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	46
Program Memory Size	288KB (288K 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/upsd3434eb40u6

Table 2. Pin definitions

Port pin	Signal name	80-pin No.	52-pin No.(1)	In/out	Function		
					Basic	Alternate 1	Alternate 2
MCUAD0	AD0	36	N/A	I/O	External bus multiplexed address/data bus A0/D0		
MCUAD1	AD1	37	N/A	I/O	Multiplexed address/data bus A1/D1		
MCUAD2	AD2	38	N/A	I/O	Multiplexed address/data bus A2/D2		
MCUAD3	AD3	39	N/A	I/O	Multiplexed address/data bus A3/D3		
MCUAD4	AD4	41	N/A	I/O	Multiplexed address/data bus A4/D4		
MCUAD5	AD5	43	N/A	I/O	Multiplexed address/data bus A5/D5		
MCUAD6	AD6	45	N/A	I/O	Multiplexed address/data bus A6/D6		
MCUAD7	AD7	47	N/A	I/O	Multiplexed address/data bus A7/D7		
P1.0	T2 ADC0	52	34	I/O	General I/O port pin	Timer 2 Count input (T2)	ADC Channel 0 input (ADC0)
P1.1	T2X ADC1	54	35	I/O	General I/O port pin	Timer 2 Trigger input (T2X)	ADC Channel 1 input (ADC1)
P1.2	RxD1 ADC2	56	36	I/O	General I/O port pin	UART1 or IrDA Receive (RxD1)	ADC Channel 2 input (ADC2)
P1.3	TXD1 ADC3	58	37	I/O	General I/O port pin	UART or IrDA Transmit (TxD1)	ADC Channel 3 input (ADC3)
P1.4	SPICLK ADC4	59	38	I/O	General I/O port pin	SPI Clock Out (SPICLK)	ADC Channel 4 input (ADC4)
P1.5	SPIRxD ADC5	60	39	I/O	General I/O port pin	SPI Receive (SPIRxD)	ADC Channel 5 input (ADC5)
P1.6	SPITXD ADC6	61	40	I/O	General I/O port pin	SPI Transmit (SPITxD)	ADC Channel 6 input (ADC6)
P1.7	SPISEL ADC7	64	41	I/O	General I/O port pin	SPI Slave Select (SPISEL)	ADC Channel 7 input (ADC7)
P3.0	RxD0	75	23	I/O	General I/O port pin	UART0 Receive (RxD0)	

10 UPSD34xx instruction set summary

Tables *6* through *11* list all of the instructions supported by the UPSD34xx, including the number of bytes and number of machine cycles required to implement each instruction. This is the standard 8051 instruction set.

The meaning of “machine cycles” is how many 8032 MCU core machine cycles are required to execute the instruction. The “native” duration of all machine cycles is set by the memory wait state settings in the SFR, BUSCON, and the MCU clock divider selections in the SFR, CCON0 (i.e. a machine cycle is typically set to 4 MCU clocks for a 5 V UPSD34xx). However, an individual machine cycle may grow in duration when either of two things happen:

1. a stall is imposed while loading the 8032 Pre-Fetch Queue (PFQ); or
2. the occurrence of a cache miss in the Branch Cache (BC) during a branch in program execution flow.

See *Section 5: 8032 MCU core performance enhancements on page 33* or more details.

But generally speaking, during typical program execution, the PFQ is not empty and the BC has no misses, producing very good performance without extending the duration of any machine cycles.

Table 6. Arithmetic instruction set

Mnemonic[1] and use		Description	Length/cycles
ADD	A, Rn	Add register to ACC	1 byte/1 cycle
ADD	A, Direct	Add direct byte to ACC	2 byte/1 cycle
ADD	A, @Ri	Add indirect SRAM to ACC	1 byte/1 cycle
ADD	A, #data	Add immediate data to ACC	2 byte/1 cycle
ADDC	A, Rn	Add register to ACC with carry	1 byte/1 cycle
ADDC	A, direct	Add direct byte to ACC with carry	2 byte/1 cycle
ADDC	A, @Ri	Add indirect SRAM to ACC with carry	1 byte/1 cycle
ADDC	A, #data	Add immediate data to ACC with carry	2 byte/1 cycle
SUBB	A, Rn	Subtract register from ACC with borrow	1 byte/1 cycle
SUBB	A, direct	Subtract direct byte from ACC with borrow	2 byte/1 cycle
SUBB	A, @Ri	Subtract indirect SRAM from ACC with borrow	1 byte/1 cycle
SUBB	A, #data	Subtract immediate data from ACC with borrow	2 byte/1 cycle
INC	A	Increment A	1 byte/1 cycle
INC	Rn	Increment register	1 byte/1 cycle
INC	direct	Increment direct byte	2 byte/1 cycle
INC	@Ri	Increment indirect SRAM	1 byte/1 cycle
DEC	A	Decrement ACC	1 byte/1 cycle
DEC	Rn	Decrement register	1 byte/1 cycle
DEC	direct	Decrement direct byte	2 byte/1 cycle

17.1.1 GPIO function

Ports in GPIO mode operate as quasi-bidirectional pins, consistent with standard 8051 architecture. GPIO pins are individually controlled by three SFRs:

- SFR, P1 (*Table 35 on page 81*)
- SFR, P3 (*Table 37 on page 82*)
- SFR, P4 (*Table 39 on page 82*)

These SFRs can be accessed using the Bit Addressing mode, an efficient way to control individual port pins.

