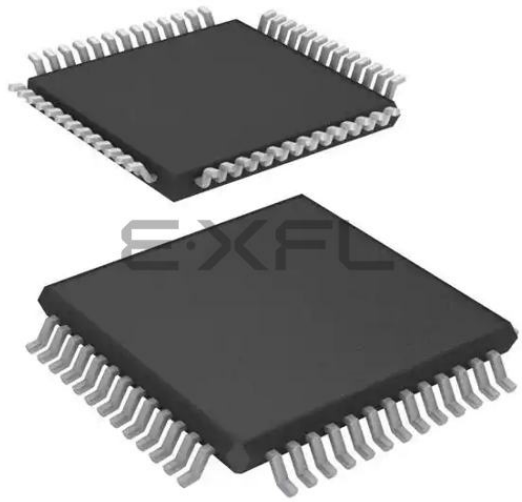


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	8032
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, IrDA, SPI, UART/USART, USB
Peripherals	LVD, POR, PWM, WDT
Number of I/O	35
Program Memory Size	160KB (160K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	52-TQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/upsd3433ev-40t6

List of figures

Table 2. Pin definitions (continued)

Port pin	Signal name	80-pin No.	52-pin No.(1)	In/out	Function		
					Basic	Alternate 1	Alternate 2
ALE		4	N/A	O	Address Latch signal, external bus		
RESET_IN		68	44	I	Active low reset input		
XTAL1		48	31	I	Oscillator input pin for system clock		
XTAL2		49	32	O	Oscillator output pin for system clock		
DEBUG		8	5	I/O	I/O to the MCU debug unit		
PA0		35	N/A	I/O	General I/O port pin		All Port A pins support: – PLD Macrocell outputs, or – PLD inputs, or – Latched address out (A0-A7), or – Peripheral I/O mode
PA1		34	N/A	I/O	General I/O port pin		
PA2		32	N/A	I/O	General I/O port pin		
PA3		28	N/A	I/O	General I/O port pin		
PA4		26	N/A	I/O	General I/O port pin		
PA5		24	N/A	I/O	General I/O port pin		
PA6		22	N/A	I/O	General I/O port pin		
PA7		21	N/A	I/O	General I/O port pin		
PB0		80	52	I/O	General I/O port pin		All Port B pins support: – PLD Macrocell outputs, or – PLD inputs, or – Latched address out (A0-A7 or A8-A15)
PB1		78	51	I/O	General I/O port pin		
PB2		76	50	I/O	General I/O port pin		
PB3		74	49	I/O	General I/O port pin		
PB4		73	48	I/O	General I/O port pin		
PB5		71	46	I/O	General I/O port pin		
PB6		67	43	I/O	General I/O port pin		
PB7		66	42	I/O	General I/O port pin		
JTAGTMS	TMS	20	13	I	JTAG pin (TMS)		
JTAGTCK	TCK	17	12	I	JTAG pin (TCK)		
PC2		16	11	I/O	General I/O port pin		PLD Macrocell output, or PLD input
PC3	TSTAT	15	N/A	I/O	General I/O port pin	Optional JTAG Status (TSTAT)	PLD, Macrocell output, or PLD input
PC4	TERR	9	N/A	I/O	General I/O port pin	Optional JTAG Status ($\overline{\text{TERR}}$)	PLD, Macrocell output, or PLD input
JTAGTDI	TDI	7	4	I	JTAG pin (TDI)		
JTAGTDO	TDO	6	3	O	JTAG pin (TDO)		

4.2.1 Program memory

External program memory is addressed by the 8032 using its 16-bit program counter (PC) and is accessed with the 8032 signal, $\overline{PSEN}$. Program memory can be present at any address in program space between 0000h and FFFFh.

After a power-up or reset, the 8032 begins program execution from location 0000h where the reset vector is stored, causing a jump to an initialization routine in firmware. At address 0003h, just following the reset vector are the interrupt service locations. Each interrupt is assigned a fixed interrupt service location in program memory. An interrupt causes the 8032 to jump to that service location, where it commences execution of the service routine. External Interrupt 0 (EXINT0), for example, is assigned to service location 0003h. If EXINT0 is going to be used, its service routine must begin at location 0003h. Interrupt service locations are spaced at 8-byte intervals: 0003h for EXINT0, 000Bh for Timer 0, 0013h for EXINT1, and so forth. If an interrupt service routine is short enough, it can reside entirely within the 8-byte interval. Longer service routines can use a jump instruction to somewhere else in program memory.

4.2.2 Data memory

External data is referred to as XDATA and is addressed by the 8032 using Indirect Addressing via its 16-bit data pointer register (DPTR) and is accessed by the 8032 signals, $\overline{RD}$ and $\overline{WR}$. XDATA can be present at any address in data space between 0000h and FFFFh.

Note: The UPSD34xx has dual data pointers (source and destination) making XDATA transfers much more efficient.

4.2.3 Memory placement

PSD module architecture allows the placement of its external memories into different combinations of program memory and data memory spaces. This means the main Flash, the secondary Flash, and the SRAM can be viewed by the 8032 MCU in various combinations of program memory or data memory as defined by PSDsoft Express.

As an example of this flexibility, for applications that require a great deal of Flash memory in data space (large lookup tables or extended data recording), the larger main Flash memory can be placed in data space and the smaller secondary Flash memory can be placed in program space. The opposite can be realized for a different application if more Flash memory is needed for code and less Flash memory for data.
By default, the SRAM and csiop memories on the PSD module must always reside in data memory space and they are treated by the 8032 as XDATA.
The main Flash and secondary Flash memories may reside in program space, data space, or both. These memory placement choices specified by PSDsoft Express are programmed into non-volatile sections of the UPSD34xx, and are active at power-up and after reset. It is possible to override these initial settings during runtime for In-Application Programming (IAP).

Standard 8032 MCU architecture cannot write to its own program memory space to prevent accidental corruption of firmware. However, this becomes an obstacle in typical 8032 systems when a remote update to firmware in Flash memory is required using IAP. The PSD module provides a solution for remote updates by allowing 8032 firmware to temporarily “reclassify” Flash memory to reside in data space during a remote update, then returning Flash memory back to program space when finished. See the VM register (*Table 160 on page 203*) in the PSD module section of this document for more details.

