 1.<br>Set to capture on negative transitions on T2EX pin if EXEN2 = 1. |

Reset Value = 0000 0000b
Bit addressable

**Table 51.** T2MOD Register
T2MOD - Timer 2 Mode Control Register (C9h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | T2OE | DCEN |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 2 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 1 | T2OE | **Timer 2 Output Enable bit**<br>Cleared to program P1.0/T2 as clock input or I/O port.<br>Set to program P1.0/T2 as clock output. |
| 0 | DCEN | **Down Counter Enable bit**<br>Cleared to disable Timer 2 as up/down counter.<br>Set to enable Timer 2 as up/down counter. |

Reset Value = XXXX XX00b
Not bit addressable

## Programmable Counter Array (PCA)

The PCA provides more timing capabilities with less CPU intervention than the standard timer/counters. Its advantages include reduced software overhead and improved accuracy. The PCA consists of a dedicated timer/counter which serves as the time base for an array of five compare/capture modules. Its clock input can be programmed to count any one of the following signals:

- Peripheral clock frequency ($F_{CLK\ PERIPH}$) ÷ 6
- Peripheral clock frequency ($F_{CLK\ PERIPH}$) ÷ 2
- Timer 0 overflow
- External input on ECI (P1.2)

Each compare/capture modules can be programmed in any one of the following modes:

- rising and/or falling edge capture,
- software timer
- high-speed output, or
- pulse width modulator

Module 4 can also be programmed as a watchdog timer (see Section "PCA Watchdog Timer", page 79).

When the compare/capture modules are programmed in the capture mode, software timer, or high speed output mode, an interrupt can be generated when the module executes its function. All five modules plus the PCA timer overflow share one interrupt vector.

The PCA timer/counter and compare/capture modules share Port 1 for external I/O. These pins are listed below. If the port is not used for the PCA, it can still be used for standard I/O.

| **PCA Component** | **External I/O Pin** |
|---|---|
| 16-bit Counter | P1.2/ECI |
| 16-bit Module 0 | P1.3/CEX0 |
| 16-bit Module 1 | P1.4/CEX1 |
| 16-bit Module 2 | P1.5/CEX2 |
| 16-bit Module 3 | P1.6/CEX3 |
| 16-bit Module 4 | P1.7/CEX4 |

The PCA timer is a common time base for all five modules (see Figure 28). The timer count source is determined from the CPS1 and CPS0 bits in the CMOD register (Table 52) and can be programmed to run at:

- 1/6 the peripheral clock frequency ($F_{CLK\ PERIPH}$).
- 1/2 the peripheral clock frequency ($F_{CLK\ PERIPH}$).
- The Timer 0 overflow
- The input on the ECI pin (P1.2)

**Figure 28.** PCA Timer/Counter

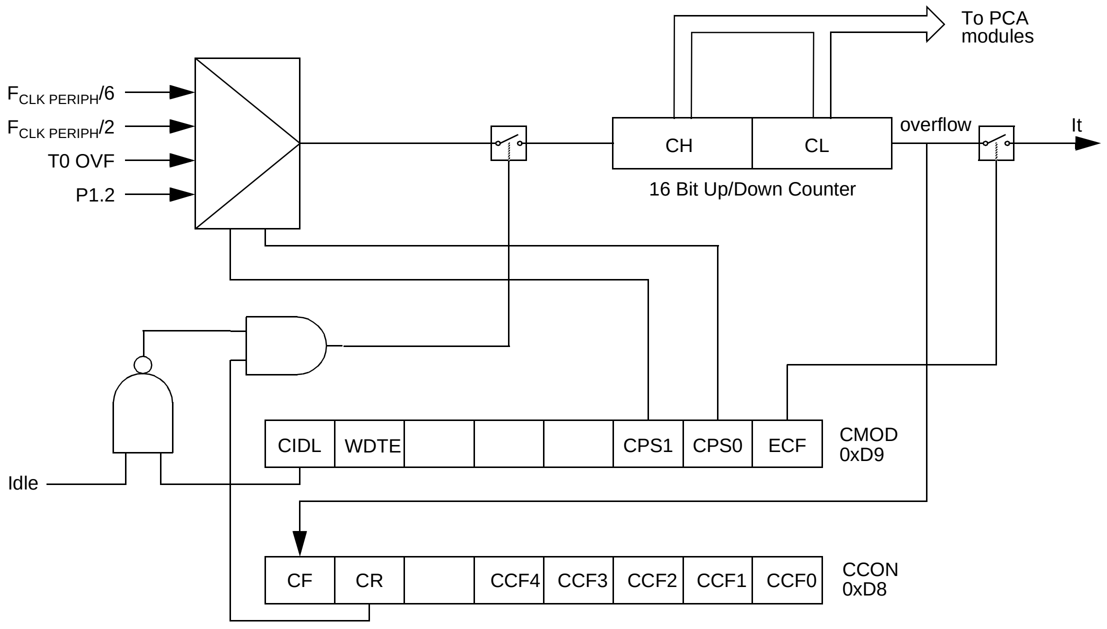


**Table 52.** CMOD Register

CMOD - PCA Counter Mode Register (D9h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **CIDL** | **WDTE** | **-** | **-** | **-** | **CPS1** | **CPS0** | **ECF** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | CIDL | **Counter Idle Control**<br>Cleared to program the PCA Counter to continue functioning during idle Mode.<br>Set to program PCA to be gated off during idle. |
| 6 | WDTE | **Watchdog Timer Enable**<br>Cleared to disable Watchdog Timer function on PCA Module 4.<br>Set to enable Watchdog Timer function on PCA Module 4. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 2 | CPS1 | **PCA Count Pulse Select** |
| 1 | CPS0 | CPS1 / CPS0 / Selected PCA input<br>0 0 Internal clock $f_{CLK\ PERIPH}/6$<br>0 1 Internal clock $f_{CLK\ PERIPH}/2$<br>1 0 Timer 0 Overflow<br>1 1 External clock at ECI/P1.2 pin (max rate = $f_{CLK\ PERIPH}/ 4$) |
| 0 | ECF | **PCA Enable Counter Overflow Interrupt**<br>Cleared to disable CF bit in CCON to inhibit an interrupt.<br>Set to enable CF bit in CCON to generate an interrupt. |

Reset Value = 00XX X000b

Not bit addressable

The CMOD register includes three additional bits associated with the PCA (See Figure 28 and Table 52).

- The CIDL bit allows the PCA to stop during idle mode.
- The WDTE bit enables or disables the watchdog function on module 4.
- The ECF bit when set causes an interrupt and the PCA overflow flag CF (in the CCON SFR) to be set when the PCA timer overflows.

The CCON register contains the run control bit for the PCA and the flags for the PCA timer (CF) and each module (see Table 53).

- Bit CR (CCON.6) must be set by software to run the PCA. The PCA is shut off by clearing this bit.
- Bit CF: The CF bit (CCON.7) is set when the PCA counter overflows and an interrupt will be generated if the ECF bit in the CMOD register is set. The CF bit can only be cleared by software.
- Bits 0 through 4 are the flags for the modules (bit 0 for module 0, bit 1 for module 1, etc.) and are set by hardware when either a match or a capture occurs. These flags can only be cleared by software.

**Table 53.** CCON Register
CCON - PCA Counter Control Register (D8h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **CF** | **CR** | – | **CCF4** | **CCF3** | **CCF2** | **CCF1** | **CCF0** |

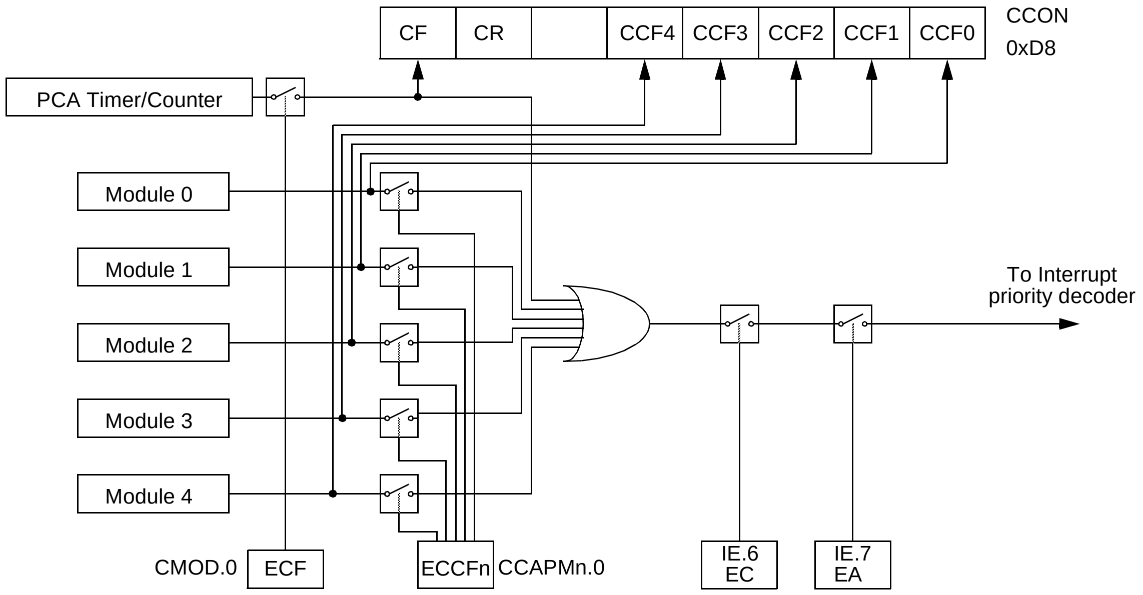
| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | CF | **PCA Counter Overflow flag**<br>Set by hardware when the counter rolls over. CF flags an interrupt if bit ECF in CMOD is set. CF may be set by either hardware or software but can only be cleared by software. |
| 6 | CR | **PCA Counter Run control bit**<br>Must be cleared by software to turn the PCA counter off.<br>Set by software to turn the PCA counter on. |
| 5 | – | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | CCF4 | **PCA Module 4 interrupt flag**<br>Must be cleared by software.<br>Set by hardware when a match or capture occurs. |
| 3 | CCF3 | **PCA Module 3 interrupt flag**<br>Must be cleared by software.<br>Set by hardware when a match or capture occurs. |
| 2 | CCF2 | **PCA Module 2 interrupt flag**<br>Must be cleared by software.<br>Set by hardware when a match or capture occurs. |
| 1 | CCF1 | **PCA Module 1 Interrupt Flag**<br>Must be cleared by software.<br>Set by hardware when a match or capture occurs. |
| 0 | CCF0 | **PCA Module 0 Interrupt Flag**<br>Must be cleared by software.<br>Set by hardware when a match or capture occurs. |

Reset Value = 000X 0000b
Not bit addressable

The watchdog timer function is implemented in module 4 (See Figure 31).

The PCA interrupt system is shown in Figure 29.

**Figure 29.** PCA Interrupt System



**PCA Modules:** each one of the five compare/capture modules has six possible functions. It can perform:

- 16-bit capture, positive-edge triggered
- 16-bit capture, negative-edge triggered
- 16-bit capture, both positive and negative-edge triggered
- 16-bit Software Timer
- 16-bit High-speed Output
- 8-bit Pulse Width Modulator

In addition, module 4 can be used as a Watchdog Timer.

Each module in the PCA has a special function register associated with it. These registers are: CCAPM0 for module 0, CCAPM1 for module 1, etc. (see Table 54). The registers contain the bits that control the mode that each module will operate in.

- The ECCF bit (CCAPMn.0 where n = 0, 1, 2, 3, or 4 depending on the module) enables the CCF flag in the CCON SFR to generate an interrupt when a match or compare occurs in the associated module.
- PWM (CCAPMn.1) enables the pulse width modulation mode.
- The TOG bit (CCAPMn.2) when set causes the CEX output associated with the module to toggle when there is a match between the PCA counter and the module's capture/compare register.
- The match bit MAT (CCAPMn.3) when set will cause the CCFn bit in the CCON register to be set when there is a match between the PCA counter and the module's capture/compare register.
- The next two bits CAPN (CCAPMn.4) and CAPP (CCAPMn.5) determine the edge that a capture input will be active on. The CAPN bit enables the negative edge, and

the CAPP bit enables the positive edge. If both bits are set both edges will be enabled and a capture will occur for either transition.

- The last bit in the register ECOM (CCAPMn.6) when set enables the comparator function.

Table 55 shows the CCAPMn settings for the various PCA functions.

**Table 54.** CCAPMn Registers (n = 0-4)

CCAPM0 - PCA Module 0 Compare/Capture Control Register (0DAh)
CCAPM1 - PCA Module 1 Compare/Capture Control Register (0DBh)
CCAPM2 - PCA Module 2 Compare/Capture Control Register (0DCh)
CCAPM3 - PCA Module 3 Compare/Capture Control Register (0DDh)
CCAPM4 - PCA Module 4 Compare/Capture Control Register (0DEh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | ECOMn | CAPPn | CAPNn | MATn | TOGn | PWMn | ECCFn |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | ECOMn | **Enable Comparator**<br>Cleared to disable the comparator function.<br>Set to enable the comparator function. |
| 5 | CAPPn | **Capture Positive**<br>Cleared to disable positive edge capture.<br>Set to enable positive edge capture. |
| 4 | CAPNn | **Capture Negative**<br>Cleared to disable negative edge capture.<br>Set to enable negative edge capture. |
| 3 | MATn | **Match**<br>When MATn = 1, a match of the PCA counter with this module’s compare/capture register causes the<br>CCFn bit in CCON to be set, flagging an interrupt. |
| 2 | TOGn | **Toggle**<br>When TOGn = 1, a match of the PCA counter with this module’s compare/capture register causes the CEXn pin to toggle. |
| 1 | PWMn | **Pulse Width Modulation Mode**<br>Cleared to disable the CEXn pin to be used as a pulse width modulated output.<br>Set to enable the CEXn pin to be used as a pulse width modulated output. |
| 0 | CCF0 | **Enable CCF Interrupt**<br>Cleared to disable compare/capture flag CCFn in the CCON register to generate an interrupt.<br>Set to enable compare/capture flag CCFn in the CCON register to generate an interrupt. |

Reset Value = X000 0000b
Not bit addressable

**Table 55.** PCA Module Modes (CCAPMn Registers)

| ECOMn | CAPPn | CAPNn | MATn | TOGn | PWMm | ECCFn | Module Function |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | No Operation |
| X | 1 | 0 | 0 | 0 | 0 | X | 16-bit capture by a positive-edge trigger on CEXn |
| X | 0 | 1 | 0 | 0 | 0 | X | 16-bit capture by a negative trigger on CEXn |
| X | 1 | 1 | 0 | 0 | 0 | X | 16-bit capture by a transition on CEXn |
| 1 | 0 | 0 | 1 | 0 | 0 | X | 16-bit Software Timer/Compare mode. |
| 1 | 0 | 0 | 1 | 1 | 0 | X | 16-bit High Speed Output |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 8-bit PWM |
| 1 | 0 | 0 | 1 | X | 0 | X | Watchdog Timer (module 4 only) |

There are two additional registers associated with each of the PCA modules. They are CCAPnH and CCAPnL and these are the registers that store the 16-bit count when a capture occurs or a compare should occur. When a module is used in the PWM mode these registers are used to control the duty cycle of the output (see Table 56 and Table 57)

**Table 56.** CCAPnH Registers (n = 0-4)

CCAP0H - PCA Module 0 Compare/Capture Control Register High (0FAh)
CCAP1H - PCA Module 1 Compare/Capture Control Register High (0FBh)
CCAP2H - PCA Module 2 Compare/Capture Control Register High (0FCh)
CCAP3H - PCA Module 3 Compare/Capture Control Register High (0FDh)
CCAP4H - PCA Module 4 Compare/Capture Control Register High (0FEh)

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 - 0 | - | **PCA Module n Compare/Capture Control**<br>CCAPnH Value |

Reset Value = 0000 0000b
Not bit addressable

**Table 57.** CCAPnL Registers (n = 0-4)

CCAP0L - PCA Module 0 Compare/Capture Control Register Low (0EAh)
CCAP1L - PCA Module 1 Compare/Capture Control Register Low (0EBh)
CCAP2L - PCA Module 2 Compare/Capture Control Register Low (0ECh)
CCAP3L - PCA Module 3 Compare/Capture Control Register Low (0EDh)
CCAP4L - PCA Module 4 Compare/Capture Control Register Low (0EEh)

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 - 0 | - | **PCA Module n Compare/Capture Control**<br>CCAPnL Value |

Reset Value = 0000 0000b
Not bit addressable

**Table 58.** CH Register

CH - PCA Counter Register High (0F9h)

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 - 0 | - | **PCA counter**<br>CH Value |

Reset Value = 0000 0000b
Not bit addressable

**Table 59.** CL Register

CL - PCA Counter Register Low (0E9h)

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

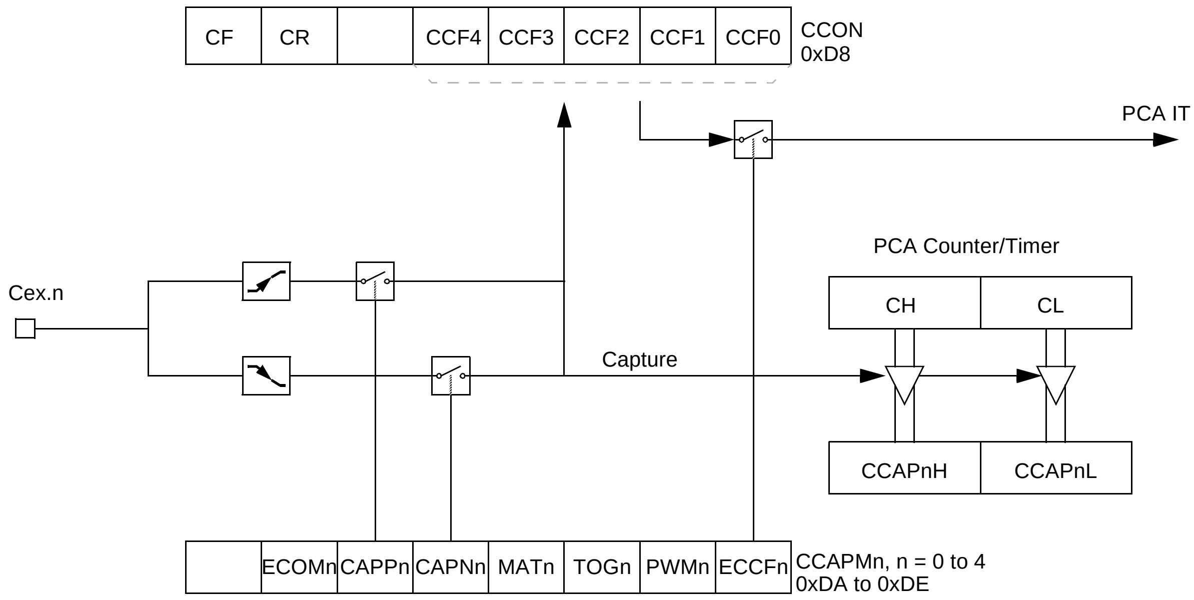
| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 - 0 | - | **PCA Counter**<br>CL Value |

Reset Value = 0000 0000b
Not bit addressable

## PCA Capture Mode

To use one of the PCA modules in the capture mode either one or both of the CCAPM bits CAPN and CAPP for that module must be set. The external CEX input for the module (on port 1) is sampled for a transition. When a valid transition occurs the PCA hardware loads the value of the PCA counter registers (CH and CL) into the module's capture registers (CCAPnL and CCAPnH). If the CCFn bit for the module in the CCON SFR and the ECCFn bit in the CCAPMn SFR are set then an interrupt will be generated (see Figure 30).

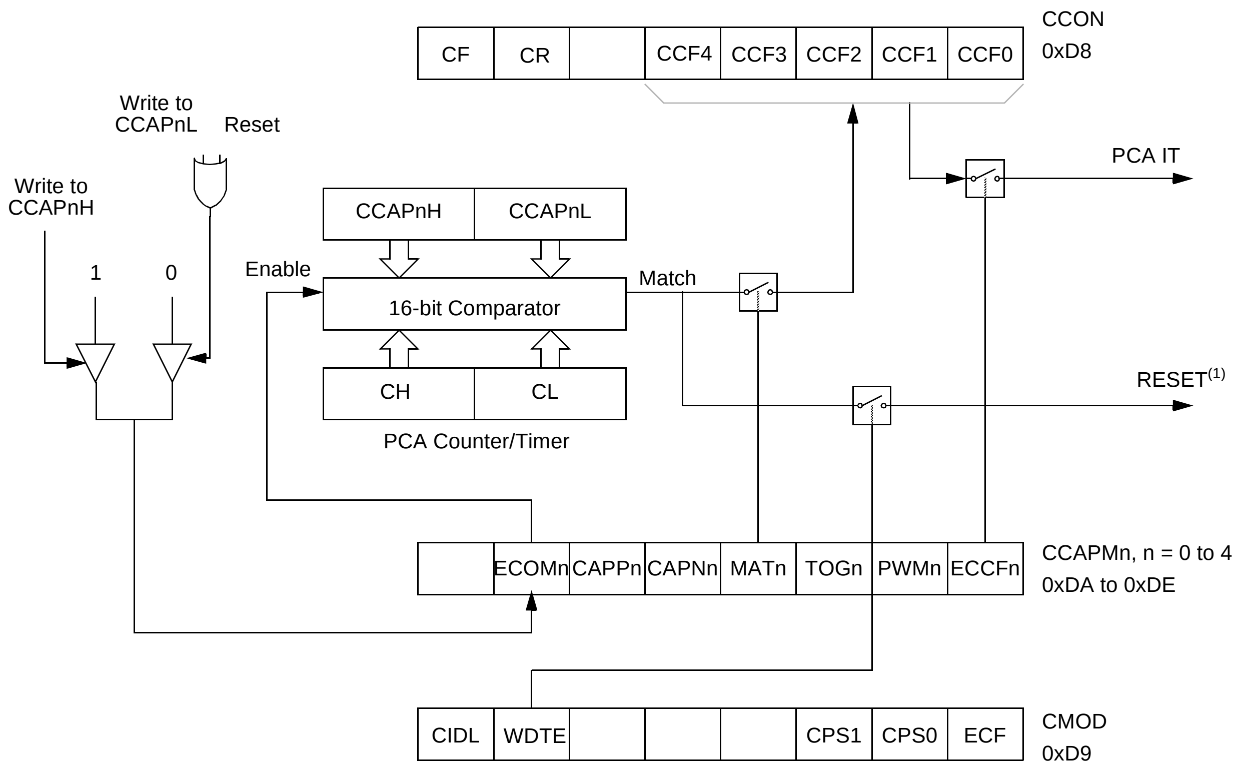
**Figure 30.** PCA Capture Mode



## 16-bit Software Timer/Compare Mode

The PCA modules can be used as software timers by setting both the ECOM and MAT bits in the modules CCAPMn register. The PCA timer will be compared to the module's capture registers and when a match occurs an interrupt will occur if the CCFn (CCON SFR) and the ECCFn (CCAPMn SFR) bits for the module are both set (see Figure 31).

**Figure 31.** PCA Compare Mode and PCA Watchdog Timer



Note: 1. Only for Module 4

Before enabling ECOM bit, CCAPnL and CCAPnH should be set with a non zero value, otherwise an unwanted match could happen. Writing to CCAPnH will set the ECOM bit.

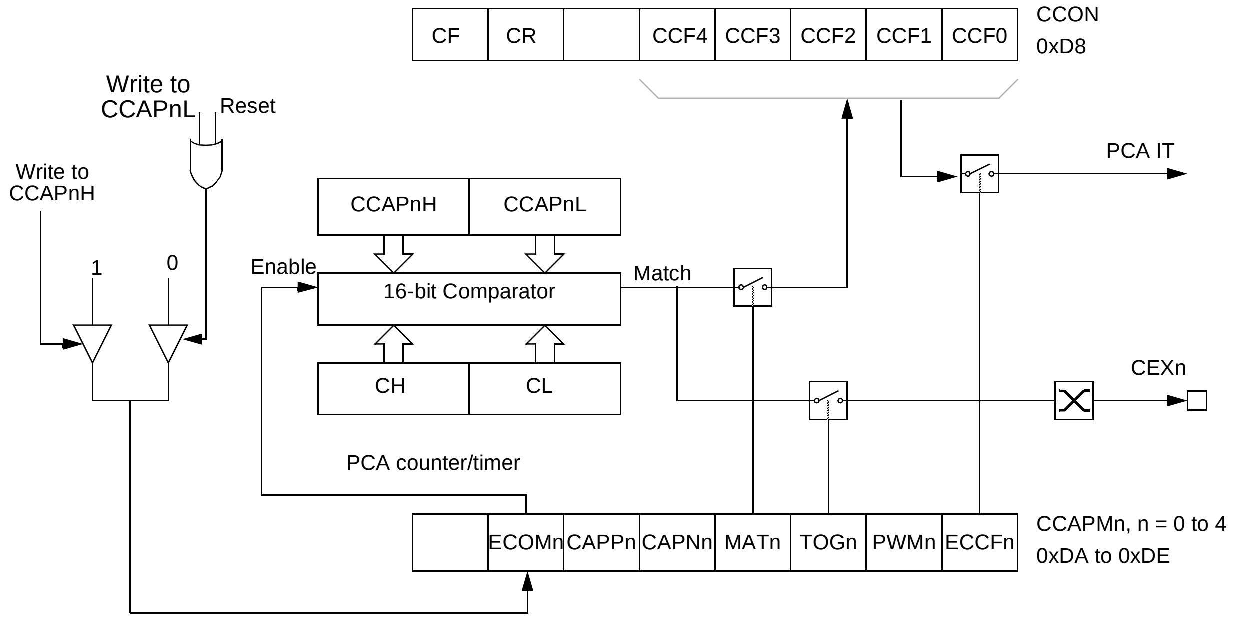
Once ECOM set, writing CCAPnL will clear ECOM so that an unwanted match doesn't occur while modifying the compare value. Writing to CCAPnH will set ECOM. For this reason, user software should write CCAPnL first, and then CCAPnH. Of course, the ECOM bit can still be controlled by accessing to CCAPMn register.

## High Speed Output Mode

In this mode, the CEX output (on port 1) associated with the PCA module will toggle each time a match occurs between the PCA counter and the module's capture registers. To activate this mode the TOG, MAT, and ECOM bits in the module's CCAPMn SFR must be set (see Figure 32).

A prior write must be done to CCAPnL and CCAPnH before writing the ECOMn bit.

**Figure 32.** PCA High-speed Output Mode



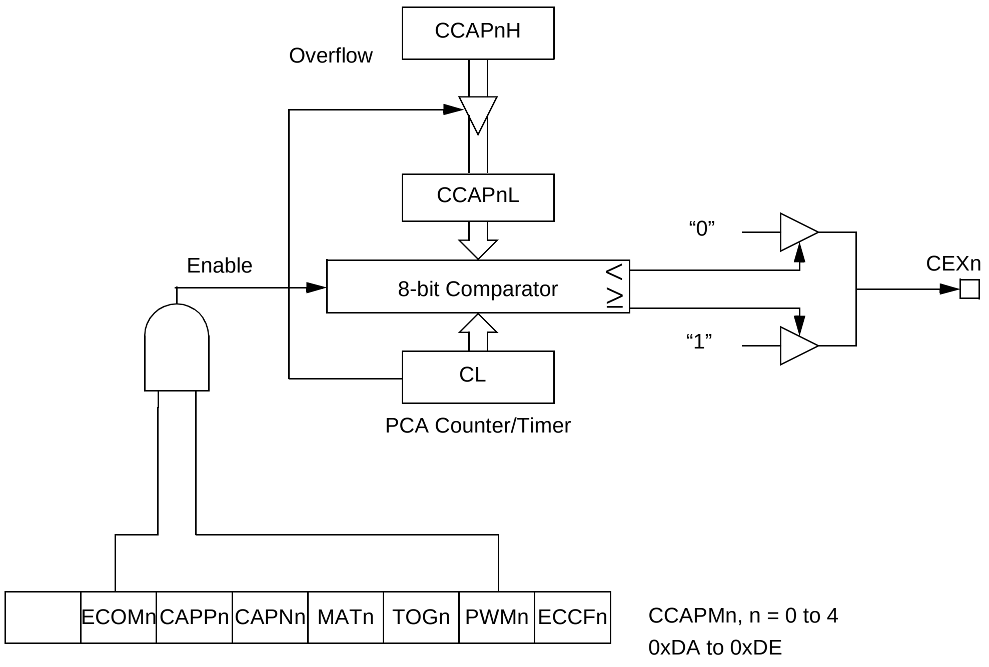
Before enabling ECOM bit, CCAPnL and CCAPnH should be set with a non zero value, otherwise an unwanted match could happen.

