Welcome to E-XFL.COM

Embedded - Microcontrollers - Application Specific: Tailored Solutions for Precision and Performance

Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering enhanced performance, efficiency, and functionality to meet the demands of specialized applications.

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Obsolete
Applications	USB Host/Function Processor
Core Processor	-
Program Memory Type	-
Controller Series	AT43USB
RAM Size	-
Interface	USB
Number of I/O	32
Voltage - Supply	3V ~ 3.6V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at43usb370e-ac

Pin Assignment

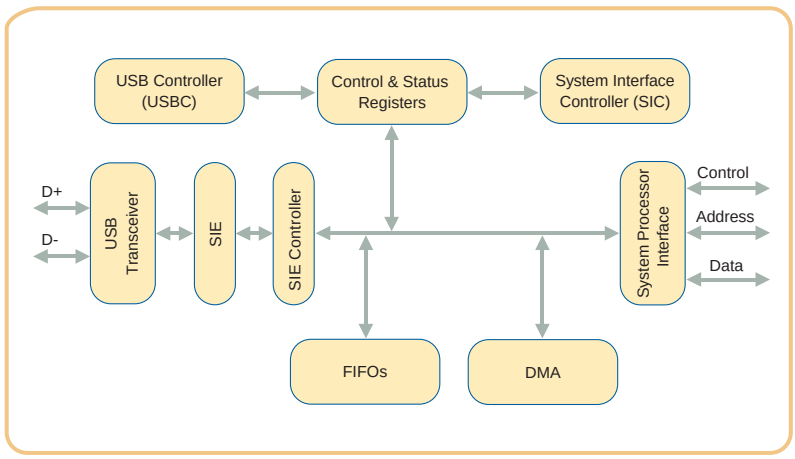
Pin #	Signal	Type
1	A7	Input
2	A6	Input
3	A5	Input
4	A4	Input
5	A3	Input
6	A2	Input
7	A1	Input
8	A0	Input
9	CS_N	Input
10	OE_N	Input
11	SELECT	Input
12	INTR_IN	Input
13	INTR_OUT	Output
14	VDD	Power Supply/Gnd
15	VSS	Power Supply/Gnd
16	VSS	Power Supply/Gnd
17	VDD18	Power Supply/Gnd
18	PROG	Input
19	MORE	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 (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



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.

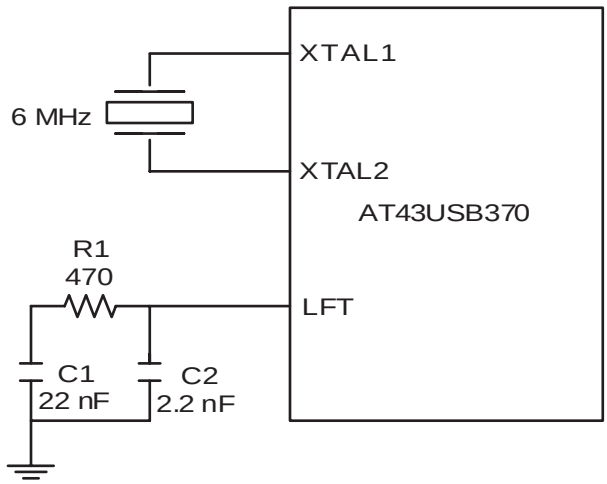
Oscillator and PLL

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



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.

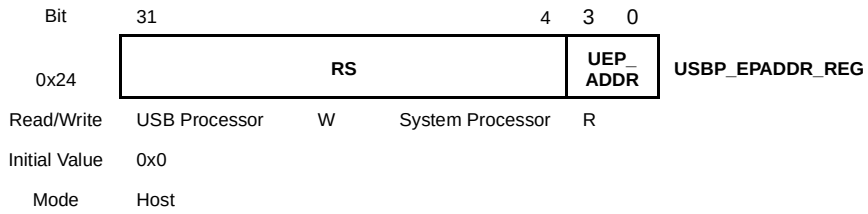
- Three different fields in the register name are separated by underscores '_'
- The first field in the register name is a prefix indicating the Write access identification literal:
  - USBP indicates the register is always written by the AT43USB370 USB Processor
  - SYSP indicates the register is always 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

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.

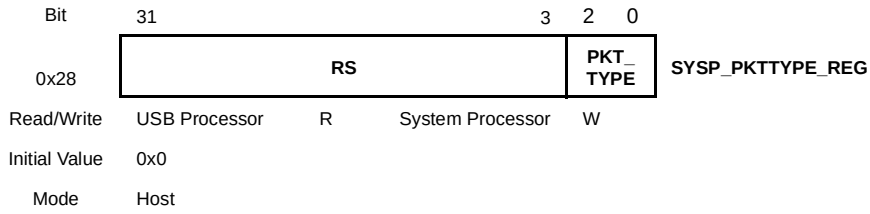
- **Bit 31:4 - RS**

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_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.