17.1.2 GPIO output

Simply stated, when a logic '0' is written to a bit in any of these port SFRs while in GPIO mode, the corresponding port pin will enable a low-side driver, which pulls the pin to ground, and at the same time releases the high-side driver and pull-ups, resulting in a logic '0' output. When a logic '1' is written to the SFR, the low-side driver is released, the high-side driver is enabled for just one MCU_CLK period to rapidly make the 0-to1 transition on the pin, while weak active pull-ups (total ~150KΩ) to V_{CC} are enabled. This structure is consistent with standard 8051 architecture. The high side driver is momentarily enabled only for 0-to-1 transitions, which is implemented with the delay function at the latch output as pictured in *Figure 16 on page 80*, *Figure 17 on page 81*, and *Figure 18 on page 81*. After the high-side driver is disabled, the two weak pull-ups remain enabled resulting in a logic '1' output at the pin, sourcing I_{OH} µA to an external device. Optionally, an external pull-up resistor can be added if additional source current is needed while outputting a logic '1.'

17.1.3 GPIO input

To use a GPIO port pin as an input, the low-side driver to ground must be disabled, or else the true logic level being driven on the pin by an external device will be masked (always reads logic '0'). So to make a port pin “input ready”, the corresponding bit in the SFR must have been set to a logic '1' prior to reading that SFR bit as an input. A reset condition forces SFRs P1, P3, and P4 to FFh, thus all three ports are input ready after reset.

When a pin is used as an input, the stronger pull-up “A” maintains a solid logic '1' until an external device drives the input pin low. At this time, pull-up “A” is automatically disabled, and only pull-up “B” will source the external device I_{IH} µA, consistent with standard 8051 architecture.

GPIO bidirectional. It is possible to operate individual port pins in bidirectional mode. For an output, firmware would simply write the corresponding SFR bit to logic '1' or '0' as needed. But before using the pin as an input, firmware must first ensure that a logic '1' was the last value written to the corresponding SFR bit prior to reading that SFR bit as an input.

GPIO current capability. A GPIO pin on Port 4 can sink twice as much current than a pin on either Port 1 or Port 3 when the low-side driver is outputting a logic '0' (I_{OL}). See the DC specifications at the end of this document for full details.

Reading port pin vs. reading port latch. When firmware reads the GPIO ports, sometimes the actual port pin is sampled in hardware, and sometimes the port SFR latch is read and not the actual pin, depending on the type of MCU instruction used. These two data paths are shown in *Figure 16 on page 80* through *Figure 18 on page 81*. SFR latches are read (and not the pins) only when the read is part of a *read-modify-write* instruction and the write destination is a bit or bits in a port SFR. These instructions are: ANL, ORL, XRL, JBC,

A[10:8] and the remaining pins can be configured for other functions such as generating chip selects to the external devices.

Figure 19. Connecting external devices using ports A and B for address AD[15:0]

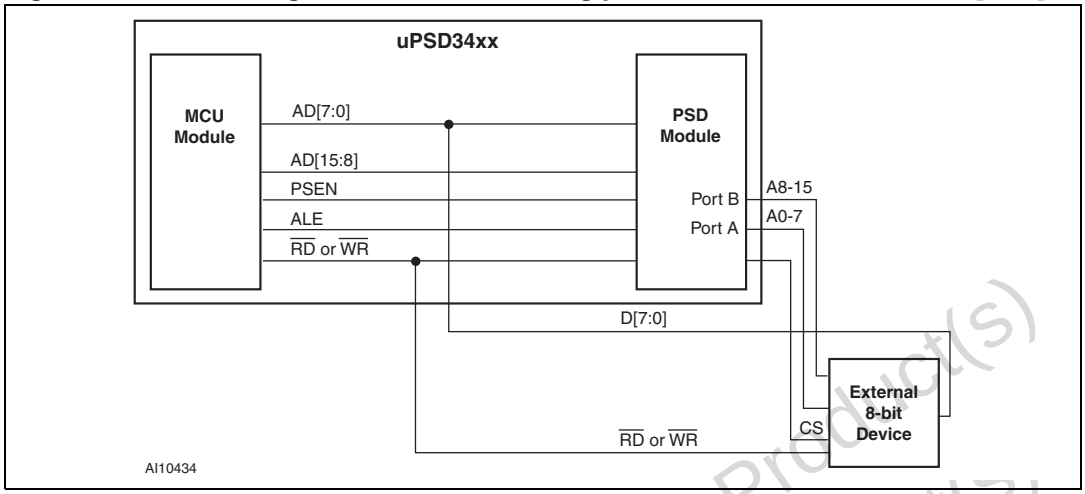
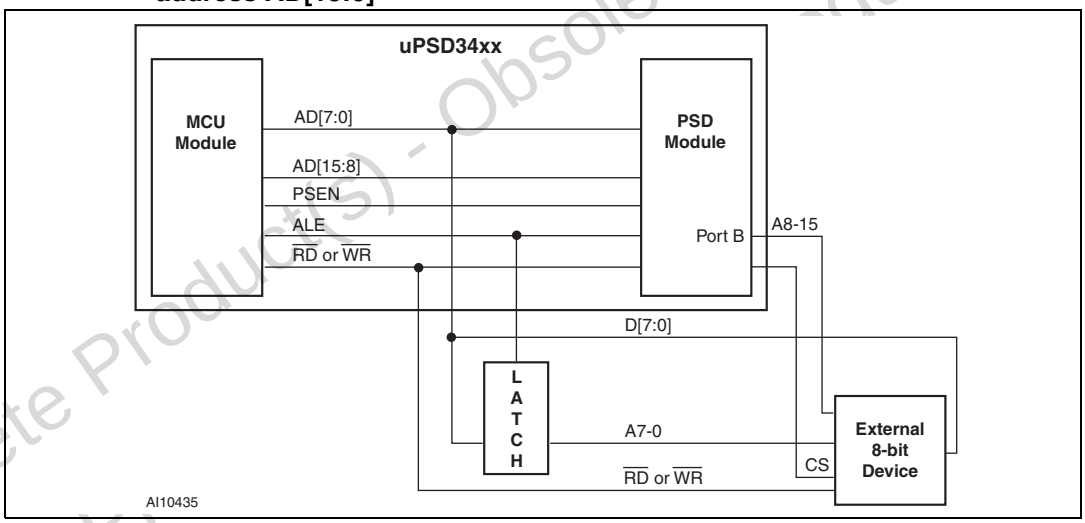