Once ECOM set, writing CCAPnL will clear ECOM so that an unwanted match doesn't occur while modifying the compare value. Writing to CCAPnH will set ECOM. For this reason, user software should write CCAPnL first, and then CCAPnH. Of course, the ECOM bit can still be controlled by accessing to CCAPMn register.

## Pulse Width Modulator Mode

All of the PCA modules can be used as PWM outputs. Figure 33 shows the PWM function. The frequency of the output depends on the source for the PCA timer. All of the modules will have the same frequency of output because they all share the PCA timer. The duty cycle of each module is independently variable using the module's capture register CCAPLn. When the value of the PCA CL SFR is less than the value in the module's CCAPLn SFR the output will be low, when it is equal to or greater than the output will be high. When CL overflows from FF to 00, CCAPLn is reloaded with the value in CCAPHn. This allows updating the PWM without glitches. The PWM and ECOM bits in the module's CCAPMn register must be set to enable the PWM mode.

**Figure 33.** PCA PWM Mode



## PCA Watchdog Timer

An on-board watchdog timer is available with the PCA to improve the reliability of the system without increasing chip count. Watchdog timers are useful for systems that are susceptible to noise, power glitches, or electrostatic discharge. Module 4 is the only PCA module that can be programmed as a watchdog. However, this module can still be used for other modes if the watchdog is not needed. Figure 31 shows a diagram of how the watchdog works. The user pre-loads a 16-bit value in the compare registers. Just like the other compare modes, this 16-bit value is compared to the PCA timer value. If a match is allowed to occur, an internal reset will be generated. This will not cause the RST pin to be driven high.

In order to hold off the reset, the user has three options:

1. Periodically change the compare value so it will never match the PCA timer
2. Periodically change the PCA timer value so it will never match the compare values, or
3. Disable the watchdog by clearing the WDTE bit before a match occurs and then re-enable it

The first two options are more reliable because the watchdog timer is never disabled as in option #3. If the program counter ever goes astray, a match will eventually occur and cause an internal reset. The second option is also not recommended if other PCA modules are being used. Remember, the PCA timer is the time base for all modules; changing the time base for other modules would not be a good idea. Thus, in most applications the first solution is the best option.

This watchdog timer won't generate a reset out on the reset pin.

## Serial I/O Port

The serial I/O port in the AT89C5131 is compatible with the serial I/O port in the 80C52. It provides both synchronous and asynchronous communication modes. It operates as an Universal Asynchronous Receiver and Transmitter (UART) in three full-duplex modes (modes 1, 2 and 3). Asynchronous transmission and reception can occur simultaneously and at different baud rates.

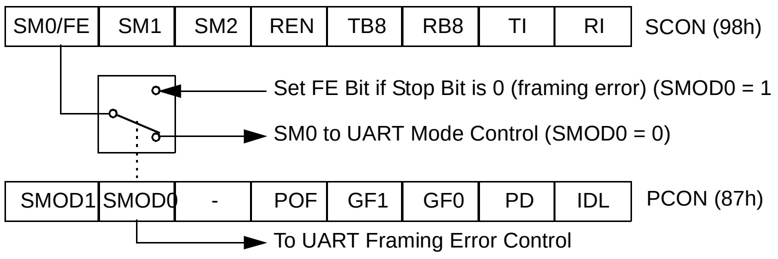
Serial I/O port includes the following enhancements:

- Framing error detection
- Automatic address recognition

### Framing Error Detection

Framing bit error detection is provided for the three asynchronous modes (modes 1, 2 and 3). To enable the framing bit error detection feature, set SMOD0 bit in PCON register (see Figure 34).

**Figure 34.** Framing Error Block Diagram


SM0/FE
SM1
SM2
REN
TB8
RB8
TI
RI
SCON (98h)
Set FE Bit if Stop Bit is 0 (framing error) (SMOD0 = 1
SM0 to UART Mode Control (SMOD0 = 0)
SMOD1
SMOD0
-
POF
GF1
GF0
PD
IDL
PCON (87h)
To UART Framing Error Control

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 CPUs. If a valid stop bit is not found, the Framing Error bit (FE) in SCON register (See Table 60) bit is set.
Software may examine FE bit after each reception to check for data errors. Once set, only software or a reset can clear FE bit. Subsequently received frames with valid stop bits cannot clear FE bit. When FE feature is enabled, RI rises on stop bit instead of the last data bit (See Figure 35 and Figure 36).

**Figure 35.** UART Timings in Mode 1

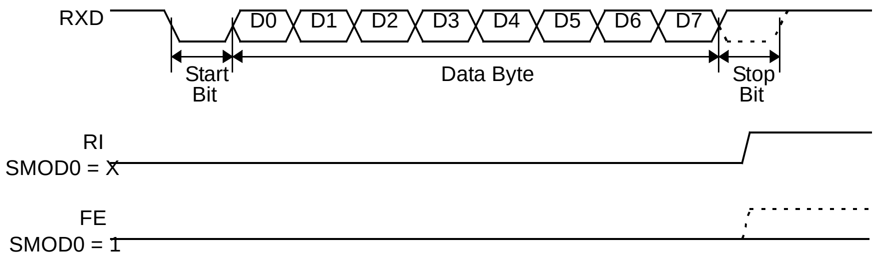
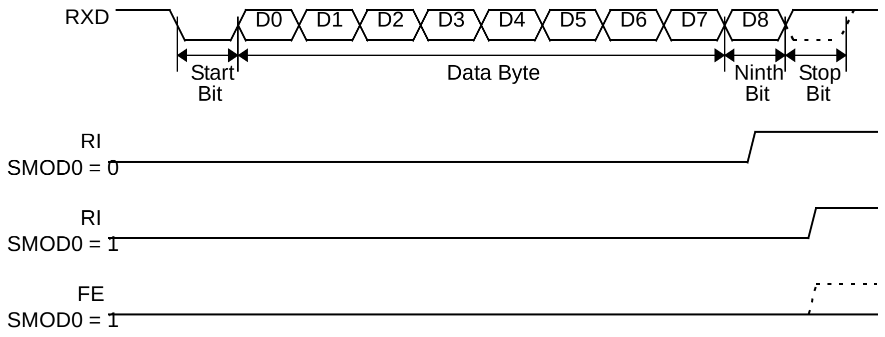


**Figure 36.** UART Timings in Modes 2 and 3



## 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, you 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:

SADDR0101 0110b
<u>SADEN1111 1100b</u>
Given0101 01XXb

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

Slave A:SADDR1111 0001b
<u>SADEN1111 1010b</u>
Given1111 0X0Xb

Slave B:SADDR1111 0011b
<u>SADEN1111 1001b</u>
Given1111 0XX1b

Slave C:SADDR1111 0010b
<u>SADEN1111 1101b</u>
Given1111 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 1; for slaves B and C, bit 1 is a don't care bit. To communicate with slaves B and C, but not slave A, 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.:

```
        SADDR0101 0110b
        SADEN1111 1100b
     Broadcast = SADDR OR SADEN1111 111Xb
```

