Details	
Product Status	Obsolete
Core Processor	8032
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, IrDA, SPI, UART/USART, USB
Peripherals	LVD, POR, PWM, WDT
Number of I/O	46
Program Memory Size	160KB (160K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/upsd3433e-40u6

1 Description

The *Turbo Plus* UPSD34xx Series combines a powerful 8051-based microcontroller with a flexible memory structure, programmable logic, and a rich peripheral mix to form an ideal embedded controller. At its core is a fast 4-cycle 8032 MCU with a 4-byte instruction prefetch queue (PFQ) and a 4-entry fully associative branching cache (BC). The MCU is connected to a 16-bit internal instruction path to maximize performance, enabling loops of code in smaller localities to execute extremely fast. The 16-bit wide instruction path in the *Turbo Plus* Series allows double-byte instructions to be fetched from memory in a single memory cycle. This keeps the average performance near its peak performance (peak performance for 5 V, 40 MHz Turbo *Plus* UPSD34xx is 10 MIPS for single-byte instructions, and average performance will be approximately 9 MIPS for mix of single- and multi-byte instructions).

USB 2.0 (full speed, 12Mbps) is included, providing 10 endpoints, each with its own 64-byte FIFO to maintain high data throughput. Endpoint 0 (control endpoint) uses two of the 10 endpoints for In and Out directions, the remaining eight endpoints may be allocated in any mix to either type of transfers: Bulk or Interrupt.

Code development is easily managed without a hardware in-circuit emulator by using the serial JTAG debug interface. JTAG is also used for in-system programming (ISP) in as little as 10 seconds, perfect for manufacturing and lab development. The 8032 core is coupled to programmable system device (PSD) architecture to optimize the 8032 memory structure, offering two independent banks of Flash memory that can be placed at virtually any address within 8032 program or data address space, and easily paged beyond 64 Kbytes using on-chip programmable decode logic.

Dual Flash memory banks provide a robust solution for remote product updates in the field through in-application programming (IAP). Dual Flash banks also support EEPROM emulation, eliminating the need for external EEPROM chips.

General-purpose programmable logic (PLD) is included to build an endless variety of glue-logic, saving external logic devices. The PLD is configured using the software development tool, PSDsoft Express, available from the web at **www.st.com/psm**, at no charge.

The UPSD34xx also includes supervisor functions such as a programmable watchdog timer and low-voltage reset.

Note: For a list of known limitations of the UPSD34xx devices, please refer to Section 34: Important notes.

9 8032 addressing modes

The 8032 MCU uses 11 different addressing modes listed below:

- Register
- Direct
- Register indirect
- Immediate
- External direct
- External indirect
- Indexed
- Relative
- Absolute
- Long
- Bit

9.1 Register addressing

This mode uses the contents of one of the registers R0 - R7 (selected by the last three bits in the instruction opcode) as the operand source or destination. This mode is very efficient since an additional instruction byte is not needed to identify the operand. For example:

```
MOV A, R7                        ; Move contents of R7 to accumulator
```

9.2 Direct addressing

This mode uses an 8-bit address, which is contained in the second byte of the instruction, to directly address an operand which resides in either 8032 DATA SRAM (internal address range 00h-07Fh) or resides in 8032 SFR (internal address range 80h-FFh). This mode is quite fast since the range limit is 256 bytes of internal 8032 SRAM. For example:

```
MOV A, 40h                       ; Move contents of DATA SRAM
                                 ; at location 40h into the accumulator
```

9.3 Register indirect addressing

This mode uses an 8-bit address contained in either register R0 or R1 to indirectly address an operand which resides in 8032 IDATA SRAM (internal address range 80h-FFh). Although 8032 SFR registers also occupy the same physical address range as IDATA, SFRs will not be accessed by register Indirect mode. SFRs may only be accesses using Direct address mode. For example:

```
MOV A, @R0                       ; Move into the accumulator the
                                 ; contents of IDATA SRAM that is
                                 ; pointed to by the address
                                 ; contained in R0.
```

Table 9. **Boolean variable manipulation instruction set**

Mnemonic[1] and use		Description	Length/cycles
CLR	C	Clear carry	1 byte/1 cycle
CLR	bit	Clear direct bit	2 byte/1 cycle
SETB	C	Set carry	1 byte/1 cycle
SETB	bit	Set direct bit	2 byte/1 cycle
CPL	C	Compliment carry	1 byte/1 cycle
CPL	bit	Compliment direct bit	2 byte/1 cycle
ANL	C, bit	AND direct bit to carry	2 byte/2 cycle
ANL	C, /bit	AND compliment of direct bit to carry	2 byte/2 cycle
ORL	C, bit	OR direct bit to carry	2 byte/2 cycle
ORL	C, /bit	OR compliment of direct bit to carry	2 byte/2 cycle
MOV	C, bit	Move direct bit to carry	2 byte/1 cycle
MOV	bit, C	Move carry to direct bit	2 byte/2 cycle
JC	rel	Jump if carry is set	2 byte/2 cycle
JNC	rel	Jump if carry is not set	2 byte/2 cycle
JB	rel	Jump if direct bit is set	3 byte/2 cycle
JNB	rel	Jump if direct bit is not set	3 byte/2 cycle
JBC	bit, rel	Jump if direct bit is set and clear bit	3 byte/2 cycle

1. All mnemonics copyrighted ©Intel Corporation 1980.

Table 10. **Program branching instruction set**

Mnemonic[1] and use		Description	Length/cycles
Program Branching Instructions			
ACALL	addr11	Absolute subroutine call	2 byte/2 cycle
LCALL	addr16	Long subroutine call	3 byte/2 cycle
RET		Return from subroutine	1 byte/2 cycle
RETI		Return from interrupt	1 byte/2 cycle
AJMP	addr11	Absolute jump	2 byte/2 cycle
LJMP	addr16	Long jump	3 byte/2 cycle
SJMP	rel	Short jump (relative addr)	2 byte/2 cycle
JMP	@A+DPTR	Jump indirect relative to the DPTR	1 byte/2 cycle
JZ	rel	Jump if ACC is zero	2 byte/2 cycle
JNZ	rel	Jump if ACC is not zero	2 byte/2 cycle
CJNE	A, direct, rel	Compare direct byte to ACC, jump if not equal	3 byte/2 cycle
CJNE	A, #data, rel	Compare immediate to ACC, jump if not equal	3 byte/2 cycle
CJNE	Rn, #data, rel	Compare immediate to register, jump if not equal	3 byte/2 cycle

Table 21. IEA register bit definition (continued)

Bit	Symbol	R/W	Function
3	–	–	Reserved, do not set to logic '1.'
2	–	–	Reserved, do not set to logic '1.'
1[(1)]	EI^{2}C	R,W	Enable I^{2}C Interrupt
0	EUSB	R,W	Enable USB Interrupt

1. 1 = Enable Interrupt, 0 = Disable Interrupt.