Figure 20. Connecting external devices using port A and an external latch for address AD[15:0]



18.4 Programmable bus timing

The length of the bus cycles are user programmable at run time. The number of MCU_CLK periods in a bus cycle can be specified in the SFR register named BUSCON (see *Table 49 on page 88*). By default, the BUSCON register is loaded with long bus cycle times (6 MCU_CLK periods) after a reset condition. It is important that the post-reset initialization firmware sets the bus cycle times appropriately to get the most performance, according to *Table 51 on page 90*. Keep in mind that the PSD module has a faster Turbo mode (default) and a slower but less power consuming Non-Turbo mode. The bus cycle times must be programmed in BUSCON to optimize for each mode as shown in *Table 51*. See *Section 28.5: PSD module detailed operation on page 207* for more details.

23.2 Communication flow

I²C data flow control is based on the fact that all I²C compatible devices will drive the bus lines with open-drain (or open-collector) line drivers pulled up with external resistors, creating a wired-AND situation. This means that either bus line (SDA or SCL) will be at a logic '1' level only when no I²C device is actively driving the line to logic '0.' The logic for handshaking, arbitration, synchronization, and collision detection is implemented by each I²C device having:

1. The ability to hold a line low against the will of the other devices who are trying to assert the line high.
2. The ability of a device to detect that another device is driving the line low against its will.

Assert high means the driver releases the line and external pull-ups passively raise the signal to logic '1.' Holding low means the open-drain driver is actively pulling the signal to ground for a logic '0.'

For example, if a Slave device cannot transmit or receive a byte because it is distracted by and interrupt or it has to wait for some process to complete, it can hold the SCL clock line low. Even though the Master device is generating the SCL clock, the Master will sense that the Slave is holding the SCL line low against the will of the Master, indicating that the Master must wait until the Slave releases SCL before proceeding with the transfer.

Another example is when two Master devices try to put information on the bus simultaneously, the first one to release the SDA data line looses arbitration while the winner continues to hold SDA low.

Two types of data transfers are possible with I²C depending on the R/$\overline{W}$ bit, see *Figure 41 on page 125*.

1. **Data transfer from master transmitter to slave receiver (R/$\overline{W}$ = 0).** In this case, the Master generates a Start condition on the bus and it generates a clock signal on the SCL line. Then the Master transmits the first byte on the SDA line containing the 7-bit Slave address plus the R/$\overline{W}$ bit. The Slave who owns that address will respond with an acknowledge bit on SDA, and all other Slave devices will not respond. Next, the Master will transmit a data byte (or bytes) that the addressed Slave must receive. The Slave will return an acknowledge bit after each data byte it successfully receives. After the final byte is transmitted by the Master, the Master will generate a Stop condition on the bus, or it will generate a Re-Start conditon and begin the next transfer. There is no limit to the number of bytes that can be transmitted during a transfer session.
2. **Data transfer from slave transmitter to master receiver (R/$\overline{W}$ = 1).** In this case, the Master generates a Start condition on the bus and it generates a clock signal on the SCL line. Then the Master transmits the first byte on the SDA line containing the 7-bit Slave address plus the R/$\overline{W}$ bit. The Slave who owns that address will respond with an acknowledge bit on SDA, and all other Slave devices will not respond. Next, the addressed Slave will transmit a data byte (or bytes) to the Master. The Master will return an acknowledge bit after each data byte it successfully receives, unless it is the last byte the Master desires. If so, the Master will not acknowledge the last byte and from this, the Slave knows to stop transmitting data bytes to the Master. The Master will then generate a Stop condition on the bus, or it will generate a Re-Start conditon and begin the next transfer. There is no limit to the number of bytes that can be transmitted during a transfer session.

The interface may operate as either a Master or a Slave within a given application, controlled by firmware writing to SFRs.

By default after a reset, the I²C interface is in Master Receiver mode, and the SDA/P3.6 and SCL/P3.7 pins default to GPIO input mode, high impedance, so there is no I²C bus interference. Before using the I²C interface, it must be initialized by firmware, and the pins must be configured. This is discussed in *Section 23.13: I2C operating sequences on page 135*.

23.4 Bus arbitration

A Master device always samples the I²C bus to ensure a bus line is high whenever that Master is asserting a logic 1. If the line is low at that time, the Master recognizes another device is overriding its own transmission.

A Master may start a transfer only if the I²C bus is not busy. However, it is possible that two or more Masters may generate a Start condition simultaneously. In this case, arbitration takes place on the SDA line each time SCL is high. The Master that first senses that its bus sample does not correspond to what it is driving (SDA line is low while it is asserting a high) will immediately change from Master-Transmitter to Slave-Receiver mode. The arbitration process can carry on for many bit times if both Masters are addressing the same Slave device, and will continue into the data bits if both Masters are trying to be Master-Transmitter. It is also possible for arbitration to carry on into the acknowledge bits if both Masters are trying to be Master-Receiver. Because address and data information on the bus is determined by the winning Master, no information is lost during the arbitration process.

