Bulk Transfers with the EZ-USB SX2™ Connected to a Hitachi SH3™ DMA Interface

Introduction

The Universal Serial Bus (USB) is an industry-standard serial interface between a host computer and peripheral devices such as Low-speed mice and keyboards, full-speed solid state mass storage media, and high-speed mass storage and imaging devices. The EZ-USB SX2™ (CY7C68001) is a high-speed device, which provides a slave interface, buffering, and master processor programmable endpoints for full- or high-speed USB interface. This application note describes a sample connection scheme between a slave SX2 and a Hitachi SuperH® RISC (SH7729) master processor. This processor is part of a family of devices, including the SH7709, which has a DMA interface. This sample interface should work with any of these Hitachi processors that include this DMA interface.

This application note shows a single DMA channel connection and usage scheme. This scheme is intended to illustrate a sample synchronous connection model. Synchronous mode has been selected in order to illustrate a fast slave interface. In order to show the flexibility of the interface, a single SH3™ DMA Channel is used for all Master/Slave transactions—data read, data write, and command/status operations—between the SX2 and SH3. This highlights the ability to change the mode of SX2 signaling pins dynamically. Alternative designs can use two SH3 DMA channels with one allocated for read operations and the second allocated for write operations, which will simplify the flag programming. However, this requires the addition of another pin connection.

This example describes SX2 programming steps for performing a Bulk loopback operation beginning with an OUT transfer from the PC Host to the SX2, followed by a DMA burst from the SX2, by the SH3 DMA, to a memory storage. This is followed by a DMA burst from the memory storage, by the SH3 DMA, to the SX2, followed by an IN packet transfer to the PC Host from the SX2. The method of signaling selected for this example is to use a flag pin (FLAGB) in order to show how to throttle the SH3 DMA activity based on data availability on OUT transfers and buffer availability on IN transfers.

This application note assumes that the reader is familiar with the SH3 DMA dual addressing modes and has reviewed the SX2 data sheet and the SH3 manual (see References). The reader should have both documents at hand, especially the DMA chapter of the Hitachi SH3 manual.

This application note is based on a concept design that does not include implementation detail and is not supported by a working example.

Connection Scheme

The example in this application demonstrates a synchronous interface using an externally generated interface clock from the SH3. The SH3 output clock (CKIO) is connected to the SX2 interface clock (IFCLK) pin for an externally supplied interface clock to the SX2. The SH3 CKIO frequency can be programmed and the SX2 slave interface IFCLK can accept any frequency in the range of 5 MHz to 50 MHz. If necessary, the SX2 can be programmed to use the inverse polarity of the SH3-provided interface clock.

SX2 Interface and Endpoint Configuration Scheme

This connection example uses the following SX2 modes:

- Synchronous slave FIFO interface
- External interface clock (IFCLK)
- Program Flag B for Direction:
  - — FIFO Empty (EF) for OUT Endpoint Data (Master data reads)
  - — FIFO Full (FF) for IN Endpoint Data (Master data writes)
- Program Flag D for Chip Select
  - — Active LOW Chip Select (CS#)
- 16-bit FIFO buffer interface for data (note: Command/Status operations are always 8-bit)
- Endpoint 2 is programmed for:
  - — OUT direction
  - — Double buffering
  - — 512-byte buffer size
  - — Bulk Type

- Endpoint 6 is programmed for:
  - IN direction
  - Double buffering
  - 512-byte buffer size
  - Bulk Type

SX2 FIFO Flag Configuration Scheme

This loopback example uses the following SX2 endpoint buffer flag settings:

- Set SX2 for inverse polarity for Empty FIFO (EF) and Full FIFO (FF) Flags (active HIGH)
- Read bursts for OUT USB data on Endpoint 2 (see *Figure 6* on page 12)
  - Set SX2 FLAGB for Endpoint 2 FIFO Empty (FLAGB now high due to inverse EF polarity setting)
  - Address SX2 Endpoint 2 FIFO
  - Throttle bursts on Data Request (DREQ0#) on SX2 FLAGB
  - Wait for FLAGB to go LOW (Host sends a packet)
  - On FLAGB LOW (FIFO not empty) throttle DMA on until FLAGB HIGH
  - On FLAGB HIGH throttle DMA off until FLAGB LOW again (when Host commits a packet)
  - Continue until transfer complete
  - FLAGB remains HIGH until new OUT or reprogrammed for IN transfers
- Write bursts for IN USB data on Endpoint 6 (see *Figure 7*)
  - Set SX2 FLAGB for Endpoint 6 FIFO Empty (FLAGB now HIGH due to empty condition and inverse EF polarity setting)
  - Address SX2 Endpoint 6 FIFO
  - Throttle bursts on Data Request (DREQ0#) on SX2 FLAGB
  - Set SX2 FLAGB for Endpoint 2 FIFO Full to force FLAGB to LOW (FLAGB now LOW due to not full condition and inverse FF polarity setting; DMA burst starts)
  - On FLAGB LOW (FIFO not full) throttle DMA on until FLAGB HIGH
  - On FLAGB HIGH throttle DMA off until FLAGB LOW again (when Host accepts a packet)
  - Continue until transfer complete
  - FLAGB remains high until new IN or reprogrammed for OUT transfers

SH3 Configuration Scheme

This loopback example uses the following SH3 modes:

- Generates external interface clock (CKIO)
- Uses CPU Programmed I/O for SX2 Command interface
  - Write Control data bytes

    (refer to Section 3.7.8.1 in SX2 Data Sheet, Reference item 1)
    1. Write Control register index byte with direction bit set for write
    2. Control register data value bytes
  - Read Status data byte

    (refer to section 3.7.8.2 in SX2 data sheet, Reference item 1)
    1. Write Control register index (descriptor index) with direction bit set for write
    2. Wait for INT Interrupt signal
    3. Read status register value byte
  - Write Descriptor bytes

    (refer to section 4.1 in SX2 data sheet, Reference item 1)
    1. Write Control register index (Descriptor index) with direction bit set for write
    2. Write descriptor length bytes
    3. Write descriptor bytes
- DMA Channel 0 is programmed for:

— Burst Mode with Level Timing
— Throttle bursts on Data Request (DREQ0#) on SX2 FLAGB
— Dual Address Transfer Mode

1. On each read during an OUT packet read burst the DMA engine will:
   a. Address SX2 Endpoint 2 FIFO using:
      - Four contiguous upper address lines
      - Address line N controls Chip Select
      - Address lines N-1, N-2, N-3 to select SX2 FIFOADR [2:0] lines
   b. Read data n
   c. Address Memory Store
   d. Write data n
2. On each write during an IN packet write burst the DMA engine will:
   a. Address Memory Store
   b. Read data n
   c. Address SX2 Endpoint 6 FIFO
      - Four contiguous upper address lines
      - Address line N controls Chip Select
        — N is allocated memory space for SX2
      - Address lines N-1, N-2, N-3 to select SX2 FIFOADR [2:0] lines
      - Write data n

Important Considerations

In 16-bit mode, the SX2 only transfers even byte packets of data. This can be an issue when a device interfaces to a Host Application that expects periodic odd byte size packets. An example of this is the Windows USB Mass Storage Class Driver for Bulk-Only Transport, which expects odd size Command Status Wrapper (CSW) packets. This requires that the Master processor reprogram the SX2 interface for 8-bit data transfer prior to writing the CSW packet contents to the SX2.