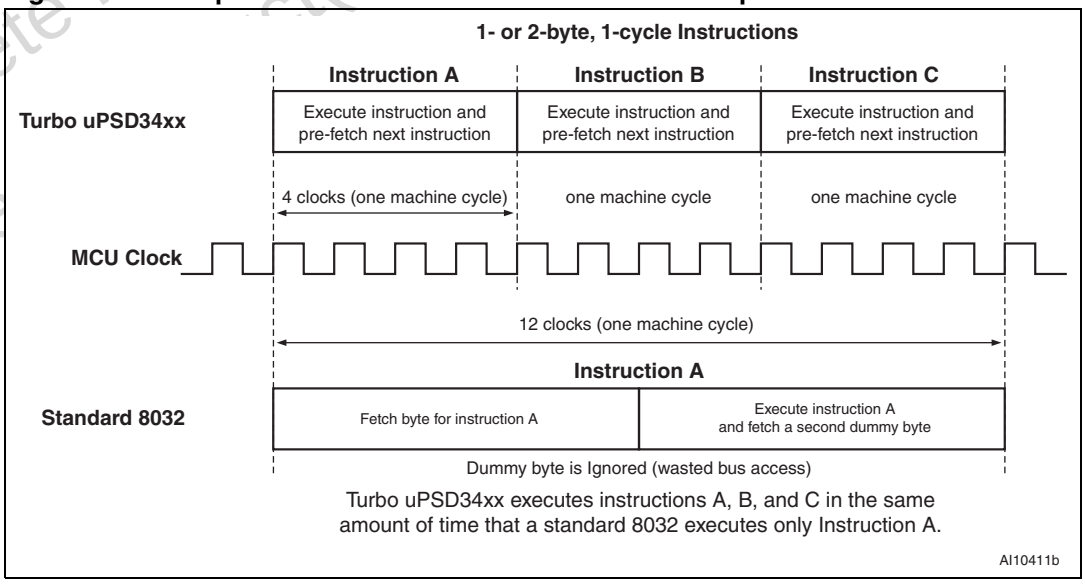
5 8032 MCU core performance enhancements

Before describing performance features of the UPSD34xx, let us first look at standard 8032 architecture. The clock source for the 8032 MCU creates a basic unit of timing called a machine-cycle, which is a period of 12 clocks for standard 8032 MCUs. The instruction set for traditional 8032 MCUs consists of 1, 2, and 3 byte instructions that execute in different combinations of 1, 2, or 4 machine-cycles. For example, there are one-byte instructions that execute in one machine-cycle (12 clocks), one-byte instructions that execute in four machine-cycles (48 clocks), two-byte, two-cycle instructions (24 clocks), and so on. In addition, standard 8032 architecture will fetch two bytes from program memory on almost every machine-cycle, regardless if it needs them or not (dummy fetch). This means for one-byte, one-cycle instructions, the second byte is ignored. These one-byte, one-cycle instructions account for half of the 8032's instructions (126 out of 255 opcodes). There are inefficiencies due to wasted bus cycles and idle bus times that can be eliminated.

The UPSD34xx 8032 MCU core offers increased performance in a number of ways, while keeping the exact same instruction set as the standard 8032 (all opcodes, the number of bytes per instruction, and the native number a machine-cycles per instruction are identical to the original 8032). The first way performance is boosted is by reducing the machine-cycle period to just 4 MCU clocks as compared to 12 MCU clocks in a standard 8032. This shortened machine-cycle improves the instruction rate for one- or two-byte, one-cycle instructions by a factor of three (*Figure 6 on page 33*) compared to standard 8051 architectures, and significantly improves performance of multiple-cycle instruction types.

The example in *Figure 6 on page 33* shows a continuous execution stream of one- or two-byte, one-cycle instructions. The 5 V UPSD34xx will yield 10 MIPS peak performance in this case while operating at 40 MHz clock rate. In a typical application however, the effective performance will be lower since programs do not use only one-cycle instructions, but special techniques are implemented in the UPSD34xx to keep the effective MIPS rate as close as possible to the peak MIPS rate at all times. This is accomplished with an instruction pre-fetch queue (PFQ), a branch cache (BC), and a 16-bit program memory bus as shown in *Figure 7 on page 34*.

Figure 6. Comparison of UPSD34xx with standard 8032 performance



11 Dual data pointers

XDATA is accessed by the External Direct addressing mode, which uses a 16-bit address stored in the DPTR register. Traditional 8032 architecture has only one DPTR register. This is a burden when transferring data between two XDATA locations because it requires heavy use of the working registers to manipulate the source and destination pointers.

However, the UPSD34xx has two data pointers, one for storing a source address and the other for storing a destination address. These pointers can be configured to automatically increment or decrement after each data transfer, further reducing the burden on the 8032 and making this kind of data movement very efficient.

11.1 Data pointer control register, DPTC (85h)

By default, the DPTR register of the UPSD34xx will behave no different than in a standard 8032 MCU. The DPSEL0 Bit of SFR register DPTC shown in *Table 13*, selects which one of the two “background” data pointer registers (DPTR0 or DPTR1) will function as the traditional DPTR register at any given time. After reset, the DPSEL0 Bit is cleared, enabling DPTR0 to function as the DPTR, and firmware may access DPTR0 by reading or writing the traditional DPTR register at SFR addresses 82h and 83h. When the DPSEL0 bit is set, then the DPTR1 register functions as DPTR, and firmware may now access DPTR1 through SFR registers at 82h and 83h. The pointer which is not selected by the DPSEL0 bit remains in the background and is not accessible by the 8032. If the DPSEL0 bit is never set, then the UPSD34xx will behave like a traditional 8032 having only one DPTR register.

To further speed XDATA to XDATA transfers, the SFR bit, AT, may be set to automatically toggle the two data pointers, DPTR0 and DPTR1, each time the standard DPTR register is accessed by a MOVX instruction. This eliminates the need for firmware to manually manipulate the DPSEL0 bit between each data transfer.

Detailed description for the SFR register DPTC is shown in *Table 13*.

Table 13. DPTC: data pointer control register (SFR 85h, reset value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	AT	–	–	–	–	–	DPSEL0

Table 14. DPTC register bit definition

Bit	Symbol	R/W	Definition
7	–	–	Reserved
6	AT	R,W	0 = Manually select data pointer 1 = Auto toggle between DPTR0 and DPTR1
5-1	–	–	Reserved
0	DPSE0	R,W	0 = DPTR0 selected for use as DPTR 1 = DPTR1 selected for use as DPTR

20 Standard 8032 timer/counters

There are three 8032-style 16-bit Timer/Counter registers (Timer 0, Timer 1, Timer 2) that can be configured to operate as timers or event counters.

