Features

- **USB 2.0 Full Speed Host/Function Processor**
  - **Real-time Host/Function Switching Capability**
  - **Internal USB and System Interface Controllers**
  - **32-bit Generic System Processor Interface with DMA**
  - **Separate TX and RX Buffers for Host and Function Operations**
  - **In-System Firmware Upgrade**
- **Autonomous USB Host Operation without System Processor Intervention**
  - **Device Enumeration**
  - **USB Protocol Management**
  - **Bus Bandwidth Reclamation**
  - **Status Handling**
  - **Control, Bulk, Interrupt and Isochronous Transfers**
  - **Supports Up to 7 USB Devices Concurrently**
- **Full-speed Function Controller**
  - **1 Bi-directional Control Endpoint**
  - **6 Programmable (Maximum Packet Size and Endpoint Type) Endpoints**
  - **Control, Interrupt, Bulk and Isochronous Transfer Support**
  - **Automatic Retry for Non-Isochronous Endpoints**
- **Integrated USB Firmware**
  - **Easy-to-use, ANSI C Compliant API USB Device Driver Development**
  - **Embedded, OS Agnostic USB Host Stack**
  - **Embedded System Interface Controller Driver**
  - **Embedded USB Hub Driver**
- **6 MHz Operation**
- **1.8 V and 3.3 V Operation**
- **100-pin LQFP Package**

Description

Atmel's AT43USB370 is a USB 2.0 compliant, dual-role, full-speed Host/Function processor designed specifically to enable point-to-point USB connectivity for embedded devices. It features an integrated USB host stack, a system interface driver, on-chip USB signaling hardware, 32-bit generic system processor interface with DMA support, and on-the-fly host/function switching capability.

The on-chip USB hardware features a USB transceiver, a serial interface engine (SIE), a SIE controller, and an SOF generation block. It supports the physical and data link layer of the USB protocol whereas the USB transaction layer is implemented in firmware.

In host mode, the integrated USB firmware consists of the Host USB Controller Driver (HUSBCD) running on the USB Controller (USBC) and the Host System Interface Controller Driver (HSICD) resident on the System Interface Controller (SIC). The HUSBCD provides complete USB protocol management including device enumeration, transaction management, scheduling and frame management, and bus reclamation. The HSICD serves as an interface between the HUSBCD and applications resident on the external system processor. It handles all of the high-level data flow management during a USB transaction. Together, the HUSBCD and the HSICD deliver complete USB host operations autonomously, without the intervention of the system processor.

USB 2.0 Full-Speed Host/Function Processor

AT43USB370

Rev. 3340B–USB–12/03

The AT43USB370 communicates with the external system processor through its generic 32-bit host processor interface. This system interface contains 2 Kbytes of FIFO and a DMA engine designed to ensure maximum bus utilization. The automatic USB retry mechanism minimizes data traffic across the system interface.

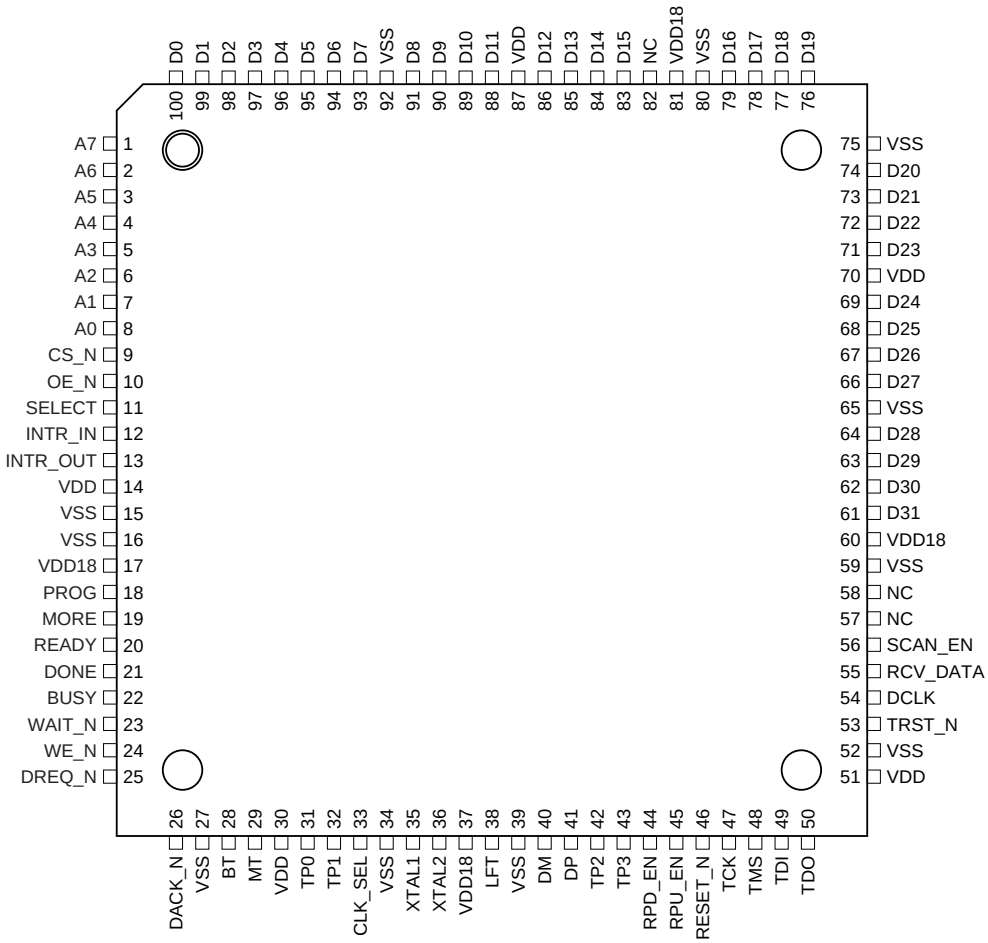
As a function, the AT43USB370 operates in full-speed mode. It supports one control endpoint and a maximum of six programmable (maximum packet size and endpoint type) endpoints. The internal USB controller runs the function firmware that manages USB enumeration and data flow control without system processor intervention.

Communication between the AT43USB370 firmware and applications resident in the system processor is realized through a small set of ANSI C compliant, system interface Application Protocol Interfaces (APIs). This API set encapsulates the complete USB functionality. It is used as the basic building blocks for constructing application specific USB device drivers of any type.

The AT43USB370, with its highly integrated USB hardware/firmware architecture, not only hides the complexity of the traditional USB design, but also frees system resources from being burdened by timing critical USB activities. It is an ideal solution for point-to-point USB connectivity in the resource constrained embedded environment.

Pin Configuration

Figure 1. AT43USB370 100-lead LQFP



Pin Assignment

Pin #	Signal	Type
1	A7	Input
2	A6	Input
3	A5	Input
4	A4	Input
5	A3	Input
6	A2	Input
7	A1	Input
8	A0	Input
9	CS_N	Input
10	OE_N	Input
11	SELECT	Input
12	INTR_IN	Input
13	INTR_OUT	Output
14	VDD	Power Supply/Gnd
15	VSS	Power Supply/Gnd
16	VSS	Power Supply/Gnd
17	VDD18	Power Supply/Gnd
18	PROG	Input
19	MORE	Input
20	READY	Output
21	DONE	Input
22	BUSY	Output
23	WAIT_N	Output
24	WE_N	Input
25	DREQ_N	Output
26	DACK_N	Input
27	VSS	Power Supply/Gnd
28	BT	Input
29	MT	Input
30	VDD	Power Supply/Gnd
31	TP0	Input
32	TP1	Output
33	CLK_SEL	Input
34	VSS	Power Supply/Gnd
35	XTAL1	Input
36	XTAL2	Output
37	VDD18	Power Supply/Gnd
38	LFT	Input
39	VSS	Power Supply/Gnd
40	DM	Bi-directional
41	DP	Bi-directional
42	TP2	Input
43	TP3	Input
44	RPD_EN	Output
45	RPU_EN	Output
46	RESET_N	Input
47	TCK	Input
48	TMS	Input
49	TDI	Input
50	TDO	Output
51	VDD	Power Supply/Gnd
52	VSS	Power Supply/Gnd
53	TRST_N	Input
54	DCLK	Output
55	RCV_DATA	Output
56	SCAN_EN	Input
57	NC	Not Connected
58	NC	Not Connected
59	VSS	Power Supply/Gnd
60	VDD18	Power Supply/Gnd
61	D31	Bi-directional
62	D30	Bi-directional
63	D29	Bi-directional
64	D28	Bi-directional
65	VSS	Power Supply/Gnd
66	D27	Bi-directional
67	D26	Bi-directional
68	D25	Bi-directional
69	D24	Bi-directional
70	VDD	Power Supply/Gnd
71	D23	Bi-directional
72	D22	Bi-directional
73	D21	Bi-directional
74	D20	Bi-directional
75	VSS	Power Supply/Gnd
76	D19	Bi-directional
77	D18	Bi-directional
78	D17	Bi-directional
79	D16	Bi-directional
80	VSS	Power Supply/Gnd
81	VDD18	Power Supply/Gnd
82	NC	Not Connected
83	D15	Bi-directional
84	D14	Bi-directional
85	D13	Bi-directional
86	D12	Bi-directional
87	VDD	Power Supply/Gnd
88	D11	Bi-directional
89	D10	Bi-directional
90	D9	Bi-directional
91	D8	Bi-directional
92	VSS	Power Supply/Gnd
93	D7	Bi-directional
94	D6	Bi-directional
95	D5	Bi-directional
96	D4	Bi-directional
97	D3	Bi-directional
98	D2	Bi-directional
99	D1	Bi-directional
100	D0	Bi-directional

Pin Description

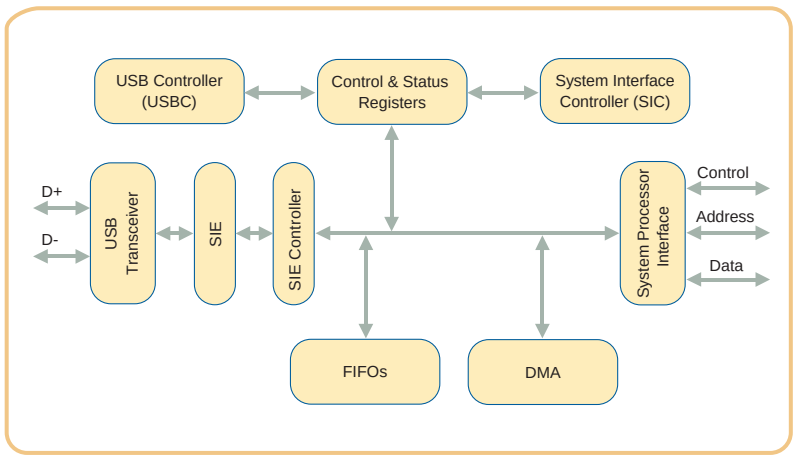
Pin Name	Type	Description
A[7:0]	Input	**ADDRESS BUS** – System Address Bus
CS_N	Input	**CHIP_SELECT** – from System Processor. Active Low
OE_N	Input	**OUTPUT_ENABLE** – from System Processor. Active Low
SELECT	Input	**PROCESSOR_SELECT** – from System Processor – used to select between the USB Controller (USBC) and System Interface Controller (SIC) when firmware is downloaded to these controllers through the System Processor. Logic “1” selects SIC and Logic “0” selects USBC.
INTR_IN	Input	**Interrupt to AT43USB370** – from System Processor. Active High
INTR_OUT	Output	**Interrupt from AT43USB370** – to System Processor. Active High
VDD	Power Supply/Gnd	**3.3V Power Supply**
VSS	Power Supply/Gnd	**Ground**
VDD18	Power Supply/Gnd	**1.8V Power Supply**
PROG	Input	**PROGRAM_LOAD_ENABLE** – from System Processor – used when the program is downloaded in the USB Controller and System Interface Controller through the System Processor. Active High
MORE	Input	**PIO Mode Handshake Signal** – from System Processor. Active High
READY	Output	**READY** – to System Processor – used when the program is downloaded in the USB Controller and System Interface Controller through the System Processor. Active High
DONE	Input	**DONE** – from System Processor – used when the program is downloaded in the USB Controller and System Interface Controller through the System Processor. Active High
BUSY	Output	**BUSY** – to System Processor – used when the System Interface Controller is busy in an interrupt service routine and does not want the System Processor to issue an interrupt. Active High
WAIT_N	Output	**WAIT** – to System Processor. Active Low
WE_N	Input	**WRITE_ENABLE** – from System Processor. Active Low
DREQ_N	Output	**DMA Request** – to System Processor – used to signal to the System Processor that the AT43USB370 wants to start a DMA transfer. Active Low
DACK_N	Input	**DMA Acknowledge** – from System Processor. Active Low
BT	Input	**BIST** – Test Signal
MT	Input	**Memory** – Test Signal
TP0	Input	**Test Pin 0**
TP1	Output	**Test Pin 1**
TP2	Input	**Test Pin 2**
TP3	Input	**Test Pin 3**
CLK_SEL	Input	**External/PLL Clock Selection** – Low selects crystal-PLL clock source while a High uses XTAL1, bypassing the PLL.
XTAL1	Input	**Oscillator Input** – Input to the inverting oscillator amplifier.
XTAL2	Output	**Oscillator Output** – Output of the inverting oscillator amplifier.
LFT	Input	**PLL Loop Filter** – For proper operation of the PLL, this pin should be connected through a 2.2 nF capacitor in parallel with a 470 Ω resistor in series with a 22 nF capacitor to ground (VSS). Both capacitors must be high quality ceramic.