The SX2 is intended for use as a high-speed “data pump” and should be included with care in designs with Control Endpoint data messaging, such as the USB Communications Class or USB Mass Storage Class (with CBI Transport protocol) where Command Block Wrapper (CBW) packets are sent via the Control Endpoint. The EZUSB FX2™ (CY7C68013) may be an alternate part selection for these applications.

The SX2 is best suited for self powered USB applications where Master conditions do not require the SX2 to enter a suspend state. Therefore, the Master processor should monitor the USB VBus condition, such that when the Host removes VBus, the Master processor can take appropriate steps to ensure that the SX2 does not attempt to drive the USB Data lines. Failure to monitor VBus conditions may cause USB Compliance Test failure during back voltage electrical testing.

Connection Diagram

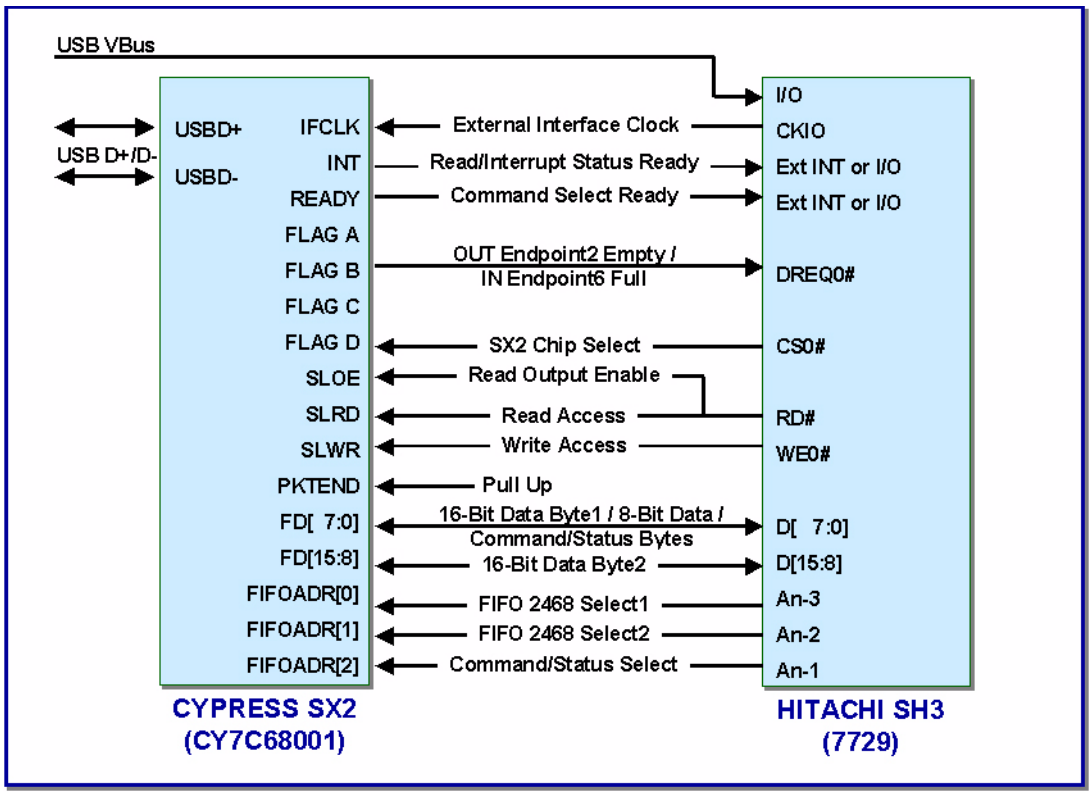

Figure 1. SX2 to SH3 Interconnect Diagram

Figure 1 shows the complete set of signal connections between the SX2 and the SH3 devices. For the SX2 READY signaling pins, the connection to the SH3 can either be to I/O or interrupt pins, however the INT signal should be connected to an interrupting pin on the SH3 to ensure timely reading of either the interrupt status or the requested register data value.

The SX2 FIFOADR [2:0] pins are connected to upper address bits and the lower address bits are reserved for DMA memory addressing for sourcing or sinking the SX2 endpoint data. The SX2 chip select (CS function of Flag D) can also be used to localize addressing of the SX2 on a common address bus.

In this scheme the SX2 Slave Output Enable (SLOE) and Slave Read (SLRD) signals are tied together and driven by the single SH3 Read (RD#) signal which the SH3 DMA drives during read burst operations.

SX2 write operations require a single Slave Write (SLWR) signal and are driven by the single SH3 Write (WE0#) signal, which the SH3 DMA drives during write burst operations.

Data is read from or written to the SX2 in either 8-bit or 16-bit operations. When programmed for 8-bit data operations, then of the 16 total data lines, only the FD [7:0] pins are required for all Command, Status, and Data operations. When programmed for 16-bit operations (as this example shows), the second byte in a 16-bit data word is carried on the upper data pins (FD [15:8]). The first byte corresponds to the Least Significant Byte (LSB) of a word and the second byte corresponds to the Most Significant Byte (MSB) of a word. Command and Status operations are always 8-bit.

The SX2 Flag B is connected to the SH3 DMA Request (DREQ0#) signal for the DMA channel. The currently addressed SX2 endpoint FIFO Flag B is used for throttling the DMA request flow.