There are two additional 16-bit Timer/Counters in the Programmable Counter Array (PCA), see *Section 27.1: PCA block on page 181* for details.

20.1 Standard timer SFRs

Timer 0 and Timer 1 have very similar functions, and they share two SFRs for control:

- TCON (*Table 56 on page 97*)
- TMOD (*Table 58 on page 99*).

Timer 0 has two SFRs that form the 16-bit counter, or that can hold reload values, or that can scale the clock depending on the timer/counter mode:

- TH0 is the high byte, address 8Ch
- TL0 is the low byte, address 8Ah

Timer 1 has two similar SFRs:

- TH1 is the high byte, address 8Dh
- TL1 is the low byte, address 8Bh

Timer 2 has one control SFR:

- T2CON (*Table 60 on page 101*)

Timer 2 has two SFRs that form the 16-bit counter, and perform other functions:

- TH2 is the high byte, address CDh
- TL2 is the low byte, address CCh

Timer 2 has two SFRs for capture and reload:

- RCAP2H is the high byte, address CBh
- RCAP2L is the low byte, address CAh

20.2 Clock sources

When enabled in the "**Timer**" function, the registers THx and TLx are incremented every 1/12 of the oscillator frequency (f_{OSC}). This timer clock source is not effected by MCU clock dividers in the CCON0, stalls from PFQ/BC, or bus transfer cycles. Timers are always clocked at 1/12 of f_{OSC}.

When enabled in the "**Counter**" function, the registers THx and TLx are incremented in response to a 1-to-0 transition sampled at their corresponding external input pin: pin C0 for Timer 0; pin C1 for Timer 1; or pin T2 for Timer 2. In this function, the external clock input pin is sampled by the counter at a rate of 1/12 of f_{OSC}. When a logic '1' is determined in one sample, and a logic '0' in the next sample period, the count is incremented at the very next sample period (period1: sample=1, period2: sample=0, period3: increment count while continuing to sample). This means the maximum count rate is 1/24 of the f_{OSC}. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be active for at least one full sample

Figure 36. UART mode 3, block diagram

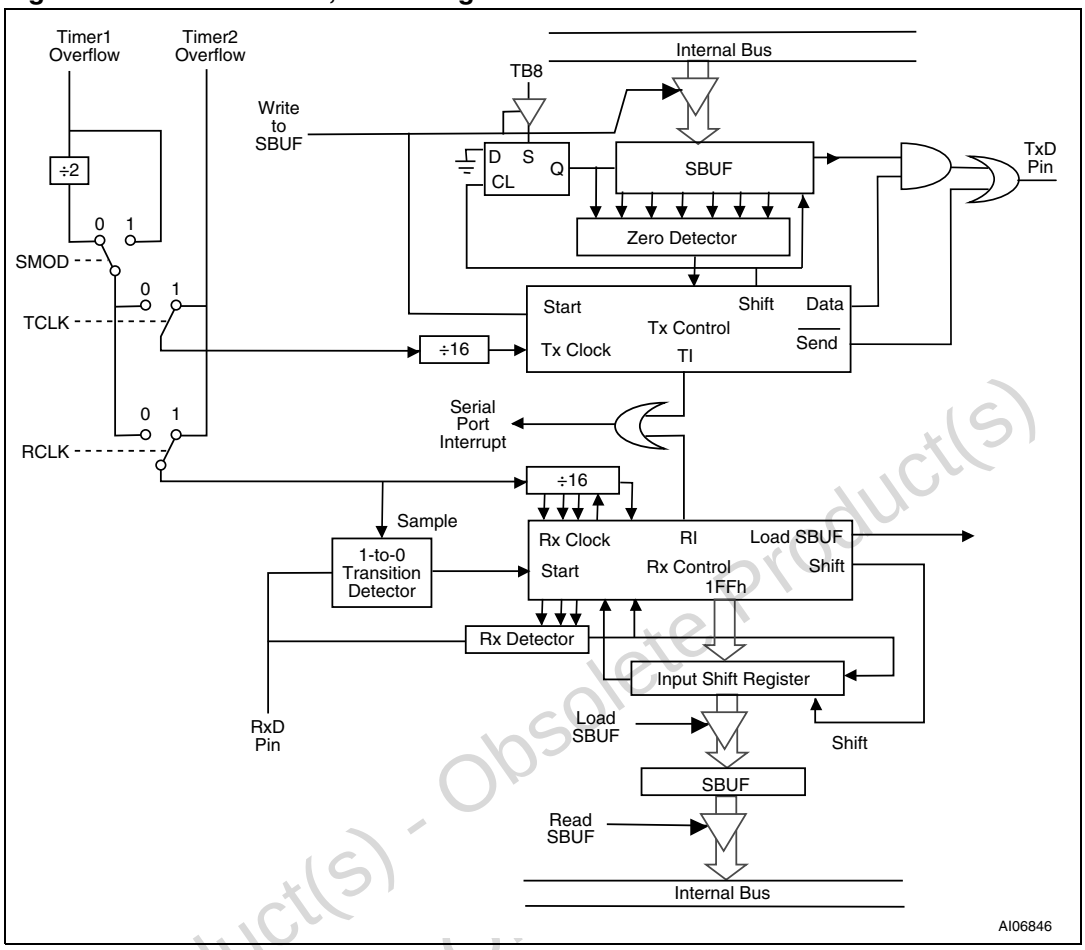
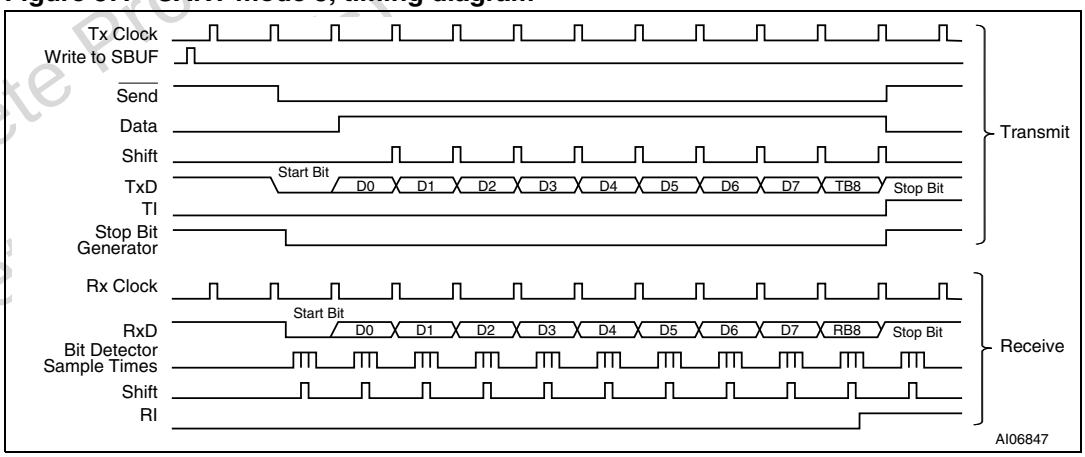