Pin Description (Continued)

Pin Name	Type	Description
DP	Bi-directional	**D+** (USB Line)
DM	Bi-directional	**D-** (USB Line)
RPD_EN	Output	**Pull Down Enable**
RPU_EN	Output	**Pull Up Enable**
RESET_N	Input	**RESET** – Active Low
TCK	Input	**JTAG Clock**
TMS	Input	**JTAG Mode Select**
TDI	Input	**JTAG Serial Data IN**
TDO	Output	**JTAG Serial Data OUT**
TRST_N	Input	**JTAG Reset** – Active Low
DCLK	Output	**Recovered SIE DPLL Clock**
RCV_DATA	Output	**Recovered Serial Data**
SCAN_EN	Input	**Scan Test Enable**
NC	–	**Not Connected**
D[31:0]	Bi-directional	**System Data Bus**

Block Diagram

Figure 2. AT43USB370 Hardware



Architectural Overview

The AT43USB370 host/function processor is available in the SRAM version. It utilizes two on-chip microcontrollers, the USB controller (USBC) and the System Interface Controller (SIC) to off-load USB related processing from the system processor. On power-up or reset, the system processor downloads the appropriate firmware into the USBC and SIC via the system bus interface.

Functionally, the USBC manages the low-level USB protocol such as enumeration, frame management and transaction scheduling, in addition to handling USB hub driver. The SIC provides data flow management to and from the system processor. It is responsible for constructing USB packets of appropriate sizes, handling retries, channeling data to and from FIFOs, and providing API support to the external system processor.

The USBC and the SIC share the same set of on-chip Control and Status Registers with the System Processor Interface. The system interface logic makes use of this register set to facilitate data exchange between the AT43USB370 and the system processor. In a typical design scenario, the AT43USB370 appears as a memory mapped peripheral to the system processor. Externally accessible registers are shown in Table 1 on page 11. The read and write accesses to the system interface registers by the system processor are made through external memory operations on the system bus.

The system processor connects to the AT43USB370 through the generic 32-bit system processor interface. The system interface signals consist of an address bus, a data bus, a chip select, a read enable and write enable. The on-chip DMA engine provides maximum data throughput between the system processor and the on-chip USB FIFO blocks. The system processor communicates with the DMA engine through standard DMA signaling.

The embedded USB hardware consists of a USB Transceiver, a Serial Interface Engine (SIE), a SIE Controller, an SOF (Start of Frame) Generation and a FIFO block. The FIFO block is divided into a 128-bytes control endpoint and 2 Kbytes of memory block dynamically configurable to support different data endpoint requirements.

The AT43USB370 can be configured to operate either in host mode or function mode. The mode of operation is determined by writing corresponding values to the specified registers and downloading the corresponding USBC and SIC firmware to the AT43USB370. The AT43USB370 commences its operation once it is configured.

The AT43USB370 requires an external 6-MHz crystal to provide a reference clock frequency for the on-chip PLL. The PLL provides all of internal clock sources required for the AT43USB370.

Please note that the AT43USB370 signals use level detection, not edge detection.

Functional Description

USB Transceiver

A Universal Serial Bus Revision 2.0 compliant transceiver is embedded in the AT43USB370. The transceiver provides the physical layer signaling and is capable of transmitting and receiving serial data at 12 Mbps and 1.5 Mbps. The driver portion of the transceiver is differential while the receiver section is comprised of a differential and two single-ended receivers. Internally, the transceiver interfaces to the Serial Interface Engine. Externally, the transceiver interfaces directly with the USB connectors and cables through external termination resistors.

Serial Interface Engine (SIE)

The SIE is implemented entirely in hardware. It performs the following functions:

- Clock and Data Recovery from incoming USB data stream
- Serial/parallel conversion
- NRZI encoding/decoding
- CRC calculation (generation and checking)
- Generating full-speed and low-speed USB physical layer signaling
- Device connection/disconnection detection
- Token generation (IN, OUT, SOF, SETUP etc.)
- Keep Alive signal for low-speed devices
- Bit stuffing and unstuffing

SIE Controller

This block serves as the interfaces between the SIE and the USBC. It decodes the commands received from the USBC and updates the status after the end of a USB transaction. This block also controls the FIFOs for data and control packets and provides the USB Controller access to these FIFOs for internal data management such as automatic retries for failed transactions.

USB Controller

This internal microcontroller is dedicated to managing the USB Protocol in both the host mode and the function mode. The Control and Status registers of the AT43USB370 are mapped into its data memory for fast and easy access. The firmware running on this controller determines its operating mode, either host or function.

System Interface Controller

This internal microcontroller serves as an interface between the USB Controller and the external system processor. Firmware running on this controller manages the data flow to and from the system processor. It also provides a generic USB device driver interface to system applications.

FIFO

The FIFO block contains one data FIFO block and one control FIFO block. The control FIFO has a 128 bytes of memory which is divided into one TX and one RX control FIFO. AT43USB370 uses this FIFO for the bi-directional control endpoint.

The data FIFO has 2 Kbytes of memory. The FIFO control logic allows for dynamic configuration of the data FIFO. In host mode, the FIFO memory is divided into 1 Kbytes of TX transfer and 1 Kbytes of RX transfer. The HUSBCD uses this memory for storing data packets. In the event of an error during a USB transaction, the SIE controller is informed of the error and the transaction is retried.

In function mode, the FIFO is divided into two 1 Kbytes blocks, one for the IN endpoints and one for the OUT endpoints. Each of the 1 Kbytes endpoint block can then be dynamically configured during runtime to support up to 3 endpoint in the same direction, but of varying maximum packet sizes.

Control and Status Registers

This block is used to configure the AT43USB370 at the start of operation. The USBC and the SIC share this register set with the system processor interface logic.

By default this block is pre-configured for Host operation with the DMA enabled for the 32-bit data bus. In function mode, this block is used to define the number and nature of the endpoints for the function. A maximum of 3 IN and 3 OUT endpoints can be specified aside from the bi-directional control endpoint. Endpoint type and, maximum packet size and other parameters are also defined using this block.

A subset of the Control and Status register set, the System Processor Interface registers, is accessible by the system processor as external memory locations. It is used to facilitate data exchange between the system processor and the AT43USB370.

System Processor Interface

The system processor interface provides 32-bit bi-directional data paths to the external processor for read and write operations to the AT43USB370's System Interface registers and FIFO. The AT43USB370 appears as a memory mapped peripheral to the external system processor. The interface logic requires a number of control lines and an 8-bit address bus.

DMA

The DMA engine provides DMA support for the system processor to transfer data between the processor's memory and the AT43USB370's internal FIFO. The system processor's DMA controller controls the DMA operation through standard DMA Request and Acknowledgement signals. The AT43USB370 can only operate as a DMA slave.

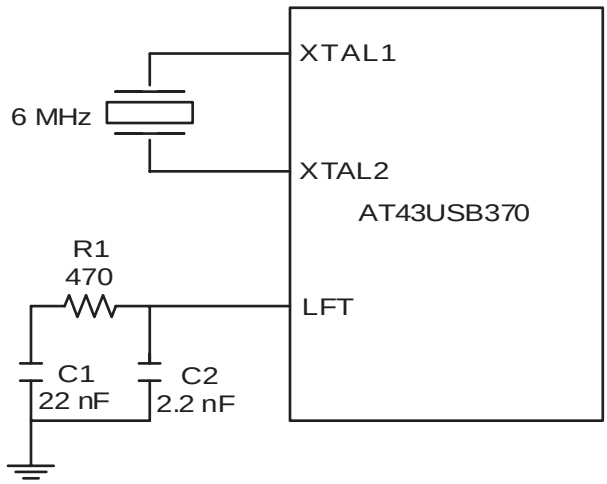
Oscillator and PLL

XTAL1 and XTAL2 are the clock pins to the AT43USB370. An external oscillator or a crystal can be connected to these pins. All clock signals required to operate the AT43USB370 are derived from the on-chip PLL. The on-chip PLL is of a special, low-drive type, designed to operate with most of the 6-Mhz crystals without any external components. The crystal must be of the parallel resonance type requiring a load capacitance of about 10 pF. If the crystal requires a higher value capacitance, external capacitors can be added to the two terminals of the crystal and ground to meet the required value. To assure a quick start-up, a crystal with a high Q, or low ESR, should be used.

The 48-MHz clock can also be externally sourced. In this case, the clock source is connected to XTAL1 pin with XTAL2 pin left open and the CLK_SEL pin tied to logic “1”.

For proper operation of the PLL, an external RC filter consisting of a series RC network of 470 Ω and 22 nF in parallel with a 2.2 nF capacitor must be connected from the LFT pin to V_{SS}. Only high-quality ceramic capacitors are recommended. Figure 3 shows the required crystal and external circuitry.

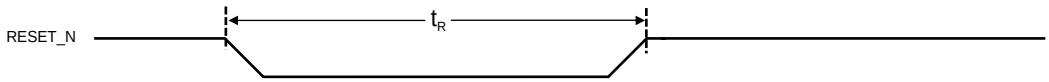
Figure 3. Oscillator and PLL



Reset

The reset signal to the AT43USB370 is active low. On reset, both the USBC and SIC are initialized and the System Processor Interface Registers are restored to their default values. Figure 4 shows the Reset timing diagram. For Reset timing, see Table 5 on page 47.

Figure 4. Reset Timing



Power Supply

The AT43USB370 requires an external supply of 3.3 V and 1.8 V.

Firmware Download Mechanism

The AT43USB370 provides an in-system programming of the USBC and the SIC through the external system processor. Programming requires SELECT, PROG, READY and DONE control signals. These control I/Os are dedicated to in-system firmware download and are not used during normal operation.

The firmware is downloaded in the program memories (SRAM) of the internal controllers after power-up or reset. The SELECT signal is used to select the USBC or the SIC for programming. The PROG signal is used to mark the start and end of firmware download. The READY and DONE signals are used for handshaking during successive programming write cycles.

The programming sequence of an internal controller is described as follows:

1. One of the controllers is selected using the SELECT pin of the AT43USB370. Logic low selects the USBC and logic high selects the SIC. The order of programming of the controllers is immaterial. Any of the controllers can be programmed first.
2. The PROG pin is asserted high by the system processor to indicate the start of programming.
3. The system processor writes to a dummy address (anywhere in the AT43USB370 addressable space of the memory mapping) to assert CS_N and WE_N to AT43USB370.
4. The system processor writes the 32-bit program word on the data bus.
5. The system processor waits for READY to be asserted high by the AT43USB370.
6. AT43USB370 asserts READY logic high to signal to the system processor that the 32-bit data word has been written to the program memory of the selected controller.
7. The system processor asserts the DONE signal high after detecting logic high on READY
8. AT43USB370 asserts READY logic low.
9. The system processor asserts logic low on DONE.
10. This completes one 32-bit write cycle of the controller's programming. Steps 2 to 8 repeated until the entire firmware is downloaded in the program memory of the selected controller.
11. Step 1 is then repeated to select the remaining controller. Step 2 to 9 are repeated to program the remaining controller.
12. The PROG is de-asserted by the system processor once the firmware download is complete. This signals an end of in-system programming of the AT43USB370.

