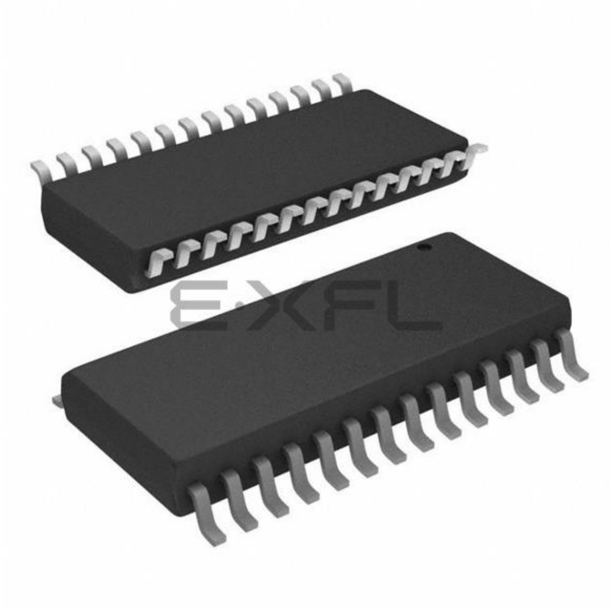


Welcome to E-XFL.COM

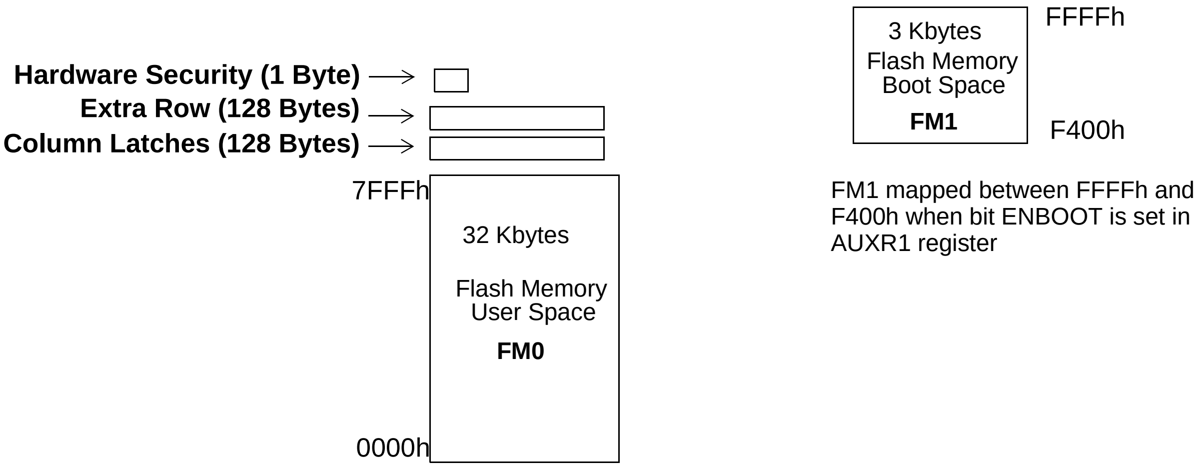
What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	80C51
Core Size	8-Bit
Speed	48MHz
Connectivity	I²C, SPI, UART/USART, USB
Peripherals	LED, POR, PWM, WDT
Number of I/O	18
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	4K x 8
RAM Size	1.25K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at89c5131-tisil

Figure 14. Flash Memory Architecture



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.

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:

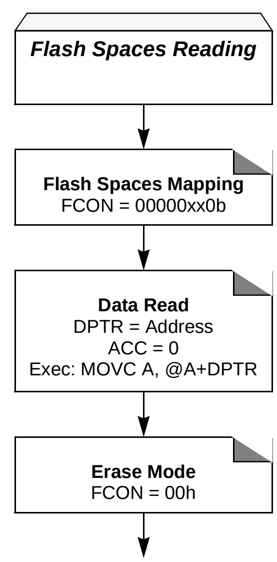
- Map the Extra Row space by writing 02h in FCON register.
- Read one byte in Accumulator by executing MOVC A, @A+DPTR with A = 0 & DPTR = FF80h to FFFFh.

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



Boot Process

Software Boot Process Example

Many algorithms can be used for the software boot process. Below are descriptions of the different flags and Bytes.

Boot Loader Jump bit (BLJB):
- This bit indicates if on RESET the user wants to jump to this application at address @0000h on FM0 or execute the boot loader at address @F400h on FM1.
- BLJB = 0 (i.e. bootloader FM1 executed after a reset) is the default Atmel factory programming.
-To read or modify this bit, the APIs are used.

Boot Vector Address (SBV):
- This byte contains the MSB of the user boot loader address in FM0.
- The default value of SBV is FFh (no user boot loader in FM0).
- To read or modify this byte, the APIs are used.

