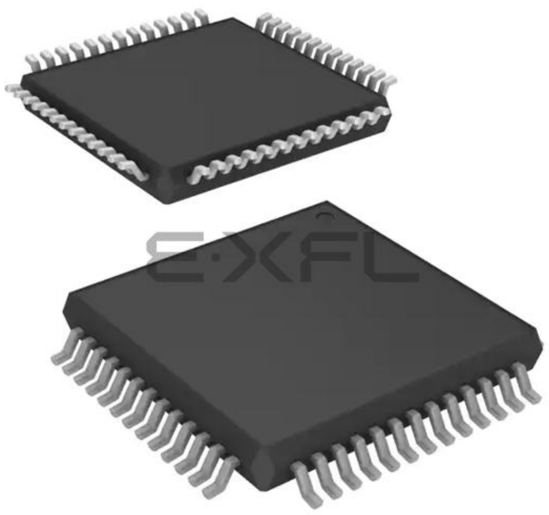
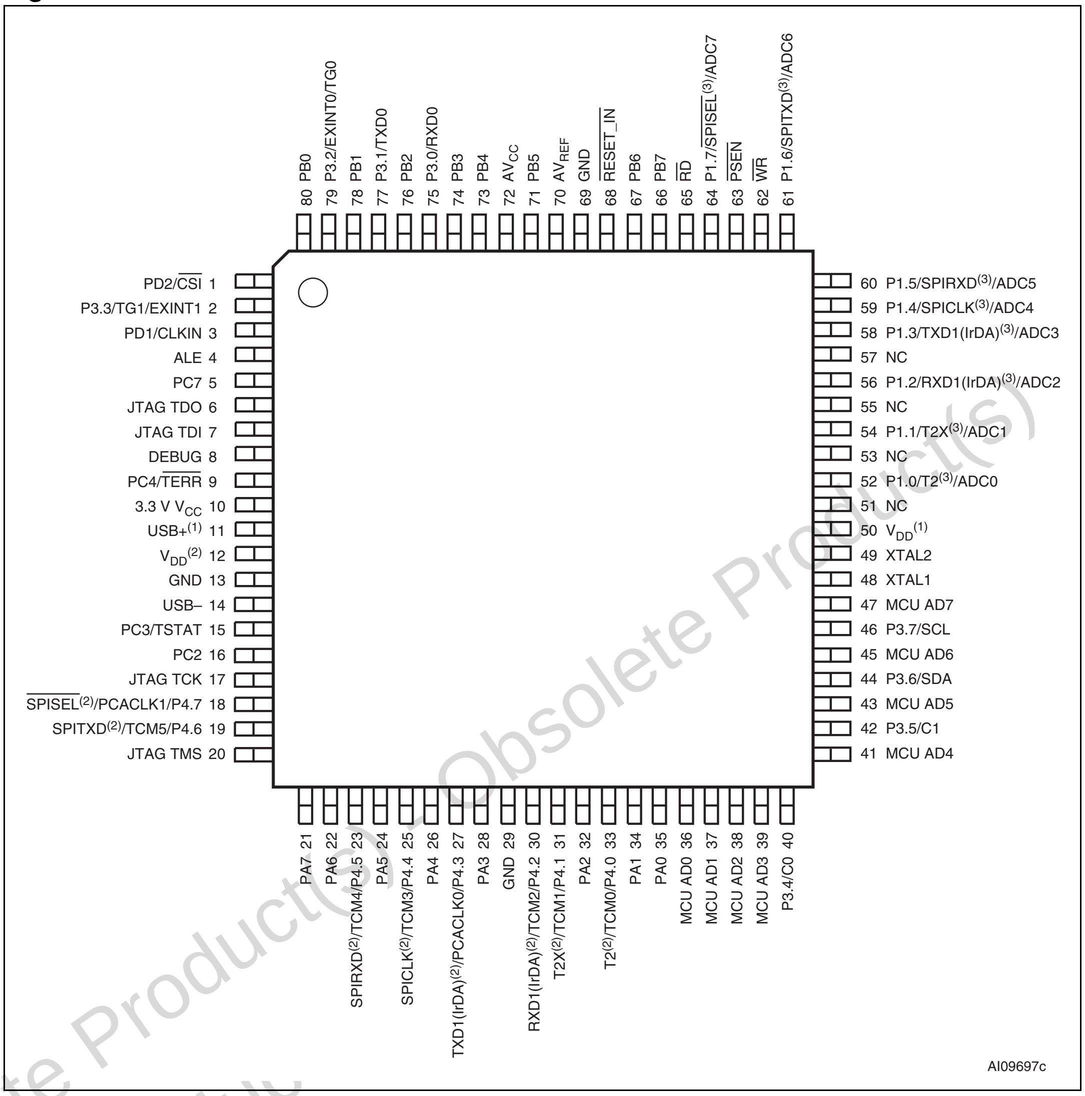