The use of don't care bits provides flexibility in defining the broadcast address, in most applications, a broadcast address is FFh. The following is an example of using broadcast addresses:

```
    Slave A:SADDR1111 0001b
        SADEN1111 1010b
        Broadcast1111 1X11b,

    Slave B:SADDR1111 0011b
        SADEN1111 1001b
        Broadcast1111 1X11B,

    Slave C:SADDR = 1111 0010b
        SADEN1111 1101b
        Broadcast1111 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 an address FFh. To communicate with slaves A and B, but not slave C, the master can send and address FBh.

**Reset Addresses**

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

SADEN - Slave Address Mask Register (B9h)

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

Reset Value = 0000 0000b
Not bit addressable

SADDR - Slave Address Register (A9h)

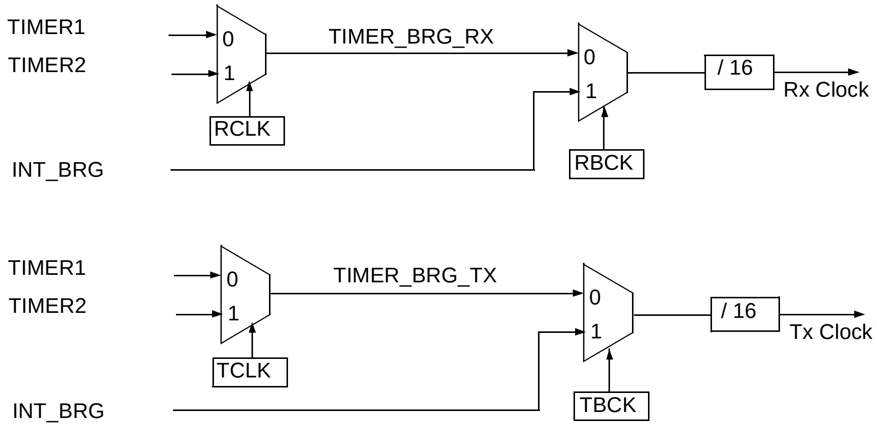
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Reset Value = 0000 0000b
Not bit addressable

## Baud Rate Selection for UART for Mode 1 and 3

The Baud Rate Generator for transmit and receive clocks can be selected separately via the T2CON and BDRCON registers.

**Figure 37.** Baud Rate Selection



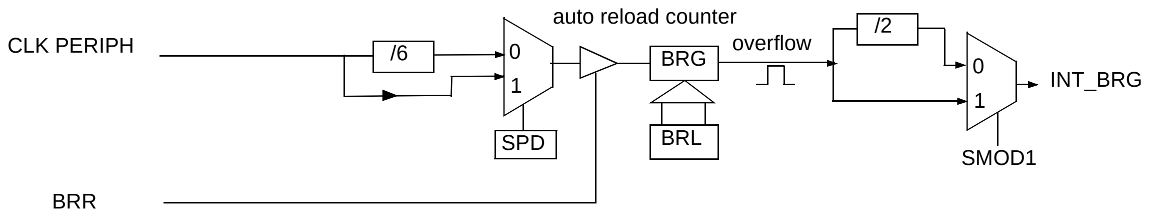
### Baud Rate Selection Table for UART

| TCLK (T2CON) | RCLK (T2CON) | TBCK (BDRCON) | RBCK (BDRCON) | Clock Source UART Tx | Clock Source UART Rx |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Timer 1 | Timer 1 |
| 1 | 0 | 0 | 0 | Timer 2 | Timer 1 |
| 0 | 1 | 0 | 0 | Timer 1 | Timer 2 |
| 1 | 1 | 0 | 0 | Timer 2 | Timer 2 |
| X | 0 | 1 | 0 | INT_BRG | Timer 1 |
| X | 1 | 1 | 0 | INT_BRG | Timer 2 |
| 0 | X | 0 | 1 | Timer 1 | INT_BRG |
| 1 | X | 0 | 1 | Timer 2 | INT_BRG |
| X | X | 1 | 1 | INT_BRG | INT_BRG |

### Internal Baud Rate Generator (BRG)

When the internal Baud Rate Generator is used, the Baud Rates are determined by the BRG overflow depending on the BRL reload value, the value of SPD bit (Speed Mode) in BDRCON register and the value of the SMOD1 bit in PCON register.

**Figure 38.** Internal Baud Rate



- The baud rate for UART is token by formula:

$$\text{Baud\_Rate} = \frac{2^{\text{SMOD1}} \times F_{\text{CLK PERIPH}}}{2 \times 2 \times 6^{(1\text{-SPD})} \times 16 \times [256 - (\text{BRL})]}$$

$$(\text{BRL}) = 256 - \frac{2^{\text{SMOD1}} \times F_{\text{CLK PERIPH}}}{2 \times 2 \times 6^{(1\text{-SPD})} \times 16 \times \text{Baud\_Rate}}$$

**Table 60.** SCON Register – SCON Serial Control Register (98h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FE/SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | FE | **Framing Error bit (SMOD0 = 1**)<br>Clear to reset the error state, not cleared by a valid stop bit.<br>Set by hardware when an invalid stop bit is detected.<br>SMOD0 must be set to enable access to the FE bit |
| | SM0 | **Serial port Mode bit 0**<br>Refer to SM1 for serial port mode selection.<br>SMOD0 must be cleared to enable access to the SM0 bit |
| 6 | SM1 | **Serial port Mode bit 1**<br>SM0 SM1 Mode Description Baud Rate<br>0 0 0 Shift Register $F_{CPU\ PERIPH}/6$<br>0 1 1 8-bit UART Variable<br>1 0 2 9-bit UART $F_{CPU\ PERIPH}/32$ or/16<br>1 1 3 9-bit UART Variable |
| 5 | SM2 | **Serial port Mode 2 bit/Multiprocessor Communication Enable bit**<br>Clear to disable multiprocessor communication feature.<br>Set to enable multiprocessor communication feature in mode 2 and 3, and eventually mode 1. This bit should be cleared in mode 0. |
| 4 | REN | **Reception Enable bit**<br>Clear to disable serial reception.<br>Set to enable serial reception. |
| 3 | TB8 | **Transmitter Bit 8/Ninth bit to Transmit in Modes 2 and 3**<br>Clear to transmit a logic 0 in the 9th bit.<br>Set to transmit a logic 1 in the 9th bit. |
| 2 | RB8 | **Receiver Bit 8/Ninth bit received in modes 2 and 3**<br>Cleared by hardware if 9th bit received is a logic 0.<br>Set by hardware if 9th bit received is a logic 1.<br>In mode 1, if SM2 = 0, RB8 is the received stop bit. In mode 0 RB8 is not used. |
| 1 | TI | **Transmit Interrupt flag**<br>Clear to acknowledge interrupt.<br>Set by hardware at the end of the 8th bit time in mode 0 or at the beginning of the stop bit in the other modes. |
| 0 | RI | **Receive Interrupt flag**<br>Clear to acknowledge interrupt.<br>Set by hardware at the end of the 8th bit time in mode 0, see Figure 35. and Figure 36. in the other modes. |

Reset Value = 0000 0000b
Bit addressable

Example of computed value when X2 = 1, SMOD1 = 1, SPD = 1

| Baud Rates | $F_{OSCA}$ = 16.384 MHz | | $F_{OSCA}$ = 24 MHz | |
|---|---|---|---|---|
| | BRL | Error (%) | BRL | Error (%) |
| 115200 | 247 | 1.23 | 243 | 0.16 |
| 57600 | 238 | 1.23 | 230 | 0.16 |
| 38400 | 229 | 1.23 | 217 | 0.16 |
| 28800 | 220 | 1.23 | 204 | 0.16 |
| 19200 | 203 | 0.63 | 178 | 0.16 |
| 9600 | 149 | 0.31 | 100 | 0.16 |
| 4800 | 43 | 1.23 | - | - |

Example of computed value when X2 = 0, SMOD1 = 0, SPD = 0

| | $F_{OSCA}$ = 16.384 MHz | | $F_{OSCA}$ = 24 MHz | |
|---|---|---|---|---|
| Baud Rates | BRL | Error (%) | BRL | Error (%) |
| 4800 | 247 | 1.23 | 243 | 0.16 |
| 2400 | 238 | 1.23 | 230 | 0.16 |
| 1200 | 220 | 1.23 | 202 | 3.55 |
| 600 | 185 | 0.16 | 152 | 0.16 |

The baud rate generator can be used for mode 1 or 3 (refer to Figure 37.), but also for mode 0 for UART, thanks to the bit SRC located in BDRCON register (Table 63.)

## UART Registers

SADEN - Slave Address Mask Register for UART (B9h)

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

Reset Value = 0000 0000b

SADDR - Slave Address Register for UART (A9h)

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

Reset Value = 0000 0000b

SBUF - Serial Buffer Register for UART (99h)

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

Reset Value = XXXX XXXXb

BRL - Baud Rate Reload Register for the internal baud rate generator, UART (9Ah)

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

Reset Value = 0000 0000b

**Table 61.** T2CON Register

T2CON - Timer 2 Control Register (C8h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **TF2** | **EXF2** | **RCLK** | **TCLK** | **EXEN2** | **TR2** | **C/T2#** | **CP/RL2#** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | TF2 | **Timer 2 overflow Flag**<br>Must be cleared by software.<br>Set by hardware on Timer 2 overflow, if RCLK = 0 and TCLK = 0. |
| 6 | EXF2 | **Timer 2 External Flag**<br>Set when a capture or a reload is caused by a negative transition on T2EX pin if EXEN2 = 1.<br>When set, causes the CPU to vector to Timer 2 interrupt routine when Timer 2 interrupt is enabled.<br>Must be cleared by software. EXF2 doesn't cause an interrupt in Up/down counter mode (DCEN = 1) |
| 5 | RCLK | **Receive Clock bit for UART**<br>Cleared to use Timer 1 overflow as receive clock for serial port in mode 1 or 3.<br>Set to use Timer 2 overflow as receive clock for serial port in mode 1 or 3. |
| 4 | TCLK | **Transmit Clock bit for UART**<br>Cleared to use Timer 1 overflow as transmit clock for serial port in mode 1 or 3.<br>Set to use Timer 2 overflow as transmit clock for serial port in mode 1 or 3. |
| 3 | EXEN2 | **Timer 2 External Enable bit**<br>Cleared to ignore events on T2EX pin for Timer 2 operation.<br>Set to cause a capture or reload when a negative transition on T2EX pin is detected, if Timer 2 is not used to clock the serial port. |
| 2 | TR2 | **Timer 2 Run control bit**<br>Cleared to turn off Timer 2.<br>Set to turn on Timer 2. |
| 1 | C/T2# | **Timer/Counter 2 select bit**<br>Cleared for timer operation (input from internal clock system: $F_{CLK\ PERIPH}$).<br>Set for counter operation (input from T2 input pin, falling edge trigger). Must be 0 for clock out mode. |
| 0 | CP/RL2# | **Timer 2 Capture/Reload bit**<br>If RCLK = 1 or TCLK = 1, CP/RL2# is ignored and timer is forced to Auto-reload on Timer 2 overflow.<br>Cleared to Auto-reload on Timer 2 overflows or negative transitions on T2EX pin if EXEN2 = 1.<br>Set to capture on negative transitions on T2EX pin if EXEN2 = 1. |

Reset Value = 0000 0000b
Bit addressable

**Table 62.** PCON Register

PCON - Power Control Register (87h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SMOD1 | SMOD0 | - | POF | GF1 | GF0 | PD | IDL |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | SMOD1 | **Serial port Mode bit 1 for UART**<br>Set to select double baud rate in mode 1, 2 or 3. |
| 6 | SMOD0 | **Serial port Mode bit 0 for UART**<br>Cleared to select SM0 bit in SCON register.<br>Set to select FE bit in SCON register. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | POF | **Power-Off Flag**<br>Cleared to recognize next reset type.<br>Set by hardware when $V_{CC}$ rises from 0 to its nominal voltage. Can also be set by software. |
| 3 | GF1 | **General-purpose Flag**<br>Cleared by user for general-purpose usage.<br>Set by user for general-purpose usage. |
| 2 | GF0 | **General-purpose Flag**<br>Cleared by user for general-purpose usage.<br>Set by user for general-purpose usage. |
| 1 | PD | **Power-down Mode Bit**<br>Cleared by hardware when reset occurs.<br>Set to enter power-down mode. |
| 0 | IDL | **Idle Mode Bit**<br>Cleared by hardware when interrupt or reset occurs.<br>Set to enter idle mode. |

Reset Value = 00X1 0000b
Not bit addressable

Power-off flag reset value will be 1 only after a power on (cold reset). A warm reset doesn't affect the value of this bit.

**Table 63.** BDRCON Register

BDRCON - Baud Rate Control Register (9Bh)

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | BRR | **Baud Rate Run Control bit**<br>Cleared to stop the internal Baud Rate Generator.<br>Set to start the internal Baud Rate Generator. |
| 3 | TBCK | **Transmission Baud rate Generator Selection bit for UART**<br>Cleared to select Timer 1 or Timer 2 for the Baud Rate Generator.<br>Set to select internal Baud Rate Generator. |
| 2 | RBCK | **Reception Baud Rate Generator Selection bit for UART**<br>Cleared to select Timer 1 or Timer 2 for the Baud Rate Generator.<br>Set to select internal Baud Rate Generator. |
| 1 | SPD | **Baud Rate Speed Control bit for UART**<br>Cleared to select the SLOW Baud Rate Generator.<br>Set to select the FAST Baud Rate Generator. |
| 0 | SRC | **Baud Rate Source select bit in Mode 0 for UART**<br>Cleared to select $F_{OSC}/12$ as the Baud Rate Generator ($F_{CLK\ PERIPH}/6$ in X2 mode).<br>Set to select the internal Baud Rate Generator for UARTs in mode 0. |

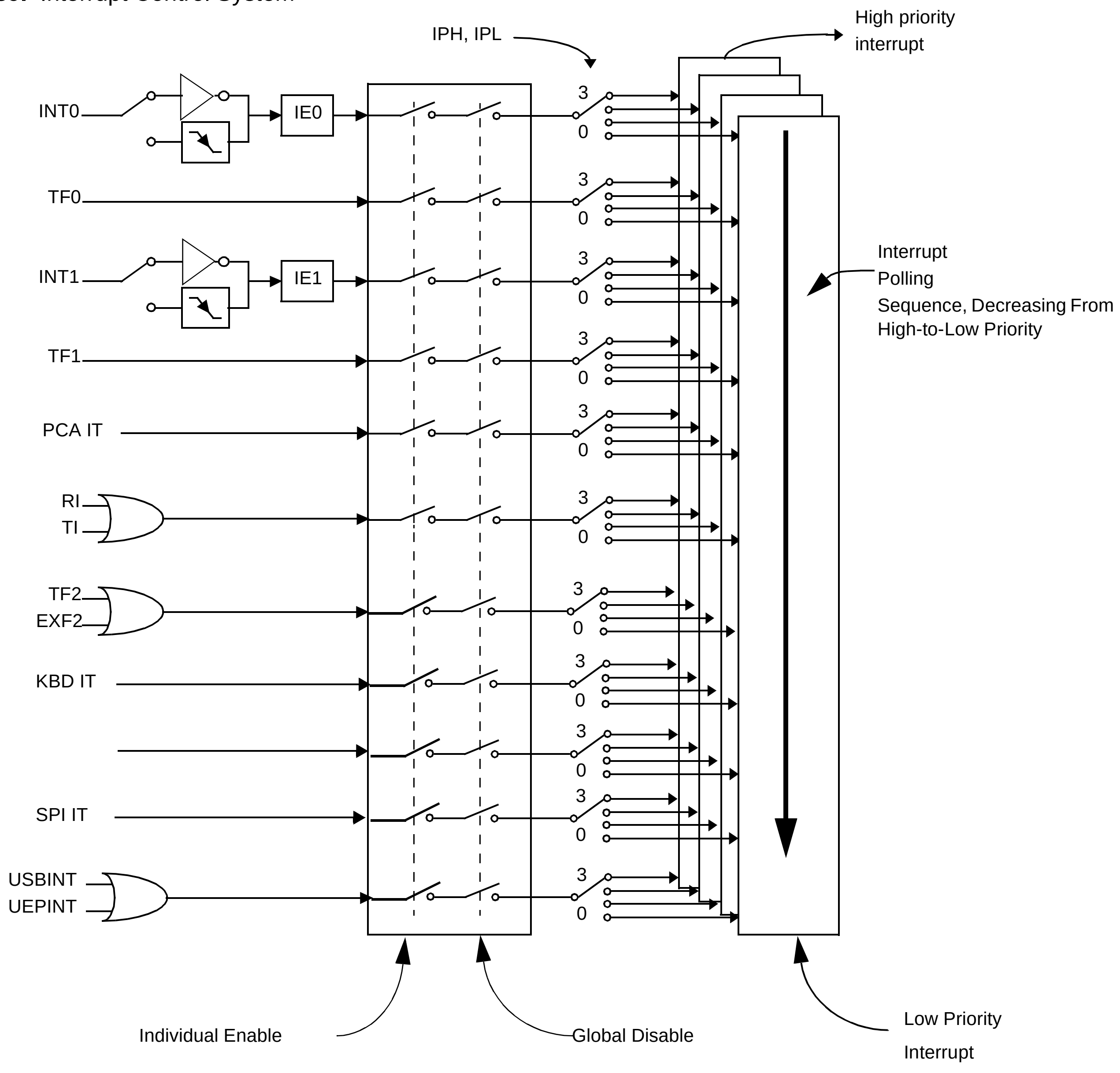
Reset Value = XXX0 0000b
Not bit addressable

## Interrupt System

### Overview

The AT89C5131 has a total of 15 interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (timers 0, 1 and 2), the serial port interrupt, SPI interrupt, Keyboard interrupt, USB interrupt and the PCA global interrupt. These interrupts are shown in Figure 39.

**Figure 39.** Interrupt Control System



Each of the interrupt sources can be individually enabled or disabled by setting or clearing a bit in the Interrupt Enable register (Table 65). This register also contains a global disable bit, which must be cleared to disable all interrupts at once.

Each interrupt source can also be individually programmed to one out of four priority levels by setting or clearing a bit in the Interrupt Priority register (Table 66.) and in the Interrupt Priority High register (Table 67). Table 64. shows the bit values and priority levels associated with each combination.

## Registers

The PCA interrupt vector is located at address 0033H, the SPI interrupt vector is located at address 0043H and Keyboard interrupt vector is located at address 004BH. All other vectors addresses are the same as standard C52 devices.

**Table 64.** Priority Level Bit Values

| IPH.x | IPL.x | Interrupt Level Priority |
|---|---|---|
| 0 | 0 | 0 (Lowest) |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 (Highest) |

A low-priority interrupt can be interrupted by a high priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

If two interrupt requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence.

**Table 65.** IE0 Register

IE0 - Interrupt Enable Register (A8h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EA | EC | ET2 | ES | ET1 | EX1 | ET0 | EX0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | EA | **Enable All interrupt bit**<br>Cleared to disable all interrupts.<br>Set to enable all interrupts. |
| 6 | EC | **PCA interrupt enable bit**<br>Cleared to disable.<br>Set to enable. |
| 5 | ET2 | **Timer 2 overflow interrupt Enable bit**<br>Cleared to disable Timer 2 overflow interrupt.<br>Set to enable Timer 2 overflow interrupt. |
| 4 | ES | **Serial port Enable bit**<br>Cleared to disable serial port interrupt.<br>Set to enable serial port interrupt. |
| 3 | ET1 | **Timer 1 overflow interrupt Enable bit**<br>Cleared to disable Timer 1 overflow interrupt.<br>Set to enable Timer 1 overflow interrupt. |
| 2 | EX1 | **External interrupt 1 Enable bit**<br>Cleared to disable external interrupt 1.<br>Set to enable external interrupt 1. |
| 1 | ET0 | **Timer 0 overflow interrupt Enable bit**<br>Cleared to disable timer 0 overflow interrupt.<br>Set to enable timer 0 overflow interrupt. |
| 0 | EX0 | **External interrupt 0 Enable bit**<br>Cleared to disable external interrupt 0.<br>Set to enable external interrupt 0. |

Reset Value = 0000 0000b
Bit addressable

**Table 66.** IPL0 Register

IPL0 - Interrupt Priority Register (B8h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | PPCL | PT2L | PSL | PT1L | PX1L | PT0L | PX0L |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | PPCL | **PCA interrupt Priority bit**<br>Refer to PPCH for priority level. |
| 5 | PT2L | **Timer 2 overflow interrupt Priority bit**<br>Refer to PT2H for priority level. |
| 4 | PSL | **Serial port Priority bit**<br>Refer to PSH for priority level. |
| 3 | PT1L | **Timer 1 overflow interrupt Priority bit**<br>Refer to PT1H for priority level. |
| 2 | PX1L | **External interrupt 1 Priority bit**<br>Refer to PX1H for priority level. |
| 1 | PT0L | **Timer 0 overflow interrupt Priority bit**<br>Refer to PT0H for priority level. |
| 0 | PX0L | **External interrupt 0 Priority bit**<br>Refer to PX0H for priority level. |

Reset Value = X000 0000b
Bit addressable

**Table 67.** IPH0 Register

IPH0 - Interrupt Priority High Register (B7h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | PPCH | PT2H | PSH | PT1H | PX1H | PT0H | PX0H |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | PPCH | **PCA interrupt Priority high bit.**<br>PPCH PPCL Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 5 | PT2H | **Timer 2 overflow interrupt Priority High bit**<br>PT2H PT2L Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 4 | PSH | **Serial port Priority High bit**<br>PSH PSL Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 3 | PT1H | **Timer 1 overflow interrupt Priority High bit**<br>PT1H PT1L Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 2 | PX1H | **External interrupt 1 Priority High bit**<br>PX1H PX1L Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 1 | PT0H | **Timer 0 overflow interrupt Priority High bit**<br>PT0H PT0L Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 0 | PX0H | **External interrupt 0 Priority High bit**<br>PX0H PX0L Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |

Reset Value = X000 0000b
Not bit addressable

**Table 68.** IE1 Register

IE1 - Interrupt Enable Register (B1h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | EUSB | - | - | - | ESPI | - | EKB |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved** |
| 6 | EUSB | **USB Interrupt Enable bit** |
| 5 | - | **Reserved** |
| 4 | - | **Reserved** |
| 3 | - | **Reserved** |
| 2 | ESPI | **SPI interrupt Enable bit**<br>Cleared to disable SPI interrupt.<br>Set to enable SPI interrupt. |
| 1 | - | **Reserved** |
| 0 | EKB | **Keyboard interrupt Enable bit**<br>Cleared to disable keyboard interrupt.<br>Set to enable keyboard interrupt. |

Reset Value = XXXX X000b
Bit addressable

**Table 69.** IPL1 Register

IPL1 - Interrupt Priority Register (B2h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | PUSBL | - | - | - | PSPIL | | PKBDL |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | PUSBL | **USB Interrupt Priority bit**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 2 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 1 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 0 | PKBL | **Keyboard Interrupt Priority bit**<br>Refer to KBDH for priority level. |

Reset Value = XXXX X000b
Bit addressable

**Table 70.** IPH1 Register

IPH1 - Interrupt Priority High Register (B3h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | PSPIH | - | PKBH |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | PUSBH | **USB Interrupt Priority High bit**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 2 | PSPIH | **SPI Interrupt Priority High bit**<br>SPIH SPIL Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |
| 1 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 0 | PKBH | **Keyboard Interrupt Priority High bit**<br>KBDH KBDL Priority Level<br>0 0 Lowest<br>0 1<br>1 0<br>1 1 Highest |

Reset Value = XXXX X000b
Not bit addressable

## Interrupt Sources and Vector Addresses

**Table 71.** Vector Table

| Number | Polling Priority | Interrupt Source | Interrupt Request | Vector Address |
|---|---|---|---|---|
| 0 | 0 | Reset | | 0000h |
| 1 | 1 | INT0 | IE0 | 0003h |
| 2 | 2 | Timer 0 | TF0 | 000Bh |
| 3 | 3 | INT1 | IE1 | 0013h |
| 4 | 4 | Timer 1 | IF1 | 001Bh |
| 5 | 6 | UART | RI+TI | 0023h |
| 6 | 7 | Timer 2 | TF2+EXF2 | 002Bh |
| 7 | 5 | PCA | CF + CCFn (n = 0-4) | 0033h |
| 8 | 8 | Keyboard | KBDIT | 003Bh |
| 9 | 9 | | | 0043h |
| 10 | 10 | SPI | SPIIT | 004Bh |
| 11 | 11 | | | 0053h |
| 12 | 12 | | | 005Bh |
| 13 | 13 | | | 0063h |
| 14 | 14 | USB | UEPINT + USBINT | 006Bh |
| 15 | 15 | | | 0073 |

# Keyboard Interface

## Introduction

The AT89C5131 implements a keyboard interface allowing the connection of a 8 x n matrix keyboard. It is based on 8 inputs with programmable interrupt capability on both high or low level. These inputs are available as an alternate function of P1 and allow to exit from idle and power down modes.

## Description

The keyboard interface communicates with the C51 core through 3 special function registers: KBLS, the Keyboard Level Selection register (Table 74), KBE, The Keyboard interrupt Enable register (Table 73), and KBF, the Keyboard Flag register (Table 72).

### Interrupt

The keyboard inputs are considered as 8 independent interrupt sources sharing the same interrupt vector. An interrupt enable bit (KBD in IE1) allows global enable or disable of the keyboard interrupt (see Figure 40). As detailed in Figure 41 each keyboard input has the capability to detect a programmable level according to KBLS.x bit value. Level detection is then reported in interrupt flags KBF.x that can be masked by software using KBE.x bits.

This structure allow keyboard arrangement from 1 by n to 8 by n matrix and allow usage of P1 inputs for other purpose.

**Figure 40.** Keyboard Interface Block Diagram

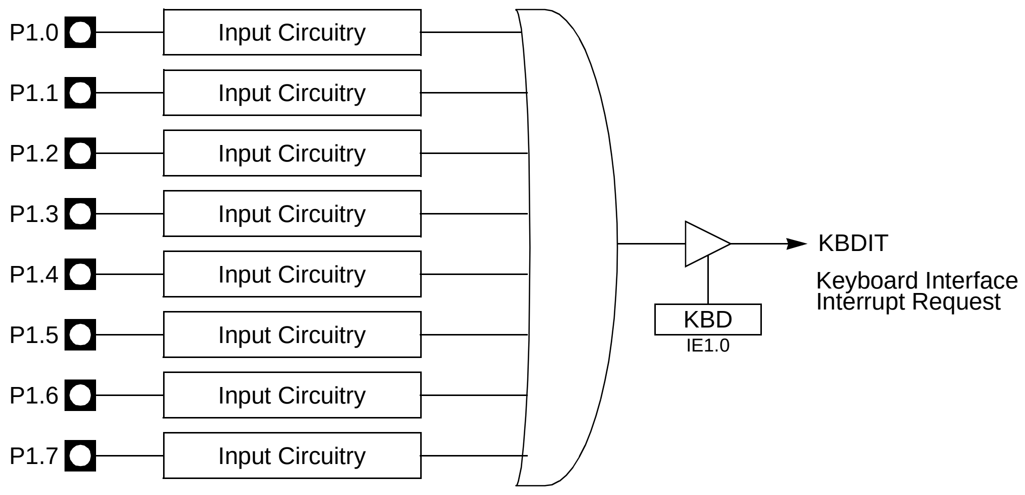
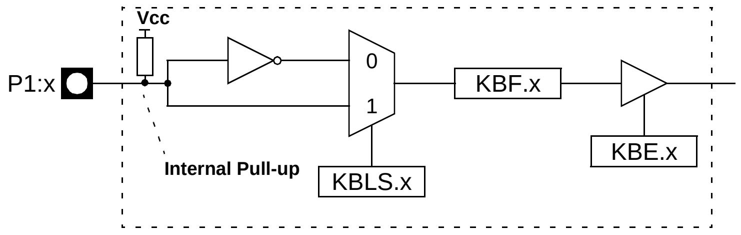


**Figure 41.** Keyboard Input Circuitry



## Power Reduction Mode

P1 inputs allow exit from idle and power down modes as detailed in section “Power-down Mode”.