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_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**

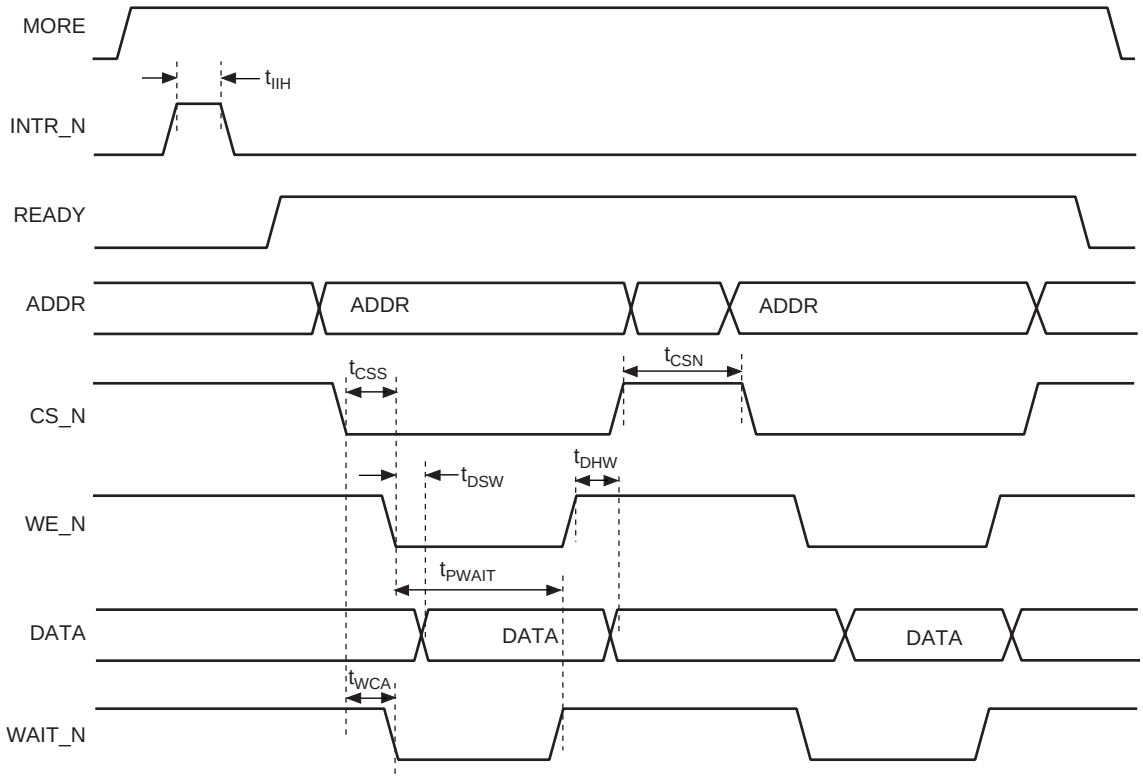
Interface number.

- **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.

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.

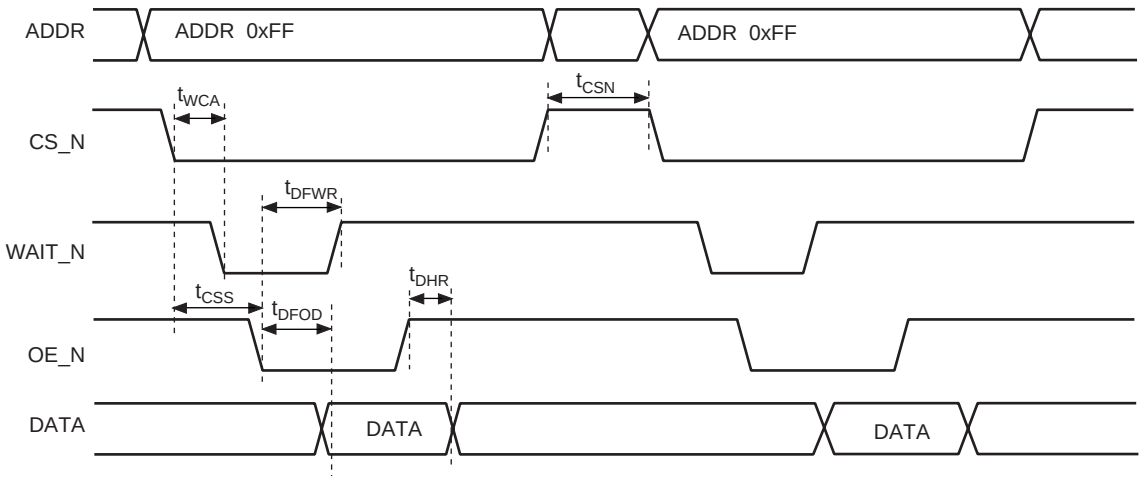
1. The system processor starts the Direct FIFO Read operation by placing the 0xFF address on the address bus.
2. The system processor asserts the CS_N signal.
3. The AT43USB370 asserts the WAIT_N signal.
4. The system processor asserts the OE_N signal.
5. The AT43USB370 reads the Direct FIFO address (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



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.