23.5 Clock synchronization

Clock synchronization is used to synchronize arbitrating Masters, or used as a handshake by a devices to slow down the data transfer.

23.5.1 Clock sync during arbitration

During bus arbitration between competing Masters, Master_X, with the longest low period on SCL, will force Master_Y to wait until Master_X finishes its low period before Master_Y proceeds to assert its high period on SCL. At this point, both Masters begin asserting their high period on SCL simultaneously, and the Master with the shortest high period will be the first to drive SCL for the next low period. In this scheme, the Master with the longest low SCL period paces low times, and the Master with the shortest high SCL period paces the high times, making synchronized arbitration possible.

23.5.2 Clock sync during handshaking

This allows receivers in different devices to handle various transfer rates, either at the byte-level, or bit-level.

At the byte-level, a device may pause the transfer between bytes by holding SCL low to have time to store the latest received byte or fetch the next byte to transmit.

At the bit-level, a Slave device may extend the low period of SCL by holding it low. Thus the speed of any Master device will adapt to the internal operation of the Slave.

25.2 Types of transfers

The USB specification defines four types of transfers, Bulk, Interrupt, Isochronous, and Control.

Note: The UPSD34xx supports all types of transfers except Isochronous.

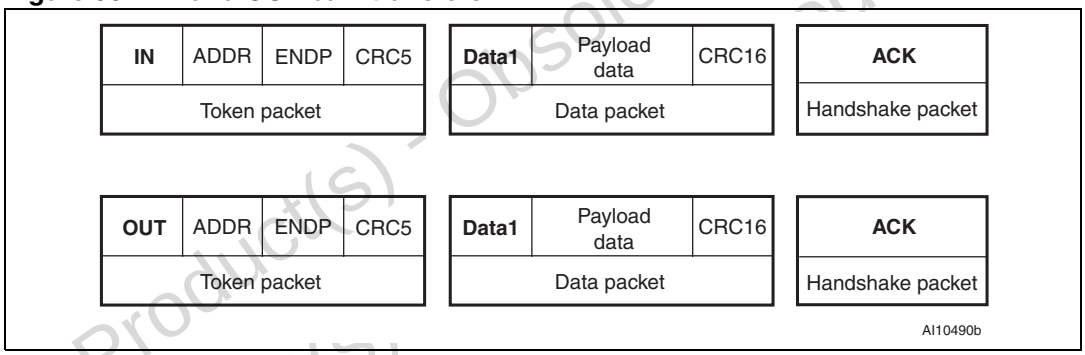
- Bulk Transfers (see *Figure 50*)

  Bulk data is transferred in both directions and is used with both IN and OUT endpoints. Packets may be 8, 16, 32, or 64 bytes in length. Bulk transfers occur in bursts, and are scheduled by the host when there is available time on the bus. While there is no guaranteed delivery time for bulk transfers, the accuracy of the data is guaranteed due to automatic retries for erroneous data. Bulk transfers are typically used for mass storage, printer, and scanner data.

- Interrupt Transfers (see *Figure 51*)

  Interrupt data is a lot like bulk data but travels only in one direction, from the device to the host, so only IN endpoints are used. Interrupt data holds packet sizes ranging from 1 to 64 bytes. Interrupt endpoints have an associated polling interval, meaning that the host sends IN tokens at a periodic interval to the host on a regular basis. Interrupt transfers are typically used for human interface devices such as keyboards, mice, and joysticks.

Figure 50. IN and OUT bulk transfers



- USB IN FIFO interrupt flag (UIF1)

  The USB IN FIFO Interrupt Flag register (see *Table 114*) contains flags that indicate when an IN Endpoint FIFO that was full becomes empty. Once set, firmware must clear the flag by writing a '0' to the appropriate bit. When FIFOs are paired, only the odd numbered FIFO Interrupt flags are active.

Table 114. USB IN FIFO interrupt flag (UIF1 0E9h, reset value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	–	–	IN4F	IN3F	IN2F	IN1F	IN0F