| **Details** | |
|---|---|
| Product Status | Obsolete |
| Core Processor | 8032 |
| Core Size | 8-Bit |
| Speed | 40MHz |
| Connectivity | I²C, IrDA, SPI, UART/USART, USB |
| Peripherals | LVD, POR, PWM, WDT |
| Number of I/O | 35 |
| Program Memory Size | 80KB (80K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 4K 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 | 52-LQFP |
| Supplier Device Package | - |
| Purchase URL | https://www.e-xfl.com/product-detail/stmicroelectronics/upsd3422eb40t6 |

**Figure 3. LQFP80 connections**



1. NC = Not connected
2. The USB+ pin needs a 1.5 kΩ pull-up resistor.
3. For 5 V applications, $V_{DD}$ must be connected to a 5.0 V source. For 3.3 V applications, $V_{DD}$ must be connected to a 3.3 V source.
4. These signals can be used on one of two different ports (Port 1 or Port 4) for flexibility. Default is Port1.

If an interrupt flag is not cleared after servicing the interrupt, an unwanted interrupt will occur upon exiting the ISR.

After the interrupt is serviced, the last instruction executed by the ISR is RETI. The RETI informs the MCU that the ISR is no longer in progress and the MCU pops the top two bytes from the stack and loads them into the PC. Execution of the interrupted program continues where it left off.