Writing Commands to the SX2

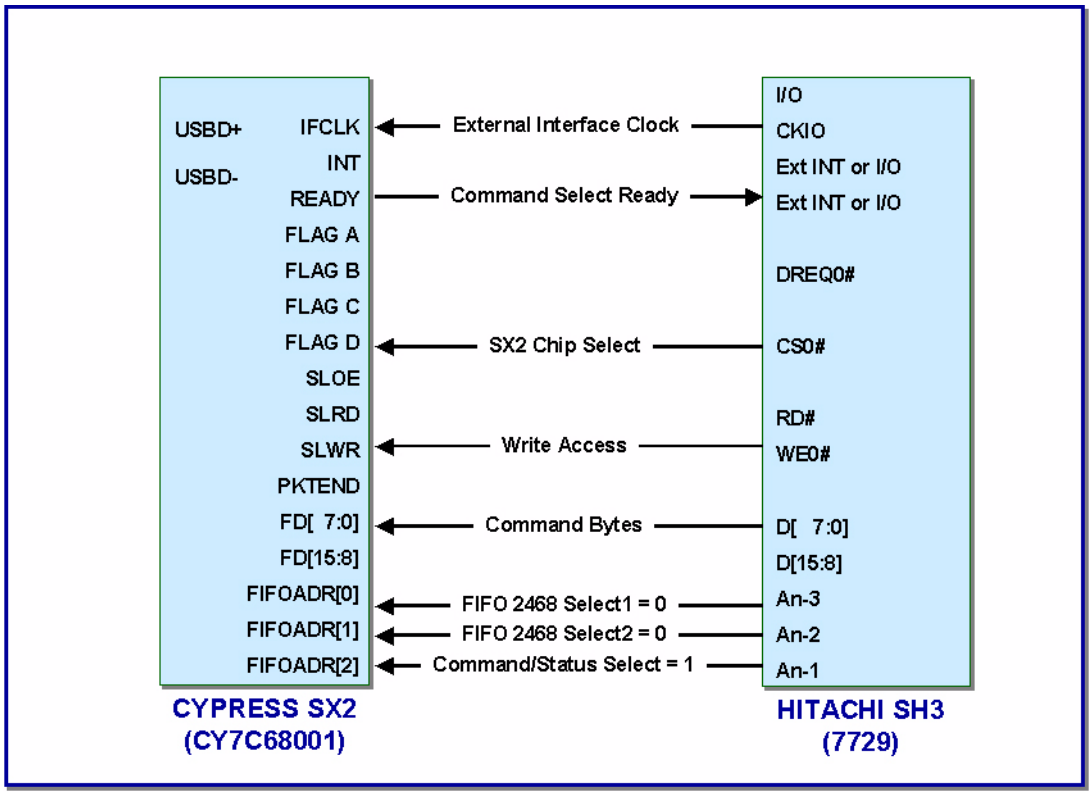

Figure 2. SX2 Command Write Signals

There are two types of Command Write sequences in a typical application: downloading vendor and application specific descriptor contents and setting indexed SX2 control register contents.

1. The first Command Write sequence is for downloading custom descriptor information into the SX2, which the SX2 will then use during subsequent USB enumerations. There are two types of descriptor downloads. The first method is for downloading only Vendor ID (VID), Product ID (PID) and Device ID (DID) information. This sequence consists of loading a six byte only sequence consisting of VID, PID, and DID. The SX2 will enumerate with its default interface and endpoint configurations. The second method is for downloading complete descriptor information including interface and endpoint configuration information. In this case, the SH3 writes a Command Byte, with the Descriptor Register index, followed by the descriptor length, and then downloads the descriptor contents. The descriptor can be up to 500 bytes in length and includes both high-speed and full-speed descriptors (see the SX2 data sheet sections 4.1 and 4.2 for downloading descriptors and section 12 for the default descriptors).
2. The second Command Write sequence is for programming the SX2 control registers. This sequence consists of writing a Command Byte followed by two data bytes. The Command Byte consists of a direction bit (for write or read register contents) and a register index. The first data byte consists of the most significant 4 bits of the control setting and the second data byte consists of the least significant 4 bits of the control setting.

Reading Status from the SX2

The SX2 INT pin provides the SH3 with two signals depending on mode. When not in a Read SX2 register sequence, the INT signals the SH3 that a USB event—SX2 internal interrupt (refer to Section 3.4 in the SX2 data sheet, Reference item 1)—has occurred. When in a Read SX2 register sequence, the INT signals the SH3 that the requested register status byte is available for reading from the SX2 FD [7:0] data bus.

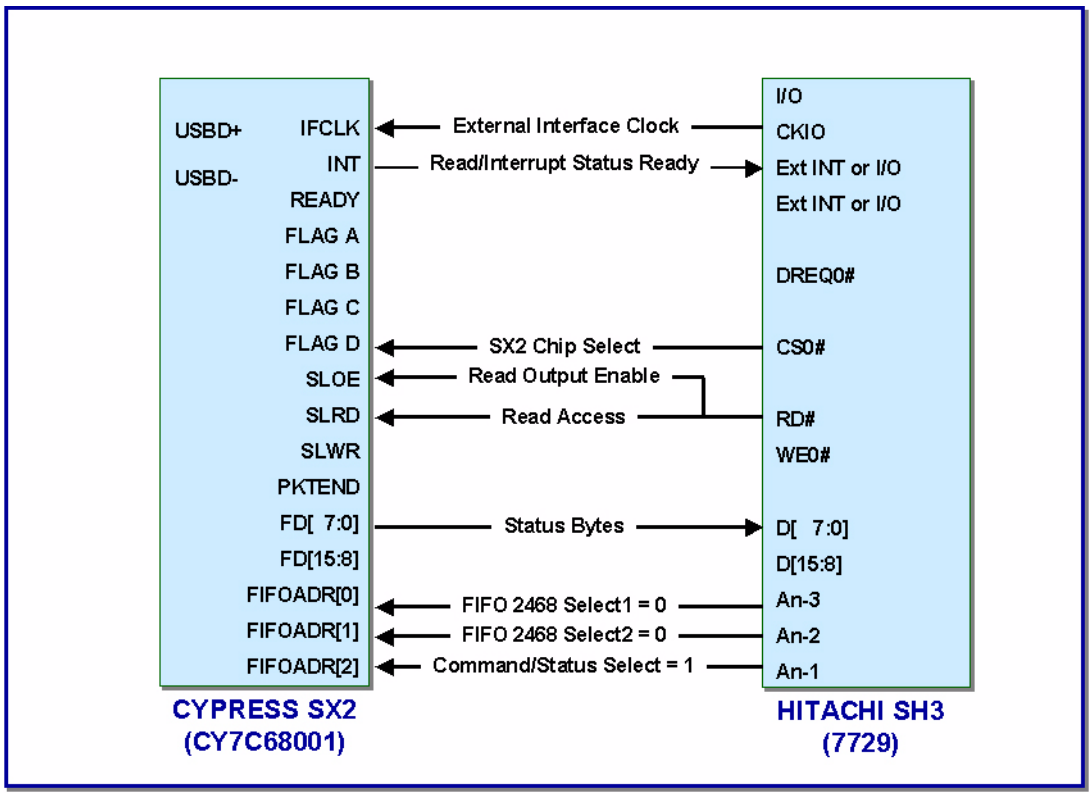

Figure 3. SX2 Read Status Signals

To Read the SX2 interrupt status register contents:

1. On an INT signal
   a. Read the Status byte

To Read an SX2 control register contents:

1. Wait for the SX2 Ready signal
2. Write the SX2 Read register Command sequence for the desired register
3. Wait for the SX2 INT signal
4. Read the Status byte

During a control register read sequence, any USB event interrupts are held off until the read sequence completes, then the INT line is reasserted for the interrupt event. Multiple interrupt events are queued up and serviced in sequence.

Reading OUT Packet Data from SX2 Endpoint Buffer

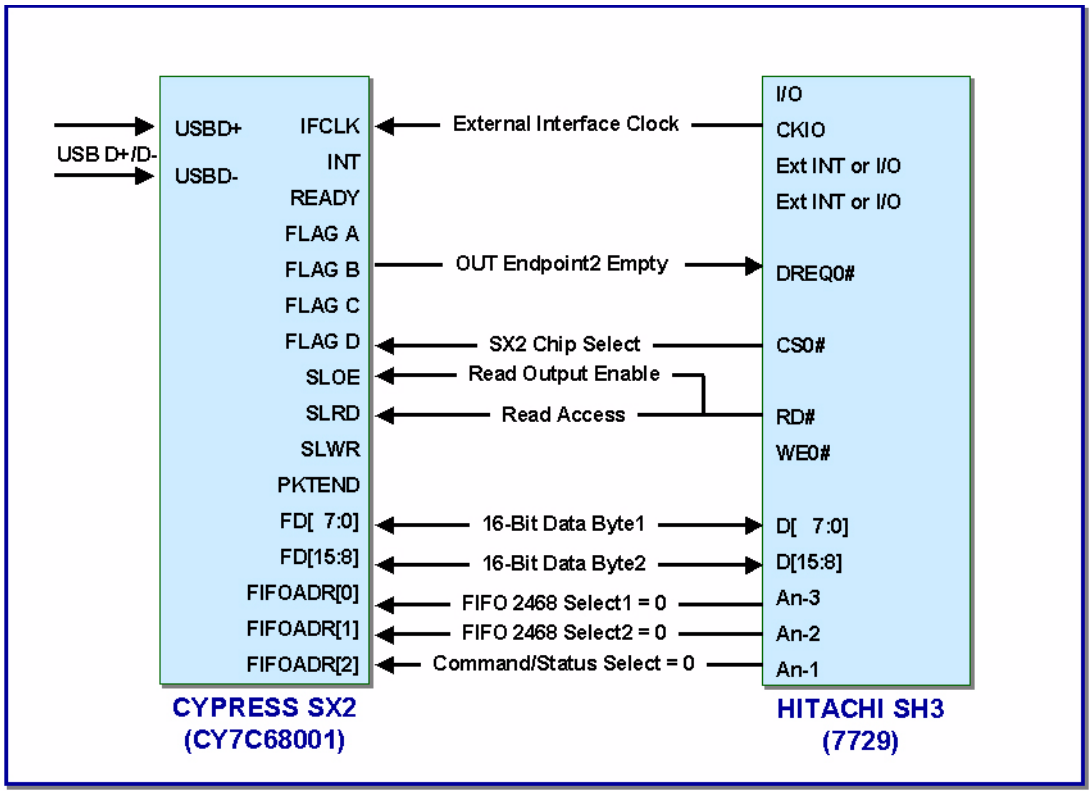

Figure 4. SX2 16-Bit Data Read Signals

The SH3 allocates the address space at A [n] to the SX2 and relies on CS generation to enable the SX2 for read portions of the dual-address DMA operation. The SH3 sets the upper address bits A [n-1: n-3] for FIFOADR [2:0] equal to (000b) to read OUT packet data from the SX2 Endpoint 2 FIFO buffers.