Table 22. IP: interrupt priority register (SFR B8h, reset value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	–	PT2	PS0	PT1	PX1	PT0	PX0

Table 23. IP register bit definition

Bit	Symbol	R/W	Function
7	–	–	Reserved
6	–	–	Reserved
5[(1)]	PT2	R,W	Timer 2 Interrupt priority level
4[(1)]	PS0	R,W	UART0 Interrupt priority level
3[(1)]	PT1	R,W	Timer 1 Interrupt priority level
2[(1)]	PX1	R,W	External Interrupt INT1 priority level
1[(1)]	PT0	R,W	Timer 0 Interrupt priority level
0[(1)]	PX0	R,W	External Interrupt INT0 priority level

1. 1 = Assigns high priority level, 0 = Assigns low priority level.

Table 24. IPA: Interrupt Priority Addition register (SFR B7h, reset value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PADC	PSPI	PPCA	PS1	–	–	PI^{2}C	PUSB

Table 25. IPA register bit definition

Bit	Symbol	R/W	Function
7[(1)]	PADC	R,W	ADC Interrupt priority level
6[(1)]	PSPI	R,W	SPI Interrupt priority level
5[(1)]	PPCA	R,W	PCA Interrupt level
4[(1)]	PS1	R,W	UART1 Interrupt priority level
3	–	–	Reserved
2	–	–	Reserved
1[(1)]	PI^{2}C	R,W	I^{2}C Interrupt priority level
0	PUSB	R,W	USB Interrupt priority level

1. 1 = Assigns high priority level, 0 = Assigns low priority level.

19.2 Low V_{CC} voltage detect, LVD

An internal reset is generated by the LVD circuit when V_{CC} drops below the reset threshold, V_{LV_THRESH}. After V_{CC} returns to the reset threshold, the MCU_RESET signal will remain asserted for t_{RST_ACTV} before it is released. The LVD circuit is always enabled (cannot be disabled by SFR), even in Idle mode and Power-down mode. The LVD input has a voltage hysteresis of V_{RST_HYS} and will reject voltage spikes less than a duration of t_{RST_FIL}.

Important note: The LVD voltage threshold is V_{LV_THRESH}, suitable for monitoring both the 3.3 V V_{CC} supply on the MCU module and the 3.3 V V_{DD} supply on the PSD module for 3.3 V UPSD34xxV devices, since these supplies are one in the same on the circuit board.

However, for 5 V UPSD34xx devices, V_{LV_THRESH} is not suitable for monitoring the 5 V V_{DD} voltage supply (V_{LV_THRESH} is too low), but good for monitoring the 3.3 V V_{CC} supply. In the case of 5 V UPSD34xx devices, an external means is required to monitor the separate 5 V V_{DD} supply, if desired.

19.3 Power-up reset

At power up, the internal reset generated by the LVD circuit is latched as a logic '1' in the POR bit of the SFR named PCON (*Table 33 on page 74*). Software can read this bit to determine whether the last MCU reset was the result of a power up (cold reset) or a reset from some other condition (warm reset). This bit must be cleared with software.

19.4 JTAG debug reset

The JTAG Debug Unit can generate a reset for debugging purposes. This reset source is also available when the MCU is in Idle mode and Power-Down mode (the user can use the JTAG debugger to exit these modes).

19.5 Watchdog timer, WDT

When enabled, the WDT will generate a reset whenever it overflows. Firmware that is behaving correctly will periodically clear the WDT before it overflows. Run-away firmware will not be able to clear the WDT, and a reset will be generated.

By default, the WDT is disabled after each reset.

Note: The WDT is not active during Idle mode or Power-down mode.

There are two SFRs that control the WDT, they are WDKEY (*Table 52 on page 94*) and WDRST (*Table 54 on page 94*).

If WDKEY contains 55h, the WDT is disabled. Any value other than 55h in WDKEY will enable the WDT. By default, after any reset condition, WDKEY is automatically loaded with 55h, disabling the WDT. It is the responsibility of initialization firmware to write some value other than 55h to WDKEY after each reset if the WDT is to be used.

The WDT consists of a 24-bit up-counter (*Figure 23*), whose initial count is 000000h by default after every reset. The most significant byte of this counter is controlled by the SFR, WDRST. After being enabled by WDKEY, the 24-bit count is increased by 1 for each MCU machine cycle. When the count overflows beyond FFFFFh (2^{24} MCU machine cycles), a reset is issued and the WDT is automatically disabled (WDKEY = 55h again).

Table 55. WDRST register bit definition

Bit	Symbol	R/W	Definition
[7:0]	WDRST	W	This SFR is the upper byte of the 24-bit WDT up-counter. Writing this SFR sets the upper byte of the counter to the written value, and clears the lower two bytes of the counter to 0000h. Counting begins when WDKEY does not contain 55h.

in use as a baud rate generator, the pin T2X can be used as an extra external interrupt, if desired.

When Timer 2 is running (TR2 = 1) in a “timer” function in the Baud rate generator mode, firmware should not read or write TH2 or TL2. Under these conditions the results of a read or write may not be accurate. However, SFRs RCAP2H and RCAP2L may be read, but should not be written, because a write might overlap a reload and cause write and/or reload errors. Timer 2 should be turned off (clear TR2) before accessing Timer 2 or registers RCAP2H and RCAP2L, in this case.