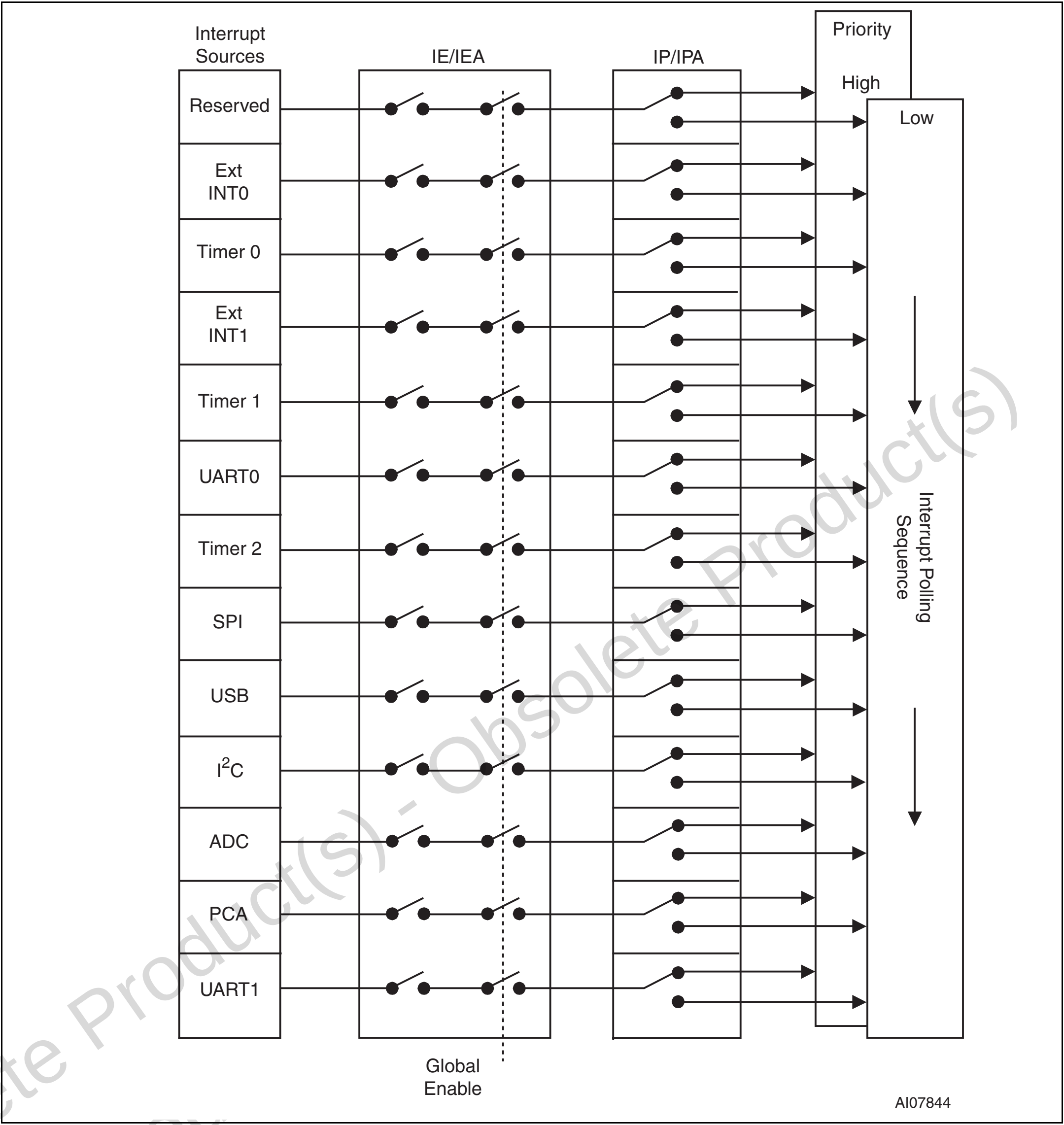
*Note: An ISR must end with a RETI instruction, not a RET. An RET will not inform the interrupt control system that the ISR is complete, leaving the MCU to think the ISR is still in progress, making future interrupts impossible.*

**Table 17. Interrupt summary**

| Interrupt source | Polling priority | Vector addr | Flag bit name (SFR.bit position) 1 = Intr pending 0 = No interrupt | Flag bit auto-cleared by hardware? | Enable bit name (SFR.bit position) 1 = Intr Enabled 0 = Intr Disabled | Priority bit name (SFR.bit position) 1= High Priority 0 = Low Priority |
|---|---|---|---|---|---|---|
| Reserved | 0 (high) | 0063h | – | – | – | – |
| External Interrupt INT0 | 1 | 0003h | IE0 (TCON.1) | Edge - Yes Level - No | EX0 (IE.0) | PX0 (IP.0) |
| Timer 0 Overflow | 2 | 000Bh | TF0 (TCON.5) | Yes | ET0 (IE.1) | PT0 (IP.1) |
| External Interrupt INT1 | 3 | 0013h | IE1 (TCON.3 | Edge - Yes Level - No | EX1 (IE.2) | PX1 (IP.2) |
| Timer 1 Overflow | 4 | 001Bh | TF1 (TCON.7) | Yes | ET1 (IE.3) | PT1 (IP.3) |
| UART0 | 5 | 0023h | RI (SCON0.0) TI (SCON0.1) | No | ES0 (IE.4) | PS0 (IP.4) |
| Timer 2 Overflow or TX2 Pin | 6 | 002Bh | TF2 (T2CON.7) EXF2 (T2CON.6) | No | ET2 (IE.5) | PT2 (IP.5) |
| SPI | 7 | 0053h | TEISF, RORISF, TISF, RISF (SPISTAT[3:0]) | Yes | ESPI (IEA.6) | PSPI (IPA.6) |
| USB | 8 | 0033h | – (1) | No | EUSB (IEA.0) | PUSB (IPA.0) |
| $I^2C$ | 9 | 0043h | INTR (S1STA.5) | Yes | $EI^2C$ (IEA.1) | $PI^2C$ (IPA.1) |
| ADC | 10 | 003Bh | AINTF (ACON.7) | No | EADC (IEA.7) | PADC (IPA.7) |
| PCA | 11 | 005Bh | OFVx, INTFx (PCASTA[0:7]) | No | EPCA (IEA.5) | PPCA (IPA.5) |
| UART1 | 12 (low) | 004Bh | RI (SCON1.0) TI (SCON1.1) | No | ES1 (IEA.4) | PS1 (IPA.4) |