The 32-bit word written by the system processor to its system bus must conform to the following format:

- Bits15:0: Address of the instruction

- Bits 31:16: The actual instruction itself

Both controllers reset internally and start executing the firmware when PROG is de-asserted. The AT43USB370 starts its operation as a USB host or USB function depending upon the firmware downloaded by the system processor.

Figure 5 and Figure 6 illustrate the programming waveform for the USBC and the SIC respectively.

Figure 5. Typical USB Controller's Programming Waveform

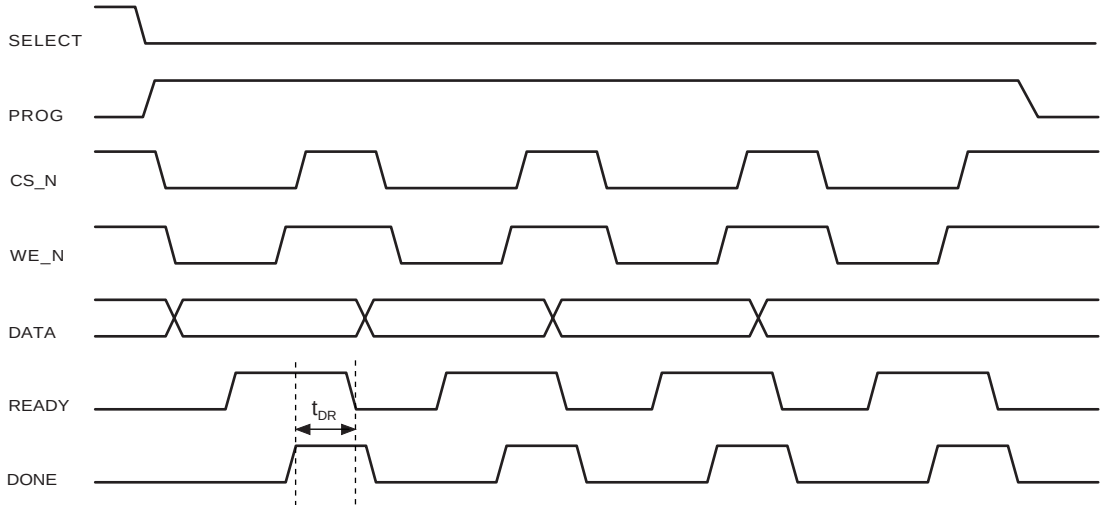
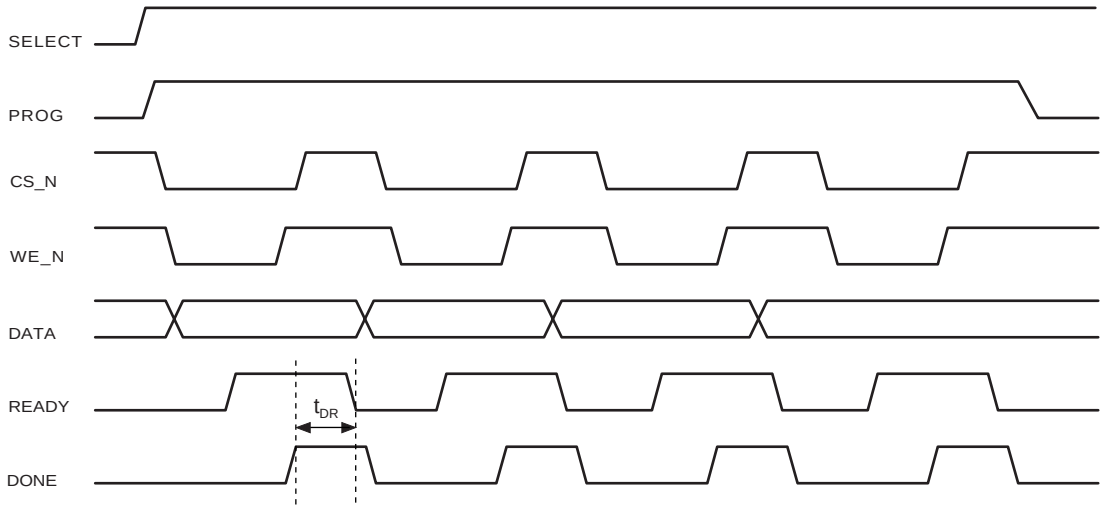


Figure 5 shows the programming waveform for the SIC.

Figure 6. Typical System Interface Controller Programming Waveform



System Processor Interface Register Set

The System Processor Interface register set is used by the AT43USB370 to interact with the system processor. The same register set is used in both the host and the function modes except where explicitly stated. All registers are 32-bit wide and require access on 4-bytes boundaries.

Reading a register for which the external system processor does not have read access will yield a zero value result. Writing to a register for which the external system processor does not have write access has no effect. For detailed usage of the registers, please refer to the *AT43USB370 Software Development Guide*.

Naming Convention

The following naming convention applies to the System Processor Interface Register Set.

- Three different fields in the register name are separated by underscores '_'
- The first field in the register name is a prefix indicating the Write access identification literal:
  - USBP indicates the register is always written by the AT43USB370 USB Processor
  - SYSP indicates the register is always written by the system processor
- The second field in the register name indicates the functionality of the register
- The third field in the register name is a suffix 'REG' common to all the registers

Table 1. System Processor Interface Register Set

S.N.	Address	Host/Device	Name	Function
1	0x00	H/D	SYSP_CMD_REG	Command Register
2	0x04	H/D	USBP_REQ_REG	Request Register
3	0x08	H	SYSP_CMDID_REG	Command ID Register (System Processor)
4	0x0C	H/D	USBP_CMDID_REG	Command ID Register (USB Processor)
5	0x10	H/D	USBP_EXECMDID_REG	Executed Command ID Register
6	0x14	H/D	USBP_CMDRESP_REG	Command Response Register
7	0x18	H	SYSP_DEVADDR_REG	Device Address Register (System Processor)
8	0x1C	H	USBP_DEVADDR_REG	Device Address Register (USB Processor)
9	0x20	H/D	SYSP_EPADDR_REG	Endpoint Address Register (System Processor)
10	0x24	H	USBP_EPADDR_REG	Endpoint Address Register (USB Processor)
11	0x28	H	SYSP_PKTTYPE_REG	Packet Type Register
12	0x2C	H	USBP_CLASSCD_REG	Class Code Register
13	0x30	H	USBP_SCLASSCD_REG	Subclass Code Register
14	0x34	H	USBP_PRTLCD_REG	Protocol Code Register
15	0x38	H	USBP_VENDID_REG	Vendor ID Register
16	0x3C	H	USBP_PRODID_REG	Product ID Register
17	0x40	H	USBP_HUBADDR_REG	Hub's Device Address Register
18	0x44	H	USBP_PORTNUM_REG	Hub's Port Number Register
19	0x48	H	SYSP_PKTSIZE_REG	Packet Size Register
20	0x4C	H	SYSP_RTYCNT_REG	Retry Count Register
21	0x50	H/D	SYSP_XFRMODE_REG	Data Transfer Mode Register

Table 1. System Processor Interface Register Set (Continued)

S.N.	Address	Host/Device	Name	Function
22	0x54	H/D	SYSP_SNDADDR_REG	Send Data Address Register
23	0x58	H/D	SYSP_SNDCNT_REG	Send Data Count Register
24	0x5C	H/D	SYSP_GETADDR_REG	Get Data Address Register
25	0x60	H/D	SYSP_GETCNT_REG	Get Data Count Register
26	0x64	H/D	USBP_XFRADDR_REG	Transfer Address Register
27	0x68	H/D	USBP_XFRCNT_REG	Transfer Count Register
28	0x6C	H	USBP_CNTXFRD_REG	Count Transferred Register
29	0x70	D	SYSP_CMDPARAM_REG	Command Parameter Register
30	0x74	D	USBP_CONGNUM_REG	Device Configuration Number Register
31	0x78	D	USBP_INTRNUM_REG	Device Interface Number Register
32	0x7C	D	USBP_ALSTNUM_REG	Device Alternate Setting Number Register
33	0x80	D	USBP_REQPARM0_REG	Request Parameter 0 Register
34	0x84	D	USBP_REQPARM1_REG	Request Parameter 1 Register
35	0x88	H/D	PRMS_HANDSHAKE_REG	Parameters Handshake Register
36	0x8C - 0xFE		Reserved	Unused
37	0xFF	H/D	SYSP_FIFODATA_REG	FIFO Data Access Register

Register Set Description

SYSP_CMD_REG – Command Register

The system processor software writes in this register.

Bit	31			8	7	0	
0x00	**RS**				**CMD_VALUE**		**SYSP_CMD_REG**
Read/Write	USB Processor	R	System Processor		W		
Initial Value	0x0						
Mode	Host/Device						

- **Bit 7:0 - CMD_VALUE**

Unique value of the command issued by the system processor software. The following values are valid.

- **Bit 31:8 - RS**

Reserved. Must be reset to zero by the system processor software.

This register is used by the system processor software to write the command code while issuing a command to the USB processor. After power-up or reset, this register will contain the value of 0x00.

USBP_REQ_REG – Request Register

The USB processor writes in this register.

Bit	31			8	7	0	
0x04	**RS**				**REQ_VALUE**		**USBP_REQ_REG**
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Host/Device						

- **Bit 7:0 - REQ_VALUE**

Unique value of the request issued by the USB processor. The following definitions are valid.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the request code while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

SYSP_CMDID_REG – Command ID Register (System Processor)

The system processor software writes in this register.

Bit	31			8	7	0	
0x08	**RS**				**SCMD_ID**		**SYSP_CMDID_REG**
Read/Write	USB Processor	R	System Processor		W		
Initial Value	0x0						
Mode	Host						

- **Bit 7:0 - SCMD_ID**

Command ID of the command referenced by the system processor software.

- **Bit 31:8 - RS**

Reserved. Must be reset to zero by the system processor software.

This register is used by the system processor software to write the Command ID of a particular command issued earlier. This may be required where a reference to some previous command is required while issuing the command. After power-up or reset, this register will contain the value of 0x00.

USBP_CMDID_REG – Command ID Register (USB Processor)

The USB processor writes in this register.

Bit	31			8	7	0	
0x0C	RS				UCMD_ID		USBP_CMDID_REG
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Host/Device						

- **Bit 7:0 - UCMD_ID**

Command ID of the command issued by the USB processor.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the Command ID in response to a command issued by the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_EXECMDID_REG – Executed Command ID Register

The USB processor writes in this register

Bit	31			8	7	0	
0x10	RS				EXE_CMD_ID		USBP_EXECMDID_REG
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Host/Device						

- **Bit 7:0 - EXE_CMD_ID**

Command ID of the command executed.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the Command ID of a particular command executed by the USB processor. After power-up or reset, this register will contain the value of 0x00.

USBP_CMDRESP_REG – Command Response Register

The USB processor writes in this register.

Bit	31			8	7	0	
0x14	**RS**				**CMD_RESP**		**USBP_CMDRESP_REG**
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Host/Device						

- **Bit 7:0 - CMD_RESP**

Response of the command executed.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

SYSP_DEVADDR_REG – Device Address Register (System Processor)

The system processor software writes in this register.

Bit	31			8	7	0	
0x18	**RS**				**SDEV_ADDR**		**SYSP_DEVADDR_REG**
Read/Write	USB Processor	R	System Processor		W		
Initial Value	0x0						
Mode	Host						

- **Bit 7:0 - SDEV_ADDR**

Device address of the target device.

- **Bit 31:8 - RS**

Reserved. Must be reset to zero by the system processor software.

This register is used by the system processor software to write the address of the device for which a command is being issued to the USB processor. After power-up or reset, this register will contain the value of 0x00.

USBP_DEVADDR_REG – Device Address Register (USB Processor)

The USB processor writes in this register.

Bit	31			8	7	0	
0x1C	RS				UDEV_ADDR		USBP_DEVADDR_REG
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Host						

- **Bit 7:0 - UDEV_ADDR**

Device address of the target device.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the address of the device for which a request is being issued to the system processor software. After power-up or reset, this register will contain the value of 0x00.