## Registers

**Table 72. KBF Register**

KBF **-** Keyboard Flag Register (9Eh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **KBF7** | **KBF6** | **KBF5** | **KBF4** | **KBF3** | **KBF2** | **KBF1** | **KBF0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | KBF7 | **Keyboard line 7 flag**<br>Set by hardware when the Port line 7 detects a programmed level. It generates a Keyboard interrupt request if the KBKBIE.7 bit in KBIE register is set.<br>Must be cleared by software. |
| 6 | KBF6 | **Keyboard line 6 flag**<br>Set by hardware when the Port line 6 detects a programmed level. It generates a Keyboard interrupt request if the KBIE.6 bit in KBIE register is set.<br>Must be cleared by software. |
| 5 | KBF5 | **Keyboard line 5 flag**<br>Set by hardware when the Port line 5 detects a programmed level. It generates a Keyboard interrupt request if the KBIE.5 bit in KBIE register is set.<br>Must be cleared by software. |
| 4 | KBF4 | **Keyboard line 4 flag**<br>Set by hardware when the Port line 4 detects a programmed level. It generates a Keyboard interrupt request if the KBIE.4 bit in KBIE register is set.<br>Must be cleared by software. |
| 3 | KBF3 | **Keyboard line 3 flag**<br>Set by hardware when the Port line 3 detects a programmed level. It generates a Keyboard interrupt request if the KBIE.3 bit in KBIE register is set.<br>Must be cleared by software. |
| 2 | KBF2 | **Keyboard line 2 flag**<br>Set by hardware when the Port line 2 detects a programmed level. It generates a Keyboard interrupt request if the KBIE.2 bit in KBIE register is set.<br>Must be cleared by software. |
| 1 | KBF1 | **Keyboard line 1 flag**<br>Set by hardware when the Port line 1 detects a programmed level. It generates a Keyboard interrupt request if the KBIE.1 bit in KBIE register is set.<br>Must be cleared by software. |
| 0 | KBF0 | **Keyboard line 0 flag**<br>Set by hardware when the Port line 0 detects a programmed level. It generates a Keyboard interrupt request if the KBIE.0 bit in KBIE register is set.<br>Must be cleared by software. |

Reset Value = 0000 0000b

**Table 73.** KBE Register

KBE **-** Keyboard Input Enable Register (9Dh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| KBE7 | KBE6 | KBE5 | KBE4 | KBE3 | KBE2 | KBE1 | KBE0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | KBE7 | **Keyboard line 7 Enable bit**<br>Cleared to enable standard I/O pin.<br>Set to enable KBF.7 bit in KBF register to generate an interrupt request. |
| 6 | KBE6 | **Keyboard line 6 Enable bit**<br>Cleared to enable standard I/O pin.<br>Set to enable KBF.6 bit in KBF register to generate an interrupt request. |
| 5 | KBE5 | **Keyboard line 5 Enable bit**<br>Cleared to enable standard I/O pin.<br>Set to enable KBF.5 bit in KBF register to generate an interrupt request. |
| 4 | KBE4 | **Keyboard line 4 Enable bit**<br>Cleared to enable standard I/O pin.<br>Set to enable KBF.4 bit in KBF register to generate an interrupt request. |
| 3 | KBE3 | **Keyboard line 3 Enable bit**<br>Cleared to enable standard I/O pin.<br>Set to enable KBF.3 bit in KBF register to generate an interrupt request. |
| 2 | KBE2 | **Keyboard line 2 Enable bit**<br>Cleared to enable standard I/O pin.<br>Set to enable KBF.2 bit in KBF register to generate an interrupt request. |
| 1 | KBE1 | **Keyboard line 1 Enable bit**<br>Cleared to enable standard I/O pin.<br>Set to enable KBF.1 bit in KBF register to generate an interrupt request. |
| 0 | KBE0 | **Keyboard line 0 Enable bit**<br>Cleared to enable standard I/O pin.<br>Set to enable KBF.0 bit in KBF register to generate an interrupt request. |

Reset Value = 0000 0000b

**Table 74.** KBLS Register

KBLS-Keyboard Level Selector Register (9Ch)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| KBLS7 | KBLS6 | KBLS5 | KBLS4 | KBLS3 | KBLS2 | KBLS1 | KBLS0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | KBLS7 | **Keyboard line 7 Level Selection bit**<br>Cleared to enable a low level detection on Port line 7.<br>Set to enable a high level detection on Port line 7. |
| 6 | KBLS6 | **Keyboard line 6 Level Selection bit**<br>Cleared to enable a low level detection on Port line 6.<br>Set to enable a high level detection on Port line 6. |
| 5 | KBLS5 | **Keyboard line 5 Level Selection bit**<br>Cleared to enable a low level detection on Port line 5.<br>Set to enable a high level detection on Port line 5. |
| 4 | KBLS4 | **Keyboard line 4 Level Selection bit**<br>Cleared to enable a low level detection on Port line 4.<br>Set to enable a high level detection on Port line 4. |
| 3 | KBLS3 | **Keyboard line 3 Level Selection bit**<br>Cleared to enable a low level detection on Port line 3.<br>Set to enable a high level detection on Port line 3. |
| 2 | KBLS2 | **Keyboard line 2 Level Selection bit**<br>Cleared to enable a low level detection on Port line 2.<br>Set to enable a high level detection on Port line 2. |
| 1 | KBLS1 | **Keyboard line 1 Level Selection bit**<br>Cleared to enable a low level detection on Port line 1.<br>Set to enable a high level detection on Port line 1. |
| 0 | KBLS0 | **Keyboard line 0 Level Selection bit**<br>Cleared to enable a low level detection on Port line 0.<br>Set to enable a high level detection on Port line 0. |

Reset Value = 0000 0000b

## Programmable LED

AT89C5131 have up to 4 programmable LED current sources, configured by the register LEDCON.

**Table 75.** LEDCON Register

LEDCON (S:F1h)
LED Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| LED3 | | LED2 | | LED1 | | LED0 | |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:6 | LED3 | <u>**Port**</u> <u>**LED0**</u> <u>**Configuration**</u><br>0 0 Standard C51 Port<br>0 1 2 mA current source when P3.7 is O<br>1 0 4 mA current source when P3.7 is O<br>1 1 10 mA current source when P3.7 is O |
| 5:4 | LED2 | <u>**Port**</u> <u>**LED0**</u> <u>**Configuration**</u><br>0 0 Standard C51 Port<br>0 1 2 mA current source when P3.6 is O<br>1 0 4 mA current source when P3.6 is O<br>1 1 10 mA current source when P3.6 is O |
| 3:2 | LED1 | <u>**Port**</u> <u>**LED0**</u> <u>**Configuration**</u><br>0 0 Standard C51 Port<br>0 1 2 mA current source when P3.5 is O<br>1 0 4 mA current source when P3.5 is O<br>1 1 10 mA current source when P3.5 is O |
| 1:0 | LED0 | <u>**Port**</u> <u>**LED0**</u> <u>**Configuration**</u><br>0 0 Standard C51 Port<br>0 1 2 mA current source when P3.3 is O<br>1 0 4 mA current source when P3.3 is O<br>1 1 10 mA current source when P3.3 is O |

Reset Value = 00h

## Serial Peripheral Interface (SPI)

The Serial Peripheral Interface module (SPI) allows full-duplex, synchronous, serial communication between the MCU and peripheral devices, including other MCUs.

### Features

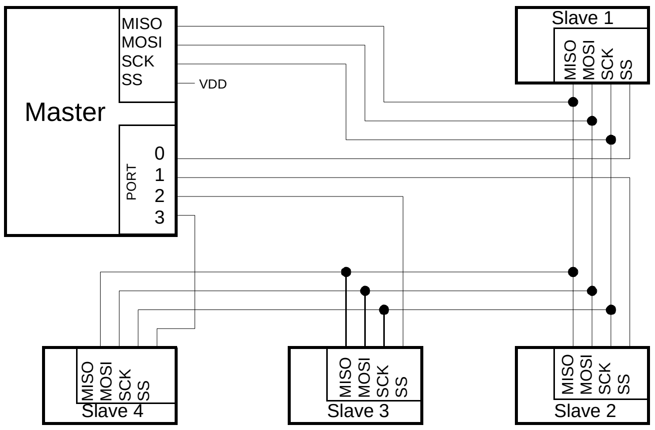
Features of the SPI module include the following:

- Full-duplex, three-wire synchronous transfers
- Master or Slave operation
- Eight programmable Master clock rates
- Serial clock with programmable polarity and phase
- Master mode fault error flag with MCU interrupt capability
- Write collision flag protection

### Signal Description

Figure 42 shows a typical SPI bus configuration using one Master controller and many Slave peripherals. The bus is made of three wires connecting all the devices:

**Figure 42.** SPI Master/Slaves Interconnection



The Master device selects the individual Slave devices by using four pins of a parallel port to control the four $\overline{SS}$ pins of the Slave devices.

#### Master Output Slave Input (MOSI)

This 1-bit signal is directly connected between the Master Device and a Slave Device. The MOSI line is used to transfer data in series from the Master to the Slave. Therefore, it is an output signal from the Master, and an input signal to a Slave. A byte (8-bit word) is transmitted most significant bit (MSB) first, least significant bit (LSB) last.

#### Master Input Slave Output (MISO)

This 1-bit signal is directly connected between the Slave Device and a Master Device. The MISO line is used to transfer data in series from the Slave to the Master. Therefore, it is an output signal from the Slave, and an input signal to the Master. A byte (8-bit word) is transmitted most significant bit (MSB) first, least significant bit (LSB) last.

#### SPI Serial Clock (SCK)

This signal is used to synchronize the data movement both in and out the devices through their MOSI and MISO lines. It is driven by the Master for eight clock cycles which allows to exchange one byte on the serial lines.

#### Slave Select ($\overline{SS}$)

Each Slave peripheral is selected by one Slave Select pin ($\overline{SS}$). This signal must stay low for any message for a Slave. It is obvious that only one Master ($\overline{SS}$ high level) can drive the network. The Master may select each Slave device by software through port pins (Figure 42). To prevent bus conflicts on the MISO line, only one slave should be selected at a time by the Master for a transmission.

In a Master configuration, the $\overline{SS}$ line can be used in conjunction with the MODF flag in the SPI Status register (SPSTA) to prevent multiple masters from driving MOSI and SCK (see Section “Error Conditions”, page 109).

A high level on the $\overline{SS}$ pin puts the MISO line of a Slave SPI in a high-impedance state.

The $\overline{SS}$ pin could be used as a general-purpose if the following conditions are met:

- The device is configured as a Master and the SSDIS control bit in SPCON is set. This kind of configuration can be found when only one Master is driving the network and there is no way that the $\overline{SS}$ pin could be pulled low. Therefore, the MODF flag in the SPSTA will never be set(1).
- The Device is configured as a Slave with CPHA and SSDIS control bits set(2) This kind of configuration can happen when the system comprises one Master and one Slave only. Therefore, the device should always be selected and there is no reason that the Master uses the $\overline{SS}$ pin to select the communicating Slave device.

Notes: 1. Clearing SSDIS control bit does not clear MODF.
2. Special care should be taken not to set SSDIS control bit when CPHA =’0’ because in this mode, the $\overline{SS}$ is used to start the transmission.

## Baud Rate

In Master mode, the baud rate can be selected from a baud rate generator which is controlled by three bits in the SPCON register: SPR2, SPR1 and SPR0. The Master clock is chosen from one of seven clock rates resulting from the division of the internal clock by 2, 4, 8, 16, 32, 64 or 128.

Table 76 gives the different clock rates selected by SPR2:SPR1:SPR0:

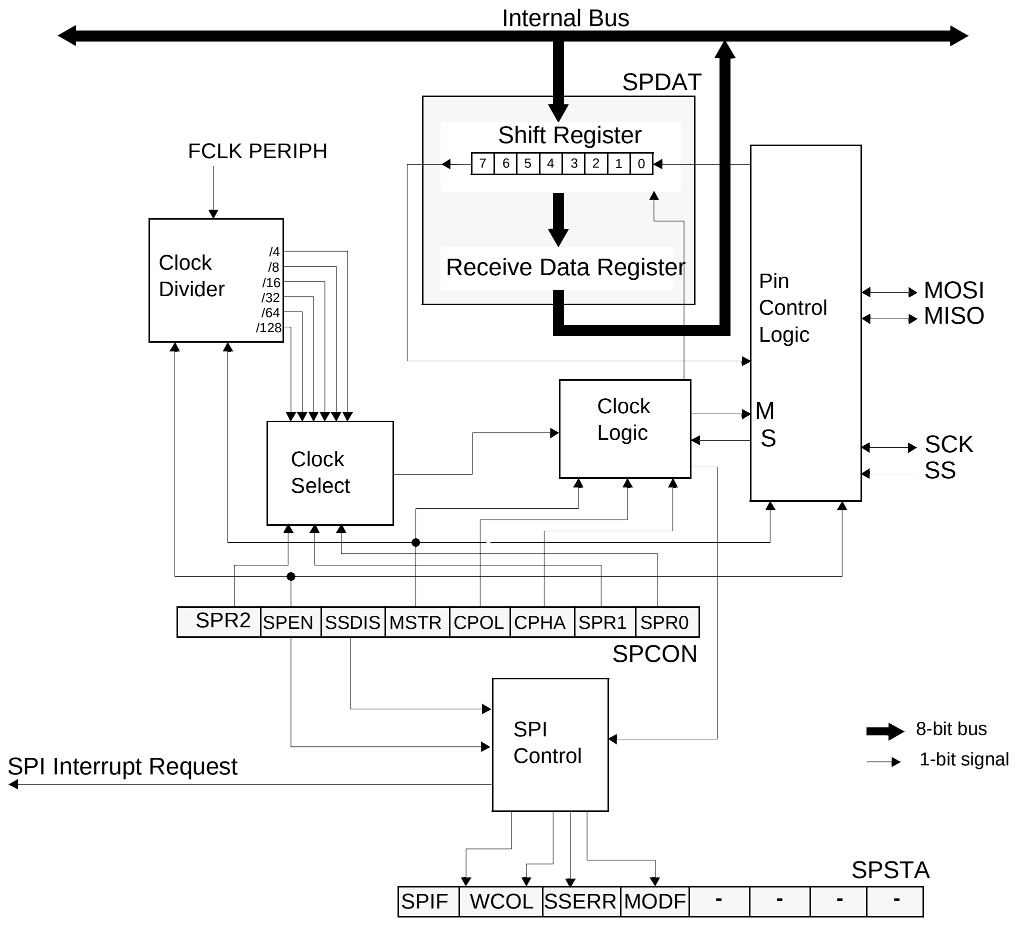
**Table 76.** SPI Master Baud Rate Selection

| SPR2 | SPR1 | SPR0 | Clock Rate | Baud Rate Divisor (BD) |
|---|---|---|---|---|
| 0 | 0 | 0 | Don’t Use | No BRG |
| 0 | 0 | 1 | $F_{CLK\ PERIPH}/4$ | 4 |
| 0 | 1 | 0 | $F_{CLK\ PERIPH}/8$ | 8 |
| 0 | 1 | 1 | $F_{CLK\ PERIPH}/16$ | 16 |
| 1 | 0 | 0 | $F_{CLK\ PERIPH}/32$ | 32 |
| 1 | 0 | 1 | $F_{CLK\ PERIPH}/64$ | 64 |
| 1 | 1 | 0 | $F_{CLK\ PERIPH}/128$ | 128 |
| 1 | 1 | 1 | Don’t Use | No BRG |

## Functional Description

Figure 43 shows a detailed structure of the SPI module.

**Figure 43.** SPI Module Block Diagram



## Operating Modes

The Serial Peripheral Interface can be configured as one of the two modes: Master mode or Slave mode. The configuration and initialization of the SPI module is made through one register:

- The Serial Peripheral CONtrol register (SPCON)

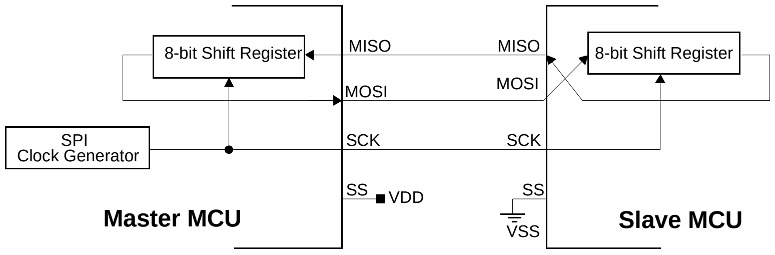
Once the SPI is configured, the data exchange is made using:

- SPCON
- The Serial Peripheral STAtus register (SPSTA)
- The Serial Peripheral DATa register (SPDAT)

During an SPI transmission, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line (SCK) synchronizes shifting and sampling on the two serial data lines (MOSI and MISO). A Slave Select line ($\overline{SS}$) allows individual selection of a Slave SPI device; Slave devices that are not selected do not interfere with SPI bus activities.

When the Master device transmits data to the Slave device via the MOSI line, the Slave device responds by sending data to the Master device via the MISO line. This implies full-duplex transmission with both data out and data in synchronized with the same clock (Figure 44).

**Figure 44.** Full-duplex Master/Slave Interconnection



*Master Mode*

The SPI operates in Master mode when the Master bit, MSTR [(1)], in the SPCON register is set. Only one Master SPI device can initiate transmissions. Software begins the transmission from a Master SPI module by writing to the Serial Peripheral Data Register (SPDAT). If the shift register is empty, the byte is immediately transferred to the shift register. The byte begins shifting out on MOSI pin under the control of the serial clock, SCK. Simultaneously, another byte shifts in from the Slave on the Master's MISO pin. The transmission ends when the Serial Peripheral transfer data flag, SPIF, in SPSTA becomes set. At the same time that SPIF becomes set, the received byte from the Slave is transferred to the receive data register in SPDAT. Software clears SPIF by reading the Serial Peripheral Status register (SPSTA) with the SPIF bit set, and then reading the SPDAT.

*Slave Mode*

The SPI operates in Slave mode when the Master bit, MSTR [(2)], in the SPCON register is cleared. Before a data transmission occurs, the Slave Select pin, $\overline{SS}$, of the Slave device must be set to'0'. $\overline{SS}$ must remain low until the transmission is complete.

In a Slave SPI module, data enters the shift register under the control of the SCK from the Master SPI module. After a byte enters the shift register, it is immediately transferred to the receive data register in SPDAT, and the SPIF bit is set. To prevent an overflow condition, Slave software must then read the SPDAT before another byte enters the shift register [(3)]. A Slave SPI must complete the write to the SPDAT (shift register) at least one bus cycle before the Master SPI starts a transmission. If the write to the data register is late, the SPI transmits the data already in the shift register from the previous transmission.

**Transmission Formats**

Software can select any of four combinations of serial clock (SCK) phase and polarity using two bits in the SPCON: the Clock POLarity (CPOL [(4)]) and the Clock PHAse (CPHA[4]). CPOL defines the default SCK line level in idle state. It has no significant effect on the transmission format. CPHA defines the edges on which the input data are sampled and the edges on which the output data are shifted (Figure 45 and Figure 46). The clock phase and polarity should be identical for the Master SPI device and the communicating Slave device.

1. The SPI module should be configured as a Master before it is enabled (SPEN set). Also the Master SPI should be configured before the Slave SPI.
2. The SPI module should be configured as a Slave before it is enabled (SPEN set).
3. The maximum frequency of the SCK for an SPI configured as a Slave is the bus clock speed.
4. Before writing to the CPOL and CPHA bits, the SPI should be disabled (SPEN ='0').

**Figure 45.** Data Transmission Format (CPHA = 0)

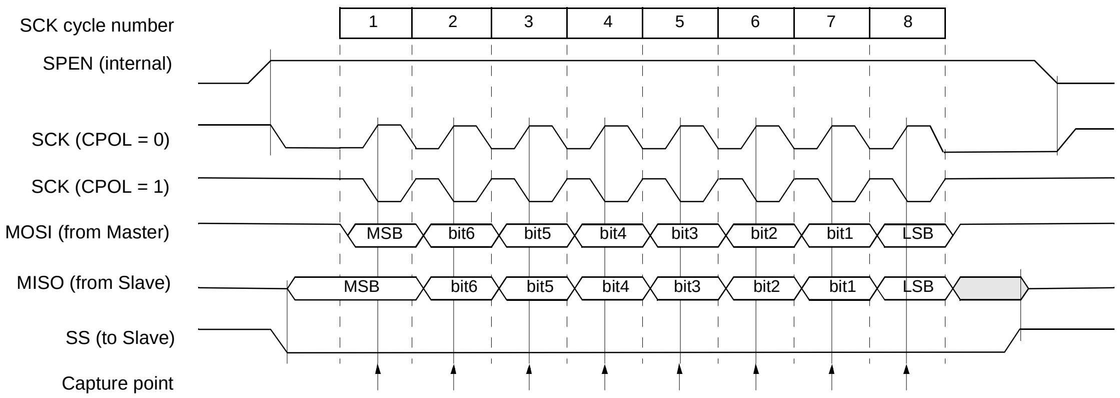


**Figure 46.** Data Transmission Format (CPHA = 1)

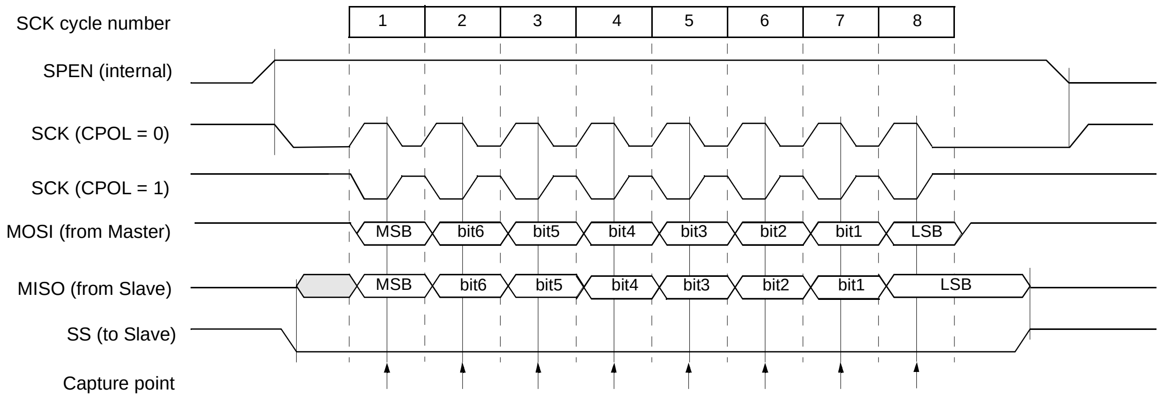
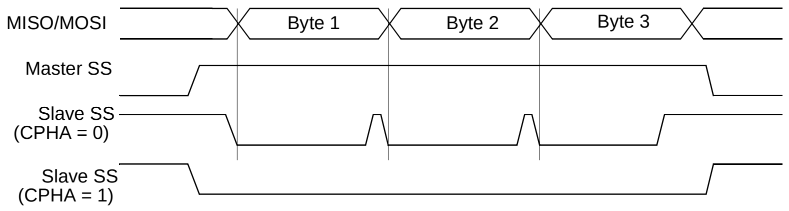


**Figure 47.** CPHA/$\overline{\text{SS}}$ Timing



As shown in Figure 46, the first SCK edge is the MSB capture strobe. Therefore the Slave must begin driving its data before the first SCK edge, and a falling edge on the $\overline{\text{SS}}$ pin is used to start the transmission. The $\overline{\text{SS}}$ pin must be toggled high and then low between each byte transmitted (Figure 43).

Figure 47 shows an SPI transmission in which CPHA is'1'. In this case, the Master begins driving its MOSI pin on the first SCK edge. Therefore the Slave uses the first SCK edge as a start transmission signal. The $\overline{\text{SS}}$ pin can remain low between transmissions (Figure 42). This format may be preferable in systems having only one Master and only one Slave driving the MISO data line.

## Error Conditions

The following flags in the SPSTA signal SPI error conditions:

### *Mode Fault (MODF)*

Mode Fault error in Master mode SPI indicates that the level on the Slave Select ($\overline{SS}$) pin is inconsistent with the actual mode of the device. MODF is set to warn that there may have a multi-master conflict for system control. In this case, the SPI system is affected in the following ways:

- An SPI receiver/error CPU interrupt request is generated,
- The SPEN bit in SPCON is cleared. This disable the SPI,
- The MSTR bit in SPCON is cleared

When $\overline{SS}$ $\overline{DI}$Sable (SSDIS) bit in the SPCON register is cleared, the MODF flag is set when the $\overline{SS}$ signal becomes “0”.

However, as stated before, for a system with one Master, if the $\overline{SS}$ pin of the Master device is pulled low, there is no way that another Master attempt to drive the network. In this case, to prevent the MODF flag from being set, software can set the SSDIS bit in the SPCON register and therefore making the $\overline{SS}$ pin as a general-purpose I/O pin.

Clearing the MODF bit is accomplished by a read of SPSTA register with MODF bit set, followed by a write to the SPCON register. SPEN Control bit may be restored to its original set state after the MODF bit has been cleared.

### *Write Collision (WCOL)*

A Write Collision (WCOL) flag in the SPSTA is set when a write to the SPDAT register is done during a transmit sequence.

WCOL does not cause an interruption, and the transfer continues uninterrupted.

Clearing the WCOL bit is done through a software sequence of an access to SPSTA and an access to SPDAT.

### *Overrun Condition*

An overrun condition occurs when the Master device tries to send several data bytes and the Slave devise has not cleared the SPIF bit issuing from the previous data byte transmitted. In this case, the receiver buffer contains the byte sent after the SPIF bit was last cleared. A read of the SPDAT returns this byte. All others bytes are lost.

This condition is not detected by the SPI peripheral.

## Interrupts

Two SPI status flags can generate a CPU interrupt requests:

**Table 77.** SPI Interrupts

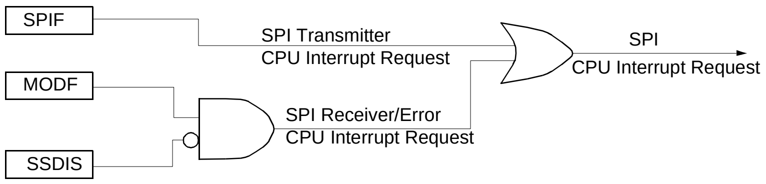
| Flag | Request |
|---|---|
| SPIF (SP Data Transfer) | SPI Transmitter Interrupt request |
| MODF (Mode Fault) | SPI Receiver/Error Interrupt Request (if SSDIS = “0”) |

Serial Peripheral data transfer flag, SPIF: This bit is set by hardware when a transfer has been completed. SPIF bit generates transmitter CPU interrupt requests.

Mode Fault flag, MODF: This bit becomes set to indicate that the level on the SS is inconsistent with the mode of the SPI. MODF with SSDIS reset, generates receiver/error CPU interrupt requests.

Figure 48 gives a logical view of the above statements.

**Figure 48.** SPI Interrupt Requests Generation



## Registers

There are three registers in the module that provide control, status and data storage functions. These registers are describes in the following paragraphs.

### *Serial Peripheral Control Register (SPCON)*

- The Serial Peripheral Control Register does the following:
  - Selects one of the Master clock rates
  - Configure the SPI module as Master or Slave
  - Selects serial clock polarity and phase