Writing IN Packet Data to SX2 Endpoint Buffer

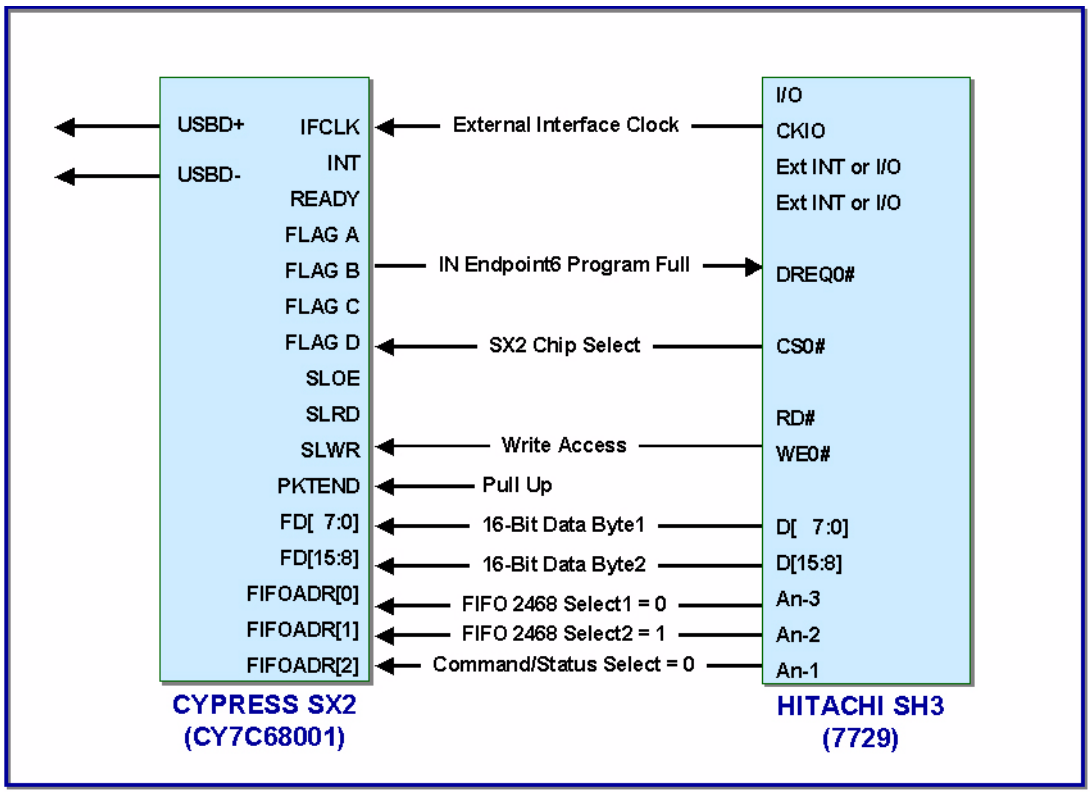

Figure 5. SX2 16-Bit Data Write Signals

The SH3 allocates the address space at A[n] to the SX2 and relies on CS generation to enable the SX2 for write portions of the dual-address DMA operation. The SH3 sets the upper address bits A [n-1: n-3] for FIFOADR [2:0] equal to (010b) to write IN packet data to the SX2 Endpoint 6 FIFO buffers.

Note: setting WORDWIDE in either the Endpoint 2 or Endpoint 6 FIFO configuration register enables 16-bit data mode. Setting either endpoint configuration to WORDWIDE enables the 16-bit interface. 16-bit transfers occur only if the WORDWIDE bit is set for the currently addressed endpoint buffer, otherwise default 8-bit transfers are performed.

Example SX2 Control Sequence for Bulk Loop Back

The following commands are sent to the SX2 to initialize it and dynamically control it for a loopback set-up and operation on Endpoints 2 and 6. Endpoint 2 is initialized for OUT packet transactions and Endpoint 6 is initialized for IN packet transactions. Endpoints 4 and 8 are programmed off and are not used in this example. The sample descriptor at the end of this application note shows the settings for Endpoints 2 and 6 for both High-Speed and Full (Other) Speed operations. Notice that Endpoints 4 and 8 are not included in the descriptors.

The length of the OUT packet transactions in total bytes is assumed known for the example. In typical operations the method for communicating total OUT transfer size is communicated through either a control packet or in a wrapper packet. The actual implementation is application specific.

SX2 Initiation sequence (“Reads” are Command/Status operations and “Programs” are Command/Write operations):

1. The SX2 powers up in async mode, so initial access to configure it to go into sync mode has to be async. Configure SH3 for async access mode
2. Release reset on SX2
3. Wait for Interrupt Status (INT) assertion
4. Read Interrupt Status Byte and check for READY interrupt
5. Program SX2 interface configuration (IFCONFIG register) for:

a. IFCLKSRC = 0 - external clock source
b. 3048MHZ = 0 - ignored for external clock source
c. IFCLKOE = 0 - ignored for external clock source
d. IFCLKPOL = 0 - if normal clock polarity required, else = 1 for inverted polarity
e. ASYNC = 0 - synchronous operation (changes from async to sync)
f. STANDBY = 0 - this bit can be set to put the SX2 into low-powered mode when the BUSACTIVITY bit indicates an absence of Bus Activity. This is not typically used for self-powered designs unless battery conservation is a requirement.
g. FLAGD/CS# = 1 - use Flag D as chip select
h. DISCON = 0 - to remain disconnected from the USB.

6. Set SH3 to sync mode operation.
7. Download descriptors (see example descriptor at end of this application note):
   a. Write Command Byte = address transfer, write request, descriptor register index (30h)
   b. Wait for READY
   c. Write descriptor size, LSB then MSB (in nibble format)
      i. Write LSB high nibble
      ii. Wait for READY
      iii. Write LSB low nibble
      iv. Wait for READY
      v. Write MSB high nibble
      vi. Wait for READY
      vii. Write MSB low nibble
   d. For each descriptor byte
      i. Wait for READY
      ii. Write high nibble
      iii. Wait for READY
      iv. Write low nibble
8. Wait for Interrupt Status (INT) assertion.
9. Read Interrupt Status Byte and check for enumeration (ENUMOK interrupt).
10. Read SX2 USB Function (FNADDR) register and test USB Speed (HSGRANT) bit
    a. Save speed status for use in programming SX2)
    b. If HSGRANT is True then set an SH3 firmware variable MaxPacketSize = 512
    c. Otherwise set the SH3 firmware variable MaxPacketSize = 64.
11. Program OUT Endpoint 2 FIFO configuration (EP2PKTLENH, EP2PKTLENL registers) for:
    a. INFM1 = 0 - ignored for OUT endpoint
    b. OEP1 = 1 - OUT endpoint flags occur 1 sample early to meet SH3 DMA timing (see SH3 Hardware Manual Figure 12.15 in Reference item 2)
    c. ZEROLEN = 0 - ignored for OUT endpoint
    d. WORDWIDE = 1 - for 16-bit data interface transfers
    e. PL[X:0] bits need not be set for OUT transfers, SIE will handle automatically
12. Program IN Endpoint 6 FIFO configuration (EP6PKTLENH, EP6PKTLENL registers) for:
    a. INFM1 - IN endpoint flags occur 1 sample early to meet SH3 DMA timing (see SH3 Hardware Manual Figure 12.15 in Reference item 2)
    b. OEP1 = 0 - ignore for IN endpoint
    c. ZEROLEN = 1 - SX2 sends a zero length IN packet when no data is in the buffer and INPKTEND is asserted
    d. WORDWIDE = 1 - 16-bit data interface transfers
    e. If the SH3 firmware variable MaxPacketSize is equal to 512 then
       i. PL [9:8] = 10b - Most significant bits of 512-byte packet size

   ii. PL [7:0] = 0000 0000b - Least significant byte of 512-byte packet size. (Note: for WORDWIDE transfers, EP6PKTLENL must be even)

f. Otherwise

   i. PL [9:8] = 00b - Most significant bits of 64-byte packet size

   ii. PL [7:0] = 0100 0000b - Least significant byte of 64-byte packet size. (Note: for WORDWIDE transfers, EP6PKTLENL must be even)