1. See USB interrupt flag registers UIF0-3.

Figure 12. Enabling and polling interrupts



## 13.1 Individual interrupt sources

### 13.1.1 External interrupts Int0 and Int1

External interrupt inputs on pins EXTINT0 and EXTINT1 (pins 3.2 and 3.3) are either edge-triggered or level-triggered, depending on bits IT0 and IT1 in the SFR named TCON.

When an external interrupt is generated from an edge-triggered (falling-edge) source, the appropriate flag bit (IE0 or IE1) is automatically cleared by hardware upon entering the ISR.

When an external interrupt is generated from a level-triggered (low-level) source, the appropriate flag bit (IE0 or IE1) is NOT automatically cleared by hardware.

Table 41. P3SFS: Port 3 special function select register (SFR 91h, reset value 00h)

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| P3SFS7 | P3SFS6 | P3SFS5 | P3SFS4 | P3SFS3 | P3SFS2 | P3SFS1 | P3SFS0 |

Table 42. P3SFS register bit definition

| Port 3 Pin | R/W | Default port function | Alternate port function |
|---|---|---|---|
| | | P3SFS[i] - 0; Port 3 Pin, i = 0..7 | P3SFS[i] - 1; Port 3 Pin, i = 0..7 |
| 0 | R,W | GPIO | UART0 Receive, RXD0 |
| 1 | R,W | GPIO | UART0 Transmit, TXD0 |
| 2 | R,W | GPIO | Ext Intr 0/Timer 0 Gate, EXT0INT/TG0 |
| 3 | R,W | GPIO | Ext Intr 1/Timer 1 Gate, EXT1INT/TG1 |
| 4 | R,W | GPIO | Counter 0 Input, C0 |
| 5 | R,W | GPIO | Counter 0 Input, C1 |
| 6 | R,W | GPIO | $I^2C$ Data, I2CSDA |
| 7 | R,W | GPIO | $I^2C$ Clock, I2CCL |

Table 43. P1SFS0: Port 1 special function select 0 register (SFR 8Eh, reset value 00h)

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| P1SF07 | P1SF06 | P1SF05 | P1SF04 | P1SF03 | P1SF02 | P1SF01 | P1SF00 |

Table 44. P1SFS1: Port 1 special function select 1 register (SFR 8Fh, reset value 00h)

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| P1SF17 | P1SF16 | P1SF15 | P1SF14 | P1SF13 | P1SF12 | P1SF11 | P1SF10 |

Table 45. P1SFS0 and P1SFS1 details

| Port 1 Pin | R/W | Default port function | Alternate 1 port function | Alternate 2 port function |
|---|---|---|---|---|
| | | P1SFS0[i] = 0<br>P1SFS1[i] = x | P1SFS0[i] = 1<br>P1SFS1[i] = 0 | P1SFS0[i] = 1<br>P1SFS1[i] = 1 |
| | | Port 1 Pin, i = 0.. 7 | Port 1 Pin, i = 0.. 7 | Port 1 Pin, i = 0.. 7 |
| 0 | R,W | GPIO | Timer 2 count input, T2 | ADC Chn 0 input, ADC0 |
| 1 | R,W | GPIO | Timer 2 trigger input, TX2 | ADC Chn 1 input, ADC1 |
| 2 | R,W | GPIO | UART1 Receive, RXD1 | ADC Chn 2 input, ADC2 |
| 3 | R,W | GPIO | UART1 Transmit, TXD1 | ADC Chn 3 input, ADC3 |
| 4 | R,W | GPIO | SPI Clock, SPICLK | ADC Chn 4 input, ADC4 |
| 5 | R,W | GPIO | SPI Receive, SPIRXD | ADC Chn 5 input, ADC5 |