Table 63 on page 104 shows commonly used baud rates and how they can be obtained from Timer 2, with T2CON = 34h.

Table 63. Commonly used baud rates generated from timer2 (T2CON = 34h)

f_{OSC} MHz	Desired baud rate	Timer 2 SFRs		Resulting baud rate	Baud rate deviation
		RCAP2H (hex)	RCAP2L(hex)		
40.0	115200	FF	F5	113636	-1.36%
40.0	57600	FF	EA	56818	-1.36%
40.0	28800	FF	D5	29070	0.94%
40.0	19200	FF	BF	19231	0.16%
40.0	9600	FF	7E	9615	0.16%
36.864	115200	FF	F6	115200	0
36.864	57600	FF	EC	57600	0
36.864	28800	FF	D8	28800	0
36.864	19200	FF	C4	19200	0
36.864	9600	FF	88	9600	0
36.0	28800	FF	D9	28846	0.16%
36.0	19200	FF	C5	19067	-0.69%
36.0	9600	FF	8B	9615	0.16%
24.0	57600	FF	F3	57692	0.16%
24.0	28800	FF	E6	28846	0.16%
24.0	19200	FF	D9	19231	0.16%
24.0	9600	FF	B2	9615	0.16%
12.0	28800	FF	F3	28846	0.16%
12.0	9600	FF	D9	9615	0.16%
11.0592	115200	FF	FD	115200	0
11.0592	57600	FF	FA	57600	0
11.0592	28800	FF	F4	28800	0
11.0592	19200	FF	EE	19200	0
11.0592	9600	FF	DC	9600	0

23 I^2C interface

UPSD34xx devices support one serial I^2C interface. This is a two-wire communication channel, having a bidirectional data signal (SDA, pin P3.6) and a clock signal (SCL, pin P3.7) based on open-drain line drivers, requiring external pull-up resistors, R_P each with a typical value of 4.7kΩ (see *Figure 40*).

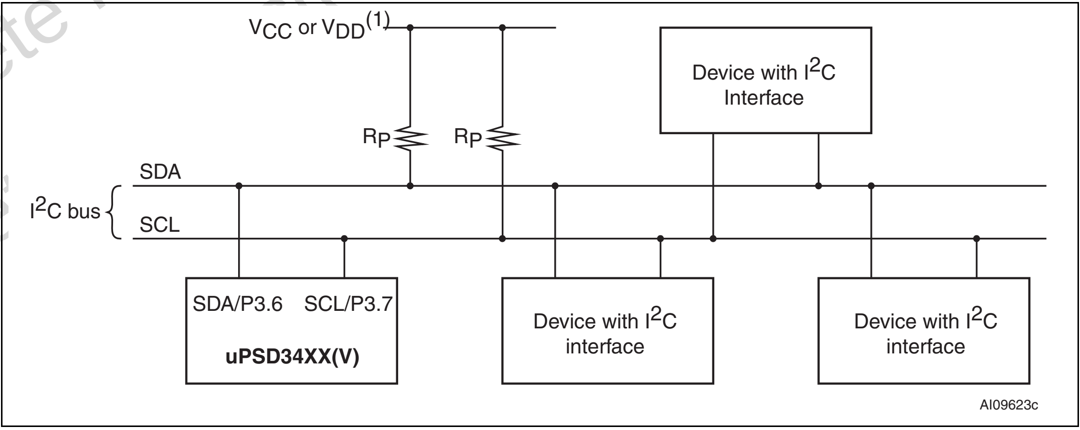
23.1 I^2C interface main features

Byte-wide data is transferred, MSB first, between a Master device and a Slave device on two wires. More than one bus Master is allowed, but only one Master may control the bus at any given time. Data is not lost when another Master requests the use of a busy bus because I^2C supports collision detection and arbitration. The bus Master initiates all data movement and generates the clock that permits the transfer. Once a transfer is initiated by the Master, any device addressed is considered a Slave. Automatic clock synchronization allows I^2C devices with different bit rates to communicate on the same physical bus. A single device can play the role of Master or Slave, or a single device can be a Slave only. Each Slave device on the bus has a unique address, and a general broadcast address is also available. A Master or Slave device has the ability to suspend data transfers if the device needs more time to transmit or receive data.

This I^2C interface has the following features:

- Serial I/O Engine (SIOE): serial/parallel conversion; bus arbitration; clock generation and synchronization; and handshaking are all performed in hardware
- Interrupt or Polled operation
- Multi-master capability
- 7-bit Addressing
- Supports standard speed I^2C (SCL up to 100kHz), fast mode I^2C (101KHz to 400kHz), and high-speed mode I^2C (401KHz to 833kHz)

Figure 40. Typical I^2C bus configuration


VCC or VDD(1)
Device with I2C Interface
RP
RP
SDA
I2C bus
SCL
SDA/P3.6
SCL/P3.7
uPSD34XX(V)
Device with I2C interface
Device with I2C interface
AI09623c

1. For 3.3 V system, connect R_P to 3.3 V V_{CC}. For 5.0 V system, connect R_P to 5.0 V V_{DD}.

```
Enable individual I2C interrupt and set priority
    –   SFR IEA.I2C = 1
    –   SFR IPA.I2C = 1 if high priority is desired
Set the Device address for Slave mode
    –   SFR S1ADR = XXh, desired address
Enable SIOE (as Slave) to return an ACK signal
    –   SFR S1CON.AA = 1
```

Master-Transmitter

```
Disable all interrupts
    –   SFR IE.EA = 0
Set pointer to global data xmit buffer, set count
    –   *xmit_buf = *pointer to data
    –   buf_length = number of bytes to xmit
Set global variables to indicate Master-Xmitter
    –   I2C_master = 1, I2C_xmitter = 1
Disable Master from returning an ACK
    –   SFR S1CON.AA = 0
Enable I2C SIOE
    –   SFR S1CON.INI1 = 1
Transmit Address and R/W bit = 0 to Slave
    –   Is bus not busy? (SFR S1STA.BBUSY = 0?)
    <If busy, then test until not busy>
    –   SFR S1DAT[7:0] = Load Slave Address & FEh
    –   SFR S1CON.STA = 1, send Start on bus
    <bus transmission begins>
Enable All Interrupts and go do something else
    –   SFR IE.EA = 1
```