13. Program OUT Endpoint 2 configuration (EP2CFG register) for:
    a. VALID = 1 - to enable Endpoint 2
    b. DIR = 0 - for OUT direction
    c. Type = 10b - for BULK type
    d. SIZE = 0 - for 512-Byte buffer size
    e. STALL = 0 - initial condition is to clear the endpoint stall bit; during runtime, the master processor may set this bit to stall the endpoint (under error conditions per the USB2.0 specification)
    f. BUF = 10b - use double buffering.
14. Program IN Endpoint 6 configuration (EP6CFG register) for:
    a. VALID = 1 - to enable Endpoint 6
    b. DIR = 1 - for IN direction
    c. Type = 10b - for BULK type
    d. SIZE = 0 - for 512-Byte buffer size
    e. STALL = 0 - initial condition is to clear the endpoint stall bit; during runtime, the master processor may set this bit to stall the endpoint (under error conditions per the USB2.0 specification)
    f. BUF = 10b - use double FIFO buffering.
15. Program unused Endpoint 4 (EP4CFG register) for:
    b. VALID = 0 to disable Endpoint 4
    c. All other bits ignored.
16. Program unused Endpoint 8 (EP8CFG register) for:
    a. VALID = 0 to disable Endpoint 8
    b. All other bits ignored.
17. Program SX2 to flush the endpoint FIFO buffers to ensure proper flags and endpoint FIFO buffer operation (INPKTEND/FLUSH register):
    a. FIFO8 = 1 - Flush Endpoint 8 FIFO buffers
    b. FIFO6 = 1 - Flush Endpoint 6 FIFO buffers
    c. FIFO4 = 1 - Flush Endpoint 4 FIFO buffers
    d. FIFO2 = 1 - Flush Endpoint 2 FIFO buffers
    e. EP[3:0] = 0 - Do not force packet end on Endpoint 2,4,6,8
18. Program FIFO Empty (EF) Flags (POLAR register) for invert (Low to High assertion):
    a. WUPOL = 0 - Wake-up Pin polarity normal (High to Low)
    b. PKTEND = 0 - Packet End Pin polarity normal (High to Low)
    c. SLOE = 0 - Slave Output Enable Pin polarity normal (High to Low). Configurable via EEPROM bit only
    d. SLRD = 0 - Slave Read Pin polarity normal (High to Low). Configurable via EEPROM bit only
    e. SLWR = 0 - Slave Write Pin polarity normal (High to Low). Configurable via EEPROM bit only
    f. EF = 1 - FIFO Empty Flag Pins (FLAGB in this example) polarity inverted (Low to High)
    g. FF = 1 - FIFO Full Flag Pins (FLAGB in this example) polarity inverted (Low to High).
19. Program the SX2 FLAGB (FLAGSAB register, FLAGCD register ignored) for OUT Endpoint 2 Empty:
    a. FLAGA [3:0] = 0000b - FLAGA pin ignored
    b. FLAGB [3:0] = 1000b - FLAGB pin active on Endpoint 2 FIFO Empty (EF) Flag
       i. At this point the FLAGB pin should be High (OUT Endpoint 2 FIFO is empty - polarity is invert).

20. Set SH3 DMA Channel0 for read bursts to SX2:
    a. Set transfer count (TC) to 1024-bytes of data
    b. Source is SX2 FIFO Endpoint 2 address (SX2 FIFOADR [2:0] = 000b)
    c. Destination is memory buffer loop back area.
21. Using EZ-USB Control Panel manually send:
    a. If High Speed connection
        i. Two 512-byte OUT packets of Bulk data on Endpoint 2.
    b. Otherwise Full Speed connection
        i. Eight 64-byte OUT packets of Bulk data on Endpoint 2.
22. The SH3 DMA should run twice in high-speed operations or eight times in full-speed operations
    a. When the Host commits a packet, the FLAGB (OUT Endpoint 2 Empty) goes to not true (LOW) creating a HIGH-to-LOW transition. The SH3 DMA senses this condition on DREQ0# and bursts data until the Endpoint 2 FIFO is empty. The FLAGB signals empty is true (HIGH) and the DMA transfer pauses (since the SH3 DMA Transfer Count is not zero).
    b. This repeats until all packets have been committed by the Host and the SH3 DMA Transfer Count (TC) = 0
23. Set SH3 DMA Channel0 for write bursts to SX2:
    a. Transfer count (TC) to 1024-bytes of data
    b. Destination is SX2 Endpoint 6 address (SX2 FIFOADR [2:0] = 010b)
    c. Source is memory buffer loop back area.
24. Program the SX2 FLAGB (FLAGSAB register, FLAGCD register ignored) for IN Endpoint 6 Empty:
    a. FLAGA [3:0] = 0000b - FLAGA pin ignored
    b. FLAGB [3:0] = 1010b - FLAGB pin active on Endpoint 6 FIFO Empty (EF) Flag
        i. At this point the FLAGB pin should be HIGH (IN Endpoint 6 FIFO is empty—polarity is invert).
25. Program the SX2 FLAGB (FLAGSAB register, FLAGCD register ignored) for IN Endpoint 6 Full:
    a. FLAGA [3:0] = 0000b - FLAGA pin ignored
    b. FLAGB [3:0] = 1110b - FLAGB pin active on Endpoint 6 FIFO Full (FF) Flag
        i. At this point the FLAGB pin should be LOW (IN Endpoint 6 FIFO is not full—polarity is invert).
26. The SH3 DMA should run once in high-speed operations or five times in full-speed operations
    a. If high-speed connection
        i. Since Endpoint 6 is double buffered with buffer size of 512 bytes, a single DMA burst fills both buffers with the complete 1024 bytes of loopback data. The SX2 FLAGB goes to HIGH (Endpoint 6 FIFO Full), the DMA TC = 0, and the DMA transfer terminates.
    b. Otherwise full-speed
        i. Since Endpoint 6 is double buffered with buffer size of 64 bytes, a single DMA burst fills both buffers with 128 bytes of loopback data, the SX2 FLAGB goes to HIGH (Endpoint 6 FIFO Full), and the DMA pauses by throttling off.
27. Using EZ-USB® Control Panel manually get:
    a. If high-speed connection
        i. Two 512-byte IN packets of Bulk data on Endpoint 6.
        ii. This completes the loopback for high-speed operations.
    b. Otherwise full-speed connection
        i. Eight 64-byte IN packets of Bulk data on Endpoint 6.
        ii. After each of the first six 64-byte packets, the SH3 DMA will throttle back on and burst another 64-bytes of loop back data.
        iii. After the first six packets are transferred, the SH3 DMA TC = 0 and the DMA operations terminate.

        Repeat at step 15 for multiple packet loopback transactions.