Table 50. **BUSCON register bit definition**

| Bit | Symbol | R/W | Definition |
|---|---|---|---|
| 7 | EPFQ | R,W | Enable pre-fetch queue<br>0 = PFQ is disabled<br>1 = PFQ is enabled (default) |
| 6 | EBC | R,W | Enable branch cache<br>0 = BC is disabled<br>1 = BC is enabled (default) |
| 5:4 | WRW[1:0] | R,W | $\overline{WR}$ Wait, number of MCU_CLK periods for $\overline{WR}$ write bus cycle during any MOVX instruction<br>00b: 4 clock periods<br>01b: 5 clock periods<br>10b: 6 clock periods (default)<br>11b: 7 clock periods |
| 3:2 | RDW[1:0] | R,W | $\overline{RD}$ Wait, number of MCU_CLK periods for $\overline{RD}$ read bus cycle during any MOVX instruction<br>00b: 4 clock periods<br>01b: 5 clock periods<br>10b: 6 clock periods (default)<br>11b: 7 clock periods |
| 1:0 | CW[1:0] | R,W | Code Wait, number of MCU_CLK periods for $\overline{PSEN}$ read bus cycle during any code byte fetch or during any MOVC code byte read instruction. Periods will increase with PFQ stall<br>00b: 3 clock periods - exception, for MOVC instructions this setting results 4 clock periods<br>01b: 4 clock periods<br>10b: 5 clock periods<br>11b: 6 clock periods (default) |

**Table 69. Commonly used baud rates generated from timer 1 (continued)**

| UART mode | $f_{OSC}$ MHz | Desired baud rate | Resultant baud rate | Baud rate deviation | SMOD bit in PCON | Timer 1 | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | C/$\overline{T}$ Bit in TMOD | Timer mode in TMOD | TH1 reload value (hex) |
| Modes 1 or 3 | 3.6864 | 19200 | 19200 | 0 | 1 | 0 | 2 | FF |
| Modes 1 or 3 | 3.6864 | 9600 | 9600 | 0 | 1 | 0 | 2 | FE |
| Modes 1 or 3 | 1.8432 | 9600 | 9600 | 0 | 1 | 0 | 2 | FF |
| Modes 1 or 3 | 1.8432 | 4800 | 4800 | 0 | 1 | 0 | 2 | FE |

## 21.4 More about UART mode 0

Refer to the block diagram in *Figure 30 on page 113*, and timing diagram in *Figure 31 on page 113*.

Transmission is initiated by any instruction which writes to the SFR named SBUF. At the end of a write operation to SBUF, a 1 is loaded into the 9th position of the transmit shift register and tells the TX Control unit to begin a transmission. Transmission begins on the following MCU machine cycle, when the “SEND” signal is active in *Figure 31*.

SEND enables the output of the shift register to the alternate function on the port containing pin RxD, and also enables the SHIFT CLOCK signal to the alternate function on the port containing the pin, TxD. At the end of each SHIFT CLOCK in which SEND is active, the contents of the transmit shift register are shifted to the right one position.

As data bits shift out to the right, zeros come in from the left. When the MSB of the data byte is at the output position of the shift register, then the '1' that was initially loaded into the 9th position, is just to the left of the MSB, and all positions to the left of that contain zeros. This condition flags the TX Control unit to do one last shift, then deactivate SEND, and then set the interrupt flag TI. Both of these actions occur at S1P1.

Reception is initiated by the condition REN = 1 and RI = 0. At the end of the next MCU machine cycle, the RX Control unit writes the bits 11111110 to the receive shift register, and in the next clock phase activates RECEIVE. RECEIVE enables the SHIFT CLOCK signal to the alternate function on the port containing the pin, TxD. Each pulse of SHIFT CLOCK moves the contents of the receive shift register one position to the left while RECEIVE is active. The value that comes in from the right is the value that was sampled at the RxD pin. As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the right-most position arrives at the left-most position in the shift register, it flags the RX Control unit to do one last shift, and then it loads SBUF. After this, RECEIVE is cleared, and the receive interrupt flag RI is set.