Master-Receiver

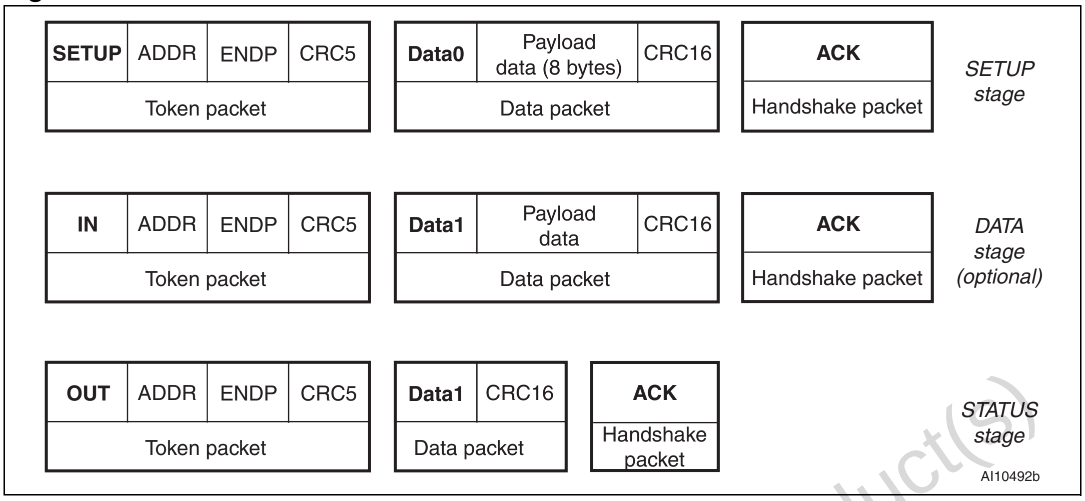
```
Disable all interrupts
    –   SFR IE.EA = 0
Set pointer to global data recv buffer, set count
    –   *recv_buf = *pointer to data
    –   buf_length = number of bytes to recv
Set global variables to indicate Master-Xmitter
    –   I2C_master = 1, I2C_xmitter = 0
Disable Master from returning an ACK
    –   SFR S1CON.AA = 0
```

```
Is this the next to last byte to receive from Slave?
       If this is the next to last byte, do not allow Master to ACK
       on next interrupt.
    -  S1CON.AA = 0, don't let Master return ACK
    -  Exit ISR, now ready to recv last byte from Slv
       If this is not next to last byte, let Master send ACK to
       Slave
       <S1CON.AA is already 1>
    -  Exit ISR, ready to recv more bytes from Slave
```

Else If mode is Slave-Transmitter:

```
Is this Intr from SIOE detecting a Stop on bus?
       If Yes, a Stop was detected:
    -  S1DAT = dummy, write to release bus
    -  Exit ISR, Master needs no more data bytes
       If No, a Stop was not detected, continue:
ACK recvd from Master? (status.ACK_RESP=0?)
       If No, an ACK was not received:
    -  S1DAT = dummy, write to release bus
    -  Exit ISR, Master needs no more data bytes
       If Yes, ACK was received, then continue:
    -  S1DAT = xmit_buf[buffer_index], transmit byte
    -  Exit ISR, transmit next byte on next interrupt
Else If mode is Slave-Receiver:
Is this Intr from SIOE detecting a Stop on bus?
       If Yes, a Stop was detected:
    -  recv_buf[buffer_index] = S1DAT, get last byte
    -  Exit ISR, Master has sent last byte
       If No, a Stop was not detected, continue:
Determine if this Interrupt is from receiving an address or a data
byte from a Master.
Is (S1CON.ADDR = 1 and S1CON.AA =1)?
       If No, intr is from receiving data, goto C:
       If Yes, intr is from an address, continue:
    -  slave_is_adressed = 1, local variable set true
       <indicates Master selected this slave>
    -  S1CON.ADDR = 0, clear address match flag
Determine if R/W bit indicates transmit or receive.
```

Figure 52. Control transfer



25.3 Endpoint FIFOs

The UPSD34xx's USB module includes 5 endpoints and 10 FIFOs. Each endpoint has two FIFOs with one for IN and the other for OUT transactions.   Each FIFO is 64 bytes long and is selectively made visible in a 64-byte XDATA segment for CPU access. For efficient data transfers, the FIFOs may be paired for double buffering. With double buffering, the CPU may operate on the contents in one buffer while the SIE is transmitting or receiving data in the paired buffer. UPSD34xx supported endpoints and FIFOs are shown in *Table 98*

25.3.1 Busy bit (BSY) operation

Each FIFO has a busy bit (BSY) that indicates when the USB SIE has ownership of the FIFO. When the SIE has ownership of the FIFO, it is either writing data to or reading data from the FIFO. The SIE writes data to the FIFO when it is receiving an OUT packet and reads data from the FIFO when it is sending data in response to an IN packet. The CPU is only permitted to access the FIFO when it is not busy and accesses to it while busy are ignored. Once the IN FIFO has been written with data by the CPU, the CPU updates the USIZE register with the number of bytes written to the FIFO. The value written to the USIZE register tells the SIE the number of bytes to send to the host in response to an IN packet. Once the USIZE register is written, the FIFOs busy bit is set and remains set until the data has been transmitted in response to an IN packet. The busy bit for an OUT FIFO is set as soon as the SIE starts receiving an OUT packet from the host. Once all the data has been received and written to the FIFO, the SIE clears the busy bit and writes the number of bytes received to the USIZE register.

25.3.2 Busy bit and interrupts

When the FIFO's interrupt is enabled, a transition of the busy bit from a '1' to a '0' (when ownership of the FIFO changes from the SIE to the CPU) generates a USB interrupt with the corresponding flag set. For an interrupt on an IN FIFO, the CPU must fill the FIFO with the next set of data to be sent and then update the USIZE register with the number of bytes to send. For an interrupt on an OUT FIFO, the CPU reads the USIZE register to determine the number of bytes received and then reads that number of data bytes out of the FIFO.