Figure 37. UART mode 3, timing diagram



The UART1 serial channel can operate in one of four different modes as shown in *Table 64 on page 108* in *Section 21: Serial UART interfaces on page 107*. However, when UART1 is used for IrDA communication, UART1 must operate in Mode 1 only, to be compatible with IrDA protocol up to 115.2k bps. The IrDA interface will support baud rates generated from Timer 1 or Timer 2, just like standard UART serial communication, but with one restriction. The transmit baud rate and receive baud rate must be the same (cannot be different rates as is allowed by standard UART communications).

The IrDA Interface is disabled after a reset and is enabled by setting the IRDAEN Bit in the SFR named IRDACON (*Table 70 on page 120*). When IrDA is disabled, the UART1's RxD and TxD signals will bypass the internal IrDA logic and instead they are routed directly to the pins RxD1 and TxD1 respectively. When IrDA is enabled, the IrDA pulse shaping logic is active and resides between UART1 and the pins RxD1 and TxD1 as shown in *Figure 38 on page 119*.

22.1 Baud rate selection

The IrDA standard only supports 2.4, 9.6, 19.2, and 115.2kbps. *Table 73 on page 121* informs the IrDA Interface of the baud rate of UART#1 so that it can perform pulse modulation properly. It may not be necessary to implement the BR[3:0] bits in the IRDACON register if the IrDA Interface obtains the proper timing from UART#1.

Table 70. IRDACON register bit definition (SFR CEh, reset value 0Fh)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	IRDAEN	PULSE	CDIV4	CDIV3	CDIV2	CDIV1	CDIV0

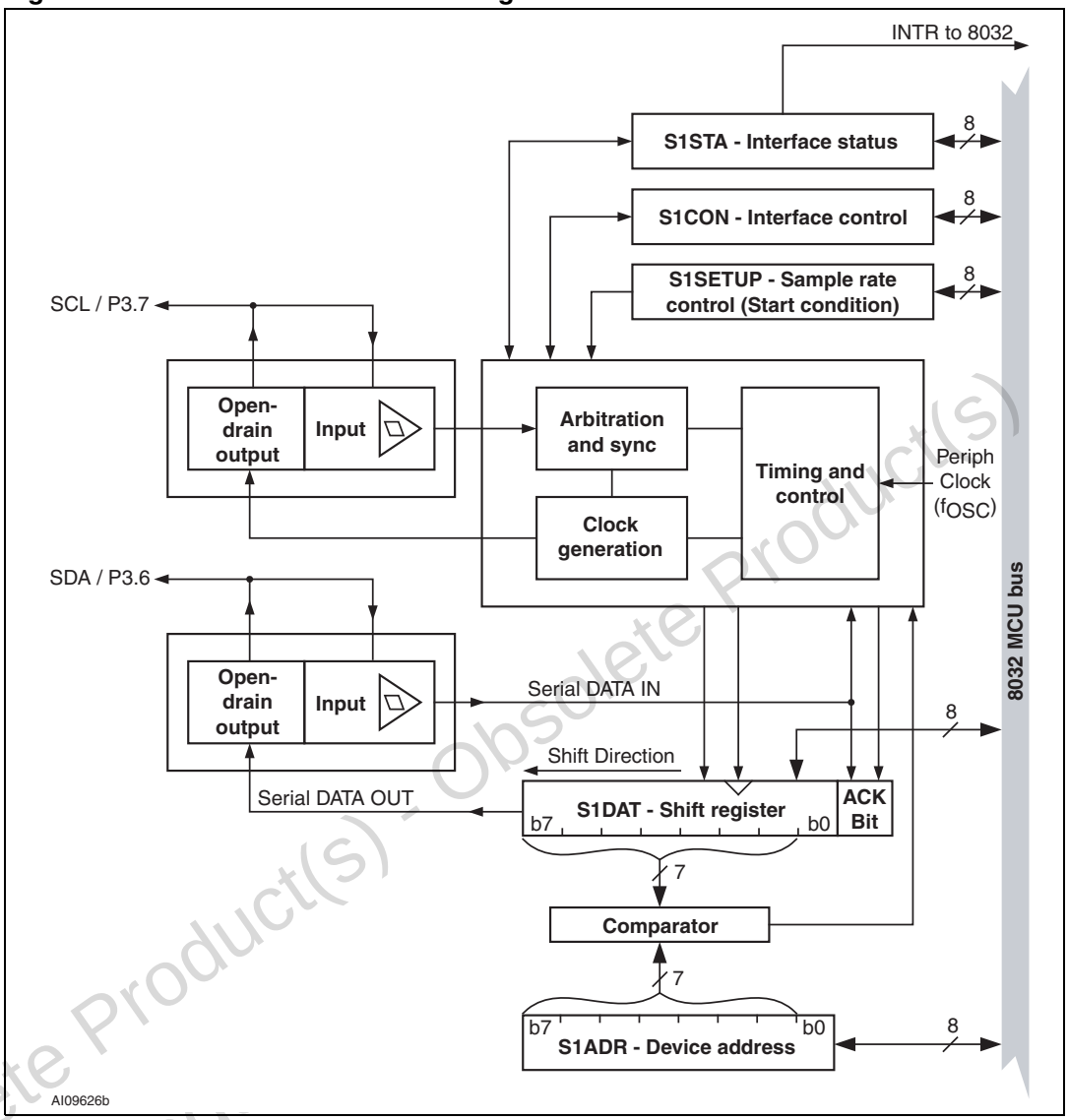
Table 71. IRDACON register bit definition

Bit	Symbol	R/W	Definition
7	–	–	Reserved
6	IRDAEN	RW	IrDA Enable 0 = IrDA Interface is disabled 1 = IrDA is enabled, UART1 outputs are disconnected from Port 1 (or Port 4)
5	PULSE	RW	IrDA Pulse Modulation Select 0 = 1.627µs 1 = 3/16 bit time pulses
4-0	CDIV[4:0]	RW	Specify Clock Divider (see *Table 74 on page 122*)