Extra Byte (EB) & Boot Status Byte (BSB):
- These Bytes are reserved for customer use.
- To read or modify these Bytes, the APIs are used.

Figure 21. Hardware Boot Process Algorithm

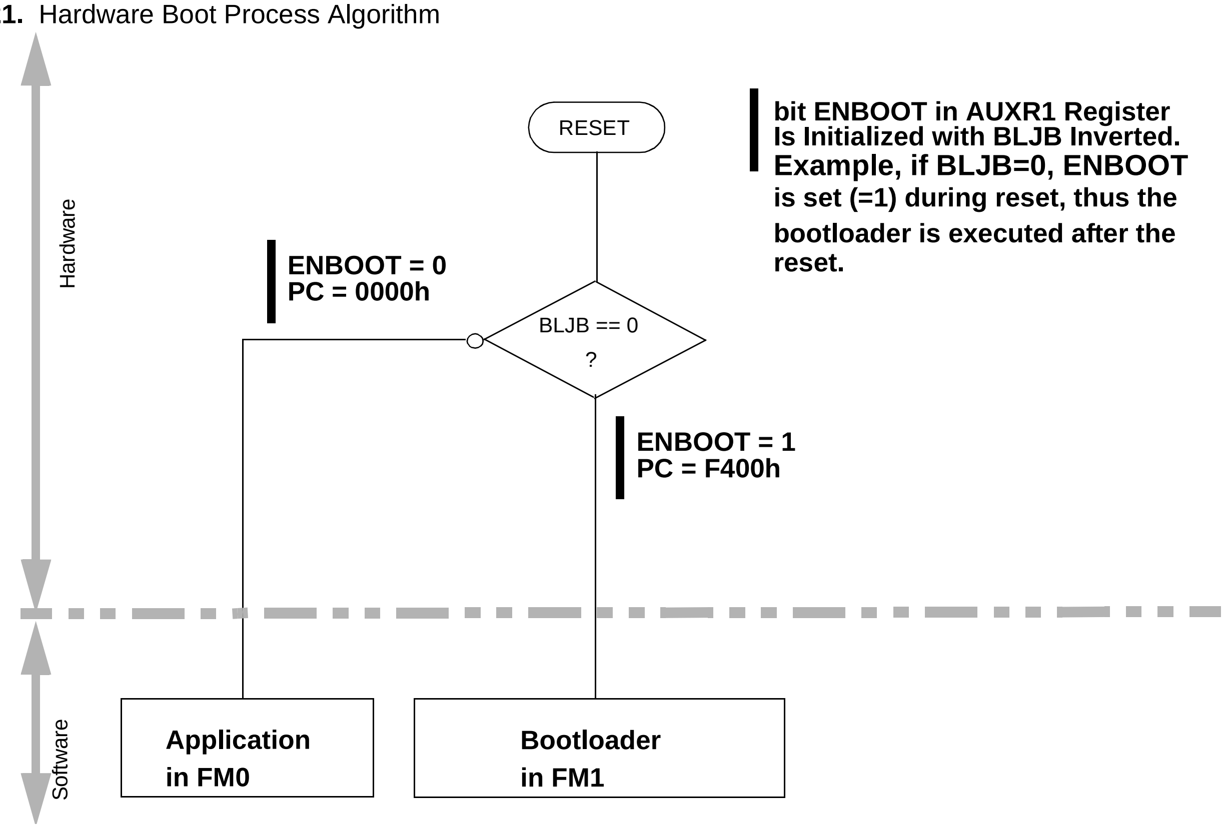


Table 47. T2CON Register

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

Reset Value = 0000 0000b
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 64).

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 pin 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 25). The timer count source is determined from the CPS1 and CPS0 bits in the CMOD register (Table 49) 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)

Table 56. CL Register

CL - PCA Counter Register Low (0E9h)