  - Enables the SPI module
  - Frees the SS pin for a general-purpose

Table 78 describes this register and explains the use of each bit.

**Table 78.** SPCON Register
SPCON - Serial Peripheral Control Register (0C3H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **SPR2** | **SPEN** | **SSDIS** | **MSTR** | **CPOL** | **CPHA** | **SPR1** | **SPR0** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | SPR2 | **Serial Peripheral Rate 2**<br>Bit with SPR1 and SPR0 define the clock rate. |
| 6 | SPEN | **Serial Peripheral Enable**<br>Cleared to disable the SPI interface.<br>Set to enable the SPI interface. |
| 5 | SSDIS | **$\overline{\text{SS}}$ Disable**<br>Cleared to enable $\overline{\text{SS}}$ in both Master and Slave modes.<br>Set to disable $\overline{\text{SS}}$ in both Master and Slave modes. In Slave mode, this bit has no effect if CPHA = “0”. |
| 5 | MSTR | **Serial Peripheral Master**<br>Cleared to configure the SPI as a Slave.<br>Set to configure the SPI as a Master. |
| 4 | CPOL | **Clock Polarity**<br>Cleared to have the SCK set to “0” in idle state.<br>Set to have the SCK set to “1” in idle low. |
| 3 | CPHA | **Clock Phase**<br>Cleared to have the data sampled when the SCK leaves the idle state (see CPOL).<br>Set to have the data sampled when the SCK returns to idle state (see CPOL). |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 2 | SPR1 | **SPR2** **SPR1** **SPR0** **Serial Peripheral Rate**<br>0 0 0 Invalid<br>0 0 1 $F_{CLK\ PERIPH}/4$<br>0 1 0 $F_{CLK\ PERIPH}/8$ |
| 1 | SPR0 | 0 1 1 $F_{CLK\ PERIPH}/16$<br>1 0 0 $F_{CLK\ PERIPH}/32$<br>1 0 1 $F_{CLK\ PERIPH}/64$<br>1 1 0 $F_{CLK\ PERIPH}/128$<br>1 1 1 Invalid |

Reset Value = 0001 0100b

Not bit addressable

*Serial Peripheral Status Register (SPSTA)*

The Serial Peripheral Status Register contains flags to signal the following conditions:

- Data transfer complete
- Write collision
- Inconsistent logic level on $\overline{SS}$ pin (mode fault error)

Table 79 describes the SPSTA register and explains the use of every bit in the register.

**Table 79.** SPSTA Register

SPSTA - Serial Peripheral Status and Control register (0C4H)

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | SPIF | **Serial Peripheral data transfer flag**<br>Cleared by hardware to indicate data transfer is in progress or has been approved by a clearing sequence.<br>Set by hardware to indicate that the data transfer has been completed. |
| 6 | WCOL | **Write Collision flag**<br>Cleared by hardware to indicate that no collision has occurred or has been approved by a clearing sequence.<br>Set by hardware to indicate that a collision has been detected. |
| 5 | SSERR | **Synchronous Serial Slave Error flag**<br>Set by hardware when $\overline{SS}$ is de-<br>asserted before the end of a received data.<br>Cleared by disabling the SPI (clearing SPEN bit in SPCON). |
| 4 | MODF | **Mode Fault**<br>Cleared by hardware to indicate that the $\overline{SS}$ pin is at appropriate logic level, or has been approved by a clearing sequence.<br>Set by hardware to indicate that the $\overline{SS}$ pin is at inappropriate logic level. |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 2 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit |
| 1 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 0 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |

Reset Value = 00X0 XXXXb
Not Bit addressable

*Serial Peripheral Data Register (SPDAT)*

The Serial Peripheral Data Register (Table 80) is a read/write buffer for the receive data register. A write to SPDAT places data directly into the shift register. No transmit buffer is available in this model.

A Read of the SPDAT returns the value located in the receive buffer and not the content of the shift register.

**Table 80.** SPDAT Register

SPDAT - Serial Peripheral Data Register (0C5H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |

Reset Value = Indeterminate

R7:R0: Receive data bits

SPCON, SPSTA and SPDAT registers may be read and written at any time while there is no on-going exchange. However, special care should be taken when writing to them while a transmission is on-going:

- Do not change SPR2, SPR1 and SPR0
- Do not change CPHA and CPOL
- Do not change MSTR
- Clearing SPEN would immediately disable the peripheral
- Writing to the SPDAT will cause an overflow

# USB Controller

## Introduction

The AT89C5131 implements a USB device controller supporting full speed data transfer. In addition to the default control endpoint 0, it provides 6 other endpoints, which can be configured in control, bulk, interrupt or isochronous modes:

- Endpoint 0:32-byte FIFO, default control endpoint
- Endpoint 1, 2, 3: 32-byte FIFO
- Endpoint 4, 5: 2 x 64-byte Ping-pong FIFO
- Endpoint 6: 2 x 512-byte Ping-pong FIFO

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

- 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 Human Interface Device Class, Version 1.1 - April 7, 1999
- USB Device Firmware Upgrade Class, Revision 1.0 - May 13, 1999

### USB Mass Storage Classes

#### *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 configuration adheres to these requirements:

- Endpoint 0: 8 bytes, Control In-Out
- Endpoint 4: 64 bytes, Bulk-out
- Endpoint 5: 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 configuration adheres to these requirements:

- Endpoint 0: 8 bytes, Control In-Out
- Endpoint 4: 64 bytes, Bulk-out
- Endpoint 5: 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 AT89C5131. This allows the implementation of 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-exists with the usual functions of the device, which may be USB Mass Storage for the AT89C5131. 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 wMaxPacketSize in the DFU configuration is 32 bytes, which is the size of the FIFO implemented for endpoint 0.

## Description

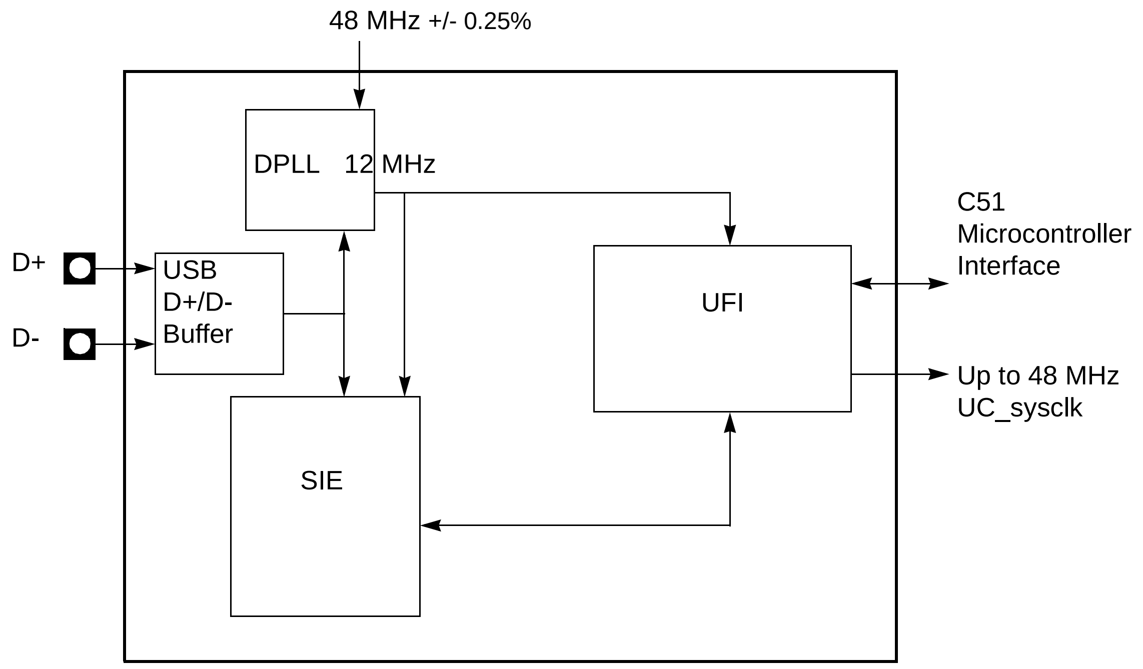
The USB device controller provides the hardware that the AT89C5131 needs to interface a USB link to a data flow stored in a double port memory (DPRAM).

The USB controller requires a 48 MHz ±0.25% reference clock, which is the output of the AT89C5131 PLL (see Section "PLL", page 20) divided by a clock prescaler. This clock is used to generate a 12 MHz Full-speed bit clock from the received USB differential data and to transmit data according to full speed USB device tolerance. Clock recovery is done by a Digital Phase Locked Loop (DPLL) block, which is compliant with the jitter specification of the USB bus.

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) realizes the interface between the data flow and the Dual Port RAM.

**Figure 49.** USB Device Controller Block Diagram

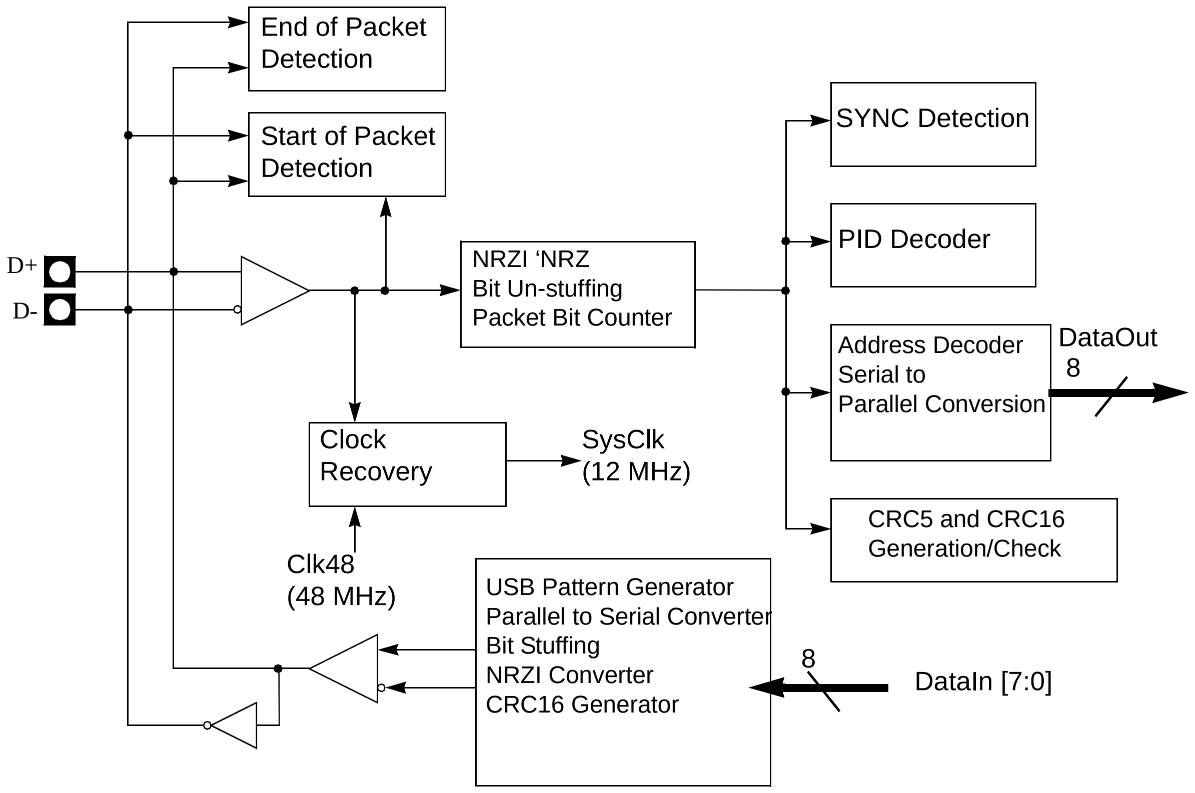


## Serial Interface Engine (SIE)

The SIE performs the following functions:

- NRZI data encoding and decoding.
- Bit stuffing and un-stuffing.
- CRC generation and checking.
- Handshakes.
- TOKEN type identifying.
- Address checking.
- Clock generation (via DPLL).

**Figure 50.** SIE Block Diagram


End of Packet Detection
Start of Packet Detection
D+
D-
NRZI 'NRZ Bit Un-stuffing Packet Bit Counter
SYNC Detection
PID Decoder
Address Decoder Serial to Parallel Conversion
DataOut
8
Clock Recovery
SysClk (12 MHz)
Clk48 (48 MHz)
CRC5 and CRC16 Generation/Check
USB Pattern Generator Parallel to Serial Converter Bit Stuffing NRZI Converter CRC16 Generator
8
DataIn [7:0]

**Function Interface Unit (FIU)**

The Function Interface Unit provides the interface between the AT89C5131 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 51.** UFI Block Diagram

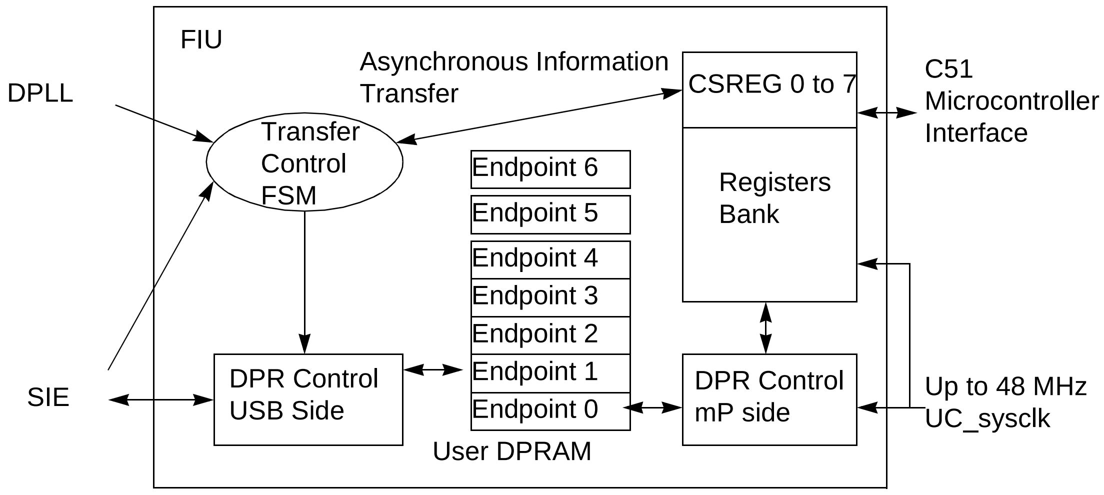
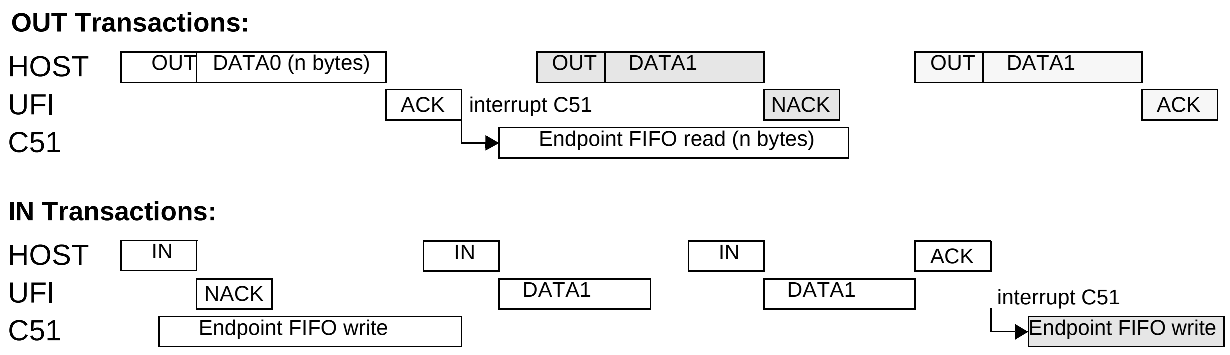


**Figure 52.** Minimum Intervention from the USB Device Firmware



## Configuration

### General Configuration

- USB controller enable

  Before any USB transaction, the 48 MHz required by the USB controller must be correctly generated (See “Clock Controller” on page 19).

  The USB controller will be then enabled by setting the EUSB bit in the USBCON register.
- Set address

  After a Reset or a USB reset, the software has to set the FEN (Function Enable) bit in the USBADDR register. This action will allow the USB controller to answer to the requests sent at the address 0.

  When a SET_ADDRESS request has been received, the USB controller must only answer to the address defined by the request. The new address will be stored in the USBADDR register. The FEN bit and the FADDEN bit in the USBCON register will be set to allow the USB controller to answer only to requests sent at the new address.
- Set configuration

  The CONFG bit in the USBCON register will be set after a SET_CONFIGURATION request with a non-zero value. Otherwise, this bit will be cleared.

### Endpoint Configuration

- Selection of an Endpoint

  The endpoint register access is performed using the UEPNUM register. The registers

  – UEPSTAX

  – UEPCONX

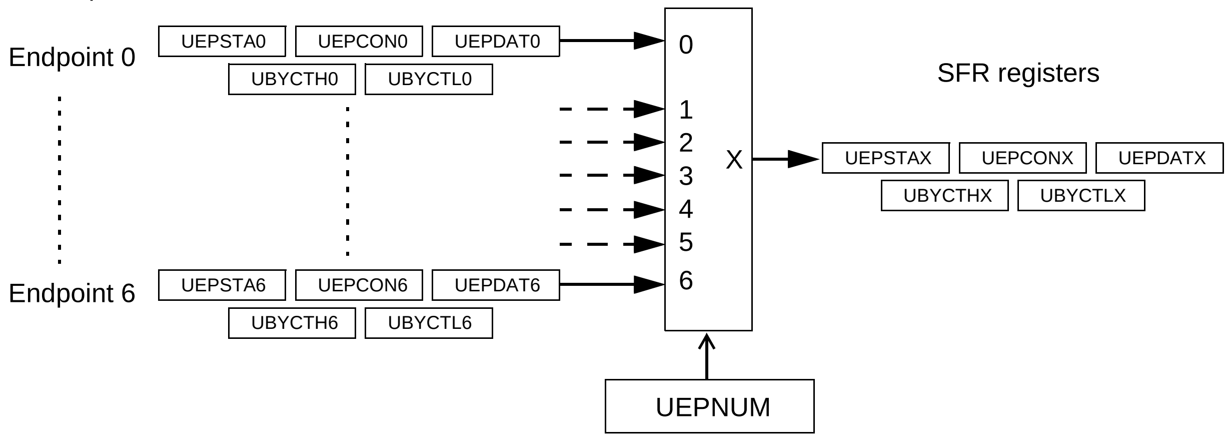
  – UEPDATX

  – UBYCTLX

  – UBYCTHX

  These registers correspond to the endpoint whose number is stored in the UEP-NUM register. To select an Endpoint, the firmware has to write the endpoint number in the UEPNUM register.

**Figure 53.** Endpoint Selection



- Endpoint enable

  Before using an endpoint, this one will be enabled by setting the EPEN bit in the UEPCONX register.

  An endpoint which is not enabled won't answer to any USB request. The Default Control Endpoint (Endpoint 0) will always be enabled in order to answer to USB standard requests.

- Endpoint type configuration

  All Standard Endpoints can be configured in Control, Bulk, Interrupt or Isochronous mode. The Ping-pong Endpoints can be configured in Bulk, Interrupt or Isochronous mode. The configuration of an endpoint is performed by setting the field EPTYPE with the following values:

  – Control:EPTYPE = 00b

  – Isochronous:EPTYPE = 01b

  – Bulk:EPTYPE = 10b

  – Interrupt:EPTYPE = 11b

  The Endpoint 0 is the Default Control Endpoint and will always be configured in Control type.

- Endpoint direction configuration

  For Bulk, Interrupt and Isochronous endpoints, the direction is defined with the EPDIR bit of the UEPCONX register with the following values:

  – IN:EPDIR = 1b

  – OUT:EPDIR = 0b

  For Control endpoints, the EPDIR bit has no effect.

- Summary of Endpoint Configuration:

  Do not forget to select the correct endpoint number in the UEPNUM register before accessing to endpoint specific registers.

**Table 81.** Summary of Endpoint Configuration

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

- Endpoint FIFO reset

  Before using an endpoint, its FIFO will be reset. This action resets the FIFO pointer to its original value, resets the byte counter of the endpoint (UBYCTLX and UBYCTHX registers), and resets the data toggle bit (DTGL bit in UEPCONX).

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

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

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

## Read/Write Data FIFO

### Read Data FIFO

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

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

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

### Write Data FIFO

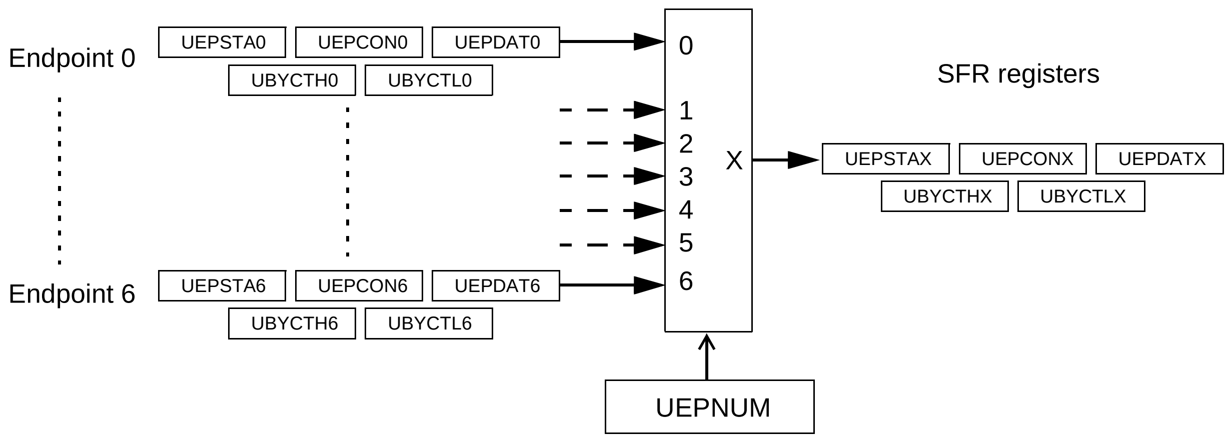
The write access for each IN endpoint is performed using the UEPDATX register.

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

Warning 1: The byte counter is not updated.
Warning 2: Do not write more bytes than supported by the corresponding endpoint.

### FIFO Mapping

**Figure 54.** Endpoint FIFO Configuration

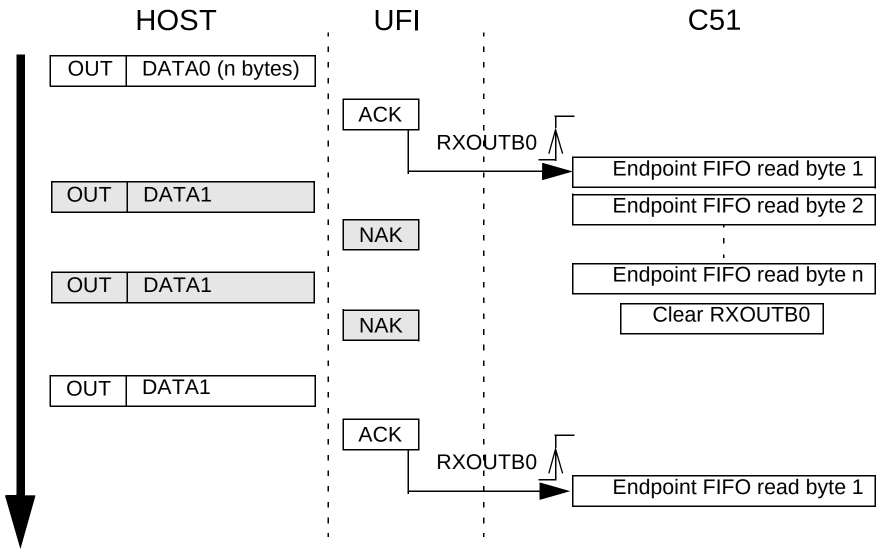


## Bulk/Interrupt Transactions

Bulk and Interrupt transactions are managed in the same way.

### Bulk/Interrupt OUT Transactions in Standard Mode

**Figure 55.** Bulk/Interrupt OUT transactions in Standard Mode


HOST
UFI
C51
OUT
DATA0 (n bytes)
ACK
RXOUTB0
Endpoint FIFO read byte 1
OUT
DATA1
Endpoint FIFO read byte 2
NAK
OUT
DATA1
Endpoint FIFO read byte n
Clear RXOUTB0
NAK
OUT
DATA1
ACK
RXOUTB0
Endpoint FIFO read byte 1

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

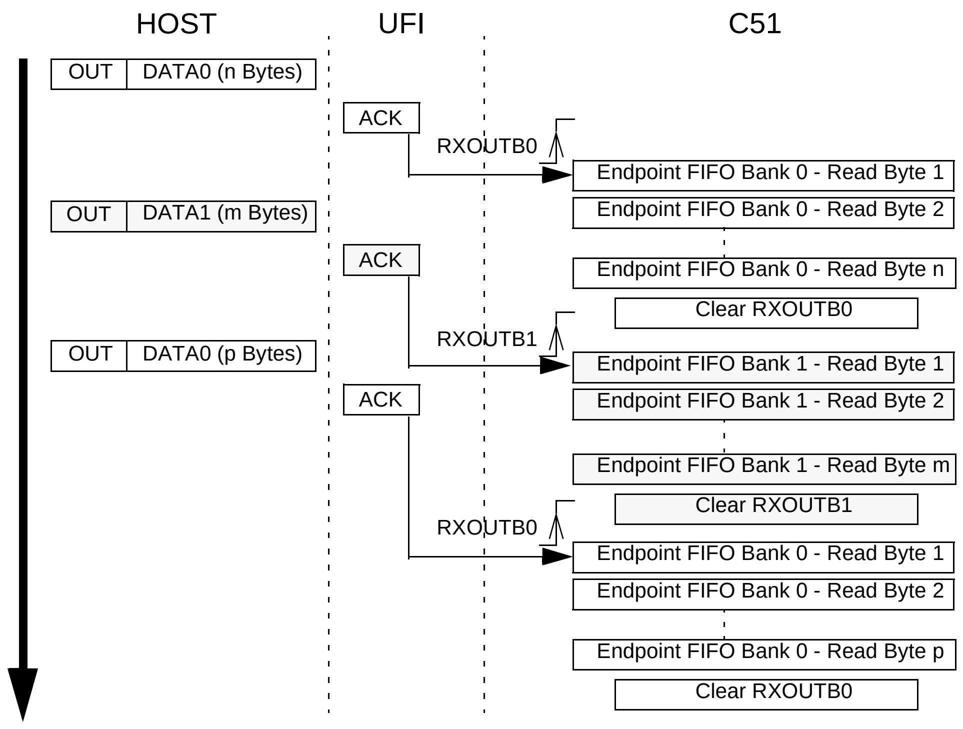
When a valid OUT packet is received on an endpoint, the RXOUTB0 bit is set by the USB controller. This triggers an interrupt if enabled. The firmware has to select the corresponding endpoint, store the number of data bytes by reading the UBYCTLX and UBYCTHX registers. If the received packet is a ZLP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

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

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

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

**Figure 56.** Bulk/Interrupt OUT Transactions in Ping-pong Mode


HOST
UFI
C51
OUT
DATA0 (n Bytes)
ACK
RXOUTB0
Endpoint FIFO Bank 0 - Read Byte 1
OUT
DATA1 (m Bytes)
Endpoint FIFO Bank 0 - Read Byte 2
ACK
Endpoint FIFO Bank 0 - Read Byte n
Clear RXOUTB0
RXOUTB1
OUT
DATA0 (p Bytes)
Endpoint FIFO Bank 1 - Read Byte 1
ACK
Endpoint FIFO Bank 1 - Read Byte 2
Endpoint FIFO Bank 1 - Read Byte m
Clear RXOUTB1
RXOUTB0
Endpoint FIFO Bank 0 - Read Byte 1
Endpoint FIFO Bank 0 - Read Byte 2
Endpoint FIFO Bank 0 - Read Byte p
Clear RXOUTB0

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

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

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

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

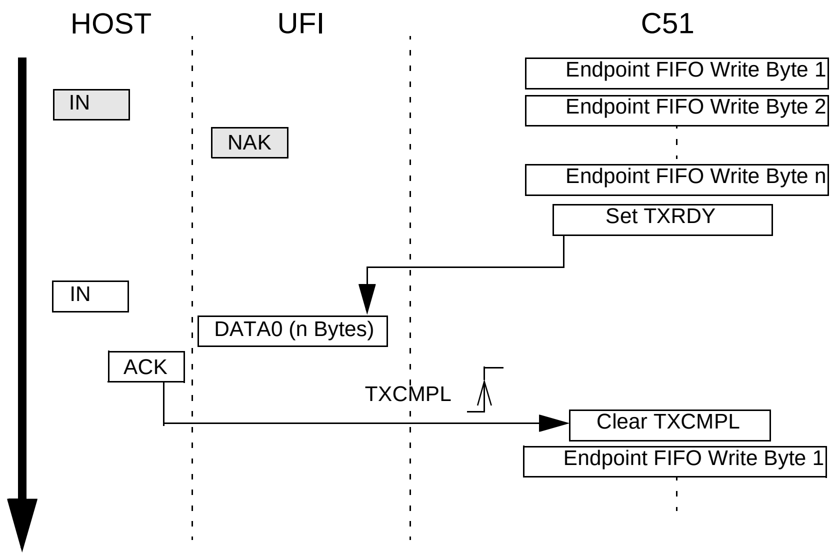
The RXOUTB0 and RXOUTB1 bits are alternatively set by the USB controller at each new valid packet receipt.
The firmware has to clear one of these two bits after having read all the data FIFO to allow a new valid packet to be stored in the corresponding bank.

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

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

## Bulk/Interrupt IN Transactions in Standard Mode

**Figure 57.** Bulk/Interrupt IN Transactions in Standard Mode


HOST
UFI
C51
IN
NAK
Endpoint FIFO Write Byte 1
Endpoint FIFO Write Byte 2
Endpoint FIFO Write Byte n
Set TXRDY
IN
DATA0 (n Bytes)
ACK
TXCMPL
Clear TXCMPL
Endpoint FIFO Write Byte 1

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

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

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

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

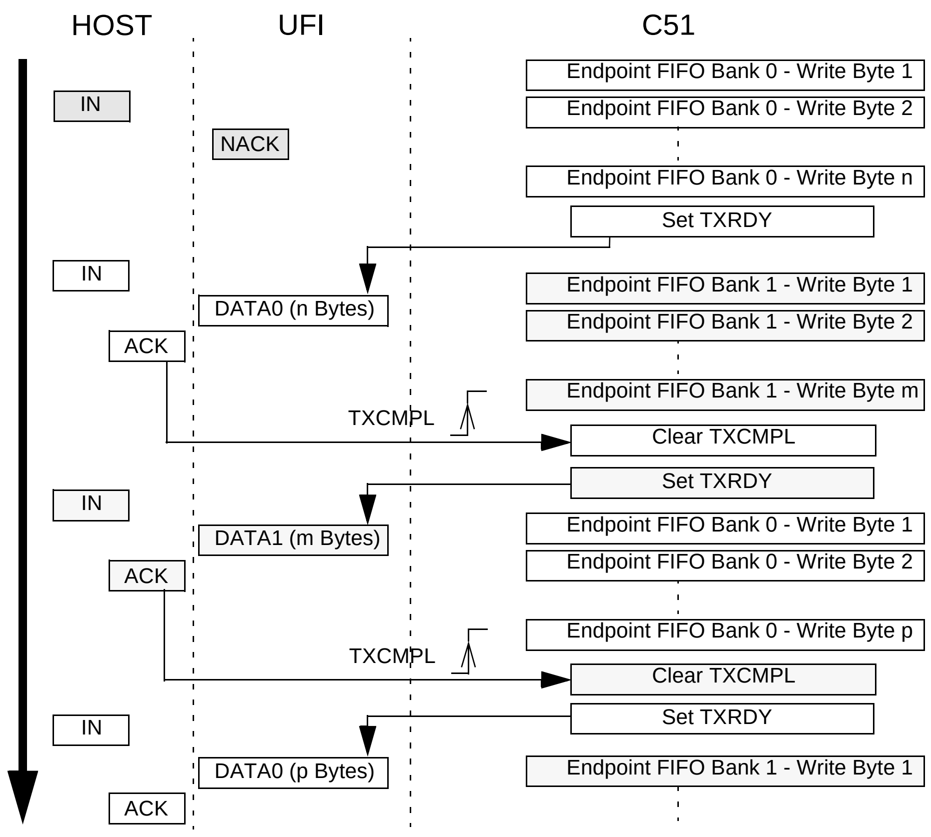
When the IN packet has been sent and acknowledged by the Host, the TXCMPL bit in the UEPSTAX register is set by the USB controller. This triggers a USB interrupt if enabled. The firmware will clear the TXCMPL bit before filling the endpoint FIFO with new data.

The firmware will never write more bytes than supported by the endpoint FIFO.

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

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

**Figure 58.** Bulk/Interrupt IN Transactions in Ping-pong Mode


HOST
UFI
C51
IN
NACK
Endpoint FIFO Bank 0 - Write Byte 1
Endpoint FIFO Bank 0 - Write Byte 2
Endpoint FIFO Bank 0 - Write Byte n
Set TXRDY
IN
DATA0 (n Bytes)
ACK
Endpoint FIFO Bank 1 - Write Byte 1
Endpoint FIFO Bank 1 - Write Byte 2
Endpoint FIFO Bank 1 - Write Byte m
TXCMPL
Clear TXCMPL
Set TXRDY
IN
DATA1 (m Bytes)
ACK
Endpoint FIFO Bank 0 - Write Byte 1
Endpoint FIFO Bank 0 - Write Byte 2
Endpoint FIFO Bank 0 - Write Byte p
TXCMPL
Clear TXCMPL
Set TXRDY
IN
DATA0 (p Bytes)
ACK
Endpoint FIFO Bank 1 - Write Byte 1

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

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

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

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

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

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

The firmware will never write more bytes than supported by the endpoint FIFO.

## Control Transactions

### Setup Stage

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

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

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

### Data Stage: Control Endpoint Direction

The data stage management is similar to Bulk management.

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

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

To send a STALL handshake, see “STALL Handshake” on page 127.

### Status Stage

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

The status stage management is similar to Bulk management.

- For a Control Write transaction or a No-Data Control transaction, the status stage consists of a IN Zero Length Packet (see “Bulk/Interrupt IN Transactions in Standard Mode” on page 122). To send a STALL handshake, see “STALL Handshake” on page 127.
- For a Control Read transaction, the status stage consists of a OUT Zero Length Packet (see “Bulk/Interrupt OUT Transactions in Standard Mode” on page 120).

## Isochronous Transactions

### Isochronous OUT Transactions in Standard Mode

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

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

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

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

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

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

### Isochronous OUT Transactions in Ping-pong Mode

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

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

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

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

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

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

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

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

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

**Isochronous IN Transactions in Standard Mode**

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

The firmware will fill the FIFO with the data to be sent and set the TXRDY bit in the UEPSTAX register to allow the USB controller to send the data stored in FIFO at the next IN request concerning this endpoint.
If the TXRDY bit is not set when the IN request occurs, nothing will be sent by the USB controller.

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

The firmware will never write more bytes than supported by the endpoint FIFO

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

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

The firmware will fill the FIFO bank 0 with the data to be sent and set the TXRDY bit in the UEPSTAX register to allow the USB controller to send the data stored in FIFO at the next IN request concerning the endpoint. The FIFO banks are automatically switched, and the firmware can immediately write into the endpoint FIFO bank 1.
If the TXRDY bit is not set when the IN request occurs, nothing will be sent by the USB controller.

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

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

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

The firmware will never write more bytes than supported by the endpoint FIFO.

## Miscellaneous

**USB Reset**

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

**STALL Handshake**

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

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