SYSP_EPADDR_REG – Endpoint Address Register (System Processor)

The system processor software writes in this register.

Bit	31			4	3	0	
0x20	RS				SEP_ ADDR		SYSP_EPADDR_REG
Read/Write	USB Processor	R	System Processor		W		
Initial Value	0x0						
Mode	Host/Device						

- **Bit 3:0 - SEP_ADDR**

Endpoint address of the target endpoint.

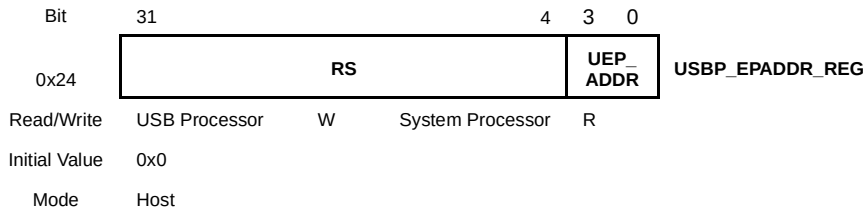
- **Bit 31:4 - RS**

Reserved. Must be reset to zero by the system processor software.

This register is used by the system processor software to specify the endpoint address. After power-up or reset, this register will contain the value of 0x00

USBP_EPADDR_REG – Endpoint Address Register (USB Processor)

The USB processor writes in this register.



- **Bit 3:0 - UEP_ADDR**

Endpoint address of the target endpoint.

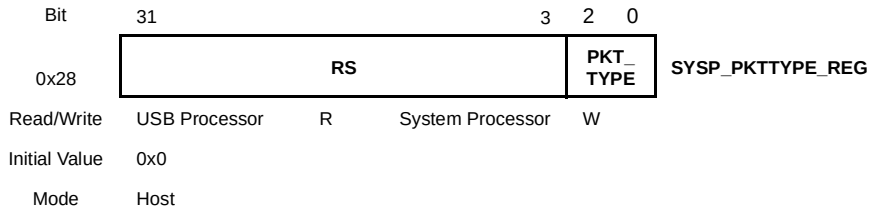
- **Bit 31:4 - RS**

Reserved. Must be reset to zero by the USB processor.

This register is used by the USB processor to specify the endpoint address. After power-up or reset, this register will contain the value of 0x00.

SYSP_PKTTYPE_REG – Packet Type Register

The system processor software writes in this register.



- **Bit 2:0 - PKT_TYPE**

Packet type (IN/OUT/SETUP) with data toggle (0/1) value of the packet. The following definitions are valid.

PKT_TYPE	Value (Hex)	Description
PKT_NO_DT	00	Packet Type - Data Toggle Value Not Specified. Data Toggle will be managed internally by the USB processor
PKT_DATA_0	01	Packet Type - Data Toggle 0
PKT_DATA_1	02	Packet Type - Data Toggle 1
PKT_SETUP	03	Packet Type - Setup

- **Bit 31:3 - RS**

Reserved. Must be reset to zero by the system processor software.

This register is used by the system processor software to write the packet type and data toggle value while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_CLASSCD_REG – Class Code Register

The USB processor writes in this register.

Bit	31			8	7	0	
0x2C	RS				CLASS_CD		USBP_CLASSCD_REG
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Host						

- **Bit 7:0 - CLASS_CD**

Class code value of the device.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the class code value while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_SCLASSCD_REG – Subclass Code Register

The USB processor writes in this register.

Bit	31			8	7	0	
0x30	RS				SCLASS_CD		USBP_SCLASSCD_REG
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Host						

- **Bit 7:0 - SCLASS_CD**

Subclass code value of the device.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the sub-class code value while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_PRTLCD_REG – Protocol Code Register

The USB processor writes in this register.

Bit	31			8	7	0	
0x34	**RS**				**PRTL_CD**		**USBP_PRTLCD_REG**
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Host						

- **Bit 7:0 - PRTL_CD**

Protocol code value of the device.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the protocol code value while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_VENDID_REG – Vendor ID Register

The USB processor writes in this register.

Bit	31			16	15	0	
0x38	**RS**				**VEND_ID**		**USBP_VENDID_REG**
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Host						

- **Bit 15:0 - VEND_ID**

Vendor ID of the USB device.

- **Bit 31:16 - RS**

Reserved. Reset to zero by the HSCID.

This register is used by the USB processor to specify the Vendor ID while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_PRODID_REG – Product ID Register

The USB processor writes in this register.

Bit	31		16	15		0	
0x3C	**RS**			**PROD_ID**			**USBP_PRODID_REG**
Read/Write	USB Processor	W		System Processor	R		
Initial Value	0x0						
Mode	Host						

- **Bit 15:0 - PROD_ID**

Product ID of the USB device.

- **Bit 31:16 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to specify the Product ID while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_HUBADDR_REG – Hub's Device Address Register

The USB processor writes in this register.

Bit	31		8	7		0	
0x40	**RS**			**HUB_ADDR**			**USBP_HUBADDR_REG**
Read/Write	USB Processor	W	System Processor	R			
Initial Value	0x0						
Mode	Host						

- **Bit 7:0 - HUB_ADDR**

Device address of the hub to which the USB device is connected.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the device address of the hub to which a USB device is connected while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_PORTNUM_REG – Hub's Port Number Register

The USB processor writes in this register.

Bit	31			8	7	0	
0x44	RS				PORT_NUM		USBP_PORTNUM_REG
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Host						

- **Bit 7:0 - PORT_NUM**

Port number of the hub to which the USB device is connected.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the port number of the hub to which a USB device is connected while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

SYSP_PKTSIZE – Packet Size Register

The system processor software writes in this register.

Bit	31		16	15		0	
0x48	RS			PKT_SIZE			USBP_PKTSIZE_REG
Read/Write	USB Processor	R		System Processor	W		
Initial Value	0x0						
Mode	Host						

- **Bit 15:0 - PKT_SIZE**

Packet size in bytes.

- **Bit 31:16 - RS**

Reserved. Must be reset to zero by the system processor software.

This register is used by the system processor software to specify the Packet Size while issuing a command to the USB processor. This packet size is used by the USB processor for every transaction associated with this command. After power-up or reset, this register will contain the value of 0x00

SYSP_RTYCNT_REG – Retry Count Register

The system processor software writes in this register.

Bit	31			8	7	0	
0x4C	RS				RTY_CNT		SYSP_RTYCNT_REG
Read/Write	USB Processor	R	System Processor		W		
Initial Value	0x0						
Mode	Host						

- **Bit 7:0 - CMD_VALUE**

Retry Count for every transaction associated with this command.

- **Bit 31:8 - RS**

Reserved. Must be reset to zero by the system processor software.

This register is used by the system processor software to specify the retry count for every transaction associated with this command while issuing a command to the USB processor. After power-up or reset, this register will contain the value of 0x00.

SYSP_XFRMODE_REG – Data Transfer Mode Register

The system processor software writes in this register.

Bit	31			1	0	
0x50	RS			TMODE		SYSP_XFRMODE_REG
Read/Write	HSICD	R	System Processor	W		
Initial Value	0x0					
Mode	Host/Device					

- **Bit 1:0 - TMODE**

Data Transfer Mode.

TMODE	Value (Hex)	Description
XFRMODE_DMA	01	(Data) Transfer Mode - DMA
XFRMODE_DMA	02	(Data) Transfer Mode - Direct FIFO

- **Bit 31:2 - RS**

Reserved. Must be reset to zero by the system processor software.

This register is used by the system processor to specify the mode with which it wants to transfer data while issuing a command to the HSICD. After power-up or reset, this register will contain the value of 0x00.

SYSP_SNDADDR_REG – Send Data Address Register

The system processor software writes in this register.

Bit	31				0	
0x54	**SND_ADDR**					**SYSP_SNDADDR_REG**
Read/Write	USB Processor	R	System Processor	W		
Initial Value	0x0					
Mode	Host/Device					

- **Bit 31:0 - SND_ADDR**

Start Address of the buffer for sending data.

This register is used by the system processor software to specify the start address of the data buffer while issuing a command to the USB processor to transfer data from the system processor software's memory to a USB device. After power-up or reset, this register will contain the value of 0x00.

SYSP_SNDCNT_REG – Send Data Count Register

The system processor software writes in this register.

Bit	31				0	
0x58	**SND_CNT**					**SYSP_SNDCNT_REG**
Read/Write	USB Processor	R	System Processor	W		
Initial Value	0x0					
Mode	Host/Device					

- **Bit 31:0 - SND_CNT**

Count of the buffer for sending data.

This register is used by the system processor software to specify the size of the data buffer while issuing a command to the USB processor to transfer data from the system processor software's memory to the USB device. This is the size of the buffer whose address is specified in Send Data Address Register. After power-up or reset, this register will contain the value of 0x00.

SYSP_GETADDR_REG – Get Data Address Register

The system processor software writes in this register.

Bit	31				0	
0x5C	GET_ADDR					SYSP_GETADDR_REG
Read/Write	USB Processor	R	System Processor	W		
Initial Value	0x0					
Mode	Host/Device					

- **Bit 31:0 - GET_ADDR**

Start Address of the buffer for storing data.

This register is used by the system processor software to specify the start address of the data buffer while issuing a command to the USB processor to transfer data from the USB device to the system processor software's memory. After power-up or reset, this register will contain the value of 0x00.

SYSP_GETCNT_REG – Get Data Count Register

The system processor software writes in this register.

Bit	31				0	
0x60	GET_CNT					SYSP_GETCNT_REG
Read/Write	USB Processor	R	System Processor	W		
Initial Value	0x0					
Mode	Host/Device					

- **Bit 31:0 - GET_CNT**

Count of the data buffer for receiving data.

This register is used by the system processor software to specify the size of the data buffer while issuing a command to the USB processor to transfer data from the USB device to the system processor software's memory. This is the size of the buffer specified in Get Data Address Register. After power-up or reset, this register will contain the value of 0x00.

USBP_XFRADDR_REG – Transfer Address Register

The USB processor writes in this register.

Bit	31				0	
0x64	**XFR_ADDR**					**USBP_XFRADDR_REG**
Read/Write	USB Processor	W	System Processor	R		
Initial Value	0x0					
Mode	Host/Device					

- **Bit 31:0 - XFR_ADDR**

Address for the data transfer.

This register is used by the USB processor to specify the start address of the memory while issuing a request to system processor software to transfer data. After power-up or reset, this register will contain the value of 0x00.

USBP_XFRCNT_REG – Transfer Count Register

The USB processor writes in this register.

Bit	31				0	
0x68	**XFR_CNT**					**USBP_XFRCNT_REG**
Read/Write	USB Processor	W	System Processor	R		
Initial Value	0x0					
Mode	Host/Device					

- **Bit 31:0 - XFR_CNT**

Transfer count in bytes.

This register is used by the USB processor to specify the number of bytes while issuing a request to the system processor software to transfer data. This register specifies the count to be transferred from the location specified in the Transfer Address Register. After power-up or reset, this register will contain the value of 0x00.

USBP_CNTXFRD_REG – Count Transferred Register

The USB processor writes in this register.

Bit	31				0	
0x6C	**CNT_XFRD**					**USBP_CNTXFRD_REG**
Read/Write	USB Processor	W	System Processor	R		
Initial Value	0x0					
Mode	Host					

- **Bit 31:0 - CNT_XFRD**

Count Transferred in bytes.

This register is used by the USB processor to specify the number of bytes actually transferred while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

SYSP_CMDPARAM_REG – Command Parameter Register

The system processor software writes in this register.

Bit	31				8	7	0	
0x70	**RS**					**CMD_PRM**		**SYSP_CMDPARAM_REG**
Read/Write	USB Processor	R	System Processor	W				
Initial Value	0x0							
Mode	Device							

- **Bit 7:0 - CMD_PRM**

Command-specific Parameter.

- **Bit 31:8 - RS**

Reserved. Must be reset to zero by the system processor software.

This register is used by the system processor software to write command-specific parameters while issuing a command to the USB processor. After power-up or reset, this register will contain the value of 0x00.

USBP_CONGNUM_REG – Device Configuration Number Register

The USB processor writes in this register.