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

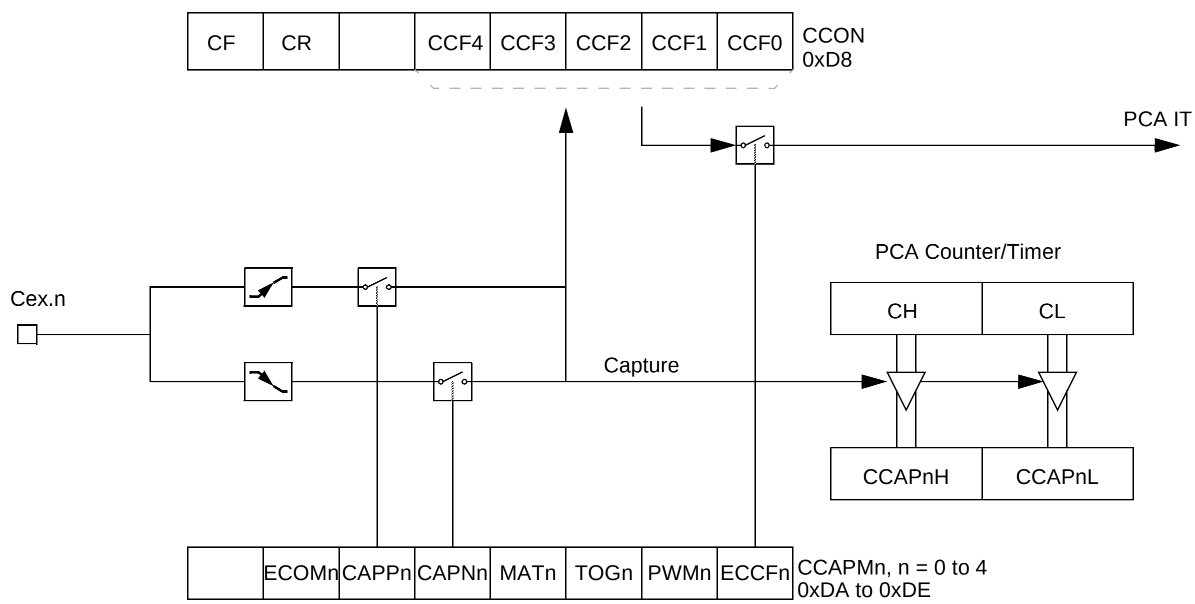
Bit Number	Bit Mnemonic	Description
7 - 0	-	**PCA Counter** CL Value

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 27).

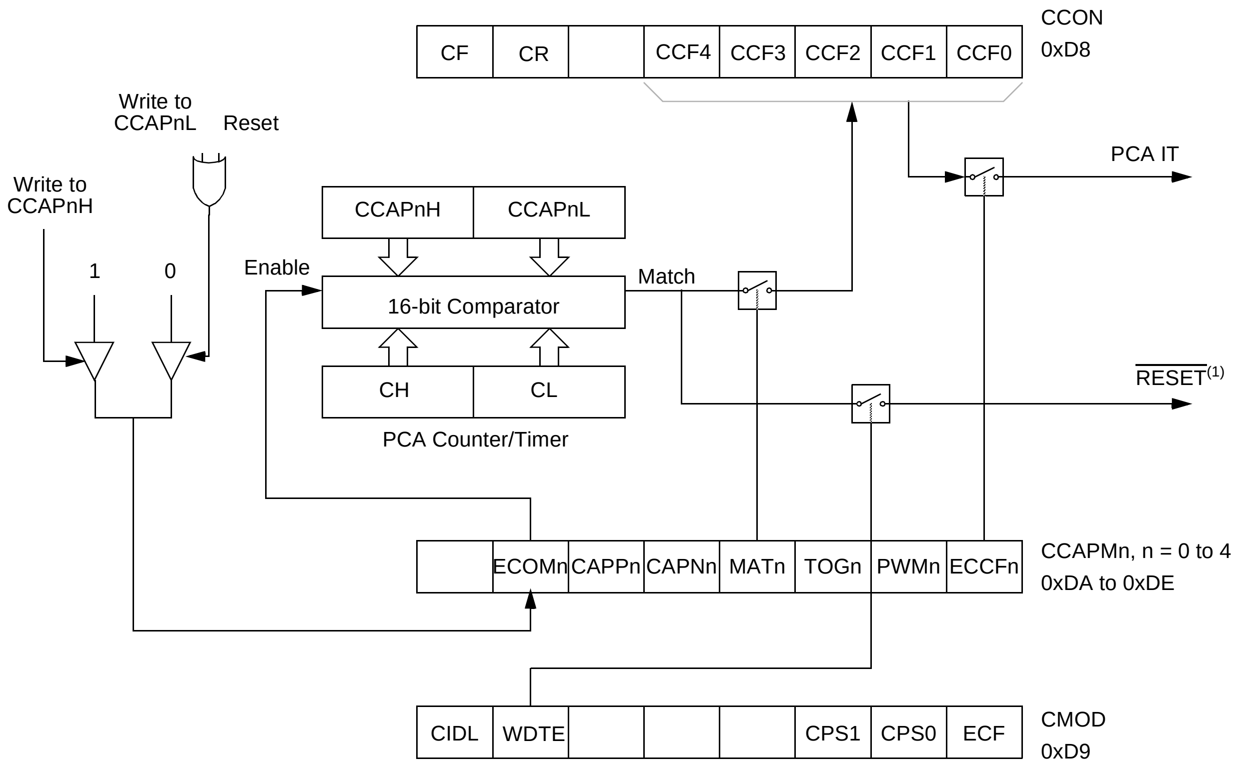
Figure 27. PCA Capture Mode



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 28).