Table 98. UPSD34xx supported endpoints

Endpoint	Function	Max packet size (FIFO size)	Supported directions
0	Control	64 bytes	OUT
0	Control	64 bytes	IN
1	Bulk/Interrupt OUT	64 bytes	OUT
1	Bulk/Interrupt IN	64 bytes	IN
2	Bulk/Interrupt OUT	64 bytes	OUT
2	Bulk/Interrupt IN	64 bytes	IN
3	Bulk/Interrupt OUT	64 bytes	OUT
3	Bulk/Interrupt IN	64 bytes	IN
4	Bulk/Interrupt OUT	64 bytes	OUT
4	Bulk/Interrupt In	64 bytes	IN

25.3.3 FIFO pairing

The FIFOs on endpoints 1 through 4 may be used independently as shown in *Figure 53* as FIFOs with no Pairing or they may be selectively paired to provide double buffering (see *Figure 54 on page 159*). Double buffering provides an efficient way to optimize data transfer rates with bulk transfers. Double buffering allows the CPU to process a data packet for an Endpoint while the SIE is receiving or transmitting another packet of data on the same Endpoint and direction. FIFO pairing is controlled by the USB pairing control register (see UPAIR, *Table 102 on page 162*). FIFO pairing options are listed below:

- IN FIFO 1 and 2
- OUT FIFO 1 and 2
- IN FIFO 3 and 4
- OUT FIFO 3 and 4

Note: *When the FIFOs are paired, the CPU must access the odd numbered FIFO while the even numbered FIFOs are no longer available for use. Also when they are paired, the active FIFO is automatically toggled by the update of USIZE.*

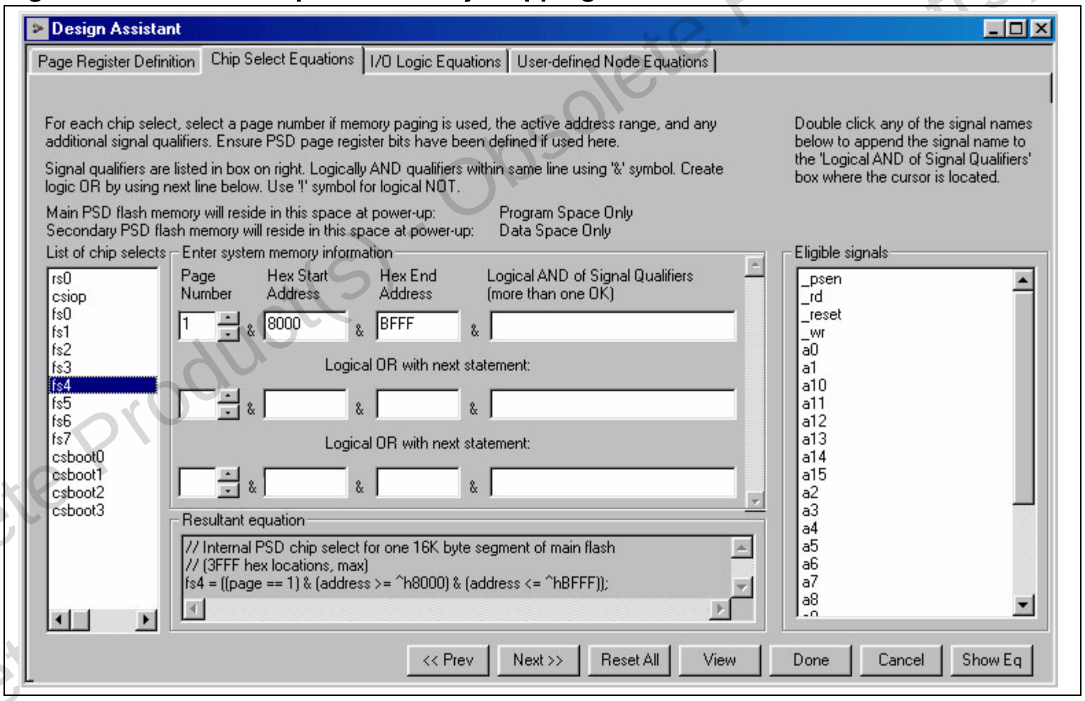
- Non-pairing FIFOs Example

  Consider a case where the device needs to send 1024 bytes of data to the host. Without FIFO pairing (see *Figure 53*), the CPU loads the IN Endpoint0 FIFO with 64 bytes of data and waits until the host sends an IN token to Endpoint0, and the SIE transfers the data to the host. Once all 64 bytes have been transferred by the SIE, the FIFO becomes empty and the CPU starts writing the next 64 bytes of data to the FIFO. While the CPU is writing the data to the FIFO, the host is sending IN tokens to Endpoint0, requesting the next 64 bytes of data, but only gets NAKs while the FIFO is being loaded. Once the FIFO has been loaded by the CPU, the SIE starts sending the data to the host with the next IN Endpoint0 token. Again, the CPU waits until the SIE transfers the 64 bytes of data to the host. This is repeated until all 1024 bytes have been transferred.