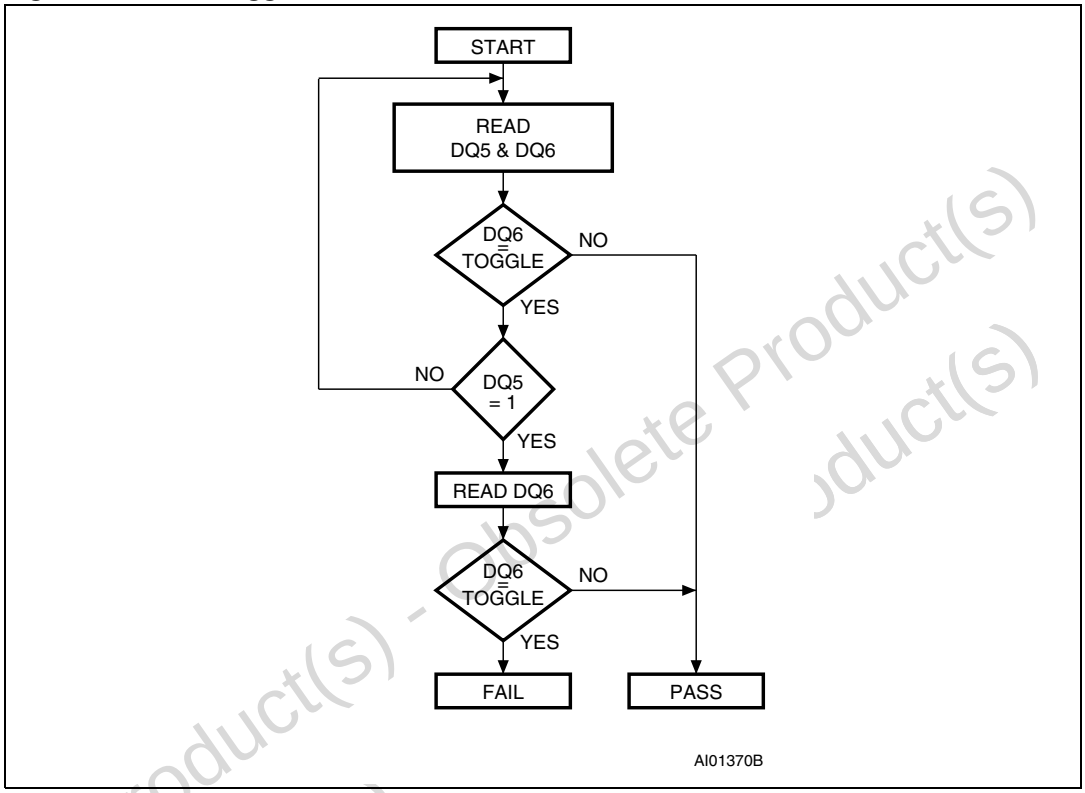
Table 115. UIF1 register bit definition

Bit	Symbol	R/W	Definition
7	–	–	Reserved
6	–	–	Reserved
5	–	–	Reserved
4	IN4F	R/W	Endpoint 4 IN FIFO Interrupt flag This bit is set when the FIFO status changes from full to empty.
3	IN3F	R/W	Endpoint 3 IN FIFO Interrupt flag This bit is set when the FIFO status changes from full to empty.
2	IN2F	R/W	Endpoint 2 IN FIFO Interrupt flag This bit is set when the FIFO status changes from full to empty.
1	IN1F	R/W	Endpoint 1 IN FIFO Interrupt flag This bit is set when the FIFO status changes from full to empty.
0	IN0F	R/W	Endpoint 0 IN FIFO Interrupt flag This bit is set when the FIFO status changes from full to empty.

Bit (DQ5) indicates a time-out condition on the Erase cycle, a ’0’ indicates no error. The 8032 can read any location within the sector being erased to get the Toggle Flag Bit (DQ6) and the Error Flag Bit (DQ5).

PSDsoft Express generates ANSI C code functions the user may use to implement these Data Toggling algorithms.

Figure 72. Data toggle flowchart



28.5.13 Ready/$\overline{\text{Busy}}$ (PC3)

This signal can be used to output the Ready/$\overline{\text{Busy}}$ status of a program or erase operation on either Flash memory. The output on the Ready/$\overline{\text{Busy}}$ pin is a ’0’ (Busy) when either Flash memory array is being written, *or* when either Flash memory array is being erased. The output is a ’1’ (Ready) when no program or erase operation is in progress. To activate this function on this pin, the user must select the “Ready/$\overline{\text{Busy}}$” selection in PSDsoft Express when configuring pin PC3. This pin may be polled by the 8032 or used as a 8032 interrupt to indicate when an erase or program operation is complete (requires routing the signal on PC board from PC3 back into a pin on the MCU module). This signal is also available internally on the PSD module as an input to both PLDs (without routing a signal externally on PC board) and its signal name is “rd_bsy”. The Ready/$\overline{\text{Busy}}$ output can be probed during lab development to check the timing of Flash memory programming in the system at run-time.

28.5.14 Bypassed unlock sequence

The Bypass Unlock mode allows the 8032 to program bytes in the Flash memories faster than using the standard Flash program instruction sequences because the typical AAh, 55h unlock bus cycles are bypassed for each byte that is programmed. Bypassing the unlock

28.5.20 Reset Flash

The Reset Flash instruction sequence resets the embedded algorithm running on the state machine in the targeted Flash memory (Main or Secondary) and the memory goes into Read Array mode. The Reset Flash instruction consists of one bus WRITE cycle as shown in *Table 163 on page 209*, and it must be executed after any error condition that has occurred during a Flash memory Program or Erase operation.

It may take the Flash memory up to 25µs to complete the Reset cycle. The Reset Flash instruction sequence is ignored when it is issued during a Program or Bulk Erase operation. The Reset Flash instruction sequence aborts any on-going Sector Erase operation and returns the Flash memory to Read Array mode within 25µs.

28.5.21 Reset signal applied to Flash memory

Whenever the PSD module receives a reset signal from the MCU module, any operation that is occurring in either Flash memory array will be aborted and the array(s) will go to Read Array mode. It may take up to 25µs to abort an operation and achieve Read Array mode.

A reset from the MCU module will result from any of these events: an active signal on the UPSD34xx $\overline{\text{RESET\_IN}}$ input pin, a watchdog timer time-out, detection of low V_{CC}, or a JTAG debug channel reset event.

28.5.22 Flash memory sector protection

Each Flash memory sector can be separately protected against program and erase operations. This mode can be activated (or deactivated) by selecting this feature in PSDsoft Express and then programming through the JTAG Port. Sector protection can be selected for individual sectors, and the 8032 cannot override the protection during run-time. The 8032 can read, but not change, sector protection.

Any attempt to program or erase a protected Flash memory sector is ignored. The 8032 may read the contents of a Flash sector even when a sector is protected.

Sector protection status is not read using Flash memory instruction sequences, but instead this status is read by the 8032 reading two registers within csiop address space shown in Table *165* and Table *166*.