DREQ0# Waveform Generation with FLAGB

OUT Packet Read Bursts

The FLAGB line is used to create a waveform to throttle a DMA read burst of data from the SX2 to the SH3 when accessing OUT endpoint packet contents. The FLAGB polarity is programmed for inverse (active HIGH) and the pin is programmed for Endpoint 2 FIFO Empty at the beginning of an OUT packet read, this creates a HIGH-to-LOW transition when the Endpoint 2 FIFO buffer goes non-empty creating a DMA request (SH3 DREQ0# line). When the buffer goes empty, a LOW-to-HIGH transition occurs, creating a DMA request termination. This set of conditions throttles the DMA request line and this continues indefinitely, or until the DMA has transferred a preset amount of data, stops, and notifies the SH3 CPU that the complete transfer is done. Since the DMA performs one additional data move after the deassertion of the DMA request (FLAGB), the SX2 is programmed to deassert FLAGB (EF) one read early. At the end of the transfer the master sets FLAGB to EP6EF to handle the IN data burst.

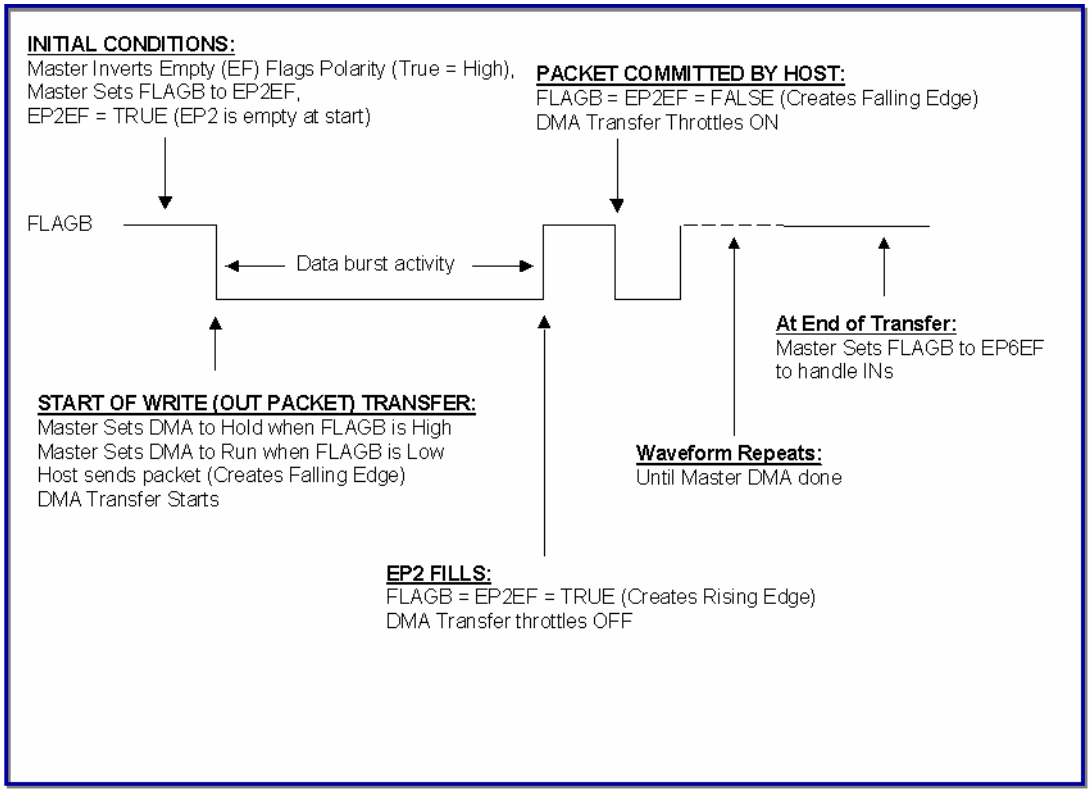

Figure 6. DREQ0# Waveform for OUT Packet DMA Read Bursts

IN Packet Write Bursts

The FLAGB line is also used to create a waveform to throttle a DMA burst write from the SH3 to the SX2 when writing IN endpoint packet contents. FLAGB is first programmed for endpoint 6 FIFO empty with active HIGH polarity. This ensures that FLAGB is HIGH at the start. Then, FLAGB is programmed for Endpoint 6 FIFO full (this creates the necessary HIGH-to-LOW transition for the SH3's DREQ0# signal) and will remain LOW until the Full Flag condition occurs. Then, FLAGB will assert and cause the DMA burst to throttle. Since the SH3 DMA performs one additional data move after the deassertion of the DMA request (FLAGB), the SX2 is programmed to deassert FLAGB (FF) one write early. At the end of the transfer the master sets FLAGB to EP2EF to handle OUTs again.