Table 159. HDL statement example generated from PSDsoft express for memory map

```
rs0     = ((address ≥ ^h0000)  &  (address ≤^h1FFF));
csiop   = ((address ≥ ^h2000)  &  (address ≤^h20FF));
fs0     = ((address ≥ ^h0000)  &  (address ≤^h3FFF));
fs1     = ((address ≥ ^h4000)  &  (address ≤^h7FFF));
fs2     = ((page == 0)         &  (address ≥ ^h8000)   &  (address ≤^hBFFF));
fs3     = ((page == 0)         &  (address ≥ ^hC000)   &  (address ≤^hFFFF));
fs4     = ((page == 1)         &  (address ≥ ^h8000)   &  (address ≤^hBFFF));
fs5     = ((page == 1)         &  (address ≥ ^hC000)   &  (address ≤^hFFFF));
fs6     = ((page == 2)         &  (address ≥ ^h8000)   &  (address ≤^hBFFF));
fs7     = ((page == 2)         &  (address ≥ ^hC000)   &  (address ≤^hFFFF));
csboot0 = ((address ≥ ^h8000)  &  (address ≤^h9FFF));
csboot1 = ((address ≥ ^hA000)  &  (address ≤^hBFFF));
csboot2 = ((address ≥ ^hC000)  &  (address ≤^hDFFF));
csboot3 = ((address ≥ ^hE000)  &  (address ≤^hFFFF));
```

Figure 64. PSDsoft express memory mapping



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

Main PSD flash memory will reside in this space at power-up: Program Space Only
Secondary PSD flash memory will reside in this space at power-up: Data Space Only

List of chip selects: rs0, csiop, fs0, fs1, fs2, fs3, fs4, fs5, fs6, fs7, csboot0, csboot1, csboot2, csboot3

Enter system memory information

Page Number		Hex Start Address		Hex End Address		Logical AND of Signal Qualifiers (more than one OK)
1	&	8000	&	BFFF	&	

Logical OR with next statement:

Logical OR with next statement:

Eligible signals: _psen, _rd, _reset, _wr, a0, a1, a10, a11, a12, a13, a14, a15, a2, a3, a4, a5, a6, a7, a8

Resultant equation

```
// Internal PSD chip select for one 16K byte segment of main flash
// (3FFF hex locations, max)
fs4 = ((page == 1) & (address >= ^h8000) & (address <= ^hBFFF));
```

28.2.4 EEPROM emulation

EEPROM emulation is needed if it is desired to repeatedly change only a small number of bytes of data in Flash memory. In this case EEPROM emulation is needed because although Flash memory can be written byte-by-byte, it must be erased sector-by-sector, it is not erasable byte-by-byte (unlike EEPROM which is written AND erased byte-by-byte). So changing one or two bytes in Flash memory typically requires erasing an entire sector each time only one byte is changed within that sector.

However, two of the 8 Kbyte sectors of Secondary Flash memory may be used to emulate EEPROM by using a linked-list software technique to create a small data set that is

28.5.9 Erase time-out flag (DQ3)

The Erase Time-out Flag Bit (DQ3) reflects the time-out period allowed between two consecutive sector erase instruction sequence bytes. If multiple sector erase commands are desired, the additional sector erase commands (30h) must be sent by the 8032 within 80us after the previous sector erase command. DQ3 is 0 before this time period has expired, indicating it is OK to issue additional sector erase commands. DQ3 will go to logic ’1’ if the time has been longer than 80µs since the previous sector erase command (time has expired), indication that is not OK to send another sector erase command. In this case, the 8032 must start a new sector erase instruction sequence (unlock and command) beginning again after the current sector erase operation has completed.

28.5.10 Programming Flash memory

When a byte of Flash memory is programmed, individual bits are programmed to logic '0.' cannot program a bit in Flash memory to a logic ’1’ once it has been programmed to a logic '0.' A bit must be erased to logic ’1’, and programmed to logic '0.' That means Flash memory must be erased prior to being programmed. A byte of Flash memory is erased to all 1s (FFh). The 8032 may erase the entire Flash memory array all at once, or erase individual sector-by-sector, but not erase byte-by-byte. However, even though the Flash memories cannot be *erased* byte-by-byte, the 8032 may *program* Flash memory byte-by-byte. This means the 8032 does not need to program group of bytes (64, 128, etc.) at one time, like some Flash memories.

Each Flash memory requires the 8032 to send an instruction sequence to program a byte or to erase sectors (see *Table 163 on page 209*).

If the byte to be programmed is in a protected Flash memory sector, the instruction sequence is ignored.

Important note: It is mandatory that a chip-select signal is active for the Flash sector where a programming instruction sequence is targeted. The user must make sure that the correct chip-select equation, FSx or CSBOOTx specified in PSDsoft Express matches the address range that the 8032 firmware is accessing, otherwise the instruction sequence will not be recognized by the Flash array. If memory paging is used, be sure that the 8032 firmware sets the page register to the correct page number before issuing an instruction sequence to the Flash memory segment on a particular memory page, otherwise the correct sector select signal will not become active.

Once the 8032 issues a Flash memory program or erase instruction sequence, it must check the status bits for completion. The embedded algorithms that are invoked inside a Flash memory array provide several ways to give status to the 8032. Status may be checked using any of three methods: Data Polling, Data Toggle, or Ready/$\overline{\text{Busy}}$ (pin PC3).

Table 164. Flash memory status bit definition[(1)] [(2)]

Functional block	FSx, or CSBOOTx	DQ7	DQ6	DQ5	DQ4	DQ3	DQ2	DQ1	DQ0
Flash memory	Active (the desired segment is selected)	Data polling	Toggle flag	Error flag	X	Erase timeout	X	X	X

1. X = Not guaranteed value, can be read either '1' or '0.'
2. DQ7-DQ0 represent the 8032 data bus bits, D7-D0.

Table 168. OMC port and data bit assignments (continued)

OMC	Port assignment(1),(2)	Native product terms from AND-OR array	Maximum borrowed product terms	Data bit on 8032 data bus for loading or reading OMC
MCELLBC3	Port B3 or C3	4	5	D3
MCELLBC4	Port B4 or C4	4	6	D4
MCELLBC5	Port B5	4	6	D5
MCELLBC6	Port B6	4	6	D6
MCELLBC7	Port B7 orC7	4	6	D7

1. MCELLAB0-MCELLAB7 can be output to Port A pins only on 80-pin devices. Port A is not available on 52-pin devices.
2. Port pins PC0, PC1, PC5, and PC6 are dedicated JTAG pins and are not available as outputs for MCELLBC 0, 1, 5, or 6.