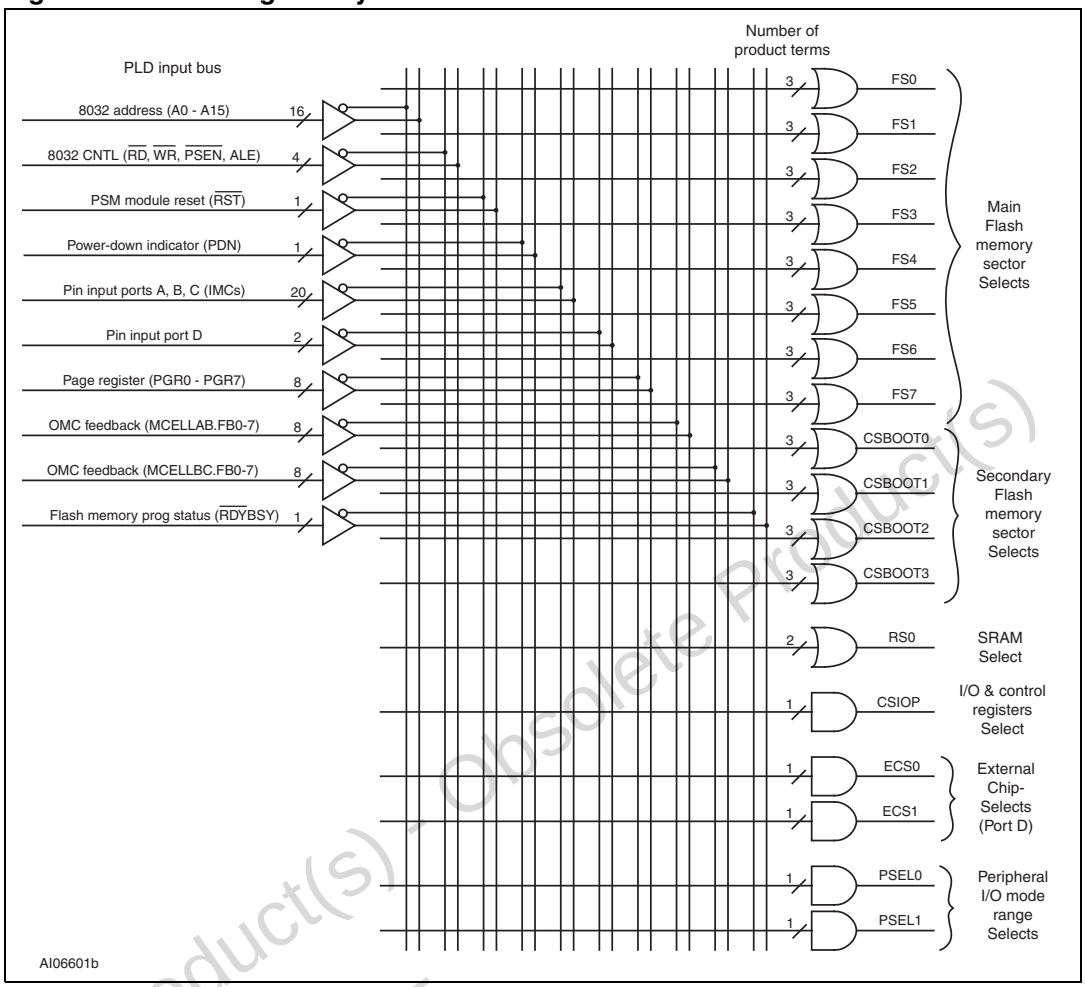
28.5.23 Flash memory protection during power-up

Flash memory WRITE operations are automatically prevented while V_{DD} is ramping up until it rises above V_{LKO} voltage threshold at which time Flash memory WRITE operations are allowed.

28.5.24 PSD module security bit

A programmable security bit in the PSD module protects its contents from unauthorized viewing and copying. The security bit is set using PSDsoft Express and programmed into the PSD module with JTAG. When set, the security bit will block access of JTAG programming equipment from reading or modifying the PSD module Flash memory and PLD configuration. The security bit also blocks JTAG access to the MCU module for debugging. The only way to defeat the security bit is to erase the entire PSD module using JTAG (erase is the only JTAG operation allowed while security bit is set), after which the device is blank and may be used again. The 8032 MCU will always have access to Flash

Figure 74. DPLD logic array



28.5.28 General PLD (GPLD)

The GPLD is used to create general system logic. *Figure 73 on page 221* shows the architecture of the entire GPLD, and *Figure 75 on page 224* shows the relationship between one OMC, one IMC, and one I/O port pin, which is representative of pins on Ports A, B, and C. It is important to understand how these elements work together. A more detailed description will follow for the three major blocks (OMC, IMC, I/O Port) shown in *Figure 75*. *Figure 75* also shows which csiop registers to access for various PLD and I/O functions.

The GPLD contains:

- 16 Output Macrocells (OMC)
- 20 Input Macrocells (IMC)
- OMC Allocator
- Product Term Allocator inside each OMC
- AND-OR Array capable of generating up to 137 product terms
- Three I/O Ports, A, B, and C