Bit	31			8	7	0	
0x74	RS				CONG_NUM		USBP_CONGNUM_REG
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Device						

- **Bit 7:0 - CONG_NUM**

Configuration number.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the configuration number while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_INTRNUM_REG – Device Interface Number Register

The USB processor writes in this register.

Bit	31			8	7	0	
0x78	RS				INTR_NUM		USBP_INTRNUM_REG
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Device						

- **Bit 7:0 - INTR_NUM**

Interface number.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the interface number while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_ALSTNUM_REG – Device Alternate Setting Number Register

The USB processor writes in this register.

Bit	31			8	7	0	
0x7C	RS				ALST_NUM		USBP_ALSTNUM_REG
Read/Write	USB Processor	W	System Processor		R		
Initial Value	0x0						
Mode	Device						

- **Bit 7:0 - ALST_NUM**

Alternate setting number.

- **Bit 31:8 - RS**

Reserved. Reset to zero by the USB processor.

This register is used by the USB processor to write the interface alternate setting number while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_REQPARM0_REG – Request Parameter 0 Register

The USB processor writes in this register.

Bit	31			0	
0x80	REQ_PRM0				USBP_REQPARM0_REG
Read/Write	USB Processor	W	System Processor	R	
Initial Value	0x0				
Mode	Device				

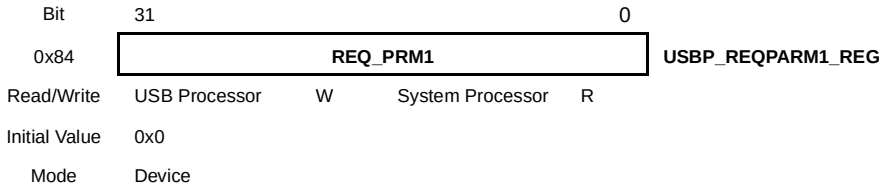
- **Bit 31:0 - REQ_PRM0**

Request-specific parameter 0 Value.

This register is used by the USB processor to write the request-specific parameters while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

USBP_REQPARM1_REG – Request Parameter 1 Register

The USB processor writes in this register.



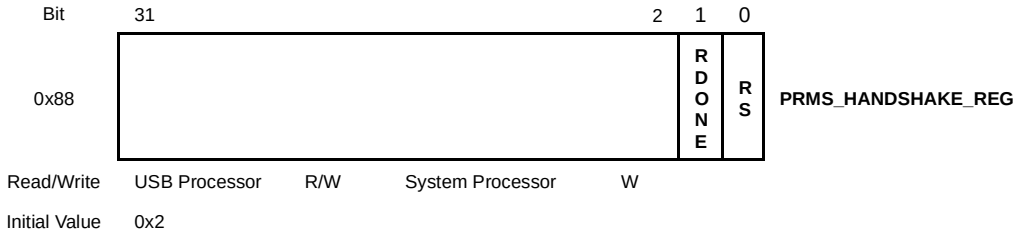
- **Bit 31:0 - REQ_PRM1**

Request-specific parameter 1 Value.

This register is used by the USB processor to write the request-specific parameters while issuing a request to the system processor software. After power-up or reset, this register will contain the value of 0x00.

PRMS_HANDSHAKE_REG – Parameters Handshake Register

The system processor and the AT43USB370 write in this register.



- **Bit 0 - RS**

Reserved. Reset to zero by the USB Processor.

- **Bit 1 - RDONE**

Request Done. This bit must be set by the system processor whenever it has finished reading the request registers on an Interrupt from the USB processor. The USB processor polls for this bit to be set before issuing an Interrupt to the system processor. This bit, when set, indicates to the USB processor that the system processor has read the request parameter registers of the previous interrupt. The USB processor reset this bit to 0 before issuing the next interrupt to the system processor.

- **Bit 31:2 - RS**

Reserved. Must be reset to zero by the system processor.

This register is used by the system processor and AT43USB370 as a handshake register to ensure that various registers written by one processor are not modified without being first read by the other processor. After power-up or reset, this register will contain the value of 0x02.

SYSP_FIFODATA_REG – FIFO Data Access Register

The USB processor writes in this register.

Bit	31			0	
0xFF	**FIFO Data**				**SYSP_FIFODATA_REG**
Read/Write	AT43USB370		System Processor	R/W	
Initial Value	0x0				
Mode	Host Device				

- **Bit 31:0 - FIFO Data Access Register**

Actual data to and from the FIFO.

This register is used by the system processor to either fetch the data from the AT43USB370 or push the data into the AT43USB370 FIFO. After power-up or reset, this register will contain the value of 0x00.

Data Transfer Mechanisms

The AT43USB370 supports three types of data transfer mechanisms

- Programmable IO (PIO) Interface
- Direct FIFO Interface
- DMA Interface

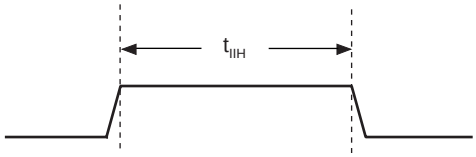
Programmable I/O Interface

The Programmable Input/Output interface is used by the system processor to perform single or multiple read/write operations to the AT43USB370's system processor interface registers. A PIO write operation allows the system processor to write to the AT43USB370 register(s). Similarly a PIO read operation allows the system processor to read the AT43USB370 register(s). PIO operation is required to set-up DMA/Direct FIFO transfer. There are two signals required to be asserted/pulsed prior to PIO operation, the MORE and INTR_IN signals.

INTR_IN

To initiate a read/write cycle to an AT43USB370's system processor interface register (PIO operation), the system processor issues an interrupt to the AT43USB370 on the INTR_IN line. Figure 7 shows the timing of INTR_IN pulse.

Figure 7. INTR_I Timing



MORE

The MORE signal is used to inform the AT43USB370 firmware when to enter and exit PIO operation. It needs to be asserted prior to the INTR_IN pulse and de-asserted at the end of the PIO operation.

PIO Read

The following sequence illustrates a typical PIO read operation by the system processor, a read from the AT43USB370 register(s). To perform more than one read operation in one PIO transaction, the system processor asserts the MORE line before the first read operation. The AT43USB370 polls this line every time a read operation is completed.

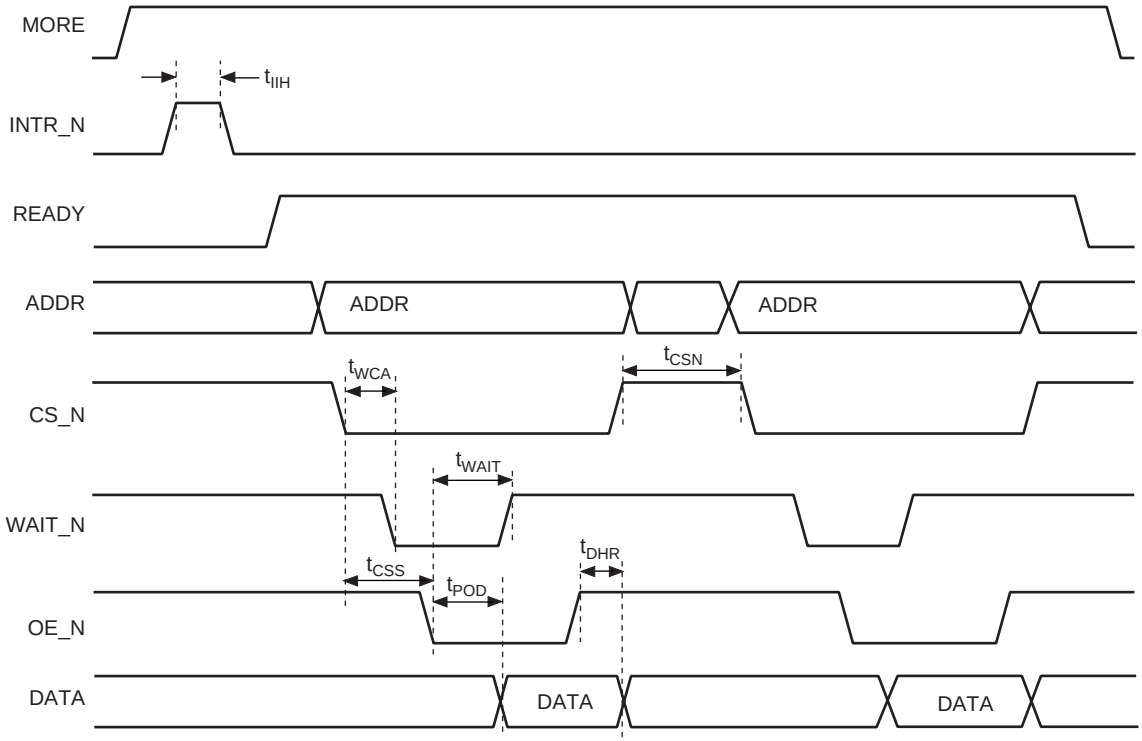
1. The system processor asserts the MORE signal to mark the start of register(s) Read operation through the PIO.
2. The system processor sends an interrupt pulse on INTR_IN to AT43USB370 to initiate a PIO Read operation.
3. The AT43USB370 asserts the READY signal and enters in the PIO scan mode.
4. The system processor places the address of the AT43USB370 register on the address bus.
5. The system processor starts the PIO Read operation by asserting the CS_N signal.
6. The AT43USB370 asserts the WAIT_N signal.
7. The system processor asserts the OE_N signal.
8. The AT43USB370 reads the register address from the address bus and puts the contents of the register on the data bus.
9. The AT43USB370 de-asserts the WAIT_N signal
10. The system processor reads the data present on the data bus.
11. The system processor de-asserts the OE_N signal

12. The system processor de-asserts the CS_N signal. This completes a single PIO Read operation.
13. The AT43USB370 samples the MORE signal:
    – If de-asserted, the AT43USB370 de-asserts the READY signal and exits the PIO mode.
    – If asserted, the AT43USB370 remains in the PIO mode and the PIO Read operation is repeated from sequence 4.

Note: The following sequences may be used interchangeably depending of the system processor used:
- Sequence 4 and 5
- Sequence 6 and 7

Figure 8 shows two consecutive PIO Read operations. For timing specifications of the PIO Transfer, please see Table 4 on page 46.

Figure 8. PIO Read Operation



PIO Write

The following sequence illustrates a typical PIO write operation by the system processor, a write to the AT43USB370 register(s). To perform multiple write operations within one PIO transaction, the system processor asserts the MORE line before the first write operation. The AT43USB370 polls this line every time a write operation is completed.