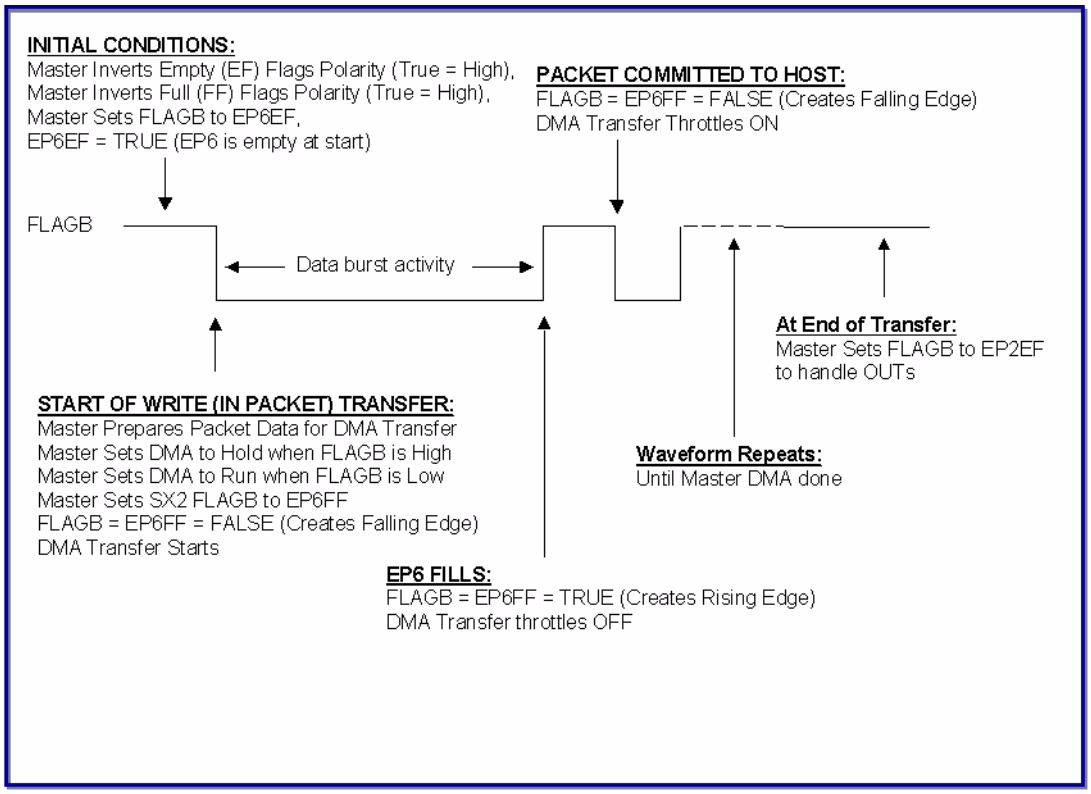

Figure 7. DREQ0# Waveform for IN Packet DMA Write Bursts

Bulk Loopback Sample Descriptors

```
//Device Descriptor
18, // Descriptor Length
1,  // Descriptor Type - Device
0x00, 0x02,// Specification Version (BCD)
0,  // Device Class
0,  // Device Sub-class
0,  // Device Sub-sub-class
64, // Control Endpoint Maximum Packet Size
0x00, 0x00// Vendor ID (example 0xB4, 0x04 = Cypress)
0x00, 0x00// Product ID (example 0x02, 0x10 = Sample Device)
0x00, 0x00// Device ID (product version number)
1,  // Manufacturer String Index
2,  // Product String Index
0,  // Serial Number String Index
1,  // Number of Configurations

// Device Qualifier Descriptor
10, // Descriptor Length
6,  // Descriptor Type - Device Qualifier
0x00, 0x02,// Specification Version (BCD)
0,  // Device Class
0,  // Device Sub-class
0,  // Device Sub-sub-class
64, // Control Endpoint Maximum Packet Size
1,  // Number of Configurations
0,  // Reserved

// High Speed Configuration Descriptor
9,  // Descriptor Length
2,  // Descriptor Type - Configuration
34, // Total Length (LSB)
0,  // Total Length (MSB)
1,  // Number of Interfaces
1,  // Configuration Number
0,  // Configuration String
0x40,// Attributes (b6 - Self Powered)
50, // Power Requirement (div 2 mA - claiming max unconfigured)

// Interface Descriptor
9,  // Descriptor Length
4,  // Descriptor Type - Interface
0,  // Zero-based Index of this Descriptor
0,  // Alternate Setting
2,  // Number of Endpoints
0xFF,// Interface Class (Vendor Specific)
0,  // Interface Sub-class
0,  // Interface Sub-sub-class
0,  // Interface Descriptor String Index

// Endpoint Descriptor 1
7,  // Descriptor Length
5,  // Descriptor Type - Endpoint
0x02,// Endpoint Number and Direction (2 OUT)
2,  // Endpoint Type (Bulk)
0x00,// Maximum Packet Size (LSB)
0x02,// Maximum Packet Size (MSB)
0,  // Polling Interval

// Endpoint Descriptor 2
7,  // Descriptor Length
5,  // Descriptor Type - Endpoint
0x86,// Endpoint Number and Direction (6 IN)
2,  // Endpoint Type (Bulk)
```

```
0x00,// Maximum Packet Size (LSB)
0x02,// Maximum Packet Size (MSB)
0,  // Polling Interval

// Full Speed Configuration Descriptor
9,  // Descriptor Length
2,  // Descriptor Type - Configuration
34, // Total Length (LSB)
0,  // Total Length (MSB)
1,  // Number of Interfaces
1,  // Configuration Number
0,  // Configuration String
0x40,// Attributes (b6 - Self Powered)
50, // Power Requirement (div 2 mA - claiming max unconfigured)

// Interface Descriptor
9,  // Descriptor Length
4,  // Descriptor Type - Interface
0,  // Zero-based Index of this Descriptor
0,  // Alternate Setting
2,  // Number of Endpoints
0xFF,// Interface Class (Vendor Specific)
0,  // Interface Sub-class
0,  // Interface Sub-sub-class
0,  // Interface Descriptor String Index

// Endpoint Descriptor 1
7,  // Descriptor Length
5,  // Descriptor Type - Endpoint
0x02,// Endpoint Number and Direction (2 OUT)
2,  // Endpoint Type (Bulk)
0x40,// Maximum Packet Size (LSB)
0x00,// Maximum Packet Size (MSB)
0,  // Polling Interval

// Endpoint Descriptor 2
7,  // Descriptor Length
5,  // Descriptor Type - Endpoint
0x86,// Endpoint Number and Direction (6 IN)
2,  // Endpoint Type (Bulk)
0x40,// Maximum Packet Size (LSB)
0x00,// Maximum Packet Size (MSB)
0,  // Polling Interval

// String Descriptors

// String Descriptor 0
4,  // Descriptor Length
3,  // Descriptor Type - String
0x09, 0x04,// US LANG ID

// String Descriptor 1
16, // Descriptor Length
3,  // Descriptor Type - String
'C', 0,
'y', 0,
'p', 0,
'r', 0,
'e', 0,
's', 0,
's', 0,

// String Descriptor 2
20, // Descriptor Length
```

```
3,  // Descriptor Type - String
'C', 0,
'Y', 0,
'7', 0,
'C', 0,
'6', 0,
'8', 0,
'0', 0,
'0', 0,
'1', 0,
```

References

Data Sheets and Manuals

1. EZ-USB SX2™ (CY7C68001) Data Sheet, "38-08013.pdf" www.cypress.com
2. Hitachi SuperH® RISC (SH7729) Hardware Manual, "e602157_sh7729.pdf" - www.hitachisemiconductor.com
3. EZ-USB FX2™ (CY7C68013) Technical Reference Manual, "FX2_TechRefManual.pdf" - www.cypress.com

Specifications

1. USB 2.0 Specification - www.usb.org
2. USB Mass Storage Class Overview - www.usb.org
3. USB Mass Storage Class Bulk-Only Transport (BOT) Protocol - www.usb.org
4. USB Mass Storage Control, Bulk, Interrupt (CBI) Transport Protocol - www.usb.org
5. USB Communication Class - www.usb.org

approved dsg 1/7/02