Table 72. Baud rate selection register (SFR xxh, reset value xxh)

Bit	Symbol	R/W	Definition
7:4	BR[3:0]	R,W	Specify Baud Rate (see *Table 73*)
3:2	PULSE	R,W	IrDA Pulse Modulation Select 0 = 3/16 bit time pulses (not recommended) 1 = 1.627µs
1:0	IRDAEN	R,W	0 = IrDA Interface is disabled 1 = IrDA is enabled, UART#1 outputs are disconnected from Port 1 (or Port 4)

Figure 42. I²C interface SIOE block diagram



23.13.1 Interrupt service routine (ISR)

A typical I²C interrupt service routine would handle a interrupt for any of the four combinations of Master/Slave and Transmitter/Receiver. In the example routines above, the firmware sets global variables, I2C_master and I2C_xmitter, before enabling interrupts. These flags tell the ISR which one of the four cases to process. Following is pseudo-code for high-level steps in the I²C ISR:

```
Begin I2C ISR <I2C interrupt just occurred>:

Clear I2C interrupt flag:
    -   S1STA.INTR = 0

Read status of SIOE, put in to variable, status
    -   status = S1STA

Read global variables that determine the mode
    -   mode <= (I2C_master, I2C_slave)

If mode is Master-Transmitter

        Bus Arbitration lost? (status.BLOST=1?)
        If Yes, Arbitration was lost:
    -   S1DAT = dummy, write to release bus
    -   Exit ISR, SIOE will switch to Slave Recv mode
        If No, Arbitration was not lost, continue:

ACK recvd from Slave? (status.ACK_RESP=0?)
        If No, an ACK was not received:
    -   S1CON.STO = 1, set Stop bus condition
    -   <Stop occurs after ISR exit>
    -   S1DAT = dummy, write to release bus
    -   Exit ISR
        If Yes, ACK was received, then continue:
    -   S1DAT = xmit_buf[buffer_index], transmit byte

Was that the last byte of data to transmit?
        If No, it was not the last byte, then:
    -   Exit ISR, transmit next byte on next interrupt
        If Yes, it was the last byte, then:
    -   S1CON.STO = 1, set Stop bus condition
    <Stop occurs after ISR exit>
    -   S1DAT = dummy, write to release bus
    -   Exit ISR
```

25.3.6 Accessing the setup command buffer

Setup Packets are sent from the host to a device's Endpoint0 and consist of 8 bytes of command data. When the SIE receives a Setup packet from the host, it stores the 8 bytes of data in the Command Buffer. The command buffer is accessed via the indexed USB Setup Command Value register (USCV). The USB Setup Command Index register (USCI) is used to select the byte from the command buffer that is read when accessing the USCV register.

25.4 USB registers

The USB module is controlled via registers mapped into the SFR space. The USB SFRs consist of the following:

- UADDR: USB device address
- UPAIR: USB FIFO pairing control
- UIE0~3: USB interrupt enable
- UIF0~3: USB interrupt flags
- UCTL: USB control
- USTA: USB status
- USEL: USB endpoint and direction select
- UCON: USB selected FIFO control register
- USIZE: USB selected FIFO size register
- UBASE: USB base address register
- USCI: USB setup command index
- USCV: USB setup command value

The memory map for the USB SFRs, the individual bit names, and the reset values are shown in *Table 99*.

Table 99. UPSD34xx USB SFR register map(1)

SFR addr (hex)	SFR name	Bit name and <bit address>								Reset value (hex)	Comment
		7	6	5	4	3	2	1	0		
E2	UADDR	–	USBADDR[6:0]							00	USB Address
E3	UPAIR	–	–	–	–	PR3OUT	PR1OUT	PR3IN	PR1IN	00	USB Pairing Control
E4	UIE0	–	–	–	–	RSTIE	SUSPNDIE	EOPIE	RESUMIE	00	USB Global Interrupt Enable
E5	UIE1	–	–	–	IN4IE	IN3IE	IN2IE	IN1IE	IN0IE	00	USB IN FIFO Interrupt Enable
E6	UIE2	–	–	–	OUT4IE	OUT3IE	OUT2IE	OUT1IE	OUT0IE	00	USB OUT FIFO Interrupt Enable
E7	UIE3	–	–	–	NAK4IE	NAK3IE	NAK2IE	NAK1IE	NAK0IE	00	USB IN FIFO NAK Int. Enable

Table 163. Flash memory instruction sequences(1)(2) (continued)

Instr. Seq.	Bus Cycle 1	Bus Cycle 2	Bus Cycle 3	Bus Cycle 4	Bus Cycle 5	Bus Cycle 6	Bus Cycle 7	Link
Resume Sector Erase	Write 30h to address that activates FSx or CSBOOTx where desired to resume erase (*command*)							*Resume sector erase on page 217*
Reset Flash	Write F0h to address that activates FSx or CSBOOTx in desired array. (*command*)							*Reset Flash on page 218*

1. All values are in hexadecimal, X = Don't care.
2. 8032 addresses A12 through A15 are "Don't care" during the instruction sequence decoding. Only address bits A0-A11 are used during decoding of Flash memory instruction sequences. The individual sector select signal (FS0 - FS7 or CSBOOT0-CSBOOT3) which is active during the instruction sequence determines the complete address.
3. Directing this command to any individual sector within a Flash memory array will invoke the bulk erase of all Flash memory sectors within that array.

28.5.3 Reading Flash memory

Under typical conditions, the 8032 may read the Flash memory using READ operations (READ bus cycles) just as it would a ROM or RAM device. Alternately, the 8032 may use READ operations to obtain status information about a Program or Erase operation that is currently in progress. The following sections describe the kinds of READ operations.

28.5.4 Read memory contents

Flash memory is placed in the Read Array mode after Power-up, after a PSD module reset event, or after receiving a Reset Flash memory instruction sequence from the 8032. The 8032 can read Flash memory contents using standard READ bus cycles anytime the Flash array is in Read Array mode. Flash memories will always be in Read Array mode when the array is not actively engaged in a program or erase operation.

28.5.5 Reading the erase/program status bits

The Flash arrays provide several status bits to be used by the 8032 to confirm the completion of an erase or program operation on Flash memory, shown in *Table 164 on page 212*. The status bits can be read as many times as needed until an operation is complete.

The 8032 performs a READ operation to obtain these status bits while an erase or program operation is being executed by the state machine inside each Flash memory array.