While transmitting, bytes are shifted out MSB first, and when receiving, bytes are shifted in MSB first, through the Acknowledge Bit register as shown in *Figure 42 on page 128*.

### 23.10.1 Bus wait condition

After the SIOE finishes receiving a byte in Receive mode, or transmitting a byte in Transmit mode, the INTR flag (in S1STA) is set and automatically a wait condition is imposed on the I²C bus (SCL held low by SIOE). In Transmit mode, this wait condition is released as soon as the MCU writes any byte to S1DAT. In Receive mode, the wait condition is released as soon as the MCU reads the S1DAT register.

This method allows the user to handle transmit and receive operations within an interrupt service routine. The SIOE will automatically stall the I²C bus at the appropriate time, giving the MCU time to get the next byte ready to transmit or time to read the byte that was just received.

**Table 80. S1DAT: I²C data shift register (SFR DEh, reset value 00h)**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| S1DAT[7:0] | | | | | | | |

**Table 81. S1DAT register bit definition**

| Bit | Symbol | R/W | Function |
|---|---|---|---|
| 7:0 | S1DAT[7:0] | R/W | Holds the data byte to be transmitted in Transmit mode, or it holds the data byte received in Receiver mode. |

## 23.11 I²C address register (S1ADR)

The S1ADR register (*Table 82*) holds the 7-bit device address used when the SIOE is operating as a Slave. When the SIOE receives an address from a Master, it will compare this address to the contents of S1ADR, as shown in *Figure 42 on page 128*.

If the 7 bits match, the INTR Interrupt flag (in S1STA) is set, and the ADDR Bit (in S1CON) is set. The SIOE cannot modify the contents S1ADR, and S1ADR is not used during Master mode.

**Table 82. S1ADR: I²C address register (SFR DFh, reset value 00h)**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| SLA6 | SLA5 | SLA4 | SLA3 | SLA2 | SLA1 | SLA0 | – |

**Table 83. S1ADR register bit definition**

| Bit | Symbol | R/W | Function |
|---|---|---|---|
| 7:1 | SLA[6:0] | R/W | Stores desired 7-bit device address, used when SIOE is in Slave mode. |
| 0 | – | – | Not used |

**Table 109. UIE2 register bit definition**

| Bit | Symbol | R/W | Definition |
|---|---|---|---|
| 7 | – | – | Reserved |
| 6 | – | – | Reserved |
| 5 | – | – | Reserved |
| 4 | OUT4IE | R/W | Enable Endpoint 4 OUT FIFO interrupt |
| 3 | OUT3IE | R/W | Enable Endpoint 3 OUT FIFO interrupt |
| 2 | OUT2IE | R/W | Enable Endpoint 2 OUT FIFO interrupt |
| 1 | OUT1IE | R/W | Enable Endpoint 1 OUT FIFO interrupt |
| 0 | OUT0IE | R/W | Enable Endpoint 0 OUT FIFO interrupt |

- USB IN FIFO NAK interrupt enable register (UIE3)

  When an endpoint's IN FIFO is empty and an IN transaction to that endpoint has been received, the SIE sends a NAK handshake token since there is no data ready for it to send.

  The UIE3 register (see *Table 110*) is used to enable each endpoint's IN FIFO NAK Interrupt.

**Table 110. USB IN FIFO NAK interrupt enable register (UIE3 0E7h, reset value 00h)**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | NAK4IE | NAK3IE | NAK2IE | NAK1IE | NAK0IE |

**Table 111. UIE3 register bit definition**

| Bit | Symbol | R/W | Definition |
|---|---|---|---|
| 7 | – | – | Reserved |
| 6 | – | – | Reserved |
| 5 | – | – | Reserved |
| 4 | NAK4IE | R/W | Enable Endpoint 4 IN FIFO NAK interrupt |
| 3 | NAK3IE | R/W | Enable Endpoint 3 IN FIFO NAK interrupt |
| 2 | NAK2IE | R/W | Enable Endpoint 2 IN FIFO NAK interrupt |
| 1 | NAK1IE | R/W | Enable Endpoint 1 IN FIFO NAK interrupt |
| 0 | NAK0IE | R/W | Enable Endpoint 0 IN FIFO NAK interrupt |