Figure 28. 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 29).

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

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 36.

Figure 36. Interrupt Control System

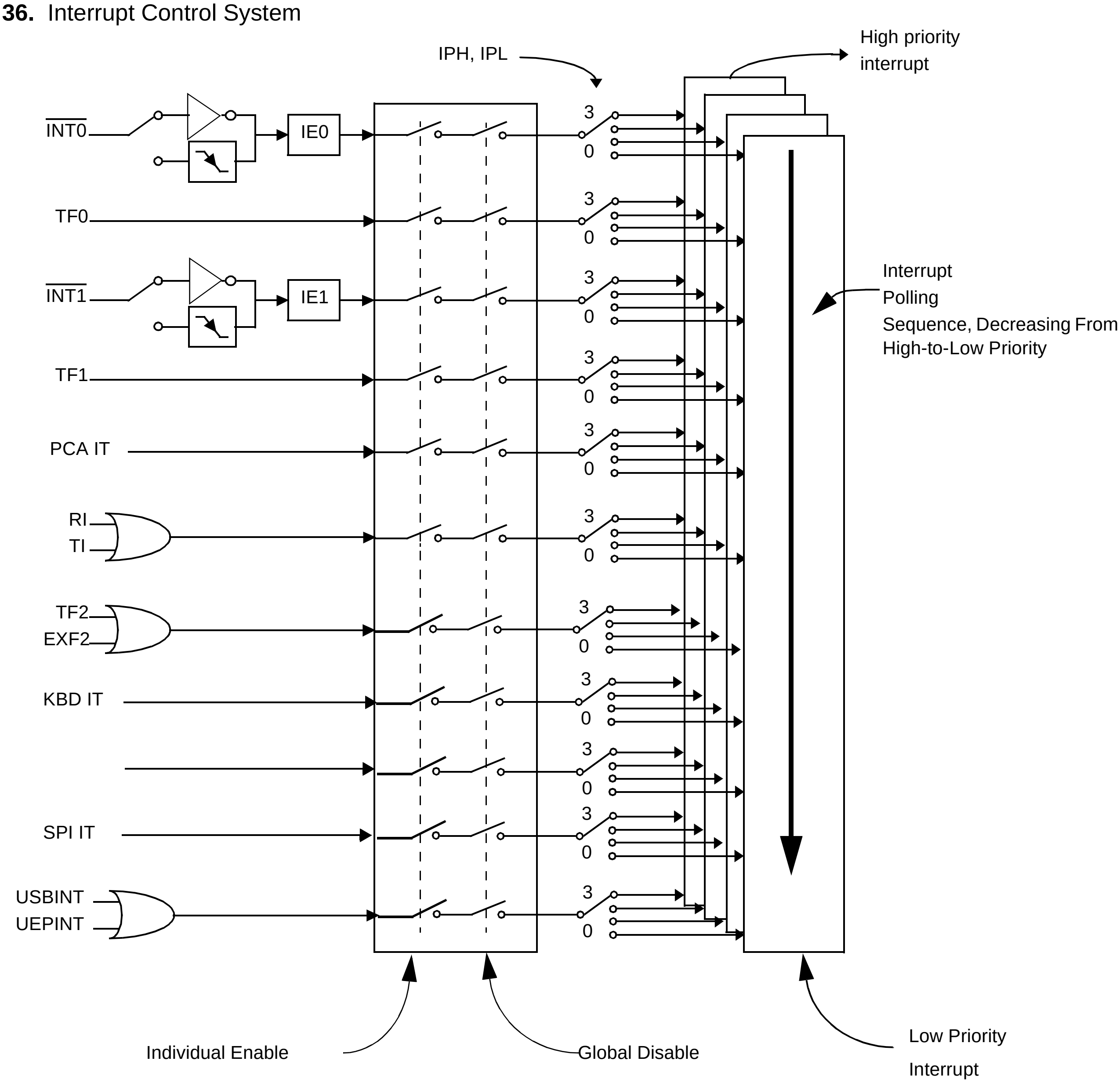


Table 65. IEN1 Register

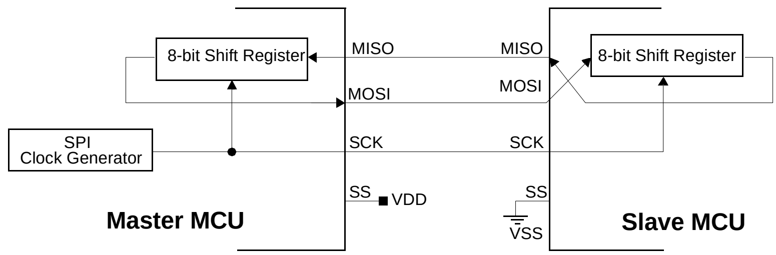
IEN1 - Interrupt Enable Register (B1h)

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** Cleared to disable SPI interrupt. Set to enable SPI interrupt.
1	-	**Reserved**
0	EKB	**Keyboard interrupt Enable bit** Cleared to disable keyboard interrupt. Set to enable keyboard interrupt.