28.5.6 Data polling flag (DQ7)

While programming either Flash memory, the 8032 may read the Data Polling Flag Bit (DQ7), which outputs the complement of the D7 Bit of the byte being programmed into Flash memory. Once the program operation is complete, DQ7 is equal to D7 of the byte just programmed into Flash memory, indicating the program cycle has completed successfully.

Native product terms come from the AND-OR Array. Each OMC may borrow product terms only from certain other OMCs, if they are not in use. Product term allocation does not add any propagation delay to the logic. The fitter report generated by PSDsoft Express will show any PT allocation that has occurred.

If an equation requires more product terms than are available to it through PT allocation, then “external” product terms are required, which consumes other OMCs. This is called product term expansion and also happens automatically in PSDsoft Express as needed. PT expansion causes additional propagation delay because an additional OMC is consumed by the expansion process and its output is rerouted (or fed back) into the AND-OR array. The user can examine the fitter report generated by PSDsoft Express to see resulting PT allocation and PT expansion (expansion will have signal names, such as ‘*.fb_0’ or ‘*.fb_1’). PSDsoft Express will always try to fit the logic design first by using PT allocation, and if that is not sufficient then PSDsoft Express will use PT expansion.

Product term expansion may occur in the DPLD for complex chip select equations for Flash memory sectors and for SRAM, but this is a rare occurence. If PSDsoft Express does use PT expansion in the DPLD, it results in an approximate 15ns additional propagation delay for that chip select signal, which gives 15ns less time for the memory to respond. Be aware of this and consider adding a wait state to the 8032 bus access (using the SFR named, BUSCON), or lower the 8032 clock frequency to avoid problems with memory access time.

Figure 77. OMC allocator

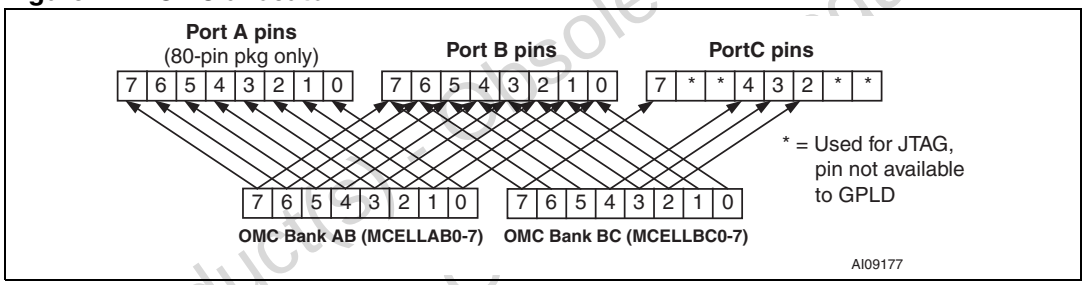


Table 168. OMC port and data bit assignments

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
MCELLAB0	Port A0 or B0	3	6	D0
MCELLAB1	Port A1 or B1	3	6	D1
MCELLAB2	Port A2 or B2	3	6	D2
MCELLAB3	Port A3 or B3	3	6	D3
MCELLAB4	Port A4 or B4	3	6	D4
MCELLAB5	Port A5 or B5	3	6	D5
MCELLAB6	Port A6 or B6	3	6	D6
MCELLAB7	Port A7 or B7	3	6	D7
MCELLBC0	Port B0	4	5	D0
MCELLBC1	Port B1	4	5	D1
MCELLBC2	Port B or C2	4	5	D2

Table 191. Latched address output, port B contro register(1) (2)(address = csiop + offset 03h)l

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PB7 (addr A7 or A15)	PB6 (addr A6 or A14)	PB5 (addr A5 or A13)	PB4 (addr A4 or A12)	PB3 (addr A3 or A11)	PB2 (Addr A2 or A10)	PB1 (addr A1 or A9)	PB0 (addr A0 or A8)

1. Default state for register is 00h after reset or power-up.
2. For each bit, 1 = drive demuxed 8032 address signal on pin, 0 = pin is default mode, MCU I/O.

28.5.41 Peripheral I/O mode

This mode will provide a data bus repeater function for the 8032 to interface with external parallel peripherals. The mode is only available on Port A (80-pin devices only) and the data bus signals, D0 - D7, are de-multiplexed (no address A0-A7). When active, this mode behaves like a bidirectional buffer, with the direction automatically controlled by the 8032 $\overline{RD}$ and $\overline{WR}$ signals for a specified address range. The DPLD signals PSEL0 and PSEL1 determine this address range. *Figure 79 on page 232* shows the action of Peripheral I/O mode on the Output Enable logic of the tri-state output driver for a single port pin. *Figure 83 on page 239* illustrates data repeater the operation. To activate this mode, choose the pin function “Peripheral I/O Mode” in PSDsoft Express on any Port A pin (all eight pins of Port A will automatically change to this mode). Next in PSDsoft, specify an address range for the PSELx signals in the “Chip-Select” section of the “Design Assistant”. The user can specify an address range for either PSEL0 or PSEL1. Always qualify the PSELx equation with “$\overline{PSEN}$ is logic ‘1’” to ensure Peripheral I/O mode is only active during 8032 data cycles, not code cycles. Only one equation is needed since PSELx signals are OR’ed together (*Figure 83*). Then in the 8032 initialization firmware, a logic ’1’ is written to the csiop VM register, Bit 7 (PIO_EN) as shown in *Table 154 on page 189*. After this, Port A will automatically perform this repeater function whenever the 8032 presents an address (and memory page number, if paging is used) that is within the range specified by PSELx. Once Port A is designated as Peripheral I/O mode in PSDsoft Express, it cannot be used for other functions.

Note: The user can alternatively connect an external parallel peripheral to the standard 8032 AD0-AD7 pins on an 80-pin uPSD device (not Port A), but these pins have multiplexed address and data signals, with a weaker fanout drive capability.