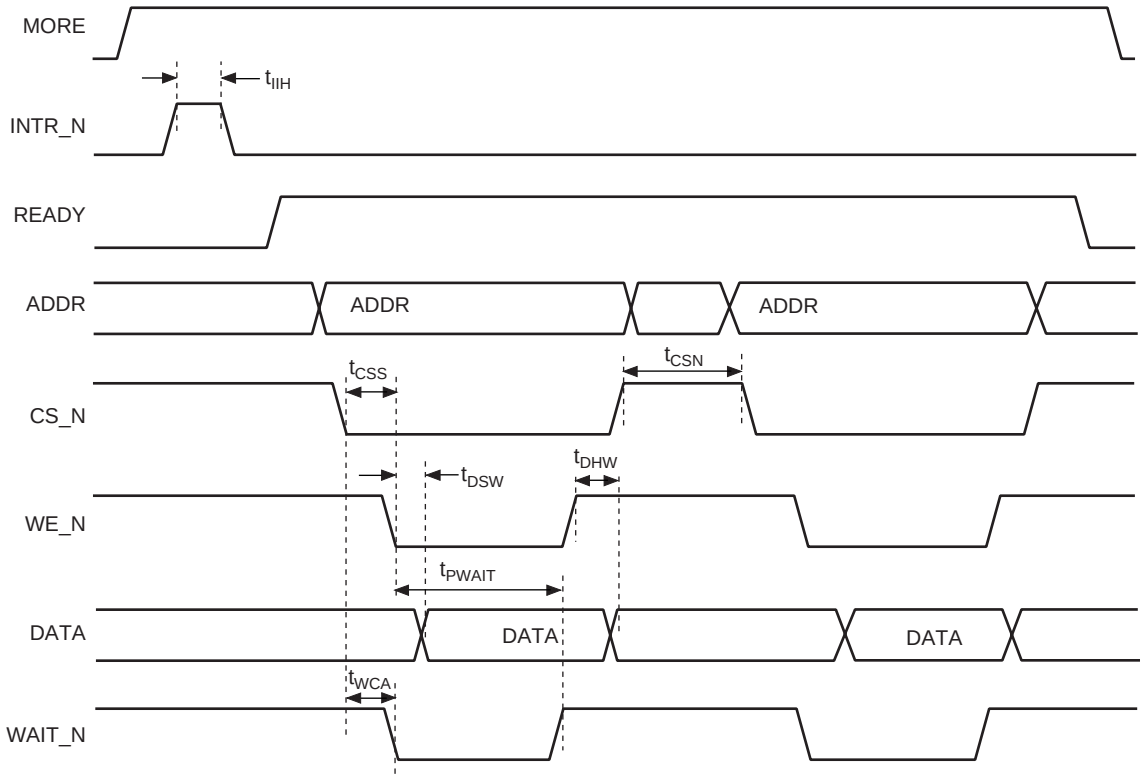
1. The system processor asserts the MORE signal to mark the start of register(s) Write operation through PIO.
2. The system processor sends an interrupt pulse on INTR_IN to AT43USB370 to initiate a PIO Write operation.
3. The AT43USB370 asserts the READY signal and enters in the PIO scan mode.
4. The system processor places the address of the AT43USB370 register on the address bus.
5. The system processor starts the PIO Write operation by asserting the CS_N signal.
6. The system processor asserts the WE_N signal.
7. The system processor puts the data on the data bus.
8. The AT43USB370 asserts the WAIT_N signal.
9. The AT43USB370 reads the register address from the address bus and copies the contents of the data bus to the target register.
10. The AT43USB370 de-asserts the WAIT_N signal
11. The system processor de-asserts the WE_N signal
12. The system processor de-asserts the CS_N signal. This completes a single PIO Write operation.
13. The AT43USB370 samples the MORE signal:
    - If de-asserted, the AT43USB370 de-asserts the READY signal and exits the PIO mode.
    - If asserted, the AT43USB370 remains in the PIO mode and the PIO Write operation is repeated from sequence 4.

Note: The following sequences may be used interchangeably depending of the system processor used:

- Sequence 4 and 5
- Sequence 6,7 and 8

Figure 9 shows two consecutive PIO Write operations. For timing specifications of the PIO Transfer, please see Table 4 on page 46.

Figure 9. PIO Write Operation



Direct FIFO Interface

The system processor can directly read from or write to the AT43USB370's on-chip FIFO through the Direct FIFO interface. The Direct FIFO interface for the AT43USB370 can be configured by the system processor by writing 0xFF on the address bus. The data can be pushed into the FIFO or read from it by the system processor using any normal memory read/write operations.

Direct FIFO Read

Figure 10 shows the timing of a FIFO read operation performed by the system processor using the Direct FIFO Interface.

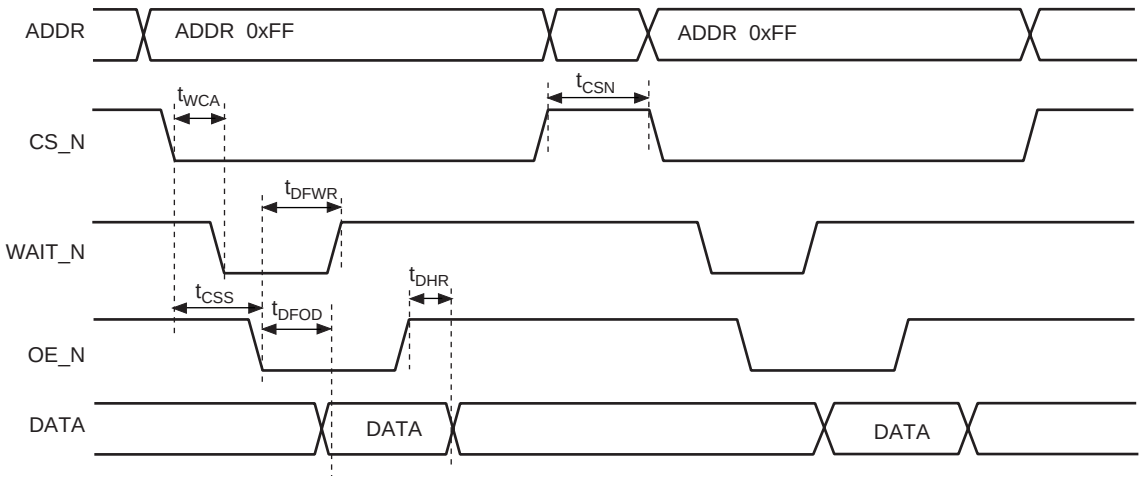
1. The system processor starts the Direct FIFO Read operation by placing the 0xFF address on the address bus.
2. The system processor asserts the CS_N signal.
3. The AT43USB370 asserts the WAIT_N signal.
4. The system processor asserts the OE_N signal.
5. The AT43USB370 reads the Direct FIFO address (0xFF) from the address bus and puts a word (32 bytes) from the FIFO on the data bus.
6. The AT43USB370 de-asserts the WAIT_N signal.
7. The system processor reads the data present on the data bus.
8. The system processor de-asserts the OE_N signal.
9. The system processor de-asserts the CS_N signal. This completes a single Direct FIFO Read operation.

Note: The following sequences may be used interchangeably depending of the system processor used:

- Sequence 3 and 4

The system processor can perform further Direct FIFO Read operations by repeating sequences 1-9. Each Direct FIFO Read operation will fetch a single word from the AT43USB370's FIFO and place it on the data bus. This applies to a 32-bit data bus.

Figure 10. Direct FIFO Read Operation



Direct FIFO Write

Figure 11 shows the timing of a FIFO write operation performed by the system processor using the Direct FIFO Interface.

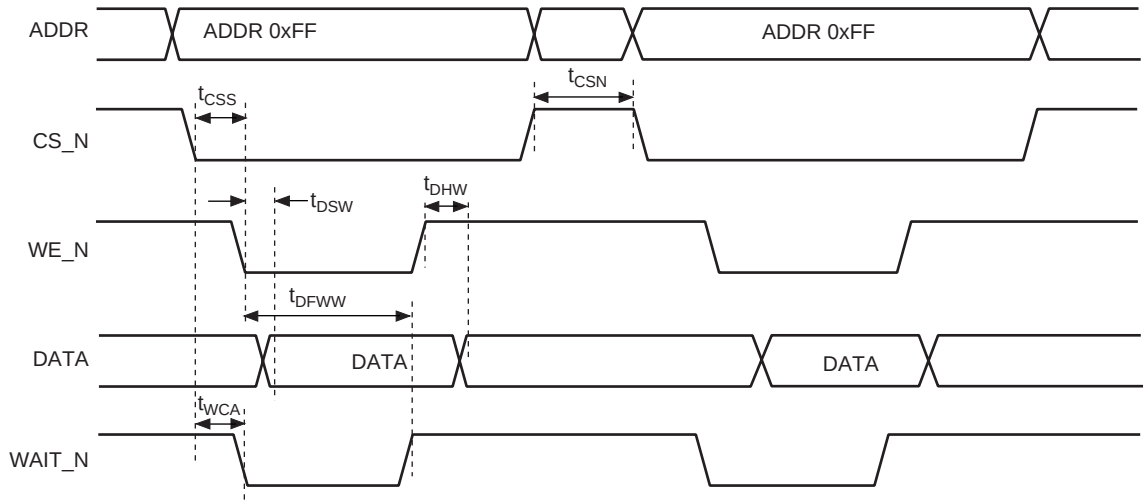
1. The system processor starts the Direct FIFO Write operation by placing the 0xFF address on the address bus.
2. The system processor asserts the CS_N signal.
3. The system processor asserts the WE_N signal.
4. The system processor puts the data on the data bus.
5. The AT43USB370 asserts the WAIT_N signal.
6. The AT43USB370 reads the Direct FIFO address (0xFF) from the address bus and copies the contents of the data bus to the FIFO.
7. The AT43USB370 de-asserts the WAIT_N signal.
8. The system processor de-asserts the WE_N signal
9. The system processor de-asserts the CS_N signal. This completes a single Direct FIFO Write operation.

Note: The following sequences may be used interchangeably depending of the system processor used:

- Sequence 3,4 and 5

The system processor can perform further Direct FIFO Write operations by repeating sequences 1-9. Each Direct FIFO Write operation will push a single word from the data bus to the AT43USB370's FIFO. This applies to a 32-bit data bus.

Figure 11. Direct FIFO Write Operation



DMA Interface

The DMA interface is used for data transfer between AT43USB370's internal FIFO and the system processor's memory. The AT43USB370 generates a request to initiate the DMA process by asserting the DREQ_N signal. It uses the block mode to transfer data through DMA. In this mode, all requested data are transferred upon the assertion of a single DMA request. The DREQ_N signal is de-asserted by AT43USB370 after the DACK_N signal has been asserted by the system processor. The DMA transfer is completed when the Transfer Count reaches zero in the DMA controller of the system processor. The DMA engine of the AT43UBS370 manages the count of the DMA transfers itself. The AT43USB370 uses a fixed source/destination address of 0x80 for RX/TX DMA transfers respectively, and the AT43USB370 can only operate as a DMA slave. Figure 12 and Figure 13 show the DMA Read and Write operations respectively.

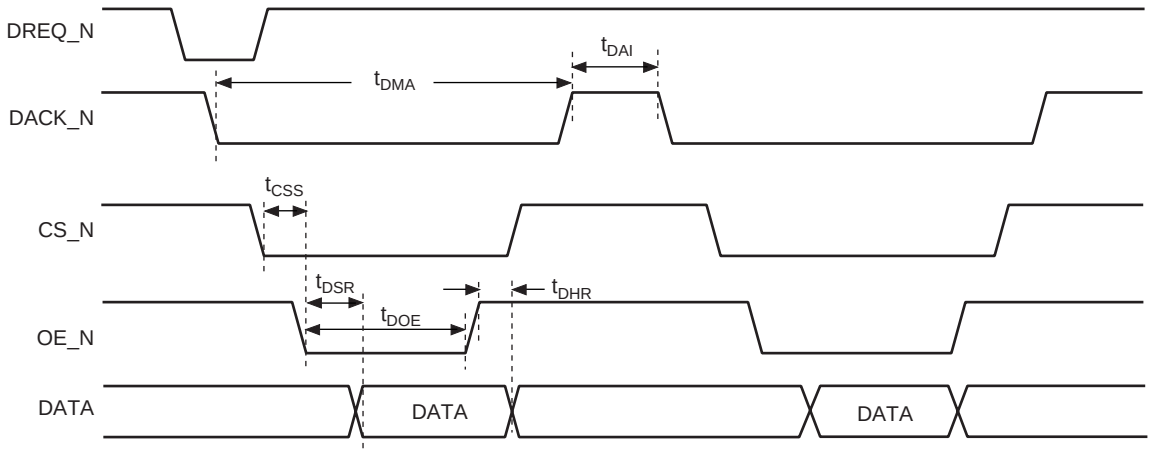
DMA Read

1. The AT43USB370 asserts the DREQ_N signal to start the DMA operation.
2. The system processor asserts the DACK_N signal.
3. The AT43USB370 de-asserts the DREQ_N signal.
4. The system processor asserts the CS_N signal.
5. The system processor asserts the OE_N signal.
6. The AT43USB370 places the contents of its FIFO on the data bus.
7. The system processor reads the data present on the data bus.
8. The system processor de-asserts the OE_N signal
9. The system processor de-asserts the CS_N signal.
10. The system processor de-asserts the DACK_N signal.

This completes a single DMA Read Cycle. If more DMA Read Cycles are to follow, the system processor asserts the DACK_N signal and sequences 4 to 10 repeat.

Notes: 1. The following sequence may be used interchangeably depending of the system processor used: Sequence 3 may occur after sequence 4 or 5.
2. WAIT_N is not used by AT43USB370 during DMA a operation.