- USB FIFO base address high and low registers (UBASEH and UBASEL)

  All 10 Endpoint FIFOs share the same 64-byte address range. The 16-bit base address for the FIFOs is specified using the USB base address registers (see Table 130 and Table 132). The USB endpoint select register (see Table 124 on page 172) selects the direction and the Endpoint for the FIFO that is accessed when addressing the 64-bytes of XDATA space starting with the base address specified in the base address registers. The base address is a 64-byte segment where the lower 6 bits of the base register are hardwired to '0.'

  *Important note: The USB FIFO base address must be set to an open 64-byte segment in the XDATA space. Care should be taken to ensure that there is no overlap of addresses between the USB FIFOs and the flash memory, SRAM, csiop registers, and anything else accessed in the XDATA space. While the logic in the PSD module handles overlap of flash memory, SRAM, and the csiop registers with a fixed priority (see Section 28.1: PSD module functional description on page 192), this is not the case with the USB FIFOs. Unpredictable results as well as potential damage to the device may occur if there is an overlap of addresses.*

**Table 130. USB FIFO base address high register (UBASEH 0F3h, reset value 00h)**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| BASEADDR[15:8] | | | | | | | |

**Table 131. UBASEH register bit definition**

| Bit | Symbol | R/W | Definition |
|---|---|---|---|
| 7:0 | BASEADDR [15:8] | R/W | The upper 8 bits of the 16-bit base address for USB FIFOs to be mapped in XDATA space |

**Table 132. USB FIFO base address low register (UBASEL 0F4h, reset value 00h)**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| BASEADDR[7:6] | | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 133. UBASEL register bit definition**

| Bit | Symbol | R/W | Definition |
|---|---|---|---|
| 7:6 | BASEADDR [7:6] | R/W | Bits 7 and 6 of the 16-bit base address for the USB FIFOs to be mapped in XDATA space |
| 5:0 | BASEADDR [5:0] | R | Hardwired '0' |

Table 153. PCASTA register bit definition (continued)

| Bit | Symbol | Function |
|---|---|---|
| 5 | INTF4 | TCM4 Interrupt flag<br>Set by hardware when a match or capture event occurs.<br>Must be clear with software. |
| 4 | INTF3 | TCM3 Interrupt flag<br>Set by hardware when a match or capture event occurs.<br>Must be clear with software. |
| 3 | OVF0 | PCA0 Counter OverFlow flag<br>Set by hardware when the counter rolls over. OVF0 flags an interrupt if Bit EOVFI in PCACON0 is set. OVF1 may be set with either hardware or software but can only be cleared with software. |
| 2 | INTF2 | TCM2 Interrupt flag<br>Set by hardware when a match or capture event occurs.<br>Must be clear with software. |
| 1 | INTF1 | TCM1 Interrupt flag<br>Set by hardware when a match or capture event occurs.<br>Must be clear with software. |
| 0 | INTF0 | TCM0 Interrupt flag<br>Set by hardware when a match or capture event occurs.<br>Must be clear with software. |

## 27.13 TCM interrupts

There are 8 TCM interrupts: 6 match or capture interrupts and two counter overflow interrupts. The 8 interrupts are “ORed” as one PCA interrupt to the CPU.

By the nature of PCA application, it is unlikely that many of the interrupts occur simultaneously. If they do, the CPU has to read the interrupt flags and determine which one to serve. The software has to clear the interrupt flag in the status register after serving the interrupt.