Figure 83. Peripheral I/O mode

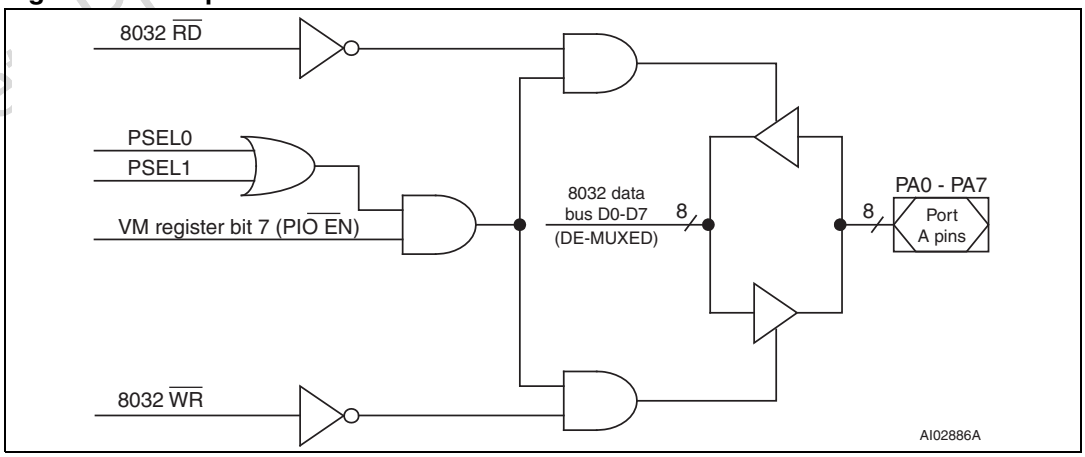


Table 193. Port B pin drive select register (address = csiop + offset 09h)(1) (2)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PB7 open drain	PB6 open drain	PB5 open drain	PB4 open drain	PB3 slew rate	PB2 slew rate	PB1 slew rate	PB0 slew rate

1. For each bit, 1 = pin drive type is selected, 0 = pin drive type is default mode, CMOS push/pull.
2. Default state for register is 00h after reset or power-up.

Table 194. Port C pin drive select register (address = csiop + offset 16h)(1) (2)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC7 open drain	N/A (JTAG)	N/A (JTAG)	PC4 open drain	PC3 open drain	PC2 open drain	N/A (JTAG)	N/A (JTAG)

1. For each bit, 1 = pin drive type is selected, 0 = pin drive type is default mode, CMOS push/pull
2. Default state for register is 00h after reset or power-up

Table 195. Port D pin drive select register (address = csiop + offset 17h)(1) (2)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
N/A	N/A	N/A	N/A	N/A	PD2(3) slew rate	PD1 slew rate	N/A

1. For each bit, 1 = pin drive type is selected, 0 = pin drive type is default mode, CMOS push/pull
2. Default state for register is 00h after reset or power-up
3. Pin is not available on 52-pin UPSD34xx devices

Table 196. Port A enable out register(1) (2)(address = csiop + offset 0Ch)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PA7 OE	PA6 OE	PA5 OE	PA4 OE	PA3 OE	PA2 OE	PA1 OE	PA0 OE

1. Port A not available on 52-pin UPSD34xx devices
2. For each bit, 1 = pin drive is enabled as an output, 0 = pin drive is off (high-impedance, pin used as input)

Table 197. Port B enable out register (address = csiop + offset 0Dh)(1)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PB7 OE	PB6 OE	PB5 OE	PB4 OE	PB3 OE	PB2 OE	PB1 OE	PB0 OE

1. For each bit, 1 = pin drive is enabled as an output, 0 = pin drive is off (high-impedance, pin used as input)

Table 198. Port C enable out register (address = csiop + offset 1Ah)(1)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC7 OE	N/A (JTAG)	N/A (JTAG)	PC4 OE	PC3 OE	PC2 OE	N/A (JTAG)	N/A (JTAG)

1. For each bit, 1 = pin drive is enabled as an output, 0 = pin drive is off (high-impedance, pin used as input)

Table 199. Port D enable out register (address = csiop + offset 1Bh)(1)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
N/A	N/A	N/A	N/A	N/A	PD2 OE(2)	PD1 OE	N/A

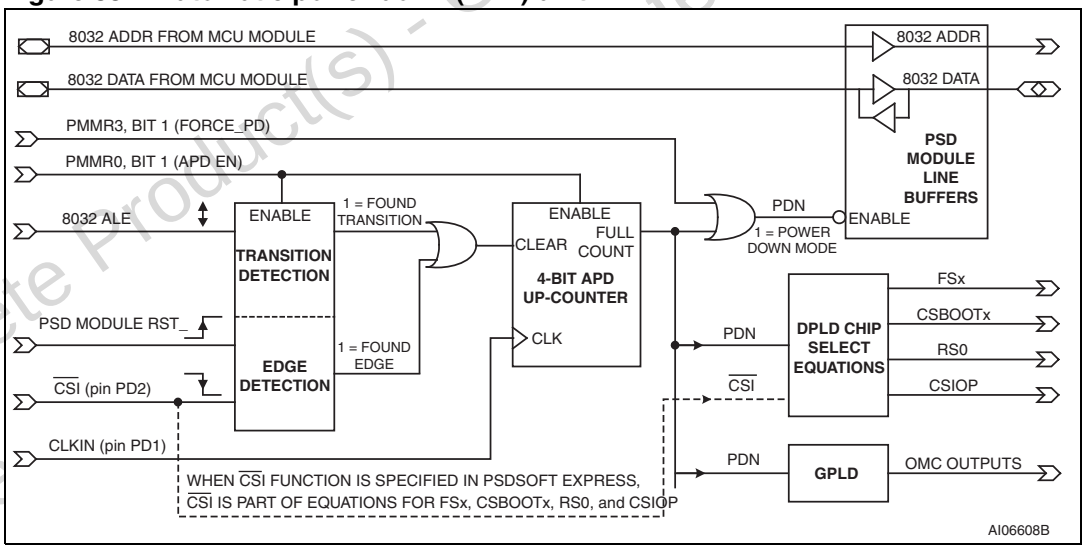
1. For each bit, 1 = pin drive is enabled as an output, 0 = pin drive is off (high-impedance, pin used as input)
2. Pin is not available on 52-pin UPSD34xx devices

assembly code example in *Table* , the PFQ will be loaded with the final instructions to command the MCU module to Power Down mode after the PDS Module goes to Power-Down mode. In this case, even though the code memory goes off-line in the PSD module, the last few MCU instruction are sourced from the PFQ.