Figure 12. DMA Read Operation



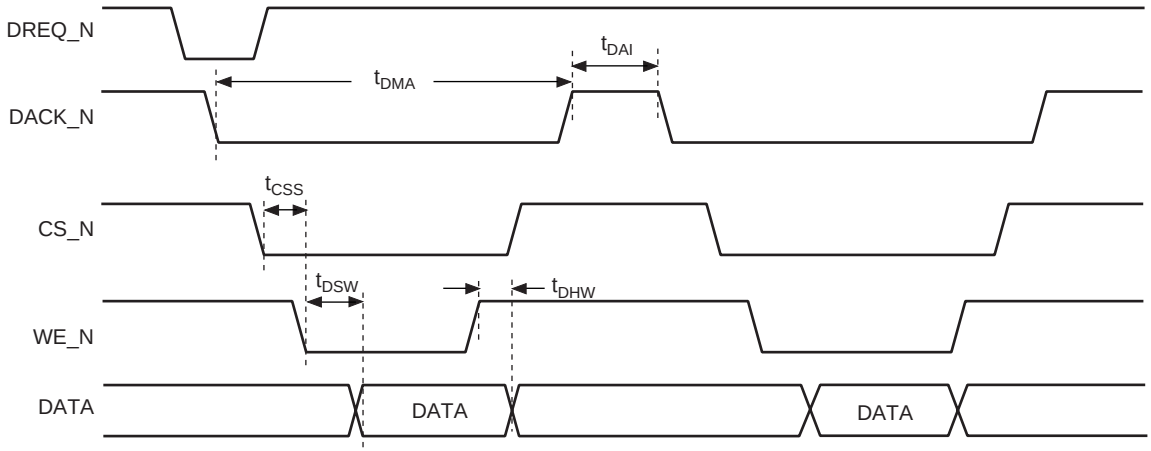
DMA Write

1. The AT43USB370 asserts the DREQ_N signal to start the DMA operation.
2. The system processor asserts the DACK_N signal.
3. The AT43USB370 de-asserts the DREQ_N signal.
4. The system processor asserts the CS_N signal.
5. The system processor asserts the WE_N signal.
6. The system processor puts the data on the data bus.
7. The AT43USB370 reads the contents of the data bus to its FIFO.
8. The system processor de-asserts the WE_N signal
9. The system processor de-asserts the CS_N signal.
10. The system processor de-asserts the DACK_N signal.

This completes a single DMA Write Cycle. If more DMA Write Cycles are to follow, the system processor asserts the DACK_N signal and sequences 4 to 10 repeat.

Notes:
1. The following sequence may be used interchangeably depending of the system processor used:
   - Sequence 5 and 6
   - Sequence 3 may occur after sequence 4 or 5
2. WAIT_N is not used by AT43USB370 during a DMA operation.

Figure 13. DMA Write Operation



Firmware Architecture

The AT43USB370 firmware model is supported by a set of USB firmware embedded on-chip and a set of system software with associated APIs running on the system processor. The APIs are used to support the development of USB device drivers of any type.

The AT43USB37 requires the host firmware when running in the host mode, and the function (or device) firmware when running in the function mode. The following sections describe in detail the firmware architecture of the AT43USB370 host/function processor.

Host Firmware

The AT43USB370 host firmware consists of the Host USB Controller Driver (HUSBCD) and Host System Interface Controller Driver (HSICD).

Host USB Controller Driver (HUSBCD)

The HUSBCD runs on the USBC when the AT43USB370 operates in the host mode. This driver performs the following tasks:

Autonomous Hub Support

The HUSBCD embeds a complete Hub Class driver to provide an autonomous Hub support.

Device Enumeration

The HUSBCD enumerates the newly connected device or hub.

Frame Management

Frame management involves calculating the time required for the next transaction and transaction completion prediction as described in USB 2.0 Specifications. It also includes the determinations the HUSBCD has to make at the time of enumeration to ensure that the requirements of the newly connected device can be met within the current power and bandwidth budget of the host.

Transaction Scheduling

The HUSBCD automatically schedules the transactions using the information that it receives from the devices during enumeration. Isochronous and Interrupt transactions are given up to 90% of the frame time. Bulk and Control transfers are guaranteed at least 10% of the bandwidth and are also allocated any available bandwidth not consumed by the Isochronous and Interrupt transfers.

Bus Reclamation

The HUSBCD implements the Bus Reclamation mechanism that allows the AT43USB370 host to maximize the bus utilization by using all the time left after servicing pending transactions to transfer bulk/control data.

Status Handling

After a transaction has been completed, the HUSBCD posts the transaction status to the HSICD. The HUSBCD also enables and disables the concerned FIFOs for data and control transfers.

Host System Interface Controller Driver (HSICD)

The HSICD runs on the System Interface Controller when the AT43USB370 is operating in the host mode. This driver performs the following tasks:

Data Transfer Management

The HSICD handles the data transfer between the USB function and the system processor memory.

High Level API Management

A set of C APIs and associated USB host system software library constitutes the basic building blocks of USB device drivers of any type. This USB host system software resides on the system processor. The HSICD manages the information exchange between the embedded AT43USB370 firmware stack and the host system software running on the system processor through the host system interface APIs. For detailed descriptions of the APIs, refer to *AT43USB370 Software Development Guide*.

Descriptor Management

The HSICD gathers the USB descriptors from the USB devices and reports back to system processor through API function calls.

USB Host System Library

The USB host system library is comprised of the AT43USB370 host system interface APIs and the underlying software library. All application-level system software communicates with the AT43USB370 through this library.

AT43USB370 USB Host System Library and APIs

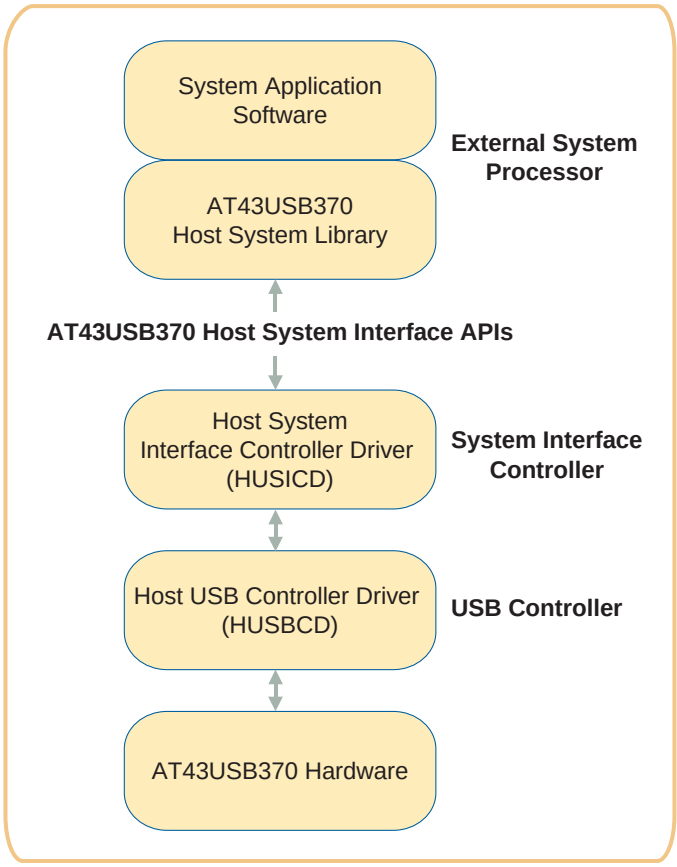
The USB host system library provides access to the AT43USB370 USB host functionality by the system processor. The corresponding host system interface API set encapsulates the complete USB functionality. It is ANSI C compliant and is used for all USB device drivers and applications development. Refer to *AT43USB370 Software Development Guide* for detailed descriptions of the AT43USB370 APIs.

System Software

System software is application specific and resides in the system processor. It communicates with the AT43USB370 through the AT43USB370 Host System Interface APIs directly or through the standard or application specific USB device drivers built on top the AT43USB370 APIs.

Figure 14 shows the AT43USB370 host firmware architecture.

Figure 14. AT43USB370 Host Firmware Architecture



Function or Device Firmware

The AT43USB370 function or device firmware includes the Device USB Controller Driver (DUSBCD) and Device System Interface Controller Driver (DSICD).

Device USB Controller Driver

The DUSBCD runs on the USB Controller. This driver interacts with the AT43USB370 hardware and performs USB protocol management relating to the function operation more specifically, the DUSBCD performs the following functions:

Device Configuration

The DUSBCD automatically configures the AT43USB370 function for the required features through a descriptor table specified by the system processor. The DUSBCD parses the USB standard and class-specific descriptors from this table and stores them in the program memory of the controller. The following features of the AT43USB370 function can be configured through the descriptor table.

- **Number of Endpoints** - Maximum of 6 data (3 IN and 3 OUT) endpoints can be supported. A bi-directional control endpoint is supported by default
- **Type of Endpoint** - Any of the Interrupt, Bulk or Isochronous endpoint may be specified
- **Maximum Packet Size** - for every endpoint including the control endpoint
- **FIFO Size** - for every endpoint
- **Remote Wake-up Support**

The format for providing the descriptor table is specified in the *AT43USB370 Software Development Guide*.

FIFO Configuration

The DUSBCD configures the FIFO according to the maximum packet sizes and endpoint types of the endpoints specified in the descriptor table. It also configures the Control and Status Registers.

Device Enumeration and Standard Request Handling

The DUSBCD generates the connect status on USB after parsing the descriptor table and automatically enumerates AT43USB370 as a device when it is connected to a USB Host. It also handles other standard USB requests issued by the USB Host.

Suspend Detection

The DUSBCD detects the suspend condition when no bus activity is reported from the SIE Controller for 3 ms.

Transaction Handling

The DUSBCD automatically handles incoming packets for OUT endpoints for expected Isochronous, Bulk and Interrupt transactions which are ultimately reported to the external system processor. Similarly, it supplies the USB Host required data from IN endpoints after getting it from the external system processor.

Device System Interface Controller Driver

The DSICD runs on the System Interface Controller. This driver is responsible for the following functions:

Data Transfer Management

The DSICD handles the data transfer between the AT43USB370 Device and the external system processor's memory. It also provides an interface to specify the descriptor table.

High Level API Management

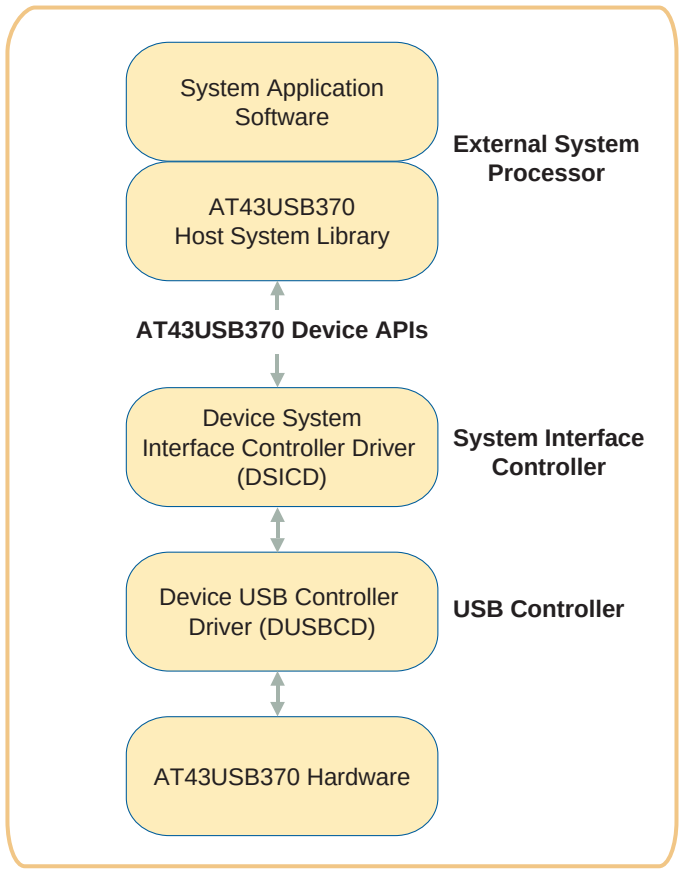
The DSICD provides a generic interface for the system processor in order to achieve maximum ease in the Device operation at a high level. The DSICD manages all the information exchange with the external system processor through this interface. In this way, the complexity in transferring data over the USB is hidden from the external system processor. For more details about the API, refer to the *AT43USB370 Software Development Guide*.