The STALLRQ bit is cleared automatically by hardware when a valid SETUP PID is received on a CONTROL type endpoint.

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

**Start of Frame Detection**

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

**Frame Number**

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

**Data Toggle Bit**

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

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

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

## Suspend/Resume Management

### Suspend

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

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

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

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

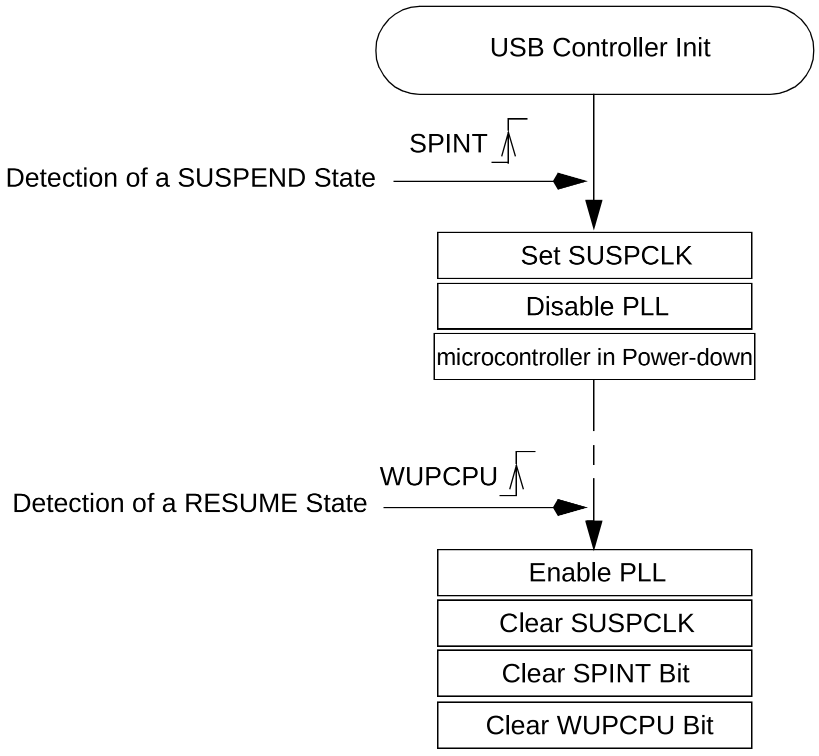
### Resume

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

The firmware has to clear the SPINT bit in the USBINT register before any other USB operation in order to wake up the USB controller from its Suspend mode.

The USB controller is then re-activated.

**Figure 59.** Example of a Suspend/Resume Management



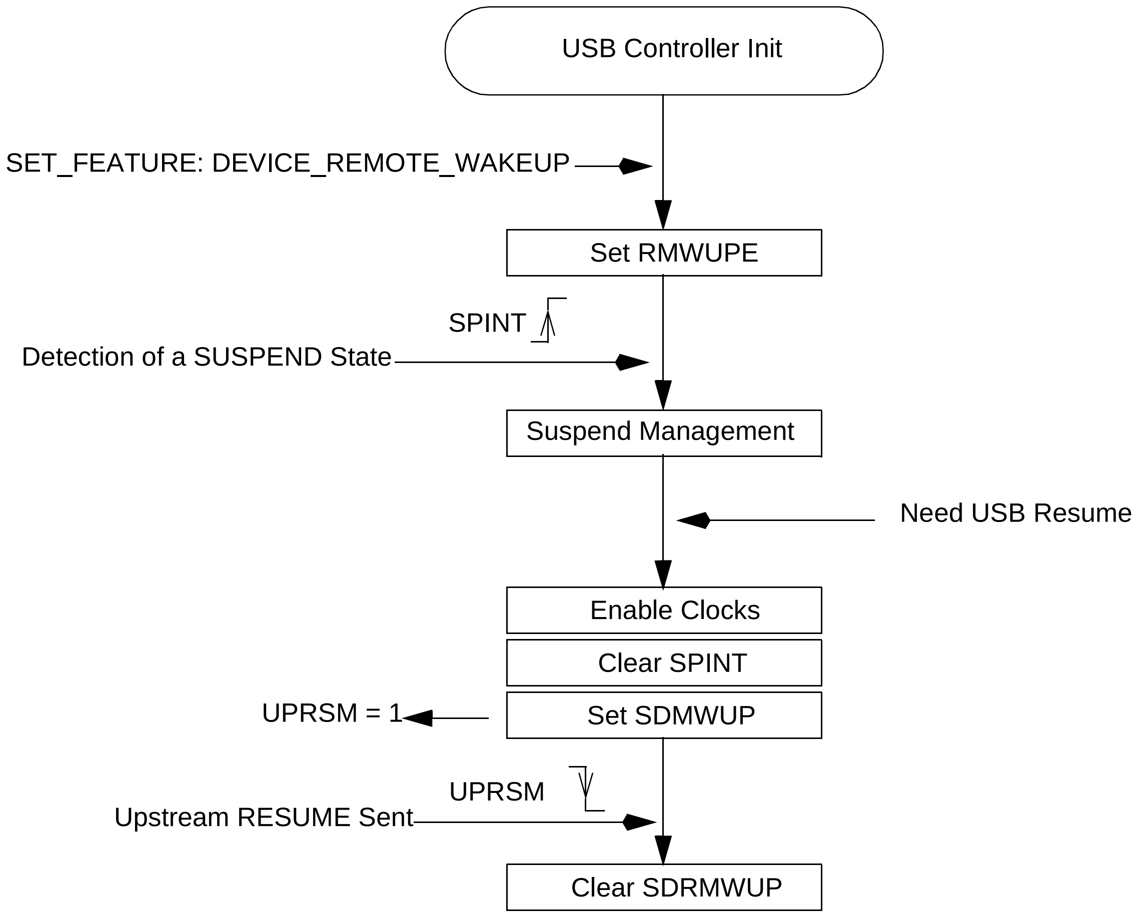
## Upstream Resume

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

When the USB controller receives the SET_FEATURE request: DEVICE_REMOTE_WAKEUP, the firmware will set to 1 the RMWUPE bit in the USBCON register to enable this functionality. RMWUPE value will be 0 in the other cases.

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

**Figure 60.** Example of REMOTE WAKEUP Management



## Detach Simulation

In order to be re-enumerated by the Host, the AT89C5131 has the possibility to simulate a DETACH - ATTACH of the USB bus.

The $V_{REF}$ output voltage is between 3.0V and 3.6V. This output can be connected to the D+ pull-up as shown in Figure 61. This output can be put in high-impedance when the DETACH bit is set to 1 in the USBCON register. Maintaining this output in high impedance for more than 3 µs will simulate the disconnection of the device. When resetting the DETACH bit, an attach is then simulated.

**Figure 61.** Example of $V_{REF}$ Connection

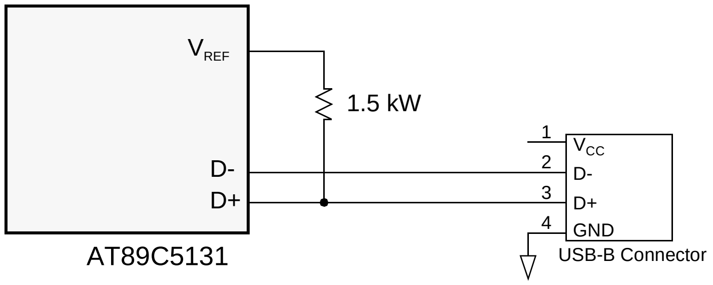
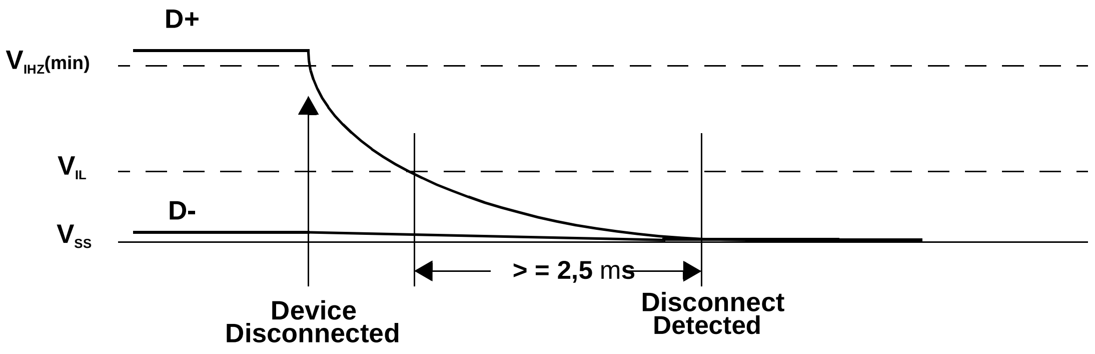


**Figure 62.** Disconnect Timing



## USB Interrupt System

### Interrupt System Priorities

**Figure 63.** USB Interrupt Control System

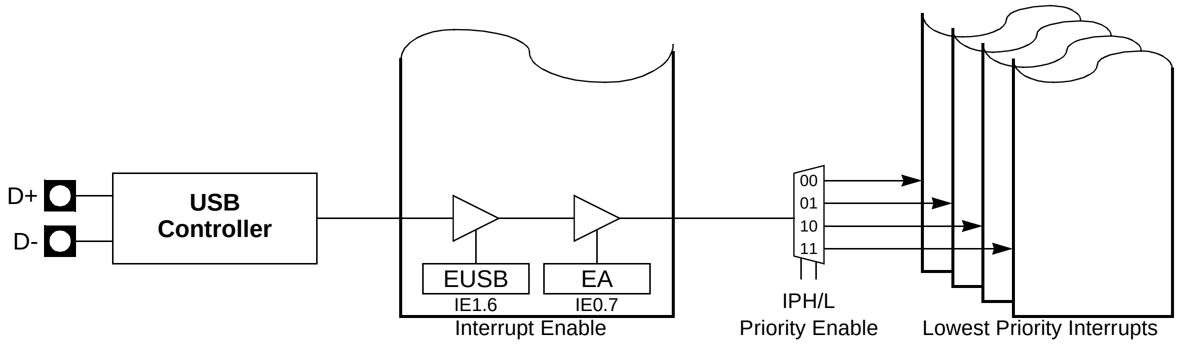


Table 82. Priority Levels

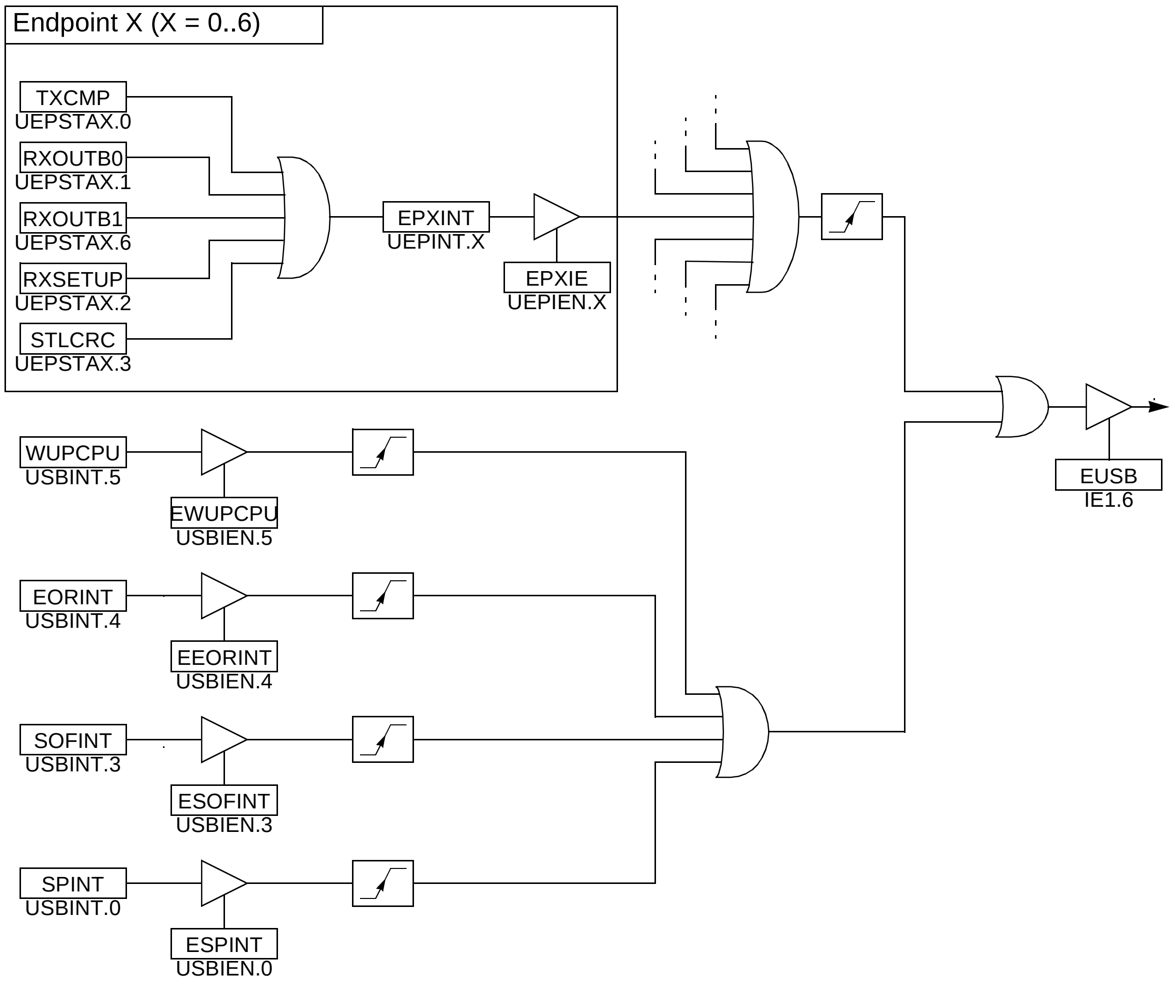
| IPHUSB | IPLUSB | USB Priority Level |
|---|---|---|
| 0 | 0 | 0 Lowest |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 Highest |

## USB Interrupt Control System

As shown in Figure 64, many events can produce a USB interrupt:

- TXCMPL: Transmitted In Data (see Table 89 on page 138). This bit is set by hardware when the Host accept a In packet.
- RXOUTB0: Received Out Data Bank 0 (see Table 89 on page 138). This bit is set by hardware when an Out packet is accepted by the endpoint and stored in bank 0.
- RXOUTB1: Received Out Data Bank 1 (only for Ping-pong endpoints) (see Table 89 on page 138). This bit is set by hardware when an Out packet is accepted by the endpoint and stored in bank 1.
- RXSETUP: Received Setup (see Table 89 on page 138). This bit is set by hardware when an SETUP packet is accepted by the endpoint.
- STLCRC: STALLED (only for Control, Bulk and Interrupt endpoints) (see Table 89 on page 138). This bit is set by hardware when a STALL handshake has been sent as requested by STALLRQ, and is reset by hardware when a SETUP packet is received.
- SOFINT: Start of Frame Interrupt (See “USBIEN Register USBIEN (S:BEh) USB Global Interrupt Enable Register” on page 135). This bit is set by hardware when a USB Start of Frame packet has been received.
- WUPCPU: Wake-Up CPU Interrupt (See “USBIEN Register USBIEN (S:BEh) USB Global Interrupt Enable Register” on page 135). This bit is set by hardware when a USB resume is detected on the USB bus, after a SUSPEND state.
- SPINT: Suspend Interrupt (See “USBIEN Register USBIEN (S:BEh) USB Global Interrupt Enable Register” on page 135). This bit is set by hardware when a USB suspend is detected on the USB bus.

**Figure 64.** USB Interrupt Control Block Diagram



## USB Registers

**Table 83.** USBCON Register
USBCON (S:BCh)
USB Global Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| USBE | SUSPCLK | SDRMWUP | DETACH | UPRSM | RMWUPE | CONFG | FADDEN |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | USBE | **USB Enable**<br>Set this bit to enable the USB controller.<br>Clear this bit to disable and reset the USB controller, to disable the USB transceiver an to disable the USB controller clock inputs. |
| 6 | SUSPCLK | **Suspend USB Clock**<br>Set this bit to disable the 48 MHz clock input (Resume Detection is still active).<br>Clear this bit to enable the 48 MHz clock input. |
| 5 | SDRMWUP | **Send Remote Wake Up**<br>Set this bit to force an external interrupt on the USB controller for Remote Wake UP purpose.<br>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.<br>This bit is cleared by software. |
| 4 | DETACH | **Detach Command**<br>Set this bit to simulate a Detach on the USB line. The $V_{REF}$ pin is then in a floating state.<br>Clear this bit to maintain $V_{REF}$ at 3.3V. |
| 3 | UPRSM | **Upstream Resume (read only)**<br>This bit is set by hardware when SDRMWUP has been set and if RMWUPE is enabled.<br>This bit is cleared by hardware after the upstream resume has been sent. |
| 2 | RMWUPE | **Remote Wake-Up Enable**<br>Set this bit to enabled request an upstream resume signaling to the host.<br>Clear this bit otherwise.<br>Note: Do not set this bit if the host has not set the DEVICE_REMOTE_WAKEUP feature for the device. |
| 1 | CONFG | **Configured**<br>This bit will be set by the device firmware after a SET_CONFIGURATION request with a non-zero value has been correctly processed.<br>It will be cleared by the device firmware when a SET_CONFIGURATION request with a zero value is received. It is cleared by hardware on hardware reset or when an USB reset is detected on the bus (SE0 state for at least 32 Full Speed bit times: typically 2.7 µs). |
| 0 | FADDEN | **Function Address Enable**<br>This bit will be set by the device firmware after a successful status phase of a SET_ADDRESS transaction.<br>It will not be cleared afterwards by the device firmware. It is cleared by hardware on hardware reset or when an USB reset is received (see above).<br>When this bit is cleared, the default function address is used (0). |

Reset Value = 00h

**Table 84.** 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 | - | **Reserved**<br>The value read from these bits is always 0. Do not set these bits. |
| 5 | WUPCPU | **Wake Up CPU Interrupt**<br>This bit is 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 Figure 85 on page 135.<br>When receiving this interrupt, user has to enable all USB clock inputs.<br>This bit will be cleared by software (USB clocks must be enabled before). |
| 4 | EORINT | **End Of Reset Interrupt**<br>This bit is 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 the Figure 85 (see Figure 85 on page 135).<br>This bit will be cleared by software. |
| 3 | SOFINT | **Start of Frame Interrupt**<br>This bit is set by hardware when an USB Start of Frame PID (SOF) has been detected. This triggers a USB interrupt when ESOFINT is set in the Figure 85 (see Figure 85 on page 135).<br>This bit will be cleared by software. |
| 2-1 | - | **Reserved**<br>The value read from these bits is always 0. Do not set these bits. |
| 0 | SPINT | **Suspend Interrupt**<br>This bit is 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 see Figure 85 on page 135.<br>This bit will be cleared by software BEFORE any other USB operation to re-activate the macro. |

Reset Value = 00h

**Table 85.** USBIEN Register
USBIEN (S:BEh)
USB Global Interrupt Enable Register

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-6 | - | **Reserved**<br>The value read from these bits is always 0. Do not set these bits. |
| 5 | EWUPCPU | **Enable Wake Up CPU Interrupt**<br>Set this bit to enable Wake Up CPU Interrupt. (See “USBIEN Register USBIEN (S:BEh) USB Global Interrupt Enable Register” on page 135)<br>Clear this bit to disable Wake Up CPU Interrupt. |
| 4 | EEOFINT | **Enable End Of Reset Interrupt**<br>Set this bit to enable End Of Reset Interrupt. (See “USBIEN Register USBIEN (S:BEh) USB Global Interrupt Enable Register” on page 135). This bit is set after reset.<br>Clear this bit to disable End Of Reset Interrupt. |
| 3 | ESOFINT | **Enable SOF Interrupt**<br>Set this bit to enable SOF Interrupt. (See “USBIEN Register USBIEN (S:BEh) USB Global Interrupt Enable Register” on page 135).<br>Clear this bit to disable SOF Interrupt. |
| 2-1 | - | **Reserved**<br>The value read from these bits is always 0. Do not set these bits. |
| 0 | ESPINT | **Enable Suspend Interrupt**<br>Set this bit to enable Suspend Interrupts (see the “USBIEN Register USBIEN (S:BEh) USB Global Interrupt Enable Register” on page 135).<br>Clear this bit to disable Suspend Interrupts. |

Reset Value = 10h

**Table 86.** 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 | **Function Enable**<br>Set this bit to enable the function. FADD is reset to 1.<br>Cleared this bit to disable the function. |
| 6-0 | UADD[6:0] | **USB Address**<br>This field contains the default address (0) after power-up or USB bus reset.<br>It will be written with the value set by a SET_ADDRESS request received by the device firmware. |

Reset Value = 80h

**Table 87.** UEPNUM Register
UEPNUM (S:C7h)
USB Endpoint Number

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | EPNUM3 | EPNUM2 | EPNUM1 | EPNUM0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-4 | - | **Reserved**<br>The value read from these bits is always 0. Do not set these bits. |
| 3-0 | EPNUM[3:0] | **Endpoint Number**<br>Set this field with the number of the endpoint which will be accessed when reading or writing to, UEPDATX Register UEPDATX (S:CFh) USB FIFO Data Endpoint X (X = EPNUM set in UEPNUM Register UEPNUM (S:C7h) USB Endpoint Number), UBYCTLX Register UBYCTLX (S:E2h) USB Byte Count Low Register X (X = EPNUM set in UEPNUM Register UEPNUM (S:C7h) USB Endpoint Number), UBYCTHX Register UBYCTHX (S:E3h) USB Byte Count High Register X (X = EPNUM set in UEPNUM Register UEPNUM (S:C7h) USB Endpoint Number) or UEPCONX Register UEPCONX (S:D4h) USB Endpoint X Control Register. This value can be 0, 1, 2, 3, 4, 5 or 6. |

Reset Value = 00h

**Table 88.** UEPCONX Register
UEPCONX (S:D4h)
USB Endpoint X Control Register

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | EPEN | **Endpoint Enable**<br>Set this bit to enable the endpoint according to the device configuration. Endpoint 0 will always be enabled after a hardware or USB bus reset and participate in the device configuration.<br>Clear this bit to disable the endpoint according to the device configuration. |
| 6-4 | - | **Reserved**<br>The value read from this bit is always 0. Do not set this bit. |
| 3 | DTGL | **Data Toggle (Read-only)**<br>This bit is set by hardware when a valid DATA0 packet is received and accepted.<br>This bit is cleared by hardware when a valid DATA1 packet is received and accepted. |
| 2 | EPDIR | **Endpoint Direction**<br>Set this bit to configure IN direction for Bulk, Interrupt and Isochronous endpoints.<br>Clear this bit to configure OUT direction for Bulk, Interrupt and Isochronous endpoints.<br>This bit has no effect for Control endpoints. |
| 1-0 | EPTYPE[1:0] | **Endpoint Type**<br>Set this field according to the endpoint configuration (Endpoint 0 will always be configured as control):<br>00Control endpoint<br>01Isochronous endpoint<br>10Bulk endpoint<br>11Interrupt endpoint |

Note: 1. (X = EPNUM set in UEPNUM Register UEPNUM (S:C7h) USB Endpoint Number)

Reset Value = 80h when UEPNUM = 0
Reset Value = 00h otherwise

**Table 89.** UEPSTAX (S:CEh) USB Endpoint X Status Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DIR | RXOUTB1 | STALLRQ | TXRDY | STL/CRC | RXSETUP | RXOUTB0 | TXCMP |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | DIR | **Control Endpoint Direction**<br>This bit is used only if the endpoint is configured in the control type (seeSection “UEPCONX Register UEPCONX (S:D4h) USB Endpoint X Control Register”).<br>This bit determines the Control data and status direction.<br>The device firmware will set this bit ONLY for the IN data stage, before any other USB operation. Otherwise, the device firmware will clear this bit. |
| 6 | RXOUTB1 | **Received OUT Data Bank 1 for Endpoints 4, 5 and 6 (Ping-pong mode)**<br>This bit is set by hardware after a new packet has been stored in the endpoint FIFO data bank 1 (only in Ping-pong mode).<br>Then, the endpoint interrupt is triggered if enabled (see“UEPINT Register UEPINT (S:F8h read-only) USB Endpoint Interrupt Register” on page 142) and all the following OUT packets to the endpoint bank 1 are rejected (NAK’ed) until this bit has been cleared, excepted for Isochronous Endpoints.<br>This bit will be cleared by the device firmware after reading the OUT data from the endpoint FIFO. |
| 5 | STALLRQ | **Stall Handshake Request**<br>Set this bit to request a STALL answer to the host for the next handshake.Clear this bit otherwise.<br>For CONTROL endpoints: cleared by hardware when a valid SETUP PID is received. |
| 4 | TXRDY | **TX Packet Ready**<br>Set this bit after a packet has been written into the endpoint FIFO for IN data transfers. Data will 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.<br>This bit is 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. When this bit is cleared, the endpoint interrupt is triggered if enabled (see“UEPINT Register UEPINT (S:F8h read-only) USB Endpoint Interrupt Register” on page 142). |
| 3 | STLCRC | **Stall Sent/CRC error flag**<br>- For Control, Bulk and Interrupt Endpoints:<br>This bit is set by hardware after a STALL handshake has been sent as requested by STALLRQ. Then, the endpoint interrupt is triggered if enabled (see“UEPINT Register UEPINT (S:F8h read-only) USB Endpoint Interrupt Register” on page 142)<br>It will be cleared by the device firmware.<br>- For Isochronous Endpoints **(Read-Only)**:<br>This bit is set by hardware if the last received data is corrupted (CRC error on data).<br>This bit is updated by hardware when a new data is received. |
| 2 | RXSETUP | **Received SETUP**<br>This bit is 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 (see“UEPINT Register UEPINT (S:F8h read-only) USB Endpoint Interrupt Register” on page 142).<br>It will be cleared by the device firmware after reading the SETUP data from the endpoint FIFO. |
| 1 | RXOUTB0 | **Received OUT Data Bank 0** (see also RXOUTB1 bit for Ping-pong Endpoints)<br>This bit is set by hardware after a new packet has been stored in the endpoint FIFO data bank 0. Then, the endpoint interrupt is triggered if enabled (see“UEPINT Register UEPINT (S:F8h read-only) USB Endpoint Interrupt Register” on page 142) and all the following OUT packets to the endpoint bank 0 are rejected (NAK’ed) until this bit has been cleared, excepted for Isochronous Endpoints. However, for control endpoints, an early SETUP transaction may overwrite the content of the endpoint FIFO, even if its Data packet is received while this bit is set.<br>This bit will be cleared by the device firmware after reading the OUT data from the endpoint FIFO. |
| 0 | TXCMPL | **Transmitted IN Data Complete**<br>This bit is 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 (see“UEPINT Register UEPINT (S:F8h read-only) USB Endpoint Interrupt Register” on page 142).<br>This bit will be cleared by the device firmware before setting TXRDY. |

Reset Value = 00h

**Table 90.** UEPDATX Register
UEPDATX (S:CFh)
USB FIFO Data Endpoint X (X = EPNUM set in UEPNUM Register UEPNUM (S:C7h)

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 - 0 | FDAT[7:0] | **Endpoint X FIFO data**<br>Data byte to be written to FIFO or data byte to be read from the FIFO, for the Endpoint X (see EPNUM). |

USB Endpoint Number)

Reset Value = XXh

**Table 91.** UBYCTLX Register
UBYCTLX (S:E2h)
USB Byte Count Low Register X (X = EPNUM set in UEPNUM Register UEPNUM

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| BYCT7 | BYCT6 | BYCT5 | BYCT4 | BYCT3 | BYCT2 | BYCT1 | BYCT0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 - 0 | BYCT[7:0] | **Byte Count LSB**<br>Least Significant Byte of the byte count of a received data packet. The most significant part is provided by the UBYCTHX Register UBYCTHX (S:E3h) USB Byte Count High Register X (X = EPNUM set in UEPNUM Register UEPNUM (S:C7h) USB Endpoint Number) (see Figure 91 on page 139). This byte count is equal to the number of data bytes received after the Data PID. |

(S:C7h) USB Endpoint Number)

Reset Value = 00h

**Table 92.** UBYCTHX Register
UBYCTHX (S:E3h)
USB Byte Count High Register X (X = EPNUM set in UEPNUM Register UEPNUM

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | BYCT9 | BYCT8 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-2 | - | **Reserved**<br>The value read from these bits is always 0. Do not set these bits. |
| 2-0 | BYCT[10:8] | **Byte Count MSB**<br>Most Significant Byte of the byte count of a received data packet. The Least significant part is provided by UBYCTLX Register UBYCTLX (S:E2h) USB Byte Count Low Register X (X = EPNUM set in UEPNUM Register UEPNUM (S:C7h) USB Endpoint Number) (see Figure 91 on page 139). |

(S:C7h) USB Endpoint Number)

Reset Value = 00h

**Table 93.** UEPRST Register
UEPRST (S:D5h)
USB Endpoint FIFO Reset Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | EP6RST | EP5RST | EP4RST | EP3RST | EP2RST | EP1RST | EP0RST |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is always 0. Do not set this bit. |
| 6 | EP6RST | **Endpoint 6 FIFO Reset**<br>Set this bit and reset the endpoint FIFO prior to any other operation, upon hardware reset or when an USB bus reset has been received.<br>Then, clear this bit to complete the reset operation and start using the FIFO. |
| 5 | EP5RST | **Endpoint 5 FIFO Reset**<br>Set this bit and reset the endpoint FIFO prior to any other operation, upon hardware reset or when an USB bus reset has been received.<br>Then, clear this bit to complete the reset operation and start using the FIFO. |
| 4 | EP4RST | **Endpoint 4 FIFO Reset**<br>Set this bit and reset the endpoint FIFO prior to any other operation, upon hardware reset or when an USB bus reset has been received.<br>Then, clear this bit to complete the reset operation and start using the FIFO. |
| 3 | EP3RST | **Endpoint 3 FIFO Reset**<br>Set this bit and reset the endpoint FIFO prior to any other operation, upon hardware reset or when an USB bus reset has been received.<br>Then, clear this bit to complete the reset operation and start using the FIFO. |
| 2 | EP2RST | **Endpoint 2 FIFO Reset**<br>Set this bit and reset the endpoint FIFO prior to any other operation, upon hardware reset or when an USB bus reset has been received.<br>Then, clear this bit to complete the reset operation and start using the FIFO. |
| 1 | EP1RST | **Endpoint 1 FIFO Reset**<br>Set this bit and reset the endpoint FIFO prior to any other operation, upon hardware reset or when an USB bus reset has been received.<br>Then, clear this bit to complete the reset operation and start using the FIFO. |