Reset Value = XXXX X000b
Bit addressable

Figure 41. 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 42 and Figure 43). 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').

If the AA bit is reset during a transfer, SSLC will transmit the last byte of the transfer and enter state C0h or C8h. SSLC is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1's as serial data. While AA is reset, SSLC does not respond to its own slave address. However, the TWI bus is still monitored and address recognition may be resume at any time by setting AA. This means that the AA bit may be used to temporarily isolate SSLC from the TWI bus.

Miscellaneous States

There are two SSCS codes that do not correspond to a define SSLC hardware state (see Table 86). These codes are discuss hereafter.

Status F8h indicates that no relevant information is available because the serial interrupt flag is not set yet. This occurs between other states and when SSLC is not involved in a serial transfer.

Status 00h indicates that a bus error has occurred during an SSLC serial transfer. A bus error is caused when a START or a STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions happen during the serial transfer of an address byte, a data byte, or an acknowledge bit. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared. This causes SSLC to enter the not addressed slave mode and to clear the STO flag (no other bits in SSCON are affected). The SDA and SCL lines are released and no STOP condition is transmitted.

Notes

SSLC interfaces to the external TWI bus via two port pins: SCL (serial clock line) and SDA (serial data line). To avoid low level asserting on these lines when SSLC is enabled, the output latches of SDA and SLC must be set to logic 1.

			Bit Frequency (kHz)		
CR2	**CR1**	**CR0**	**F_{OSCA}= 12 MHz**	**F_{OSCA} = 16 MHz**	**F_{OSCA} divided by**
0	0	0	47	62.5	256
0	0	1	53.5	71.5	224
0	1	0	62.5	83	192
0	1	1	75	100	160
1	0	0	12.5	16.5	960
1	0	1	100	-	120
1	1	0	-	-	60
1	1	1	0.5 < . < 62.5	0.67 < . < 83	96 · (256 - reload value Timer 1) (reload value range: 0-254 in mode 2)