USB Device System Library and APIs

The USB Device System Library runs on the external system processor. It provides access to USB functionality of the AT43USB370 to the system application running on the system processor. The system library interfaces to the system application through a set of high level, ANSI C compliant APIs. The API set encapsulates the USB functionality and is used as the building blocks of any USB application. Please refer to the *AT43USB370 Software Development Guide* for detailed descriptions of the AT43USB370 device APIs.

Figure 15 shows the device firmware architecture of the AT43USB370.

Figure 15. AT43USB370 Device Firmware Architecture



System Processor Connection

The AT43USB370 is typically attached to a system processor through its 32-bit bi-directional data path with additional control lines and an 8-bit address bus. The required interface signals are grouped into the following categories:

General Interface Signals

- **DATA BUS (D [31:0])** - The 32-bit bi-directional data bus is used to transfer data to and from AT43USB370. The AT43UBS370 is little-endian compatible.
- **ADDRESS BUS (A [7:0])** - This is an 8-bit address bus used to address the System Processor Interface Register set of the AT43USB370.
- **CHIP SELECT (CS_N)** - The CHIP SELECT signal is active low.
- **OUTPUT ENABLE (OE_N)** - This is the Read signal driven by the system processor to read a register or memory location. This signal is active low.
- **WRITE ENABLE (WE_N)** - This is the Write signal driven by the system processor to write an address, register or memory location. This signal is active low.
- **WAIT (WAIT_N)** - This signal is used by the AT43USB370 to assert wait states. This signal is active low, and it activates upon the assertion of AT43USB370's CS_N.
- **BUSY (BUSY)** - This signal is used by the AT43USB370 to signal the system processor not to interrupt the AT43USB370. This signal is active high.
- **MORE (MORE)** - This signal is used to inform the AT43USB370 firmware when to enter/exit a PIO operation.

Interrupt Signals

- **Interrupt In (INTR_IN)** - This is an interrupt signal from the system processor to AT43USB370. This signal is active high. See "INTR_IN" on page 31.
- **Interrupt Out (INTR_OUT)** - This is an interrupt signal from AT43USB370 to the system processor. This signal is active high. It is used to notify the system processor of an event, such as completion of enumeration, completion of commands, or more buffers required.

Programming Signals

PROG, READY, DONE, SELECT - These signals are only required for programming the AT43USB370 controllers (i.e. the USB Controller and System Interface Controller) through the external system processor. The signals can be directly connected to the processor's GPIOs. These signals are active high.

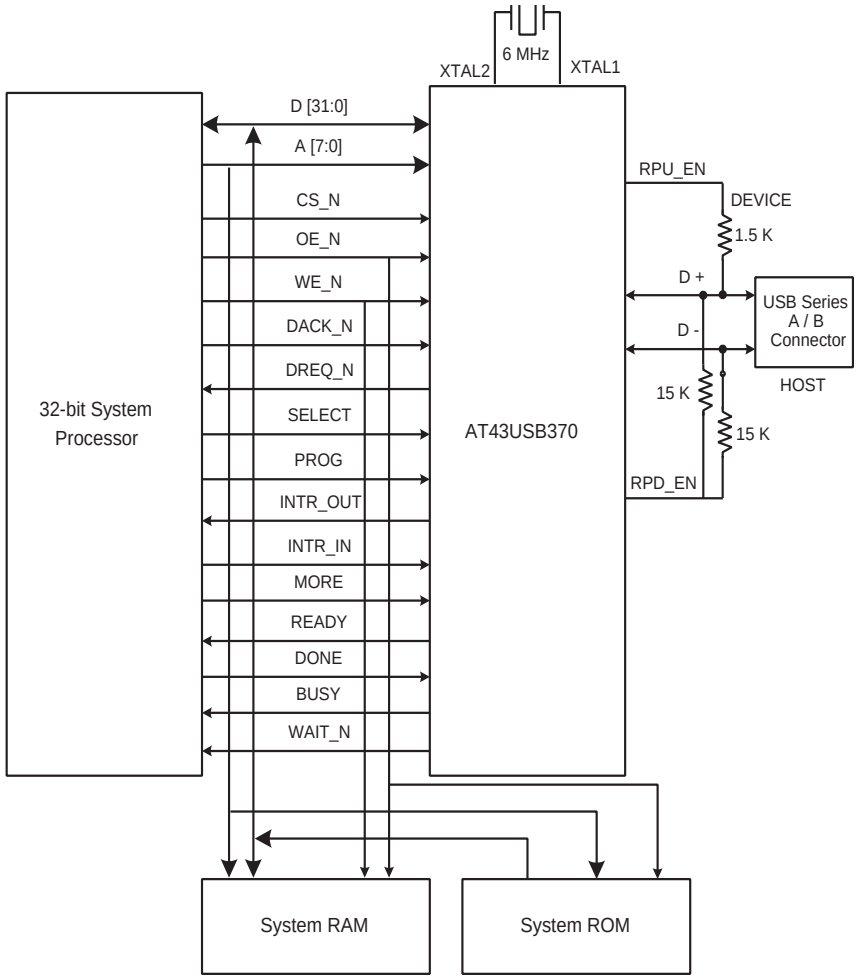
DMA Signals

DREQ_N, DACK_N - These are the standard DMA control signals used by the AT43USB370 for DMA transfers with the system processor. These signals are active low.

Power and I/O Cells

The AT43USB370 requires external 3.3V and 1.8V power supplies. All of the AT43USB370's I/O pins are +3.3V tolerant. Figure 16 shows typical connection of AT43USB370 to a 32-bit system processor.

Figure 16. Typical AT43USB370 System Processor Connection



Electrical Specification

Absolute Maximum Ratings

Table 2. Absolute Maximum Ratings*

Symbol	Parameter	Condition	Min	Max	Unit
V_{CC}	3.3V Power Supply		3.0	3.6	V
V_{IN} (3.8V)	DC Input Voltage		V_{CC} -0.3V	V_{CC} +0.3V	V
V_{IN} (1.8V)	DC Input Voltage		1.65	1.95	V
T_{ORP}	Operating Temperature		0	70	°C
T_{STG}	Storage Temperature		-40	125	°C

*Stresses beyond those listed below 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 beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Electrical Characteristics

Table 3. Power Supply

Symbol	Parameter	Condition	Min	Max	Unit
V_{CC}	Power Supply		3.0	3.6	V
V_{IL}	Input Low Voltage			0.8	V
V_{IH}	Input High Voltage		2.0		V
VT	Switching Threshold				V
V_{OL}	Output Low Voltage			0.4	V
V_{OH}	Output High Voltage		3.3		V
I_{OZ}	Tristate Output Leakage Current				µA
C_{IN}	Input Pin Capacitance			7.24	pf
C_{OUT}	Output Pin Capacitance			6.07	pf
C_{IO}	I/O Capacitance			7.27	pf

AC Electrical Characteristics

System Processor Interface Timing

Table 4. AT43USB370 - System Processor Interface Timing

Symbol	Parameter	Condition	Min	Max	Units
t_{DR}	DONE Active to READY De-active Time	Programming		42	ns
t_{IIH}	INTR_IN Active Time		504		ns
t_{WCA}	CS Active to WAIT Active	PIO, Direct FIFO		126	ns
t_{CSS}	CS Active to OE/WE Active	PIO, Direct FIFO, DMA	84		ns
$*t_{PWAIT}$	WE/OE Active to WAIT De-active	PIO R/W		2.6	us
$*t_{POD}$	PIO OE Active to Data Valid	PIO R		2.6	us
t_{DFWW}	WE Active to WAIT De-active	Direct FIFO W		336	ns
t_{DFWR}	OE Active to WAIT De-active	Direct FIFO R		252	ns
t_{DFOD}	Direct FIFO OE Active to Data Valid	Direct FIFO R		252	ns
t_{CSN}	CS In-active Time	PIO, Direct FIFO	84		ns
t_{DAI}	DMA ACK In-active Time	DMA R/W	42		ns
t_{DMA}	DACK Active Time	DMA R/W	546		ns
t_{DSW}	WE Active to Data Valid	PIO, Direct FIFO, DMA W		10	ns
t_{DHW}	Data Hold Time after WE De-active	PIO, Direct FIFO, DMA W	42		ns
t_{DHR}	Data Hold Time after OE De-active	PIO, Direct FIFO, DMA R		5	ns
t_{DSR}	DMA OE Active to Data Valid	DMA R	126		ns
t_{DOE}	DMA OE Active Time	DMA R	168		ns

Notes:
1. The timing parameters with the * are firmware dependent. The value listed is for Library 1.0 release.
2. PIO Cycle Time = $t_{WCA} + t_{PWAIT} + t_{CSN}$ ~= 2.8 µs
3. DFIFO Cycle Time (Write) = $t_{WCA} + t_{DFWW} + t_{CSN}$ = 546 ns
4. DFIFO Cycle Time (Read) = $t_{WCA} + t_{DFWR} + t_{CSN}$ = 462 ns
5. DMA Cycle Time = $t_{DMA} + t_{DAI}$ = 588 ns

Reset Timing

Table 5. Reset Timing

Symbol	Parameter	Condition	Min	Max	Units
t_R	RESET Width		200		ns

Table 6. Oscillator Signals XTAL1, XTAL2

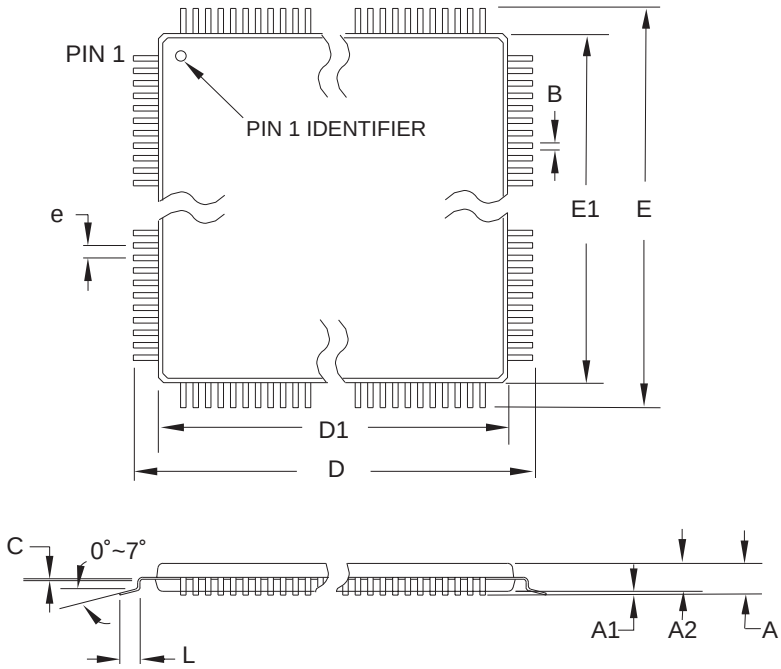
Symbol	Parameter	Condition	Min	Max	Units
V_{LH}	OSC1 Switching Level		0.47	1.20	V
V_{HL}	OSC2 Switching Level		0.67	1.44	V
CX1	Input Capacitance XTAL1			10	pf
TCX2	Output Capacitance, XTAL 2			10	pf
C12	Oscillator Capacitance			5	pf
T_{su}	Start-up Time	6 MHz, fundamental		2	ms
DL	Drive Level			50	UW

Ordering Information

Program Memory	Ordering Code	Package	Operation Range
SRAM	AT43USB370E-AC	100 LQFP	Commercial (0°C to 70°C)

Packaging Information

100-lead – LQFP



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.60	
A1	0.05	–	0.15	
A2	1.35	1.40	1.45	
D	15.75	16.00	16.25	
D1	13.90	14.00	14.10	Note 2
E	15.75	16.00	16.25	
E1	13.90	14.00	14.10	Note 2
B	0.17	–	0.27	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.50 TYP			

Notes:
1. This package conforms to JEDEC reference MS-026, Variation AED.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.08 mm maximum.

04/29/2002

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**100AA,** 100-lead, 14 x 14 mm Body Size, 1.4 mm Body Thickness, 0.5 mm Lead Pitch, Low Profile Quad Flat Pack (LQFP)	100AA	C

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
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, USA
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, USA
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

Literature Requests
www.atmel.com/literature

Disclaimer: Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

© Atmel Corporation 2003. **All rights reserved.** Atmel® and combinations thereof, are the registered trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be the trademarks of others.

Printed on recycled paper.

3340B–USB–12/03 xM