| 0 | EP0RST | **Endpoint 0 FIFO Reset**<br>Set this bit and reset the endpoint FIFO prior to any other operation, upon hardware reset or when an USB bus reset has been received.<br>Then, clear this bit to complete the reset operation and start using the FIFO. |

Reset Value = 00h

**Table 94.** UEPINT Register
UEPINT (S:F8h read-only)
USB Endpoint Interrupt Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | EP6INT | EP5INT | EP4INT | EP3INT | EP2INT | EP1INT | EP0INT |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is always 0. Do not set this bit. |
| 6 | EP6INT | **Endpoint 6 Interrupt**<br>This bit is set by hardware when an interrupt is triggered by the (see Table 89 on page 138) and this endpoint interrupt is enabled by the UEPIEN Register UEPIEN (S:C2h) USB Endpoint Interrupt Enable Register (see Figure 95 on page 143).<br>This bit is cleared by software. |
| 5 | EP5INT | **Endpoint 5 Interrupt**<br>This bit is set by hardware when an interrupt is triggered by the (see Table 89 on page 138) and this endpoint interrupt is enabled by the UEPIEN Register UEPIEN (S:C2h) USB Endpoint Interrupt Enable Register (see Figure 95 on page 143).<br>This bit is cleared by software. |
| 4 | EP4INT | **Endpoint 4 Interrupt**<br>This bit is set by hardware when an interrupt is triggered by the (see Table 89 on page 138) and this endpoint interrupt is enabled by the UEPIEN Register UEPIEN (S:C2h) USB Endpoint Interrupt Enable Register (see Figure 95 on page 143).<br>This bit is cleared by software. |
| 3 | EP3INT | **Endpoint 3 Interrupt**<br>This bit is set by hardware when an interrupt is triggered by the (see Table 89 on page 138) and this endpoint interrupt is enabled by the UEPIEN Register UEPIEN (S:C2h) USB Endpoint Interrupt Enable Register (see Figure 95 on page 143).<br>This bit is cleared by software. |
| 2 | EP2INT | **Endpoint 2 Interrupt**<br>This bit is set by hardware when an interrupt is triggered by the (see Table 89 on page 138) and this endpoint interrupt is enabled by the UEPIEN Register UEPIEN (S:C2h) USB Endpoint Interrupt Enable Register (see Figure 95 on page 143).<br>This bit is cleared by software. |
| 1 | EP1INT | **Endpoint 1 Interrupt**<br>This bit is set by hardware when an interrupt is triggered by the (see Table 89 on page 138) and this endpoint interrupt is enabled by the UEPIEN Register UEPIEN (S:C2h) USB Endpoint Interrupt Enable Register (see Figure 95 on page 143).<br>This bit is cleared by software. |
| 0 | EP0INT | **Endpoint 0 Interrupt**<br>This bit is set by hardware when an interrupt is triggered by the (see Table 89 on page 138) and this endpoint interrupt is enabled by the UEPIEN Register UEPIEN (S:C2h) USB Endpoint Interrupt Enable Register (see Figure 95 on page 143).<br>This bit is cleared by software. |

Reset Value = 00h

**Table 95.** UEPIEN Register
UEPIEN (S:C2h)
USB Endpoint Interrupt Enable Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | EP6INTE | EP5INTE | EP4INTE | EP3INTE | EP2INTE | EP1INTE | EP0INTE |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is always 0. Do not set this bit. |
| 6 | EP6INTE | **Endpoint 6 Interrupt Enable**<br>Set this bit to enable the interrupts for this endpoint.<br>Clear this bit to disable the interrupts for this endpoint. |
| 5 | EP5INTE | **Endpoint 5 Interrupt Enable**<br>Set this bit to enable the interrupts for this endpoint.<br>Clear this bit to disable the interrupts for this endpoint. |
| 4 | EP4INTE | **Endpoint 4 Interrupt Enable**<br>Set this bit to enable the interrupts for this endpoint.<br>Clear this bit to disable the interrupts for this endpoint. |
| 3 | EP3INTE | **Endpoint 3 Interrupt Enable**<br>Set this bit to enable the interrupts for this endpoint.<br>Clear this bit to disable the interrupts for this endpoint. |
| 2 | EP2INTE | **Endpoint 2 Interrupt Enable**<br>Set this bit to enable the interrupts for this endpoint.<br>Clear this bit to disable the interrupts for this endpoint. |
| 1 | EP1INTE | **Endpoint 1 Interrupt Enable**<br>Set this bit to enable the interrupts for this endpoint.<br>Clear this bit to disable the interrupts for this endpoint. |
| 0 | EP0INTE | **Endpoint 0 Interrupt Enable**<br>Set this bit to enable the interrupts for this endpoint.<br>Clear this bit to disable the interrupts for this endpoint. |

Reset Value = 00h

**Table 96.** 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 |
|---|---|---|
| 5 | CRCOK | **Frame Number CRC OK**<br>This bit is set by hardware when a new Frame Number in Start of Frame Packet is received without CRC error.<br>This bit is updated after every Start of Frame packet receipt.<br>Important note: the Start of Frame interrupt is generated just after the PID receipt. |
| 4 | CRCERR | **Frame Number CRC Error**<br>This bit is set by hardware when a corrupted Frame Number in Start of Frame packet is received.<br>This bit is updated after every Start of Frame packet receipt.<br>Important note: the Start of Frame interrupt is generated just after the PID receipt. |
| 3 | - | **Reserved**<br>The value read from this bit is always 0. Do not set this bit. |
| 2-0 | FNUM[10:8] | **Frame Number**<br>FNUM[10:8] are the upper 3 bits of the 11-bit Frame Number (see the “UFNUML Register UFNUML (S:BAh, read-only) USB Frame Number Low Register” on page 144). It is provided in the last received SOF packet (see SOFINT in the “USBIEN Register USBIEN (S:BEh) USB Global Interrupt Enable Register” on page 135). FNUM is updated if a corrupted SOF is received. |

Reset Value = 00h

**Table 97.** 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 | FNUM[7:0] | **Frame Number**<br>FNUM[7:0] are the lower 8 bits of the 11-bit Frame Number (See “UFNUMH Register UFNUMH (S:BBh, read-only) USB Frame Number High Register” on page 144). |

Reset Value = 00h

## Power Management

### Idle Mode

An instruction that sets PCON.0 indicates that it is the last instruction to be executed before going into the Idle mode. In the Idle mode, the internal clock signal is gated off to the CPU, but not to the interrupt, Timer, and Serial Port functions. The CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator and all other registers maintain their data during Idle. The port pins hold the logical states they had at the time Idle was activated. ALE and PSEN hold at logic high level.

There are two ways to terminate the Idle mode. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that put the device into idle.

The flag bits GF0 and GF1 can be used to give an indication if an interrupt occurred during normal operation or during an Idle. For example, an instruction that activates Idle can also set one or both flag bits. When Idle is terminated by an interrupt, the interrupt service routine can examine the flag bits.

The other way of terminating the Idle mode is with a hardware reset. Since the clock oscillator is still running, the hardware reset needs to be held active for only two machine cycles (24 oscillator periods) to complete the reset.

### Power-down Mode

To save maximum power, a power-down mode can be invoked by software (refer to Table 13, PCON register).

In power-down mode, the oscillator is stopped and the instruction that invoked power-down mode is the last instruction executed. The internal RAM and SFRs retain their value until the power-down mode is terminated. $V_{CC}$ can be lowered to save further power. Either a hardware reset or an external interrupt can cause an exit from power-down. To properly terminate power-down, the reset or external interrupt should not be executed before $V_{CC}$ is restored to its normal operating level and must be held active long enough for the oscillator to restart and stabilize.

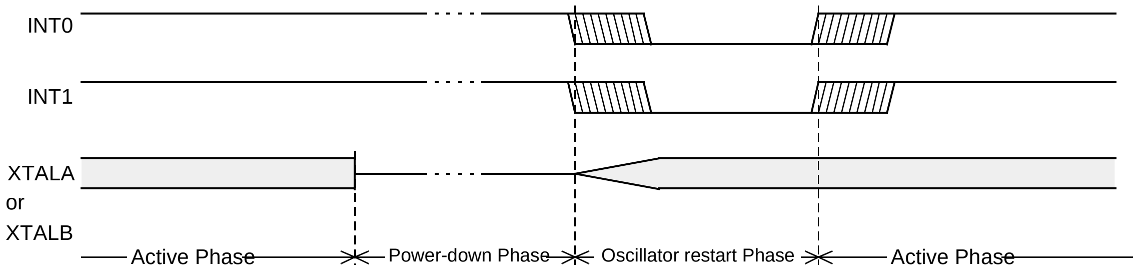
Only:

- external interrupt $\overline{\text{INT0}}$,
- external interrupt $\overline{\text{INT1}}$,
- Keyboard interrupt and
- USB Interrupt

are useful to exit from power-down. For that, interrupt must be enabled and configured as level or edge sensitive interrupt input. When Keyboard Interrupt occurs after a power down mode, 1024 clocks are necessary to exit to power-down mode and enter in operating mode.

Holding the pin low restarts the oscillator but bringing the pin high completes the exit as detailed in Figure 65. When both interrupts are enabled, the oscillator restarts as soon as one of the two inputs is held low and power-down exit will be completed when the first input is released. In this case, the higher priority interrupt service routine is executed. Once the interrupt is serviced, the next instruction to be executed after RETI will be the one following the instruction that put AT89C5131 into power-down mode.

**Figure 65.** Power-down Exit Waveform



Exit from power-down by reset redefines all the SFRs, exit from power-down by external interrupt does no affect the SFRs.

Exit from power-down by either reset or external interrupt does not affect the internal RAM content.

Note: If idle mode is activated with power-down mode (IDL and PD bits set), the exit sequence is unchanged, when execution is vectored to interrupt, PD and IDL bits are cleared and idle mode is not entered.

This table shows the state of ports during idle and power-down modes.

**Table 98.** State of Ports

| Mode | Program Memory | ALE | PSEN | PORT0 | PORT1 | PORT2 | PORT3 | PORTI2 |
|---|---|---|---|---|---|---|---|---|
| Idle | Internal | 1 | 1 | Port Data[1] | Port Data | Port Data | Port Data | Port Data |
| Idle | External | 1 | 1 | Floating | Port Data | Address | Port Data | Port Data |
| Power-down | Internal | 0 | 0 | Port Data[1] | Port Data | Port Data | Port Data | Port Data |
| Power-down | External | 0 | 0 | Floating | Port Data | Port Data | Port Data | Port Data |

Note: 1. Port 0 can force a 0 level. A “one” will leave port floating.

## Registers

**Table 99.** PCON Register

PCON (S:87h)
Power Control Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **SMOD1** | **SMOD0** | **-** | **POF** | **GF1** | **GF0** | **PD** | **IDL** |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | SMOD1 | **Serial Port Mode bit 1**<br>Set to select double baud rate in mode 1, 2 or 3. |
| 6 | SMOD0 | **Serial Port Mode bit 0**<br>Set to select FE bit in SCON register.<br>Clear to select SM0 bit in SCON register |
| 5 | - | **Reserved**<br>The value read from this bit is always 0. Do not set this bit. |
| 4 | POF | **Power-Off Flag**<br>Set by hardware when $V_{CC}$ rises from 0 to its nominal voltage. Can also be set by software.<br>Clear to recognize next reset type. |
| 3 | GF1 | **General-purpose Flag 1**<br>Set by software for general-purpose usage.<br>Cleared by software for general-purpose usage. |
| 2 | GF0 | **General-purpose Flag 0**<br>Set by software for general-purpose usage.<br>Cleared by software for general-purpose usage. |
| 1 | PD | **Power-down mode bit**<br>Set this bit to enter in power-down mode.<br>Cleared by hardware when reset occurs. |
| 0 | IDL | **Idle mode bit**<br>Set this bit to enter in Idle mode.<br>Cleared by hardware when interrupt or reset occurs. |

Reset Value = 10h

## Hardware Watchdog Timer

The WDT is intended as a recovery method in situations where the CPU may be subjected to software upset. The WDT consists of a 14-bit counter and the WatchDog Timer ReSeT (WDTRST) SFR. The WDT is by default disabled from exiting reset. To enable the WDT, user must write 01EH and 0E1H in sequence to the WDTRST, SFR location 0A6H. When WDT is enabled, it will increment every machine cycle while the oscillator is running and there is no way to disable the WDT except through reset (either hardware reset or WDT overflow reset). When WDT overflows, it will drive an output RESET HIGH pulse at the RST-pin.

## Using the WDT

To enable the WDT, user must write 01EH and 0E1H in sequence to the WDTRST, SFR location 0A6H. When WDT is enabled, the user needs to service it by writing to 01EH and 0E1H to WDTRST to avoid WDT overflow. The 14-bit counter overflows when it reaches 16383 (3FFFH) and this will reset the device. When WDT is enabled, it will increment every machine cycle while the oscillator is running. This means the user must reset the WDT at least every 16383 machine cycle. To reset the WDT the user must write 01EH and 0E1H to WDTRST. WDTRST is a write only register. The WDT counter cannot be read or written. When WDT overflows, it will generate an output RESET pulse at the RST-pin. The RESET pulse duration is 96 x $T_{CLK\ PERIPH}$, where $T_{CLK\ PERIPH} = 1/F_{CLK\ PERIPH}$. To make the best use of the WDT, it should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset.

To have a more powerful WDT, a $2^7$ counter has been added to extend the Time-out capability, ranking from 16 ms to 2s at $F_{OSCA}$ = 12 MHz. To manage this feature, refer to WDTPRG register description, Table 101.

**Table 100.** WDTRST Register
WDTRST - Watchdog Reset Register (0A6h)

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

Reset Value = XXXX XXXXb

Write only, this SFR is used to reset/enable the WDT by writing 01EH then 0E1H in sequence.

**Table 101.** WDTPRG Register
WDTPRG - Watchdog Timer Out Register (0A7h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | S2 | S1 | S0 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is undetermined. Do not try to set this bit. |
| 6 | - | |
| 5 | - | |
| 4 | - | |
| 3 | - | |
| 2 | S2 | WDT Time-out select bit 2 |
| 1 | S1 | WDT Time-out select bit 1 |
| 0 | S0 | WDT Time-out select bit 0 |
| | | <u>S2</u> <u>S1</u> <u>S0</u> <u>Selected Time-out</u><br>0 0 0 (214 - 1) machine cycles, 16.3 ms at FOSCA = 12 MHz<br>0 0 1 (215 - 1) machine cycles, 32.7 ms at FOSCA = 12 MHz<br>0 1 0 (216 - 1) machine cycles, 65.5 ms at FOSCA = 12 MHz<br>0 1 1 (217 - 1) machine cycles, 131 ms at FOSCA = 12 MHz<br>1 0 0 (218 - 1) machine cycles, 262 ms at FOSCA = 12 MHz<br>1 0 1 (219 - 1) machine cycles, 542 ms at FOSCA = 12 MHz<br>1 1 0 (220 - 1) machine cycles, 1.05 s at FOSCA = 12 MHz<br>1 1 1 (221 - 1) machine cycles, 2.09 s at FOSCA = 12 MHz |

Reset value = XXXX X000

## WDT During Power-down and Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode the user does not need to service the WDT. There are 2 methods of exiting Power-down mode: by a hardware reset or via a level activated external interrupt which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, servicing the WDT should occur as it normally should whenever the AT89C5131 is reset. Exiting Power-down with an interrupt is significantly different. The interrupt is held low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service routine.

To ensure that the WDT does not overflow within a few states of exiting of power-down, it is better to reset the WDT just before entering power-down.

In the Idle mode, the oscillator continues to run. To prevent the WDT from resetting the AT89C5131 while in Idle mode, the user should always set up a timer that will periodically exit Idle, service the WDT, and re-enter Idle mode.

## ONCE Mode (ON Chip Emulation)

The ONCE mode facilitates testing and debugging of systems using AT89C5131 without removing the circuit from the board. The ONCE mode is invoked by driving certain pins of the AT89C5131; the following sequence must be exercised:

- Pull ALE low while the device is in reset (RST high) and $\overline{\text{PSEN}}$ is high.
- Hold ALE low as RST is deactivated.

While the AT89C5131 is in ONCE mode, an emulator or test CPU can be used to drive the circuit Table 102 shows the status of the port pins during ONCE mode.

Normal operation is restored when normal reset is applied.

**Table 102.** External Pin Status during ONCE Mode

| ALE | PSEN | Port 0 | Port 1 | Port 2 | Port 3 | Port I2 | XTALA1/2 | XTALB1/2 |
|---|---|---|---|---|---|---|---|---|
| Weak pull-up | Weak pull-up | Float | Weak pull-up | Weak pull-up | Weak pull-up | Float | Active | Active |

## Reduced EMI Mode

The ALE signal is used to demultiplex address and data buses on port 0 when used with external program or data memory. Nevertheless, during internal code execution, ALE signal is still generated. In order to reduce EMI, ALE signal can be disabled by setting AO bit.

The AO bit is located in AUXR register at bit location 0. As soon as AO is set, ALE is no longer output but remains active during MOVX and MOVC instructions and external fetches. During ALE disabling, ALE pin is weakly pulled high.

**Table 103.** AUXR Register

AUXR - Auxiliary Register (8Eh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DPU | - | M0 | - | XRS1 | XRS0 | EXTRAM | AO |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | DPU | **Disable Weak Pull Up**<br>Cleared to enabled weak pull up on standard Ports<br>Set to disable weak pull up on standard Ports |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | M0 | **Pulse length**<br>Cleared to stretch MOVX control: the $\overline{RD}$ and the $\overline{WR}$ pulse length is 6 clock periods (default).<br>Set to stretch MOVX control: the $\overline{RD}$ and the $\overline{WR}$ pulse length is 30 clock periods. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | XRS1 | **XRAM Size**<br>XRS1 XRS0 XRAM size<br>0 0 256 bytes<br>0 1 512 bytes<br>1 0 768 bytes<br>1 1 1024 bytes (default) |
| 2 | XRS0 | |
| 1 | EXTRAM | **EXTRAM bit**<br>Cleared to access internal XRAM using MOVX at $\overline{Ri}$ at DPTR.<br>Set to access external memory. |
| 0 | AO | **ALE Output bit**<br>Cleared, ALE is emitted at a constant rate of 1/6 the oscillator frequency (or 1/3 if X2 mode is used) (default).<br>Set, ALE is active only during a MOVX or MOVC instruction is used. |

Reset Value = 0X0X 1100b
Not bit addressable

# Electrical Characteristics

## Absolute Maximum Ratings

| |
|---|
| Ambient Temperature Under Bias: |
| I = industrial ........................................................-40°C to 85°C |
| Storage Temperature .................................... -65°C to + 150°C |
| Voltage on $V_{CC}$ from $V_{SS}$......................................-0.5V to + 6V |
| Voltage on Any Pin from $V_{SS}$ .....................-0.5V to $V_{CC}$ + 0.2V |

Note: Stresses at or above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions may affect device reliability.

## DC Parameters for Standard Voltage

$T_A$ = -40°C to +85°C; $V_{SS}$ = 0V; $V_{CC}$ = 5V ± 10%; F = 0 to 40 MHz

| Symbol | Parameter | Min | Typ(5) | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Input Low Voltage | -0.5 | | 0.2Vcc - 0.1 | V | |
| $V_{IH}$ | Input High Voltage except XTAL1, RST | 0.2 $V_{CC}$ + 0.9 | | $V_{CC}$ + 0.5 | V | |
| $V_{IH1}$ | Input High Voltage, XTAL1, RST | 0.7 $V_{CC}$ | | $V_{CC}$ + 0.5 | V | |
| $V_{OL}$ | Output Low Voltage, ports 1, 2, 3 and 4(6) | | | 0.3<br>0.45<br>1.0 | V<br>V<br>V | $I_{OL}$ = 100 µA(4)<br>$I_{OL}$ = 1.6 mA(4)<br>$I_{OL}$ = 3.5 mA(4) |
| $V_{OL1}$ | Output Low Voltage, port 0, ALE, $\overline{PSEN}$ (6) | | | 0.3<br>0.45<br>1.0 | V<br>V<br>V | $I_{OL}$ = 200 µA(4)<br>$I_{OL}$ = 3.2 mA(4)<br>$I_{OL}$ = 7.0 mA(4) |
| $V_{OH}$ | Output High Voltage, ports 1, 2, 3, 4 and 5 | $V_{CC}$ - 0.3<br>$V_{CC}$ - 0.7<br>$V_{CC}$ - 1.5 | | | V<br>V<br>V | $I_{OH}$ = -10 µA<br>$I_{OH}$ = -30 µA<br>$I_{OH}$ = -60 µA<br>$V_{CC}$ = 5V ± 10% |