Figure 52. Format and State in the Slave Transmitter Mode

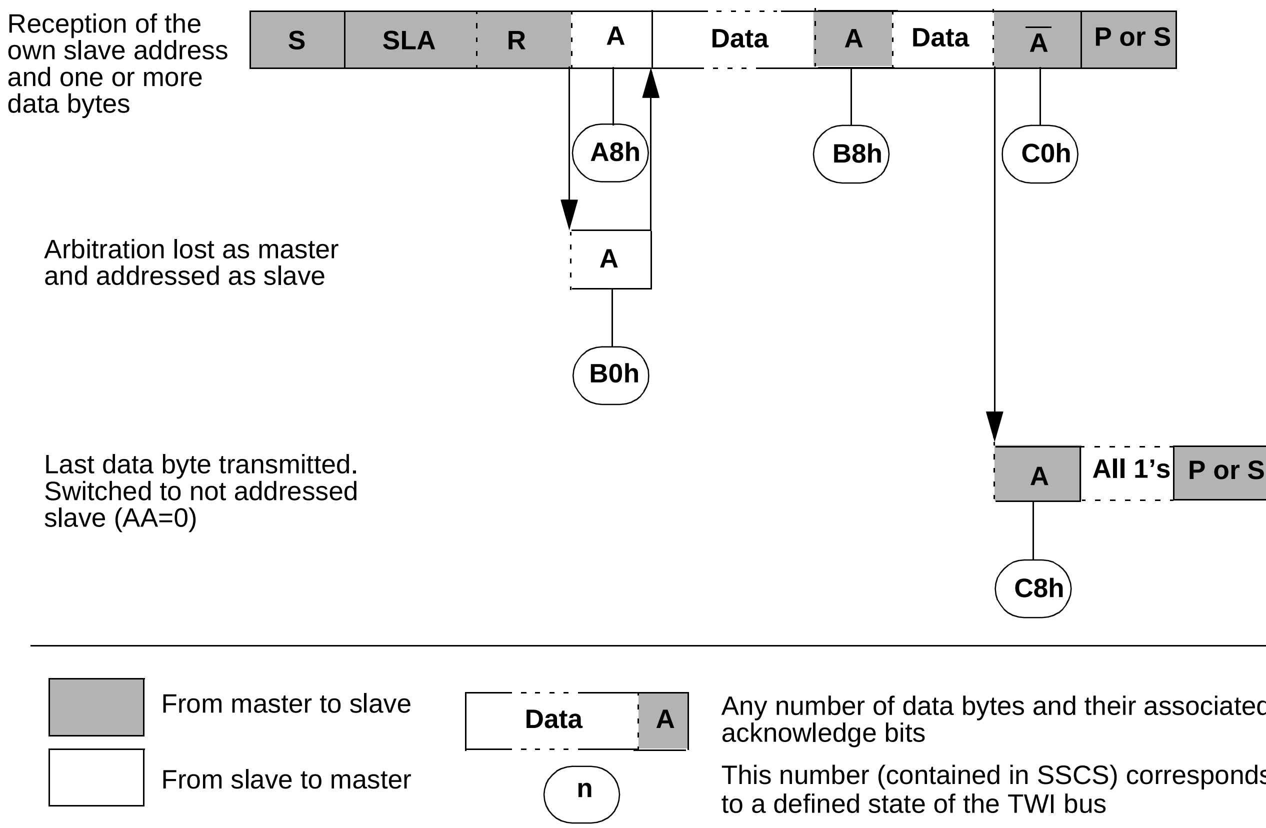


Table 83. Status for Miscellaneous States

Status Code (SSCS)	Status of the TWI bus and TWI hardware	Application Software Response: To/From SSDAT	To SSCON: STA	STO	SI	AA	Next action taken by TWI software
F8h	No relevant state information available; SI= 0	No SSDAT action	No SSCON action				Wait or proceed current transfer
00h	Bus error due to an illegal START or STOP condition	No SSDAT action	0	1	0	X	Only the internal hardware is affected, no STOP condition is sent on the bus. In all cases, the bus is released and STO is reset

Registers

Table 84. SSCON Register

SSCON - Synchronous Serial Control Register (93h)

7	6	5	4	3	2	1	0
CR2	SSIE	STA	STO	SI	AA	CR1	CR0

Bit Number	Bit Mnemonic	Description
7	CR2	**Control Rate bit 2** See .
6	SSIE	**Synchronous Serial Interface Enable bit** Clear to disable SSLC. Set to enable SSLC.
5	STA	**Start flag** Set to send a START condition on the bus.
4	ST0	**Stop flag** Set to send a STOP condition on the bus.
3	SI	**Synchronous Serial Interrupt flag** Set by hardware when a serial interrupt is requested. Must be cleared by software to acknowledge interrupt.
2	AA	**Assert Acknowledge flag** Clear in master and slave receiver modes, to force a not acknowledge (high level on SDA). Clear to disable SLA or GCA recognition. Set to recognise SLA or GCA (if GC set) for entering slave receiver or transmitter modes. Set in master and slave receiver modes, to force an acknowledge (low level on SDA). This bit has no effect when in master transmitter mode.
1	CR1	**Control Rate bit 1** See Table
0	CR0	**Control Rate bit 0** See Table

Table 85. SSDAT (095h) - Synchronous Serial Data Register (read/write)

SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
7	6	5	4	3	2	1	0

Bit Number	Bit Mnemonic	Description
7	SD7	Address bit 7 or Data bit 7.
6	SD6	Address bit 6 or Data bit 6.
5	SD5	Address bit 5 or Data bit 5.
4	SD4	Address bit 4 or Data bit 4.
3	SD3	Address bit 3 or Data bit 3.
2	SD2	Address bit 2 or Data bit 2.

USB Controller

Introduction

The AT89C5131 implements a USB device controller supporting full speed data transfer in accordance with the USB 1.1 and 2.0 Specifications. 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

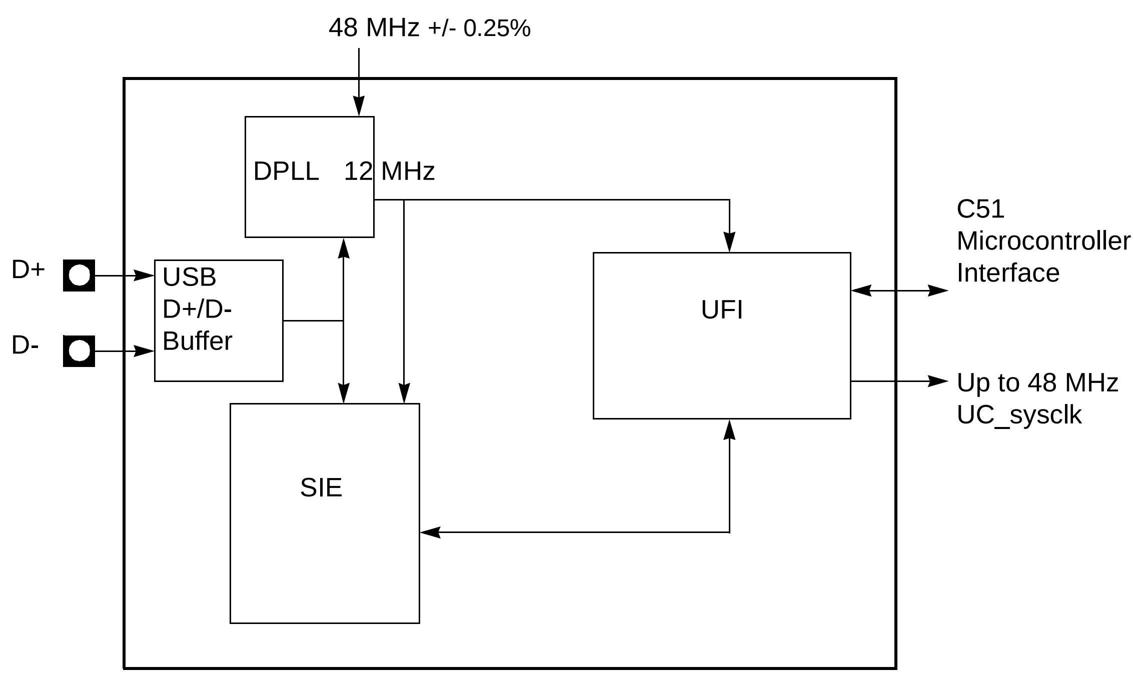
The USB device controller provides the hardware that the AT89C5131 needs to interface a USB link to a data flow stored in a double port memory (DPRAM).