28.5.32 Loading and reading OMCs

Each of the two OMC groups (eight OMCs each) occupies a byte in csiop space, named MCELLAB and MCELLBC (see *Table 169* and *Table 170*). When the 8032 writes or reads these two OMC registers in csiop it is accessing each of the OMCs through its 8-bit data bus, with the bit assignment shown in *Table 168 on page 227*. Sometimes it is important to know the bit assignment when the user builds GPLD logic that is accessed by the 8032. For example, the user may create a 4-bit counter that must be loaded and read by the 8032, so the user must know which nibble in the corresponding csiop OMC register the firmware must access. The fitter report generated by PSDsoft Express will indicate how it assigned the OMCs and data bus bits to the logic. The user can optionally force PSDsoft Express to assign logic to specific OMCs and data bus bits if desired by using the 'PROPERTY' statement in PSDsoft Express. Please see the PSDsoft Express User's Manual for more information on OMC assignments.

Loading the OMC flip-flops with data from the 8032 takes priority over the PLD logic functions. As such, the preset, clear, and clock inputs to the flip-flop can be asynchronously overridden when the 8032 writes to the csiop registers to load the individual OMCs.

Table 169. Output macrocell MCELLAB (address = csiop + offset 20h)(1)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MCELLAB 7	MCELLAB 6	MCELLAB 5	MCELLAB 4	MCELLAB 3	MCELLAB 2	MCELLAB 1	MCELLAB 0

1. All bits clear to logic '0' at power-on reset, but do not clear after warm reset conditions (non-power-on reset).

Table 170. Output macrocell MCELLBC (address = csiop + offset 21h)(1)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MCELLBC 7	MCELLBC 6	MCELLBC 5	MCELLBC 4	MCELLBC 3	MCELLBC 2	MCELLBC 1	MCELLBC 0

1. All bits clear to logic '0' at power-on reset, but do not clear after warm reset conditions (non-power-on reset).

Table 203. Function status during power-up reset, warm reset, power-down mode (continued)

Port configuration or register	Power-up reset	Warm reset	APD Power-down mode
VM register[1]	Initialized with value that was specified in PSDsoft	Initialized with value that was specified in PSDsoft	Unchanged
All other csiop registers	Cleared to 00h	Cleared to 00h	Unchanged

1. VM register Bit 7 (PIO_EN) and Bit 0 (SRAM in 8032 program space) are cleared to zero at power-up and warm reset conditions.

28.6.1 JTAG ISP and JTAG debug

An IEEE 1149.1 serial JTAG interface is used on UPSD34xx devices for ISP (in-system programming) of the PSD module, and also for debugging firmware on the MCU module. IEEE 1149.1 Boundary Scan operations are not supported in the UPSD34xx.

The main advantage of JTAG ISP is that a blank UPSD34xx device may be soldered to a circuit board and programmed with no involvement of the 8032, meaning that no 8032 firmware needs to be present for ISP. This is good for manufacturing, for field updates, and for easy code development in the lab. JTAG-based programmers and debuggers for UPSD34xx are available from STMicroelectronics and 3rd party vendors.

ISP is different than IAP (in-application programming). IAP involves the 8032 to program Flash memory over any interface supported by the 8032 (e.g., UART, SPI, I2C), which is good for remote updates over a communication channel. UPSD34xx devices support both ISP and IAP. The entire PSD module (Flash memory and PLD) may be programmed with JTAG ISP, but only the Flash memories may be programmed using IAP.

28.6.2 JTAG chaining inside the package

JTAG protocol allows serial “chaining” of more than one device in a JTAG chain. The UPSD34xx is assembled with a stacked die process combining the PSD module (one die) and the MCU module (the other die). These two die are chained together within the UPSD34xx package. The standard JTAG interface has four basic signals:

- TDI - Serial data into device
- TDO - Serial data out of device
- TCK - Common clock
- TMS - Mode Selection

Every device that supports IEEE 1149.1 JTAG communication contains a test access port (TAP) controller, which is a small state machine to manage JTAG protocol and serial streams of commands and data. Both the PSD module and the MCU module each contain a TAP controller.

Figure 90 illustrates how these die are chained within a package. JTAG programming/test equipment will connect externally to the four IEEE 1149.1 JTAG pins on Port C. The TDI pin on the UPSD34xx package goes directly to the PSD module first, then exits the PSD module through TDO. TDO of the PSD module is connected to TDI of the MCU module. The serial path is completed when TDO of the MCU module exits the UPSD34xx package through the TDO pin on Port C. The JTAG signals TCK and TMS are common to both modules as specified in IEEE 1149.1. When JTAG devices are chained, typically one devices is in BYPASS mode while another device is executing a JTAG operation. For the UPSD34xx, the

Table 204. PSD module example, typ. power calculation at V_{CC} = 5.0 V (turbo mode off) (continued)

Conditions			
	I_{CC}(MCUactive)	= 20 mA	
	I_{PD}(pwrdown)	= 250 µA	
	I_{CC}(PSDactive)	= I_{CC}(ac) + I_{CC}(dc)	
		= %flash x 2.5 mA/MHz x Freq ALE	
			+ %SRAM x 1.5 mA/MHz x Freq ALE
			+ % PLD x (from graph using Freq PLD)
		= 0.8 x 2.5 mA/MHz x 2 MHz + 0.15 x 1.5 mA/MHz x 2 MHz + 24 mA	
		= (4 + 0.45 + 24) mA	
		= 28.45 mA	
I_{CC} total	= 20 mA x 40% + 28.45 mA x 40% + 250 µA x 60%		
		= 8 mA + 11.38 mA + 150 µA	
		= 19.53 mA	
This is the operating power with no Flash memory Erase or Program cycles in progress. Calculation is based on all I/O pins being disconnected and I_{OUT} = 0 mA.			