| $V_{OH1}$ | Output High Voltage, port 0, ALE, $\overline{PSEN}$ | $V_{CC}$ - 0.3<br>$V_{CC}$ - 0.7<br>$V_{CC}$ - 1.5 | | | V<br>V<br>V | $I_{OH}$ = -200 µA<br>$I_{OH}$ = -3.2 mA<br>$I_{OH}$ = -7.0 mA<br>$V_{CC}$ = 5V ± 10% |
| $R_{RST}$ | RST Pulldown Resistor | 20 | 40 | 200 | kΩ | |
| $I_{IL}$ | Logical 0 Input Current ports 1, 2, 3 and 4 | | | -50 | µA | $V_{IN}$ = 0.45V |
| $I_{LI}$ | Input Leakage Current | | | ±10 | µA | 0.45V < $V_{IN}$ < $V_{CC}$ |
| $I_{TL}$ | Logical 1 to 0 Transition Current, ports 1, 2, 3 and 4 | | | -650 | µA | $V_{IN}$ = 2.0V |
| $C_{IO}$ | Capacitance of I/O Buffer | | | 10 | pF | Fc = 1 MHz<br>$T_A$ = 25°C |
| $I_{PD}$ | Power-down Current | | 100 | 200 | µA | 4.5V < $V_{CC}$ < 5.5V(3) |
| $I_{CC}$ | Power Supply Current | $I_{CCOP}$ = 0.7 Freq (MHz) + 3 mA<br>$I_{CCIDLE}$ = 0.6 Freq (MHz) + 2 mA | | | | $V_{CC}$ = 5.5V (1)(2) |

## DC Parameters for Low Voltage

$T_A$ = -40°C to +85°C; $V_{SS}$ = 0V; $V_{CC}$ = 3.3V ± 10%; F = 0 to 40 MHz

| Symbol | Parameter | Min | Typ[5] | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Input Low Voltage | -0.5 | | 0.2Vcc - 0.1 | V | |
| $V_{IH}$ | Input High Voltage except XTAL1, RST | 0.2 $V_{CC}$ + 0.9 | | $V_{CC}$ + 0.5 | V | |
| $V_{IH1}$ | Input High Voltage, XTAL1, RST | 0.7 $V_{CC}$ | | $V_{CC}$ + 0.5 | V | |
| $V_{OL}$ | Output Low Voltage, ports 1, 2, 3 and 4[6] | | | 0.3<br>0.45<br>1.0 | V<br>V<br>V | $I_{OL}$ = 100 μA[4]<br>$I_{OL}$ = 0.8 mA[4]<br>$I_{OL}$ = 1.6mA[4] |
| $V_{OL1}$ | Output Low Voltage, port 0, ALE, $\overline{PSEN}$ [6] | | | 0.3<br>0.45<br>1.0 | V<br>V<br>V | $I_{OL}$ = 200 μA[4]<br>$I_{OL}$ = 1.6 mA[4]<br>$I_{OL}$ = 3.5 mA[4] |
| $V_{OH}$ | Output High Voltage, ports 1, 2, 3, 4 and 5 | $V_{CC}$ - 0.3<br>$V_{CC}$ - 0.7<br>$V_{CC}$ - 1.5 | | | V<br>V<br>V | $I_{OH}$ = -10 μA<br>$I_{OH}$ = -30 μA<br>$I_{OH}$ = -60 μA<br>$V_{CC}$ = 3.3V ± 10% |
| $V_{OH1}$ | Output High Voltage, port 0, ALE, $\overline{PSEN}$ | $V_{CC}$ - 0.3<br>$V_{CC}$ - 0.7<br>$V_{CC}$ - 1.5 | | | V<br>V<br>V | $I_{OH}$ = -200 μA<br>$I_{OH}$ = -1.6 mA<br>$I_{OH}$ = -3.5 mA<br>$V_{CC}$ = 3.3V ± 10% |
| $R_{RST}$ | RST Pulldown Resistor | 20 | 40 | 200 | kΩ | |
| $I_{IL}$ | Logical 0 Input Current ports 1, 2, 3 and 4 | | | -50 | μA | Vin = 0.45V |
| $I_{LI}$ | Input Leakage Current | | | ±10 | μA | 0.45V < Vin < $V_{CC}$ |
| $I_{TL}$ | Logical 1 to 0 Transition Current, ports 1, 2, 3 and 4 | | | -650 | μA | Vin = 2.0V |
| $C_{IO}$ | Capacitance of I/O Buffer | | | 10 | pF | Fc = 1 MHz<br>$T_A$ = 25°C |
| $I_{PD}$ | Power-down Current | | 50 | 100 | μA | 3.0V < $V_{CC}$ < 3.6V[3] |
| $I_{CC}$ | Power Supply Current | $I_{CCOP}$ = 0.7 Freq (MHz) + 3 mA<br>$I_{CCIDLE}$ = 0.6 Freq (MHz) + 2 mA | | | | $V_{CC}$ = 3.3V [1][2] |
| $V_{PFDP}$ | Power Fail High Level Threshold | | | 3.0 | V | |
| $V_{PFDM}$ | Power Fail Low Level Threshold | 2.2 | | | V | |
| | Power fail hysteresis $V_{PFDP}$ - $V_{PFDM}$ | 0.15 | | | V | |

Notes:
1. Operating $I_{CC}$ is measured with all output pins disconnected; XTAL1 driven with $T_{CLCH}$, $T_{CHCL}$ = 5 ns (see Figure 69.), $V_{IL}$ = $V_{SS}$ + 0.5V,
$V_{IH}$ = $V_{CC}$ - 0.5V; XTAL2 N.C.; $\overline{EA}$ = RST = Port 0 = $V_{CC}$. $I_{CC}$ would be slightly higher if a crystal oscillator used (see Figure 66.).
2. Idle $I_{CC}$ is measured with all output pins disconnected; XTAL1 driven with $T_{CLCH}$, $T_{CHCL}$ = 5 ns, $V_{IL}$ = $V_{SS}$ + 0.5V, $V_{IH}$ = $V_{CC}$ - 0.5V; XTAL2 N.C; Port 0 = $V_{CC}$; $\overline{EA}$ = RST = $V_{SS}$ (see Figure 67).
3. Power-down $I_{CC}$ is measured with all output pins disconnected; $\overline{EA}$ = $V_{CC}$, PORT 0 = $V_{CC}$; XTAL2 NC.; RST = $V_{SS}$ (see Figure 68.). In addition, the WDT must be inactive and the POF flag must be set.
4. Capacitance loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the $V_{OLS}$ of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1 to 0 transitions during bus operation. In the worst cases (capacitive loading 100 pF), the noise pulse on the ALE line may exceed 0.45V with maxi $V_{OL}$ peak 0.6V. A Schmitt Trigger use is not necessary.
5. Typicals are based on a limited number of samples and are not guaranteed. The values listed are at room temperature.

6. Under steady state (non-transient) conditions, $I_{OL}$ must be externally limited as follows:
   Maximum $I_{OL}$ per port pin: 10 mA
   Maximum $I_{OL}$ per 8-bit port:
   Port 0: 26 mA
   Ports 1, 2 and 3: 15 mA
   Maximum total $I_{OL}$ for all output pins: 71 mA
   If $I_{OL}$ exceeds the test condition, $V_{OL}$ may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

**Figure 66.** $I_{CC}$ Test Condition, Active Mode

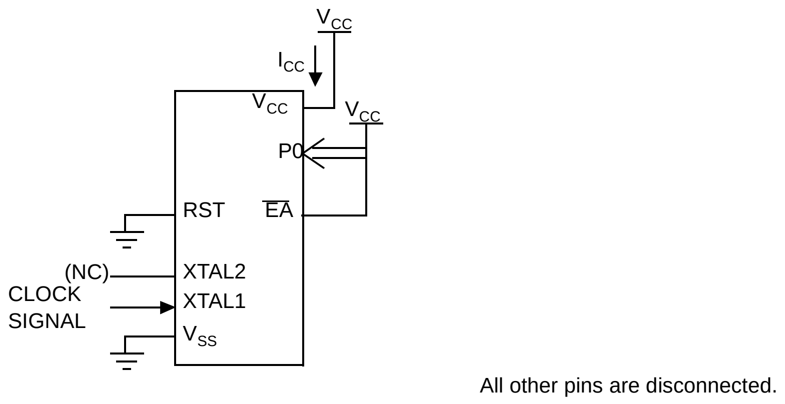


**Figure 67.** $I_{CC}$ Test Condition, Idle Mode

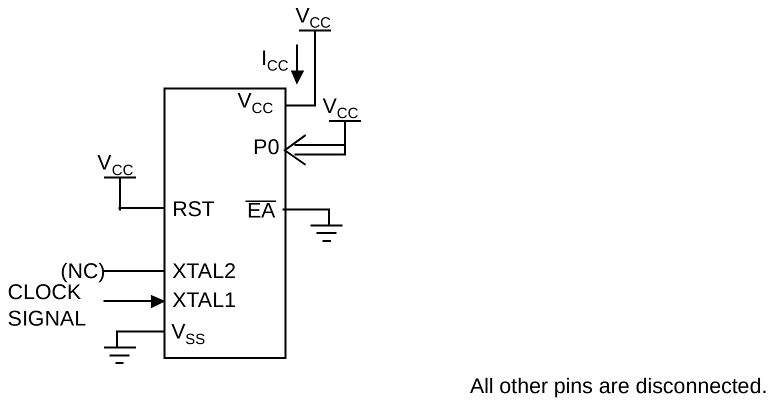


**Figure 68.** $I_{CC}$ Test Condition, Power-down Mode

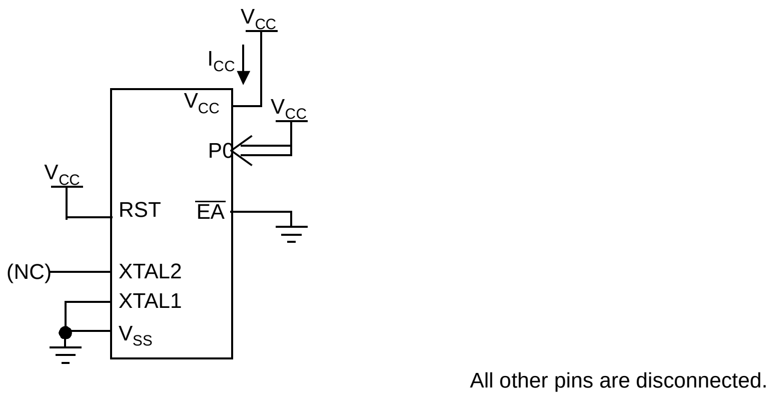
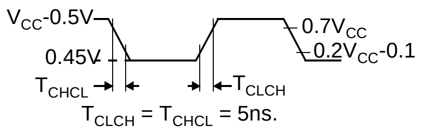
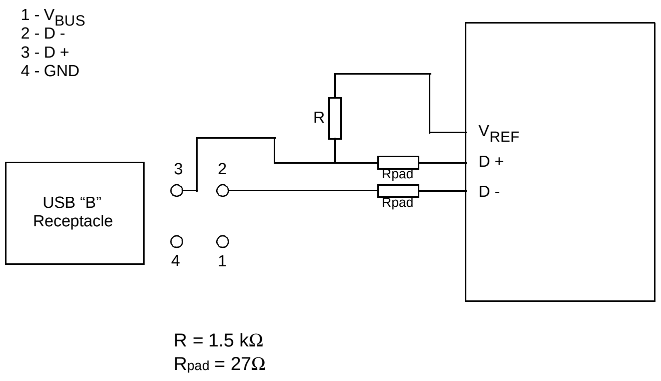


**Figure 69.** Clock Signal Waveform for $I_{CC}$ Tests in Active and Idle Modes



## USB DC Parameters



| Symbol | Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| $V_{REF}$ | USB Reference Voltage | 3.0 | | 3.6 | V |
| $V_{IH}$ | Input High Voltage for D+ and D- (Driven) | 2.0 | | | V |
| $V_{IHZ}$ | Input High Voltage for D+ and D- (Floating) | 2.7 | | 3.6 | V |
| $V_{IL}$ | Input Low Voltage for D+ and D- | | | 0.8 | V |
| $V_{OH}$ | Output High Voltage for D+ and D- | 2.8 | | 3.6 | V |
| $V_{OL}$ | Output Low Voltage for D+ and D- | 0.0 | | 0.3 | V |

## AC Parameters

### Explanation of the AC Symbols

Each timing symbol has 5 characters. The first character is always a “T” (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

Example:$T_{AVLL}$ = Time for Address Valid to ALE Low.
$T_{LLPL}$ = Time for ALE Low to $\overline{\text{PSEN}}$ Low.

$T_A$ = -40°C to +85°C; $V_{SS}$ = 0V; $V_{CC}$ = 5V ±10%; F = 0 to 40 MHz.

$T_A$ = -40°C to +85°C; $V_{SS}$ = 0V; $V_{CC}$ = 5V ± 10%.

(Load Capacitance for port 0, ALE and PSEN = 60 pF; Load Capacitance for all other outputs = 60 pF.)

Table 104, Table 107 and Table 110 give the description of each AC symbols.

Table 105, Table 109 and Table 111 give for each range the AC parameter.

Table 106, Table 109 and Table 112 give the frequency derating formula of the AC parameter for each speed range description. To calculate each AC symbols. take the x value and use this value in the formula.

Example: $T_{LLIV}$ and 20 MHz, Standard clock.
x = 30 ns
T = 50 ns
$T_{CCIV}$ = 4T - x = 170 ns

### External Program Memory Characteristics

**Table 104.** Symbol Description

| Symbol | Parameter |
|---|---|
| T | Oscillator Clock Period |
| $T_{LHLL}$ | ALE Pulse Width |
| $T_{AVLL}$ | Address Valid to ALE |
| $T_{LLAX}$ | Address Hold after ALE |
| $T_{LLIV}$ | ALE to Valid Instruction In |
| $T_{LLPL}$ | ALE to $\overline{\text{PSEN}}$ |
| $T_{PLPH}$ | $\overline{\text{PSEN}}$ Pulse Width |
| $T_{PLIV}$ | $\overline{\text{PSEN}}$ to Valid Instruction In |
| $T_{PXIX}$ | Input Instruction Hold after $\overline{\text{PSEN}}$ |
| $T_{PXIZ}$ | Input Instruction Float after $\overline{\text{PSEN}}$ |
| $T_{AVIV}$ | Address to Valid Instruction In |
| $T_{PLAZ}$ | $\overline{\text{PSEN}}$ Low to Address Float |

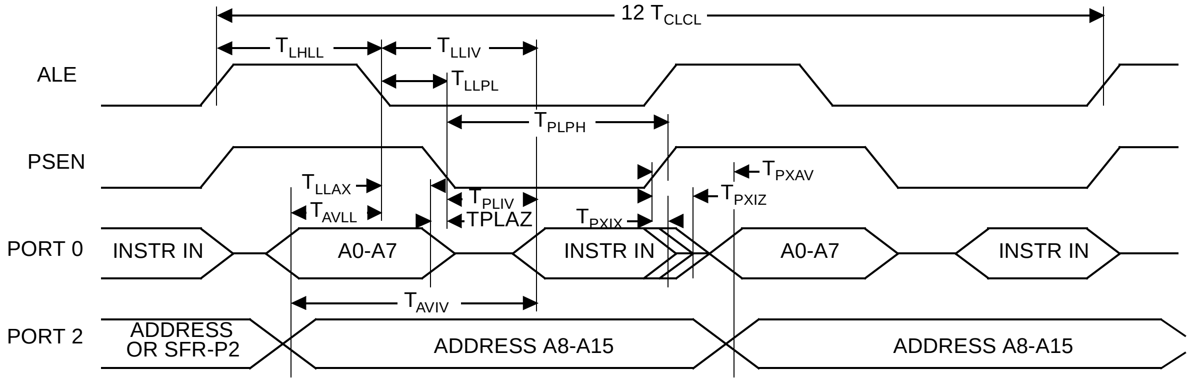
**Table 105.** AC Parameters for a Fix Clock (F = 40 MHz)

| Symbol | Min | Max | Units |
|---|---|---|---|
| T | 25 | | ns |
| $T_{LHLL}$ | 40 | | ns |
| $T_{AVLL}$ | 10 | | ns |
| $T_{LLAX}$ | 10 | | ns |
| $T_{LLIV}$ | | 70 | ns |
| $T_{LLPL}$ | 15 | | ns |
| $T_{PLPH}$ | 55 | | ns |
| $T_{PLIV}$ | | 35 | ns |
| $T_{PXIX}$ | 0 | | ns |
| $T_{PXIZ}$ | | 18 | ns |
| $T_{AVIV}$ | | 85 | ns |
| $T_{PLAZ}$ | | 10 | ns |

**Table 106.** AC Parameters for a Variable Clock

| Symbol | Type | Standard Clock | X2 Clock | X Parameter | Units |
|---|---|---|---|---|---|
| $T_{LHLL}$ | Min | 2 T - x | T - x | 10 | ns |
| $T_{AVLL}$ | Min | T - x | 0.5 T - x | 15 | ns |
| $T_{LLAX}$ | Min | T - x | 0.5 T - x | 15 | ns |
| $T_{LLIV}$ | Max | 4 T - x | 2 T - x | 30 | ns |
| $T_{LLPL}$ | Min | T - x | 0.5 T - x | 10 | ns |
| $T_{PLPH}$ | Min | 3 T - x | 1.5 T - x | 20 | ns |
| $T_{PLIV}$ | Max | 3 T - x | 1.5 T - x | 40 | ns |
| $T_{PXIX}$ | Min | x | x | 0 | ns |
| $T_{PXIZ}$ | Max | T - x | 0.5 T - x | 7 | ns |
| $T_{AVIV}$ | Max | 5 T - x | 2.5 T - x | 40 | ns |
| $T_{PLAZ}$ | Max | x | x | 10 | ns |

## External Program Memory Read Cycle



## External Data Memory Characteristics

**Table 107.** Symbol Description

| Symbol | Parameter |
|---|---|
| $T_{RLRH}$ | $\overline{RD}$ Pulse Width |
| $T_{WLWH}$ | $\overline{WR}$ Pulse Width |
| $T_{RLDV}$ | $\overline{RD}$ to Valid Data In |
| $T_{RHDX}$ | Data Hold After $\overline{RD}$ |
| $T_{RHDZ}$ | Data Float After $\overline{RD}$ |
| $T_{LLDV}$ | ALE to Valid Data In |
| $T_{AVDV}$ | Address to Valid Data In |
| $T_{LLWL}$ | ALE to $\overline{WR}$ or $\overline{RD}$ |
| $T_{AVWL}$ | Address to $\overline{WR}$ or $\overline{RD}$ |
| $T_{QVWX}$ | Data Valid to $\overline{WR}$ Transition |
| $T_{QVWH}$ | Data set-up to $\overline{WR}$ High |
| $T_{WHQX}$ | Data Hold After $\overline{WR}$ |
| $T_{RLAZ}$ | $\overline{RD}$ Low to Address Float |
| $T_{WHLH}$ | $\overline{RD}$ or $\overline{WR}$ High to ALE high |

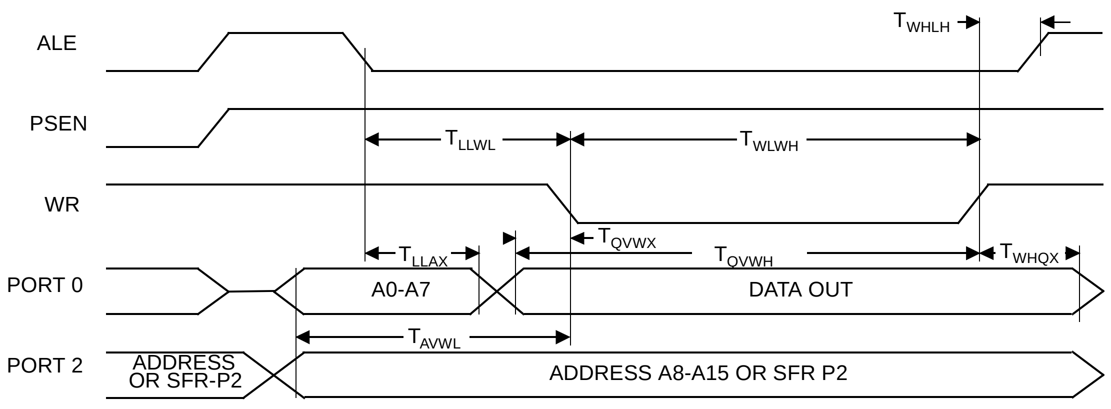
**Table 108.** AC Parameters for a Variable Clock (F = 40 MHz)

| Symbol | Min | Max | Units |
|---|---|---|---|
| $T_{RLRH}$ | 130 | | ns |
| $T_{WLWH}$ | 130 | | ns |
| $T_{RLDV}$ | | 100 | ns |
| $T_{RHDX}$ | 0 | | ns |
| $T_{RHDZ}$ | | 30 | ns |
| $T_{LLDV}$ | | 160 | ns |
| $T_{AVDV}$ | | 165 | ns |
| $T_{LLWL}$ | 50 | 100 | ns |
| $T_{AVWL}$ | 75 | | ns |
| $T_{QVWX}$ | 10 | | ns |
| $T_{QVWH}$ | 160 | | ns |
| $T_{WHQX}$ | 15 | | ns |
| $T_{RLAZ}$ | | 0 | ns |
| $T_{WHLH}$ | 10 | 40 | ns |

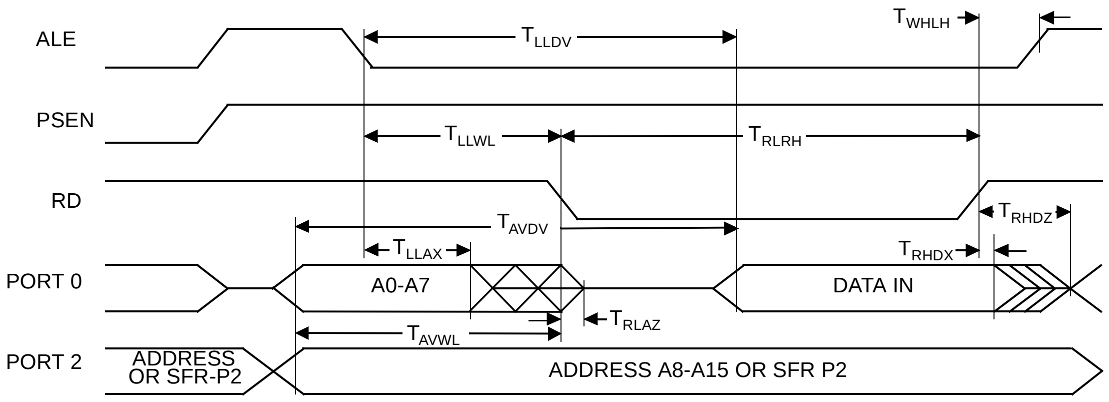
**Table 109.** AC Parameters for a Variable Clock

| Symbol | Type | Standard Clock | X2 Clock | X Parameter | Units |
|---|---|---|---|---|---|
| $T_{RLRH}$ | Min | 6 T - x | 3 T - x | 20 | ns |
| $T_{WLWH}$ | Min | 6 T - x | 3 T - x | 20 | ns |
| $T_{RLDV}$ | Max | 5 T - x | 2.5 T - x | 25 | ns |
| $T_{RHDX}$ | Min | x | x | 0 | ns |
| $T_{RHDZ}$ | Max | 2 T - x | T - x | 20 | ns |
| $T_{LLDV}$ | Max | 8 T - x | 4T -x | 40 | ns |
| $T_{AVDV}$ | Max | 9 T - x | 4.5 T - x | 60 | ns |
| $T_{LLWL}$ | Min | 3 T - x | 1.5 T - x | 25 | ns |
| $T_{LLWL}$ | Max | 3 T + x | 1.5 T + x | 25 | ns |
| $T_{AVWL}$ | Min | 4 T - x | 2 T - x | 25 | ns |
| $T_{QVWX}$ | Min | T - x | 0.5 T - x | 15 | ns |
| $T_{QVWH}$ | Min | 7 T - x | 3.5 T - x | 25 | ns |
| $T_{WHQX}$ | Min | T - x | 0.5 T - x | 10 | ns |
| $T_{RLAZ}$ | Max | x | x | 0 | ns |
| $T_{WHLH}$ | Min | T - x | 0.5 T - x | 15 | ns |
| $T_{WHLH}$ | Max | T + x | 0.5 T + x | 15 | ns |

## External Data Memory Write Cycle



## External Data Memory Read Cycle



## Serial Port Timing - Shift Register Mode

**Table 110.** Symbol Description (F = 40 MHz)

| Symbol | Parameter |
|---|---|
| $T_{XLXL}$ | Serial port clock cycle time |
| $T_{QVHX}$ | Output data set-up to clock rising edge |
| $T_{XHQX}$ | Output data hold after clock rising edge |
| $T_{XHDX}$ | Input data hold after clock rising edge |
| $T_{XHDV}$ | Clock rising edge to input data valid |

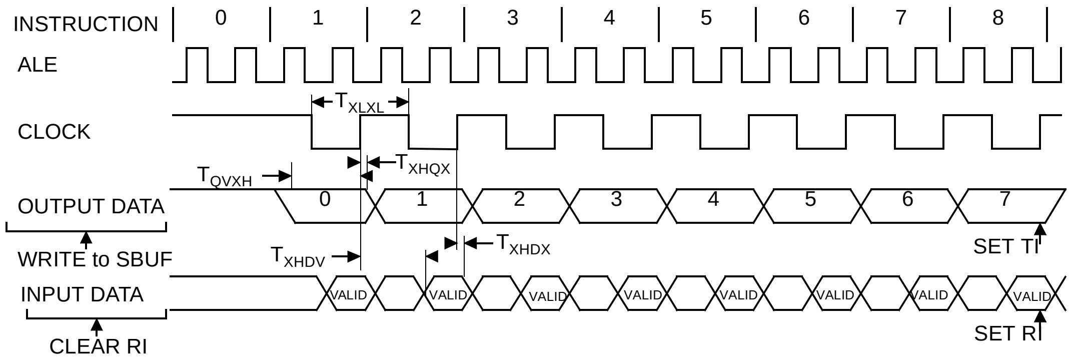
**Table 111.** AC Parameters for a Fix Clock (F = 40 MHz)

| Symbol | Min | Max | Units |
|---|---|---|---|
| $T_{XLXL}$ | 300 | | ns |
| $T_{QVHX}$ | 200 | | ns |
| $T_{XHQX}$ | 30 | | ns |
| $T_{XHDX}$ | 0 | | ns |
| $T_{XHDV}$ | | 117 | ns |

**Table 112.** AC Parameters for a Variable Clock

| Symbol | Type | Standard Clock | X2 Clock | X Parameter for -M Range | Units |
|---|---|---|---|---|---|
| $T_{XLXL}$ | Min | 12 T | 6 T | | ns |
| $T_{QVHX}$ | Min | 10 T - x | 5 T - x | 50 | ns |
| $T_{XHQX}$ | Min | 2 T - x | T - x | 20 | ns |
| $T_{XHDX}$ | Min | x | x | 0 | ns |
| $T_{XHDV}$ | Max | 10 T - x | 5 T- x | 133 | ns |

## Shift Register Timing Waveform

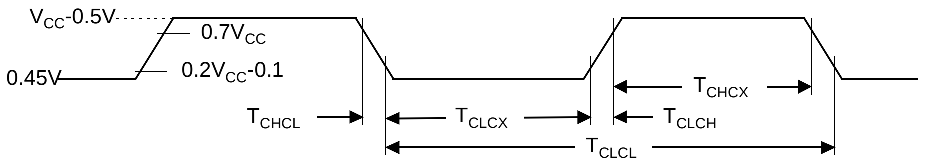


## External Clock Drive Characteristics (XTAL1)

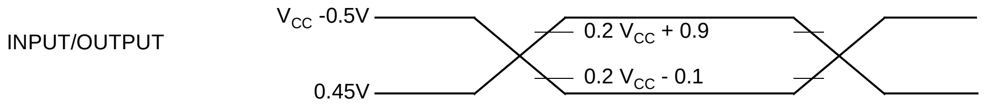
**Table 113.** AC Parameters

| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $T_{CLCL}$ | Oscillator Period | 25 | | ns |
| $T_{CHCX}$ | High Time | 5 | | ns |
| $T_{CLCX}$ | Low Time | 5 | | ns |
| $T_{CLCH}$ | Rise Time | | 5 | ns |
| $T_{CHCL}$ | Fall Time | | 5 | ns |
| $T_{CHCX}/T_{CLCX}$ | Cyclic ratio in X2 mode | 40 | 60 | % |

## External Clock Drive Waveforms

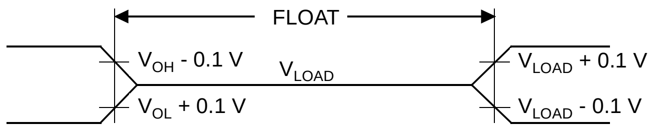


## AC Testing Input/Output Waveforms



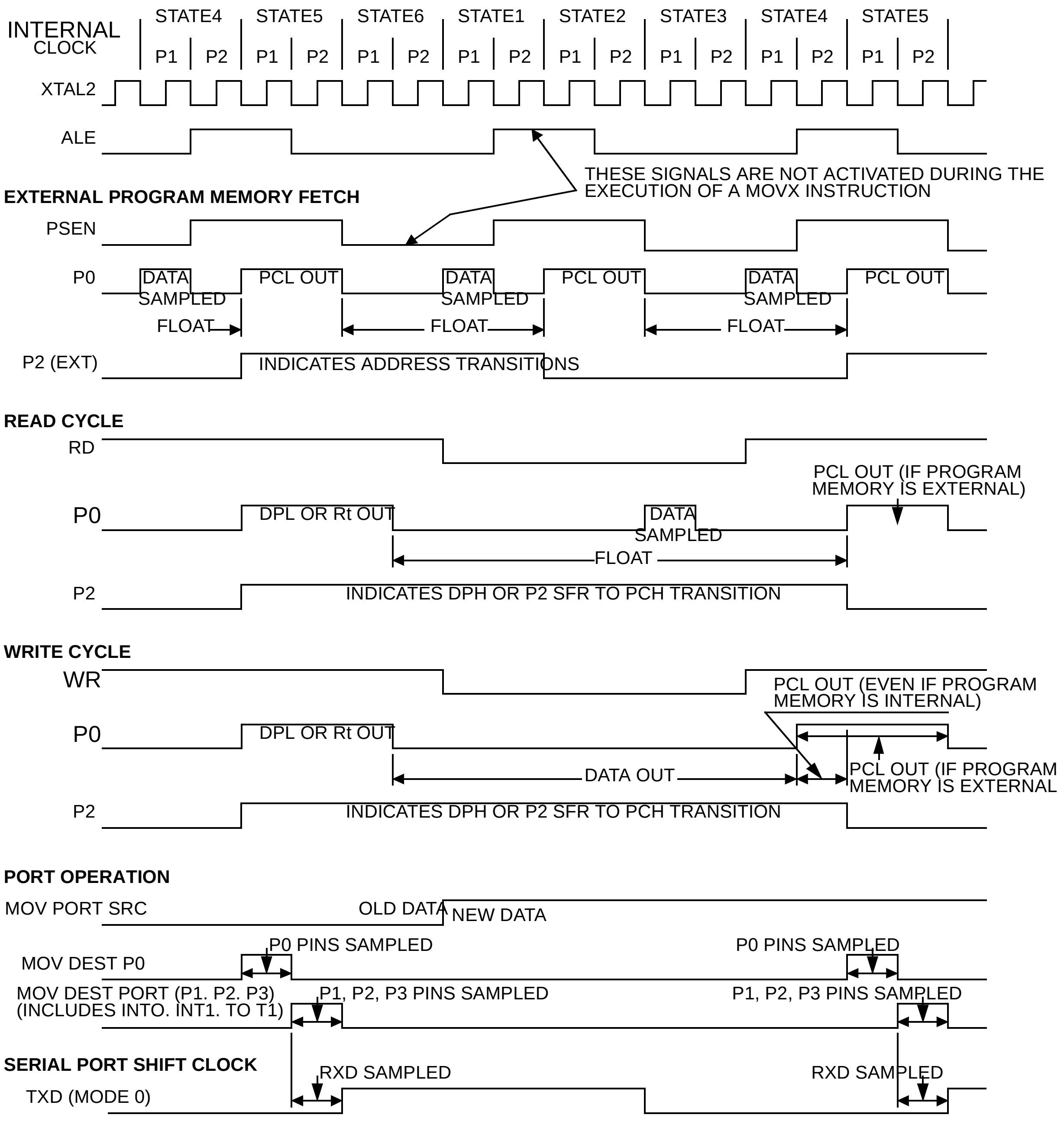
AC inputs during testing are driven at $V_{CC}$ - 0.5 for a logic “1” and 0.45V for a logic “0”. Timing measurement are made at $V_{IH}$ min for a logic “1” and $V_{IL}$ max for a logic “0”.

## Float Waveforms



For timing purposes as 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 loaded $V_{OH}/V_{OL}$ level occurs. $I_{OL}/I_{OH} \geq \pm 20$ mA.

**Clock Waveforms** Valid in normal clock mode. In X2 mode XTAL2 must be changed to XTAL2/2.



INTERNAL CLOCK
STATE4 STATE5 STATE6 STATE1 STATE2 STATE3 STATE4 STATE5
P1 P2
XTAL2
ALE
THESE SIGNALS ARE NOT ACTIVATED DURING THE EXECUTION OF A MOVX INSTRUCTION
EXTERNAL PROGRAM MEMORY FETCH
PSEN
P0
DATA SAMPLED
PCL OUT
FLOAT
P2 (EXT)
INDICATES ADDRESS TRANSITIONS
READ CYCLE
RD
PCL OUT (IF PROGRAM MEMORY IS EXTERNAL)
DPL OR Rt OUT
P2
INDICATES DPH OR P2 SFR TO PCH TRANSITION
WRITE CYCLE
WR
PCL OUT (EVEN IF PROGRAM MEMORY IS INTERNAL)
DATA OUT
PCL OUT (IF PROGRAM MEMORY IS EXTERNAL
PORT OPERATION
MOV PORT SRC
OLD DATA
NEW DATA
MOV DEST P0
P0 PINS SAMPLED
MOV DEST PORT (P1. P2. P3) (INCLUDES INTO. INT1. TO T1)
P1, P2, P3 PINS SAMPLED
SERIAL PORT SHIFT CLOCK
TXD (MODE 0)
RXD SAMPLED

This diagram indicates when signals are clocked internally. The time it takes the signals to propagate to the pins, however, ranges from 25 to 125 ns. This propagation delay is dependent on variables such as temperature and pin loading. Propagation also varies from output to output and component. Typically though ($T_A$ = 25°C fully loaded) $\overline{\text{RD}}$ and $\overline{\text{WR}}$ propagation delays are approximately 50 ns. The other signals are typically 85 ns. Propagation delays are incorporated in the AC specifications.

## Flash Memory

**Table 114.** Timing Symbol Definitions

| Signals | |
|---|---|
| S (Hardware Condition) | $\overline{\text{PSEN}}$, EA |
| R | RST |
| B | FBUSY Flag |

| Conditions | |
|---|---|
| L | Low |
| V | Valid |
| X | No Longer Valid |

**Table 115.** Memory AC Timing
VDD = 5V ± 10%, $T_A$ = -40 to +85°C

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

**Figure 70.** Flash Memory - ISP Waveforms

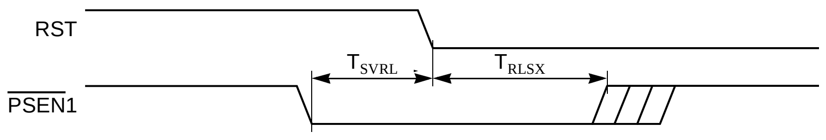
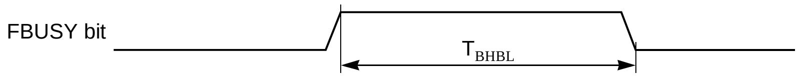


**Figure 71.** Flash Memory - Internal Busy Waveforms



## USB AC Parameters

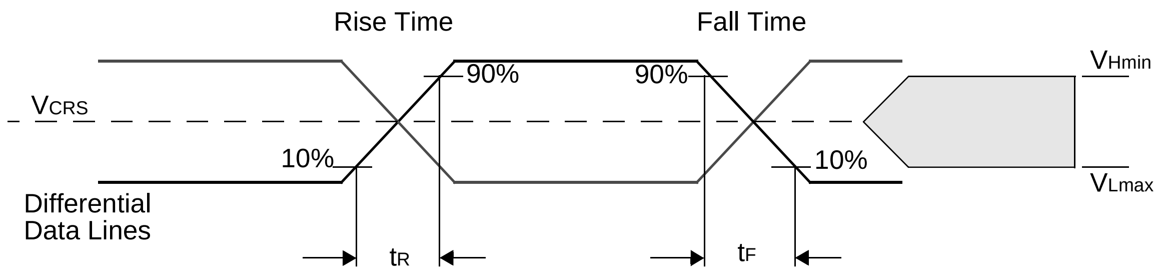


**Table 116.** USB AC Parameters

| Symbol | Parameter | Min | Typ | Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|
| $t_R$ | Rise Time | 4 | | 20 | ns | |
| $t_F$ | Fall Time | 4 | | 20 | ns | |
| $t_{FDRATE}$ | Full-speed Data Rate | 11.9700 | | 12.0300 | Mb/s | |
| $V_{CRS}$ | Crossover Voltage | 1.3 | | 2.0 | V | |
| $t_{DJ1}$ | Source Jitter Total to Next Transaction | -3.5 | | 3.5 | ns | |
| $t_{DJ2}$ | Source Jitter Total for Paired Transactions | -4 | | 4 | ns | |
| $t_{JR1}$ | Receiver Jitter to Next Transaction | -18.5 | | 18.5 | ns | |
| $t_{JR2}$ | Receiver Jitter for Paired Transactions | -9 | | 9 | ns | |

## Ordering Information

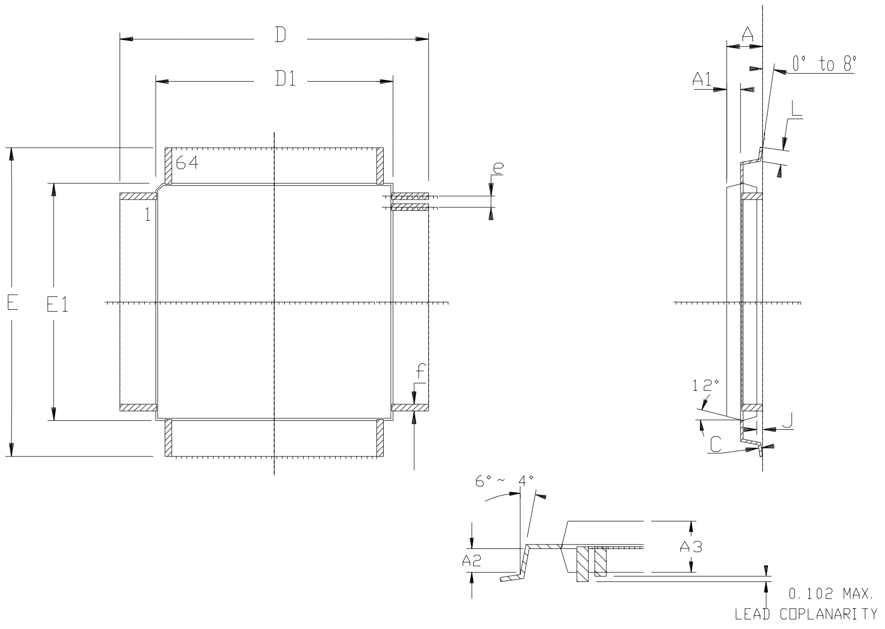
**Table 117.** Possible Order Entries

| Part Number | Memory Size (Kbytes) | Supply Voltage | Temperature Range | Package | Packing |
|---|---|---|---|---|---|
| AT89C5131-RDTEM | 32 | 3.6 to 5.5V | Enginering samples | VQFP64 | Tray |
| AT89C5131-PLSEM | 32 | 3.6 to 5.5V | Enginering samples | MLF48 | Stick |
| AT89C5131-S3SEM | 32 | 3.6 to 5.5V | Enginering samples | PLCC52 | Stick |
| AT89C5131-TISEM | 32 | 3.6 to 5.5V | Enginering samples | SO28 | Stick |
| AT89C5131-RDTIM | 32 | 3.6 to 5.5V | Industrial | VQFP64 | Tray |
| AT89C5131-PLSIM | 32 | 3.6 to 5.5V | Industrial | MLF48 | Stick |
| AT89C5131-S3SIM | 32 | 3.6 to 5.5V | Industrial | PLCC52 | Stick |
| AT89C5131-TISIM | 32 | 3.6 to 5.5V | Industrial | SO28 | Stick |
| AT89C5131-RDTEL | 32 | 3.0 to 3.6V | Enginering samples | VQFP64 | Tray |
| AT89C5131-PLSEL | 32 | 3.0 to 3.6V | Enginering samples | MLF48 | Stick |
| AT89C5131-S3SEL | 32 | 3.0 to 3.6V | Enginering samples | PLCC52 | Stick |
| AT89C5131-TISEL | 32 | 3.0 to 3.6V | Enginering samples | SO28 | Stick |
| AT89C5131-RDTIL | 32 | 3.0 to 3.6V | Industrial | VQFP64 | Tray |
| AT89C5131-PLSIL | 32 | 3.0 to 3.6V | Industrial | MLF48 | Stick |
| AT89C5131-S3SIL | 32 | 3.0 to 3.6V | Industrial | PLCC52 | Stick |
| AT89C5131-TISIL | 32 | 3.0 to 3.6V | Industrial | SO28 | Stick |

Note: 1. Optional Packing and Package options (please consult Atmel sales representative):
-Tape and Reel
-Dry Pack
-Known good dice

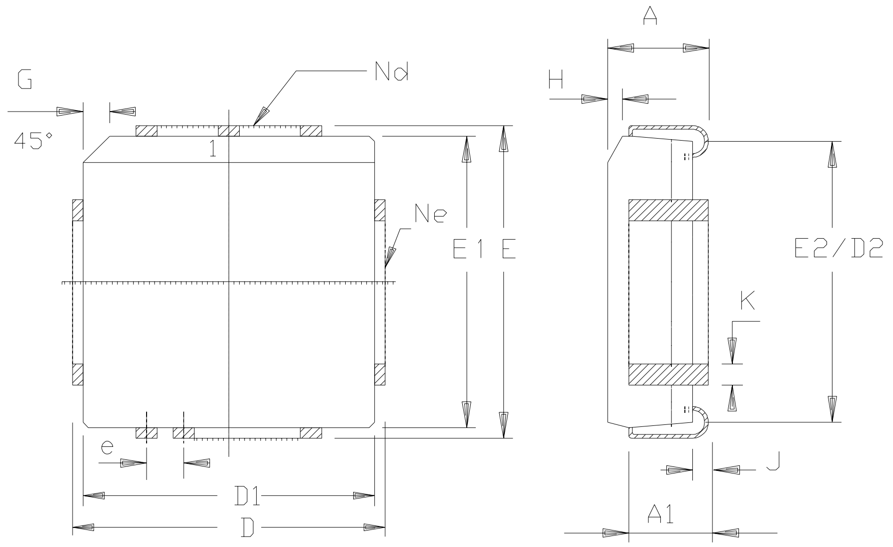
## Packaging Information

### 64-lead VQFP



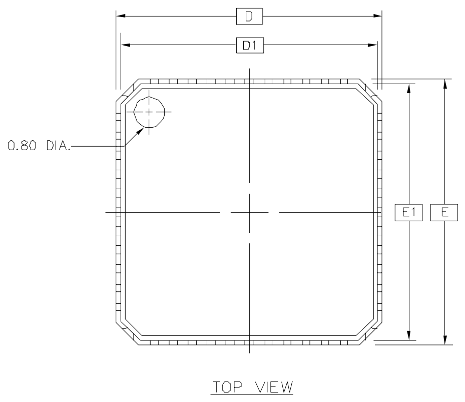
| | MM | | INCH | |
|---|---|---|---|---|
| | Min | Max | Min | Max |
| A | - | 1.60 | - | .063 |
| A1 | 0.64 REF | | .025 REF | |
| A2 | 0.64 REF | | .025 REF | |
| A3 | 1.35 | 1.45 | .053 | .057 |
| D | 11.75 | 12.25 | .463 | .483 |
| D1 | 9.90 | 10.10 | .390 | .398 |
| E | 11.75 | 12.25 | .463 | .483 |
| 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 | |

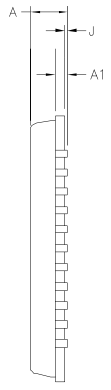
## 52-lead PLCC

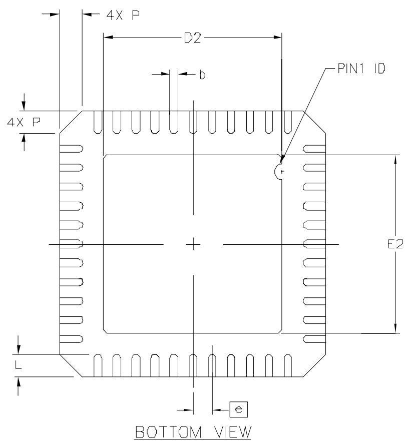


| | MM | | INCH | |
|---|---|---|---|---|
| A | 4.20 | 4.57 | .165 | .180 |
| A1 | 2.29 | 3.30 | .090 | .130 |
| D | 19.94 | 20.19 | .785 | .795 |
| D1 | 19.05 | 19.25 | .750 | .758 |
| D2 | 17.53 | 18.54 | .690 | .730 |
| E | 19.94 | 20.19 | .785 | .795 |
| E1 | 19.05 | 19.25 | .750 | .758 |
| E2 | 17.53 | 18.54 | .690 | .730 |
| 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 | 13 | | 13 | |
| Ne | 13 | | 13 | |
| PKG STD | 00 | | | |

## 48-lead MLF

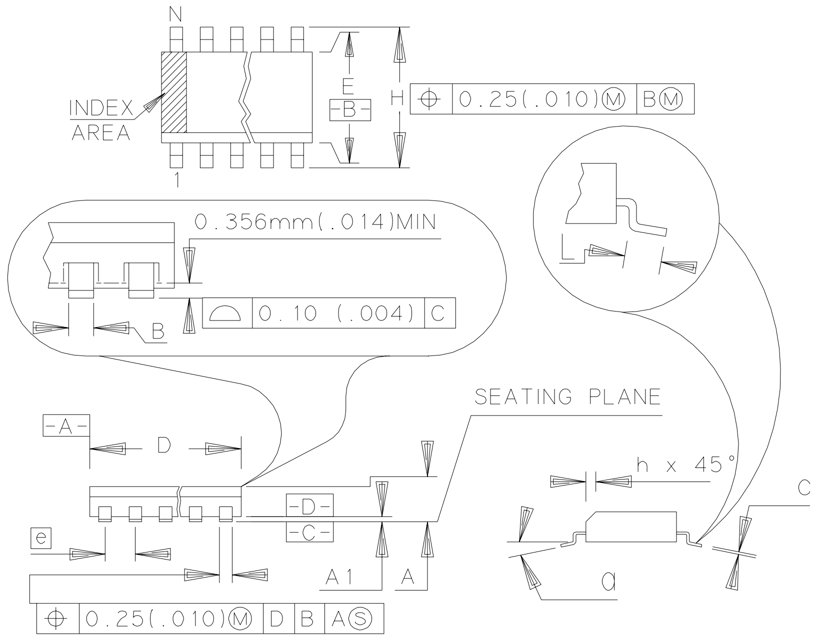






| | MM | | | INCH | | |
|---|---|---|---|---|---|---|
| | MIN | NOM | MAX | MIN | NOM | MAX |
| A | - | 0.85 | 0.90 | - | .033 | .035 |
| J | 0.00 | 0.01 | 0.05 | .000 | .000 | .002 |
| A1 | | 0.20 ref | | | .008 ref | |
| D/E | | 7.00 BSC | | | .276 BSC | |
| D1/E1 | | 6.75 BSC | | | .266 BSC | |
| D2/E2 | 4.95 | 5.10 | 5.25 | .195 | .201 | .207 |
| | | | | | | |
| P | 0.24 | 0.42 | 0.60 | .009 | .016 | .024 |
| e | | 0.50 BSC | | | .020 BSC | |
| L | 0.30 | 0.40 | 0.50 | .012 | .016 | .020 |
| b | 0.18 | 0.23 | 0.30 | .007 | .009 | .012 |

## 28-lead SO



| | MM | | INCH | |
|---|---|---|---|---|
| A | 2.35 | 2.65 | .093 | .104 |
| A1 | 0.10 | 0.30 | .004 | .012 |
| B | 0.35 | 0.49 | .014 | .019 |
| C | 0.23 | 0.32 | .009 | .013 |
| D | 17.70 | 18.10 | .697 | .713 |
| E | 7.40 | 7.60 | .291 | .299 |
| e | 1.27 | BSC | .050 | BSC |
| H | 10.00 | 10.65 | .394 | .419 |
| h | 0.25 | 0.75 | .010 | .029 |
| L | 0.40 | 1.27 | .016 | .050 |
| N | 28 | | 28 | |
| α | 0° | | 8° | |

## Table of Contents

## 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 Datacom*
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80

### *e-mail*
literature@atmel.com

### *Web Site*
http://www.atmel.com