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 53. USB Device Controller Block Diagram



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 55. UFI Block Diagram

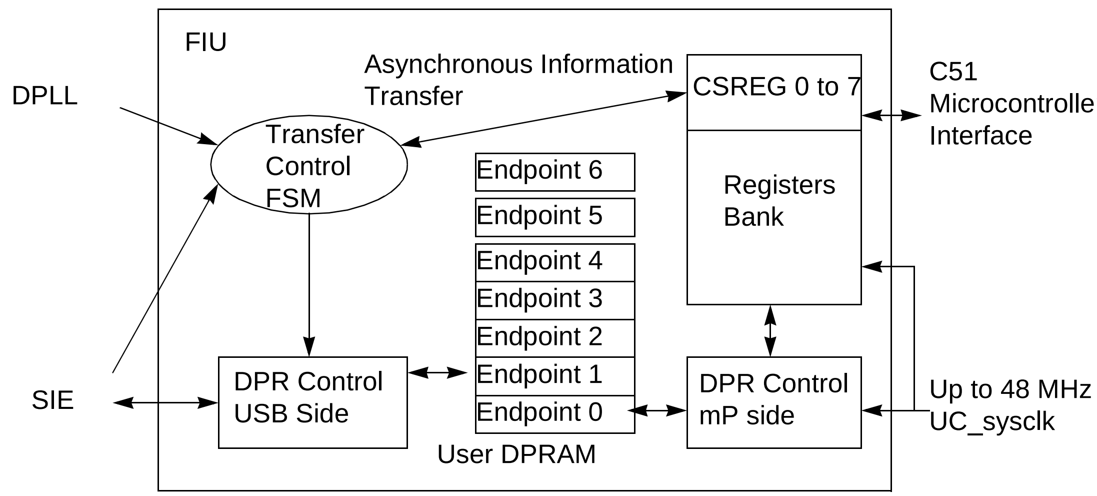
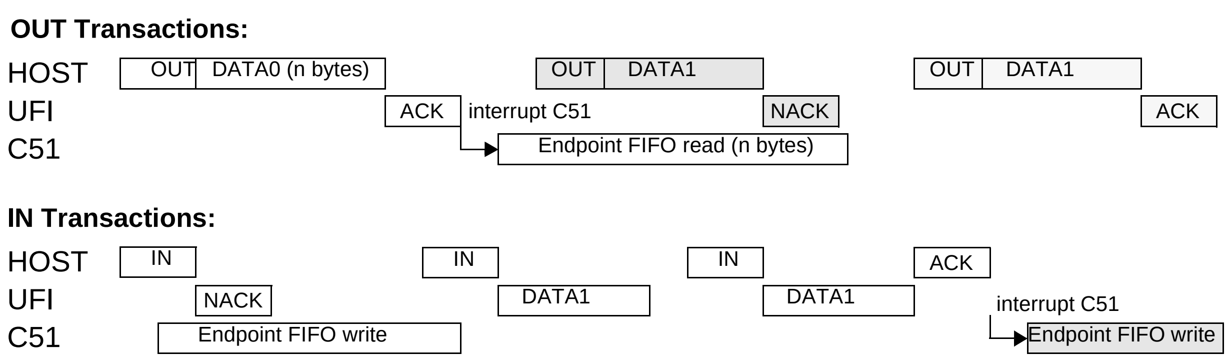
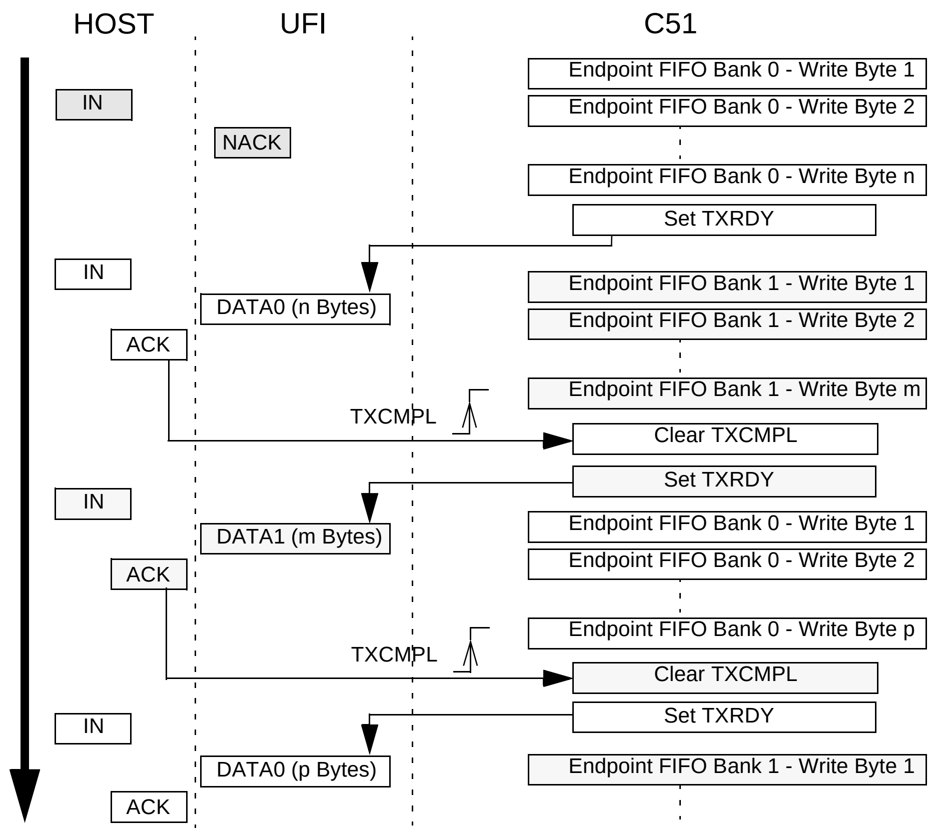