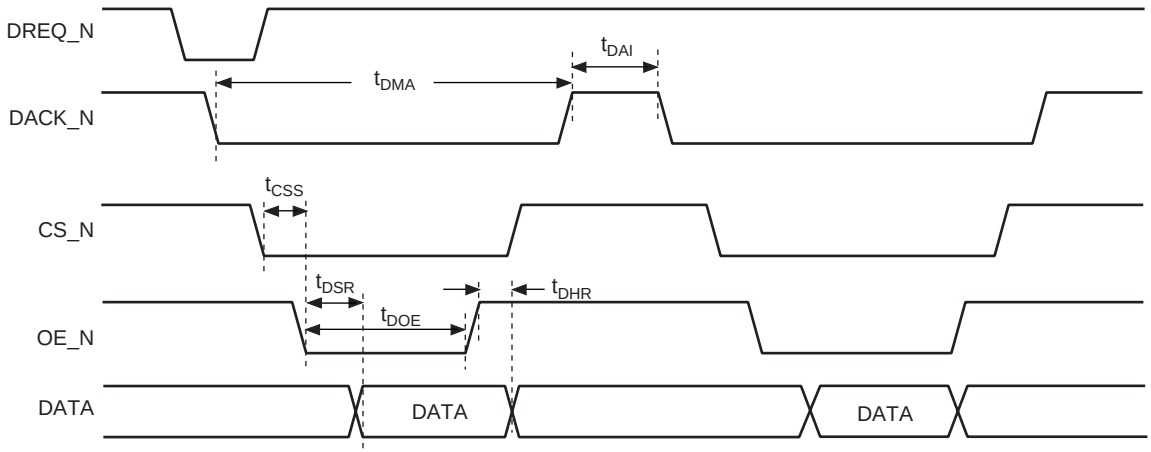
DMA Read

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



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

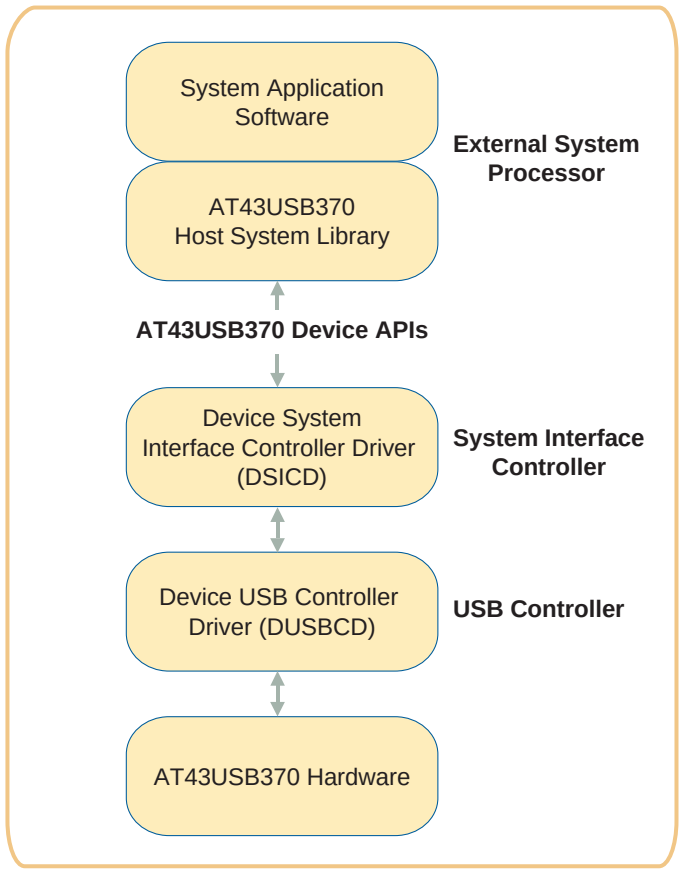
A set of C APIs and associated USB host system software library constitutes the basic building blocks of USB device drivers of any type. This USB host system software resides on the system processor. The HSICD manages the information exchange between the embedded AT43USB370 firmware stack and the host 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*.

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



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

Ordering Information

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