Forced power-down example

```
PDOWN:  ANL   A8h, #7Fh      ; disable all interrupts
        ORL   9Dh, #C0h      ; ensure PFQ and BC are enabled
        MOV   DPTR, #xxC7    ; load XDATA pointer to select PMMR3 register (xx = base
                             ; address of csiop registers)
        CLR   A              ; clear A
        JMP   LOOP           ; first loop - fill PFQ/BQ with Power Down instructions
        NOP                  ; second loop - fetch code from PFQ/BC and set Power-
                             ; Down bits for PSD module and then MCU module
LOOP:   MOVX  @DPTR, A       ; set FORCE_PD Bit in PMMR3 in PSD module in second
                             ; loop
        MOV   87h, A         ; set PD Bit in PCON register in MCU module in second
                             ; loop
        MOV   A, #02h        ; set power-down bit in the A register, but not in PMMR3 or
                             ; PCON yet in first loop
        JMP   LOOP           ; uPSD enters into Power-Down mode in second loop
```

Figure 88. Automatic power-down (APD) unit



30 Maximum rating

Stressing the device above the rating listed in the absolute maximum ratings table may cause permanent damage to the device. These are stress ratings only and operation of the device at these or any other conditions above those indicated in the operating sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability. Refer also to the STMicroelectronics SURE Program and other relevant quality documents.

Table 205. Absolute maximum ratings

Symbol	Parameter	Min.	Max.	Unit
T_{STG}	Storage temperature	–65	125	°C
T_{LEAD}	Lead temperature during soldering (20 seconds max.)(1)		235	°C
V_{IO}	Input and output voltage (Q = V_{OH} or Hi-Z)	–0.5	6.5	V
V_{CC}, V_{DD}, AV_{CC}	Supply voltage	–0.5	6.5	V
V_{ESD}	Electrostatic discharge voltage (human body model)(2)	–2000	2000	V

1. IPC/JEDEC J-STD-020A.
2. *JEDEC Std JESD22-A114A (C1=100pF, R1=1500 Ω, R2=500 Ω).*

Table 234. ISC timing (5 V PSD module) (continued)

Symbol	Parameter	Conditions	Min	Max	Unit
t_{ISCPCO}	ISC port clock to output			21	ns
t_{ISCPZV}	ISC port high-impedance to valid output			21	ns
t_{ISCPVZ}	ISC port valid output to high-impedance			21	ns

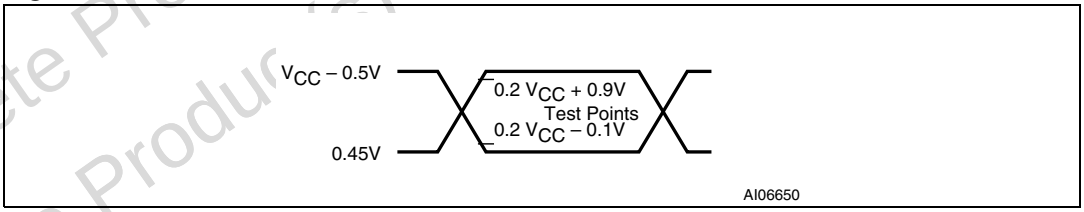
1. For non-PLD programming, erase or in ISC bypass mode.
2. For program or erase PLD only.

Table 235. ISC timing (3 V PSD module)

Symbol	Parameter	Conditions	Min	Max	Unit
t_{ISCCF}	Clock (TCK, PC1) frequency (except for PLD)	(1)		16	MHz
t_{ISCCH}	Clock (TCK, PC1) high time (except for PLD)		40		ns
t_{ISCCL}	Clock (TCK, PC1) low time (except for PLD)		40		ns
t_{ISCCFP}	Clock (TCK, PC1) frequency (PLD only)	(2)		4	MHz
t_{ISCCHP}	Clock (TCK, PC1) high time (PLD only)		90		ns
t_{ISCCLP}	Clock (TCK, PC1) low time (PLD only)		90		ns
t_{ISCPSU}	ISC port setup time		12		ns
t_{ISCPH}	ISC port hold up time		5		ns
t_{ISCPCO}	ISC port clock to output			30	ns
t_{ISCPZV}	ISC port high-impedance to valid output			30	ns
t_{ISCPVZ}	ISC port valid output to high-impedance			30	ns

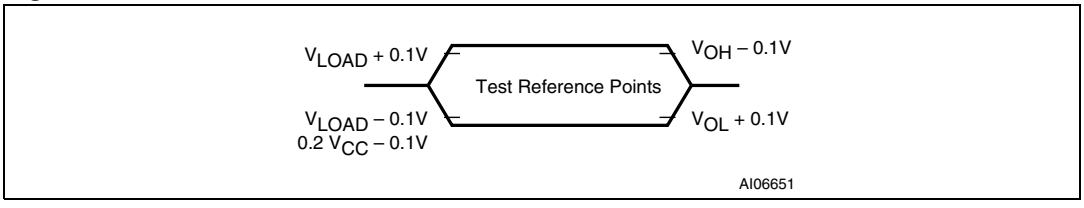
1. For non-PLD programming, erase or in ISC bypass mode.
2. For program or erase PLD only.

Figure 108. MCU module AC measurement I/O waveform



1. AC inputs during testing are driven at V_{CC}–0.5 V for a logic '1,' and 0.45 V for a logic '0.'
2. Timing measurements are made at V_{IH}(min) for a logic '1,' and V_{IL}(max) for a logic '0'

Figure 109. PSD module AC float I/O waveform



1. For timing purposes, a port pin is considered to be no longer floating when a 100 mV change from load voltage occurs, and begins to float when a 100 mV change from the loaded V_{OH} or V_{OL} level occurs
2. I_{OL} and $I_{OH} \geq 20$ mA

34 Important notes

The following sections describe the limitations that apply to the UPSD34xx devices and the differences between revision A and B silicon.

34.1 USB interrupts with idle mode

Description

An interrupt generated by a USB related event does not bring the MCU out of idle mode for processing.

Impact on application

Idle mode cannot be used with USB.

Workaround

Revision A - None identified at this time.

Revision B - Corrected in silicon so that a USB interrupt that occurs will bring the MCU out of idle mode.

34.2 USB reset interrupt

Description

When the MCU clock prescaler is set to a value other than $f_{MCU} = f_{OSC}$ (no division), a reset signal on the USB does not cause a USB interrupt to be generated.

Impact on application

An MCU clock other than that equal to the frequency of the oscillator cannot be used.

Workaround

Revision A - The CPUPS field in the CCON0 register must be set to 000b (default after reset). The 3400 USB firmware examples set CCON0 register to 000b.

Revision B - Corrected in silicon so that when an MCU clock prescaler is used, a reset signal on the USB does generate an interrupt.

34.3 USB reset

Description

A USB reset does not reset the USB SIE's registers.

Impact on application

A USB reset does not reset the USB SIE's registers as does a power-on or hardware reset.