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

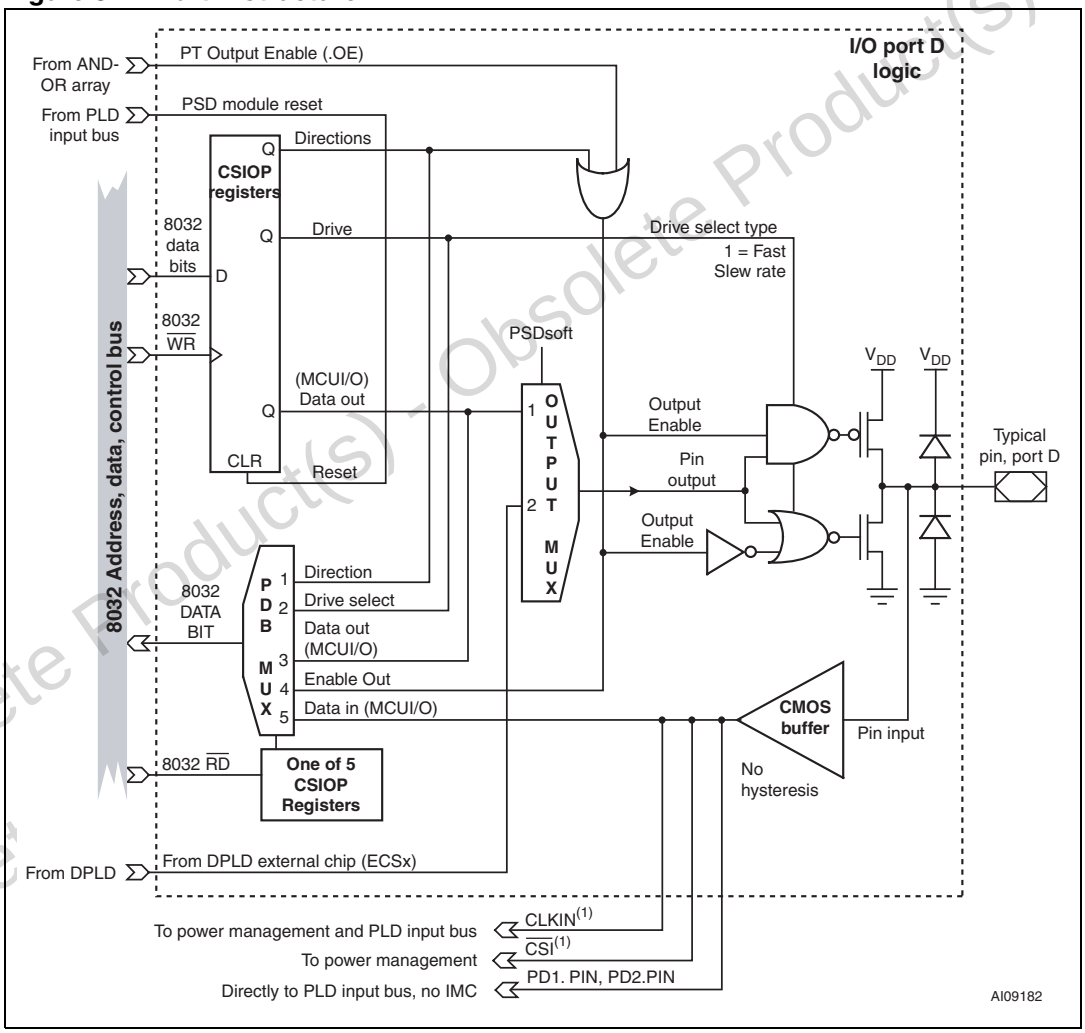
1. For each bit, 1 = pin drive is enabled as an output, 0 = pin drive is off (high-impedance, pin used as input)
2. Pin is not available on 52-pin UPSD34xx devices

Port D pins can also be configured in PSDsoft as pins for other dedicated functions:

- PD1 can be used as a common clock input to all 16 OMC Flip-flops (see *Section 28.1.11: OMCs on page 195*) and also the *Section 28.5.53: Automatic power-down (APD) on page 250*.
- PD2 can be used as a common chip select signal ($\overline{CSI}$) for the Flash and SRAM memories on the PSD module (see *Section 28.5.55: Chip select input (CSI) on page 253*). If driven to logic '1' by an external source, $\overline{CSI}$ will force the Flash memory into standby mode regardless of what other internal memory select signals are doing on the PSD module. This is specified in PSDsoft as "PSD Chip Select Input, $\overline{CSI}$".

Port D also supports the Fast Slew Rate output drive type option using the csiop Drive Select registers.

Figure 87. Port D structure



Note: 1 Optional function on a specific Port D pin.

28.5.52 Power management

The PSD module offers configurable power saving options. These options may be used individually or in combinations. A top level description for these functions is given here, then more detailed descriptions will follow.

- **Zero-Power memory:** All memory arrays (Flash and SRAM) in the PSD module are built with zero-power technology. It puts the Flash memory into standby mode (~ zero DC current) when 8032 address signals are not changing. As soon as a transition occurs on any address input, the Flash memory “wakes up”, changes and latches its outputs, then goes back to standby. The designer does not have to do anything special to achieve this memory standby mode when no inputs are changing—it happens automatically. Thus, the slower the 8032 clock, the lower the current consumption.

  Both PLDs (DPLD and GPLD) are also zero-power, but this is not the default condition. The 8032 must set a bit in one of the csiop PMMR registers at run-time to achieve zero-power.

- **Automatic Power-down (APD):** The APD feature allows the PSD module to reach its lowest current consumption levels. If enabled, the APD counter will time-out when there is a lack of 8032 bus activity for an extended amount of time (8032 asleep). After time-out occurs, all 8032 address and data buffers on the PSD module are shut down, preventing the PSD module memories and potentially the PLDs from waking up from standby, even if address inputs are changing state because of noise or any external components driving the address lines. Since the actual address and data buffers are turned off, current consumption is even further reduced.

Note: Non-address signals are still available to PLD inputs and will wake up the PLDs if these signals are changing state, but will not wake up the memories.

  The APD counter requires a relatively slow external clock input on pin PD1 that does stop when the 8032 goes to sleep mode.

- **Forced Power-down (FPD):** The MCU can put the PSD module into Power-Down mode with the same results as using APD described above, but FPD does not rely on the APD counter. Instead, FPD will force the PSD module into Power-Down mode when the MCU firmware sets a bit in one of the csiop PMMR registers. This is a good alternative to APD because no external clock is needed for the APD counter.