Figure 56. Minimum Intervention from the USB Device Firmware



Bulk/Interrupt IN Transactions in Ping-pong Mode

Figure 62. Bulk/Interrupt IN Transactions in Ping-pong Mode



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.

Table 100. 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** The value read from this bit is always 0. Do not set this bit.
6	EP6RST	**Endpoint 6 FIFO Reset** 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. Then, clear this bit to complete the reset operation and start using the FIFO.
5	EP5RST	**Endpoint 5 FIFO Reset** 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. Then, clear this bit to complete the reset operation and start using the FIFO.
4	EP4RST	**Endpoint 4 FIFO Reset** 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. Then, clear this bit to complete the reset operation and start using the FIFO.
3	EP3RST	**Endpoint 3 FIFO Reset** 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. Then, clear this bit to complete the reset operation and start using the FIFO.
2	EP2RST	**Endpoint 2 FIFO Reset** 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. Then, clear this bit to complete the reset operation and start using the FIFO.
1	EP1RST	**Endpoint 1 FIFO Reset** 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. Then, clear this bit to complete the reset operation and start using the FIFO.
0	EP0RST	**Endpoint 0 FIFO Reset** 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. Then, clear this bit to complete the reset operation and start using the FIFO.

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.

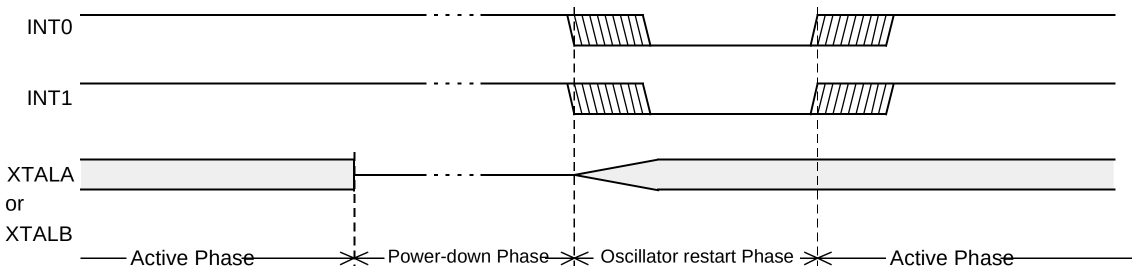
Only:

- 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 69. 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 69. 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 105. 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.