Figure 101. Synchronous Clock mode timing – PLD

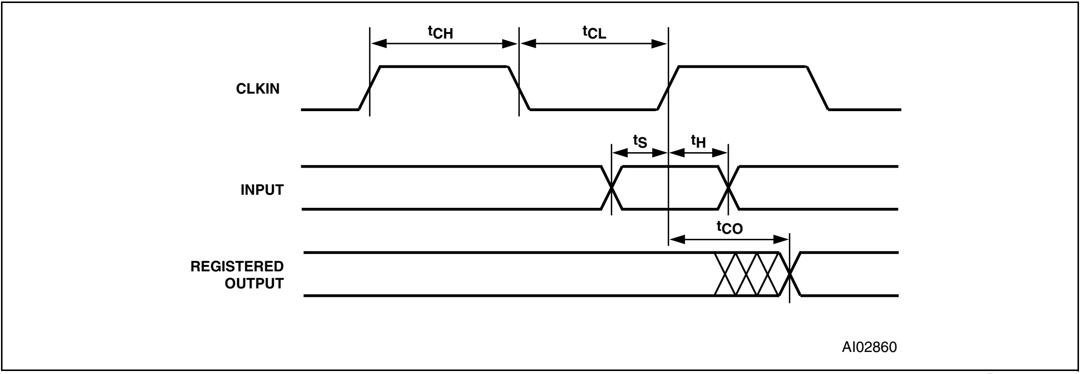


Table 222. CPLD macrocell synchronous clock mode timing (5 V PSD module)

Symbol	Parameter	Conditions	Min	Max	PT Aloc	Turbo Off	Slew rate[(1)]	Unit
f_{MAX}	Maximum frequency external feedback	$1/(t_S+t_{CO})$		40.0				MHz
	Maximum frequency internal feedback (f_{CNT})	$1/(t_S+t_{CO}-10)$		66.6				MHz
	Maximum frequency pipelined data	$1/(t_{CH}+t_{CL})$		83.3				MHz
t_S	Input setup time		12		+ 2	+ 10		ns
t_H	Input hold time		0					ns
t_{CH}	Clock high time	Clock input	6					ns
t_{CL}	Clock low time	Clock input	6					ns
t_{CO}	Clock to output delay	Clock input		13			– 2	ns
t_{ARD}	CPLD array delay	Any macrocell		11	+ 2			ns
t_{MIN}	Minimum clock period[(2)]	$t_{CH}+t_{CL}$	12					ns

1. Fast slew rate output available on PA3-PA0, PB3-PB0, and PD2-PD1. Decrement times by given amount.
2. CLKIN (PD1) $t_{CLCL} = t_{CH} + t_{CL}$.

Table 240. Order codes

Part number	Max MHz	1st Flash	2nd Flash	SRAM	GPIO	8032 bus	V_{CC}	V_{DD}	Package
		(bytes)							
UPSD3422E-40T6	40	64K	32K	4K	35	No	3.3 V	5.0 V	LQFP52
UPSD3422EV-40T6	40	64K	32K	4K	35	No	3.3 V	3.3 V	LQFP52
UPSD3422E-40U6	40	64K	32K	4K	46	Yes	3.3 V	5.0 V	LQFP80
UPSD3422EV-40U6	40	64K	32K	4K	46	Yes	3.3 V	3.3 V	LQFP80
UPSD3433E-40T6	40	128K	32K	8K	35	No	3.3 V	5.0 V	LQFP52
UPSD3433EV-40T6	40	128K	32K	8K	35	No	3.3 V	3.3 V	LQFP52
UPSD3433E-40U6	40	128K	32K	8K	46	Yes	3.3 V	5.0 V	LQFP80
UPSD3433EV-40U6	40	128K	32K	8K	46	Yes	3.3 V	3.3 V	LQFP80
UPSD3434E-40T6	40	256K	32K	8K	35	No	3.3 V	5.0 V	LQFP52
UPSD3434EV-40T6	40	256K	32K	8K	35	No	3.3 V	3.3 V	LQFP52
UPSD3434E-40U6	40	256K	32K	8K	46	Yes	3.3 V	5.0 V	LQFP80
UPSD3434EV-40U6	40	256K	32K	8K	46	Yes	3.3 V	3.3 V	LQFP80
UPSD3454E-40T6	40	256K	32K	32K	35	No	3.3 V	5.0 V	LQFP52
UPSD3454EV-40T6	40	256K	32K	32K	35	No	3.3 V	3.3 V	LQFP52
UPSD3454E-40U6	40	256K	32K	32K	46	Yes	3.3 V	5.0 V	LQFP80
UPSD3454EV-40U6	40	256K	32K	32K	46	Yes	3.3 V	3.3 V	LQFP80
UPSD3422EB40T6	40	64K	32K	4K	35	No	3.3 V	5.0 V	LQFP52
UPSD3422EVB40T6	40	64K	32K	4K	35	No	3.3 V	3.3 V	LQFP52
UPSD3422EB40U6	40	64K	32K	4K	46	Yes	3.3 V	5.0 V	LQFP80
UPSD3422EVB40U6	40	64K	32K	4K	46	Yes	3.3 V	3.3 V	LQFP80
UPSD3433EB40T6	40	128K	32K	8K	35	No	3.3 V	5.0 V	LQFP52
UPSD3433EVB40T6	40	128K	32K	8K	35	No	3.3 V	3.3 V	LQFP52
UPSD3433EB40U6	40	128K	32K	8K	46	Yes	3.3 V	5.0 V	LQFP80
UPSD3433EVB40U6	40	128K	32K	8K	46	Yes	3.3 V	3.3 V	LQFP80
UPSD3434EB40T6	40	256K	32K	8K	35	No	3.3 V	5.0 V	LQFP52
UPSD3434EVB40T6	40	256K	32K	8K	35	No	3.3 V	3.3 V	LQFP52
UPSD3434EB40U6	40	256K	32K	8K	46	Yes	3.3 V	5.0 V	LQFP80
UPSD3434EVB40U6	40	256K	32K	8K	46	Yes	3.3 V	3.3 V	LQFP80
UPSD3454EB40T6	40	256K	32K	32K	35	No	3.3 V	5.0 V	LQFP52
UPSD3454EVB40T6	40	256K	32K	32K	35	No	3.3 V	3.3 V	LQFP52
UPSD3454EB40U6	40	256K	32K	32K	46	Yes	3.3 V	5.0 V	LQFP80
UPSD3454EVB40U6	40	256K	32K	32K	46	Yes	3.3 V	3.3 V	LQFP80

Note: Operating temperature is in the Industrial range (–40 °C to 85 °C).