- **PSD module Chip Select input ($\overline{\text{CSI}}$):** This input on pin PD2 (80-pin devices only) can be used to disable the internal Flash memory, placing it in standby mode even if address inputs are changing. This feature does not block any internal signals (the address and data buffers are still on but signals are ignored) and $\overline{\text{CSI}}$ does not disable the PLDs. This is a good alternative to using the APD counter, which requires an external clock on pin PD1.

- **Non-Turbo mode:** The PLDs can operate in Turbo or non-Turbo modes. Turbo mode has the shortest signal propagation delay, but consumes more current than non-Turbo mode. A csiop register can be written by the 8032 to select modes, the default mode is with Turbo mode enabled. In non-Turbo mode, the PLDs can achieve very low standby current (~ zero DC current) while no PLD inputs are changing, and the PLDs will even use less AC current when inputs do change compared to Turbo mode.

  When the Turbo mode is enabled, there is a significant DC current component AND the AC current component is higher than non-Turbo mode, as shown in *Figure 95 on page 265* (5 V) and *Figure 96 on page 266* (3.3 V).

- **Blocking bits:** Significant power savings can be achieved by blocking 8032 bus control signals ($\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{PSEN}}$, ALE) from reaching PLD inputs, if these signals are not used in any PLD equations. Blocking is achieved by the 8032 writing to the “blocking bits” in

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

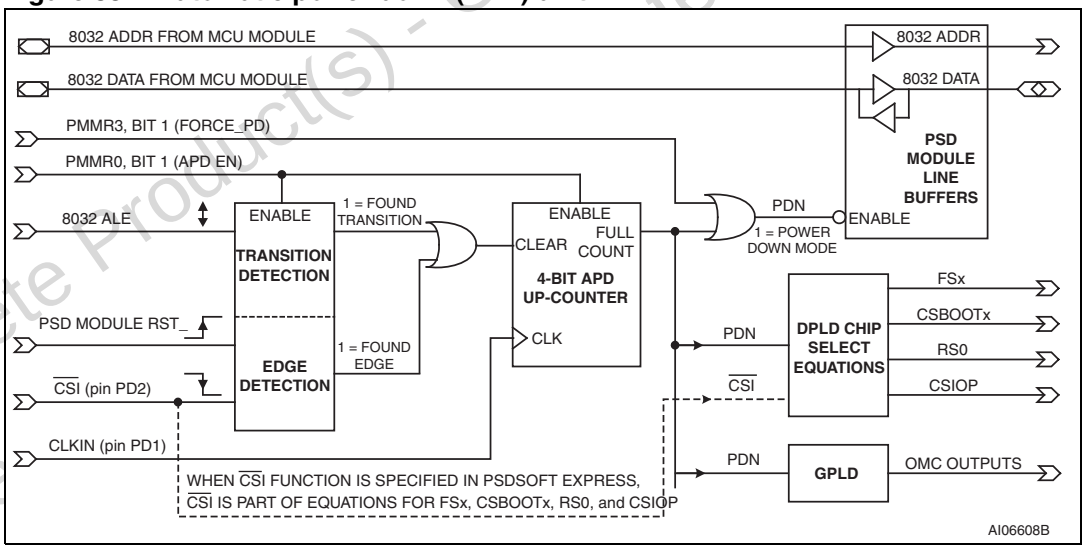


Table 217. External clock drive

Symbol	Parameter	Variable oscillator $1/t_{CLCL}$ = 3 to 40 MHz		Unit
		Min	Max	
t_{CLCL}	Oscillator period	25	333	ns
t_{CHCX}	High time	10	$t_{CLCL} - t_{CLCX}$	ns
t_{CLCX}	Low time	10	$t_{CLCL} - t_{CLCX}$	ns
t_{CLCH}	Rise time		10	ns
t_{CHCL}	Fall time		10	ns

Table 218. A/D analog specification

Symbol	Parameter	Test conditions(1)	Min.	Typ.	Max.	Unit
I_{DD}	Normal	Input = AV_{REF}		4.0		mA
	Power-down				40	µA
AV_{IN}	Analog input voltage		GND		AV_{REF}	V
AV_{REF}(2)(3)	Analog reference voltage				3.6	V
Accuracy	Resolution				10	bits
INL	Integral nonlinearity	Input = 0 to AV_{REF} (V) $f_{OSC} \leq 32MHz$			±2	LSB
DNL	Differential nonlinearity	Input = 0 to AV_{REF} (V) $f_{OSC} \leq 32MHz$			±2	LSB
SNR	Signal to noise ratio	f_{SAMPLE} = 500ksps	50	54		dB
SNDR	Signal to noise distortion ratio		48	52		dB
ACLK	ADC clock		2	8	16	MHz
t_C	Conversion time	8MHz	1	4	8	µs
t_{CAL}	Power-up time	Calibration Time		16		ms
f_{IN}	Analog input frequency				60	kHz
THD	Total harmonic distortion		50	54		dB

1. f_{IN} 2kHz, ACLK = 8MHz, $AV_{REF} = AV_{CC}$ = 3.3 V.
2. $AV_{REF} = AV_{CC}$ in 52-pin package.
3. If the A/D converter is not used, connect AV_{CC}/AV_{REF} to V_{CC}.

Table 219. USB transceiver specification

Symbol	Parameter	Test conditions(1)	Min.	Typ.	Max.	Unit
UV_{OH}	High output voltage	V_{DD} = 3.3 V; I_{OUT} = 2.2mA	3		–	V
UV_{OL}	Low output voltage	V_{DD} = 3.3 V; I_{OUT} = 2.2mA	–		0.25	V
UV_{IH}	High input voltage	V_{DD} = 3.6 V	2		–	V

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).

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.