Table 154. TCMMODE0 - TCMMODE5 (6 registers, reset value 00h)

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| EINTF | E_COMP | CAP_PE | CAP_NE | MATCH | TOGGLE | PWM[1:0] | |

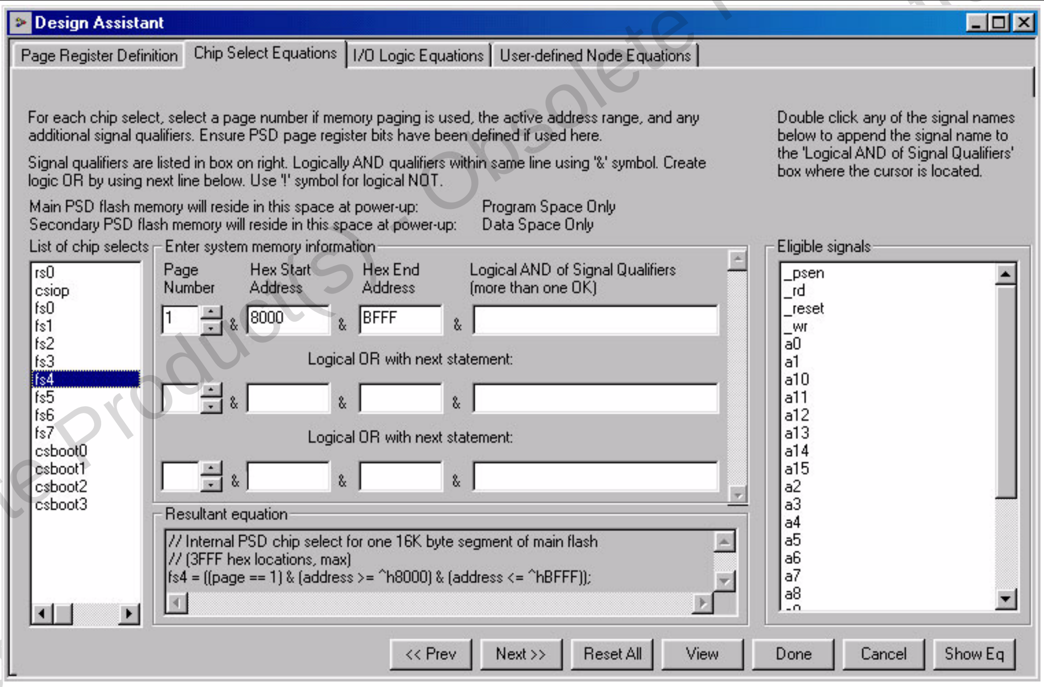
Table 155. TCMMODE0 - TCMMODE5 register bit definition

| Bit | Symbol | Function |
|---|---|---|
| 7 | EINTF | 1 - Enable the interrupt flags (INTF) in the status register to generate an interrupt. |
| 6 | E_COMP | 1 - Enable the comparator when set |
| 5 | CAP_PE | 1 - Enable Capture Mode, a positive edge on the CEXn pin. |
| 4 | CAP_NE | 1 - Enable Capture Mode, a negative edge on the CEXn pin. |
| 3 | MATCH | 1 - A match from the comparator sets the INTF bits in the status register. |

**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

This unlocking sequence is typical for many Flash memories to prevent accidental WRITEs by errant code. However, it is possible to bypass this unlocking sequence to save time while intentionally programming Flash memory.

*Important note: The 8032 may not read and execute code from the same Flash memory array for which it is directing an instruction sequence. Or more simply stated, the 8032 may not read code from the same Flash array that is writing or erasing. Instead, the 8032 must execute code from an alternate memory (like SRAM or a different Flash array) while sending instruction sequences to a given Flash array. Since the two Flash memory arrays inside the PSD module device are completely independent, the 8032 may read code from one array while sending instructions to the other. It is possible, however, to suspend a sector erase operation in one particular Flash array in order to access a different sector within that same Flash array, then resume the erase later.*

*After a Flash memory array is programmed or erased it will go to “Read Array” mode, then the 8032 can read from Flash memory just as it would read from any ROM or SRAM device.*

### 28.5.2 Flash memory instruction sequences

An instruction sequence consists of a sequence of specific byte WRITE and byte READ operations. Each byte written to either Flash memory array on the PSD module is received by a state machine inside the Flash array and sequentially decoded to execute an embedded algorithm. The algorithm is executed when the correct number of bytes are properly received and the time between two consecutive bytes is shorter than the time-out period of 80µs. Some instruction sequences are structured to include READ operations after the initial WRITE operations.

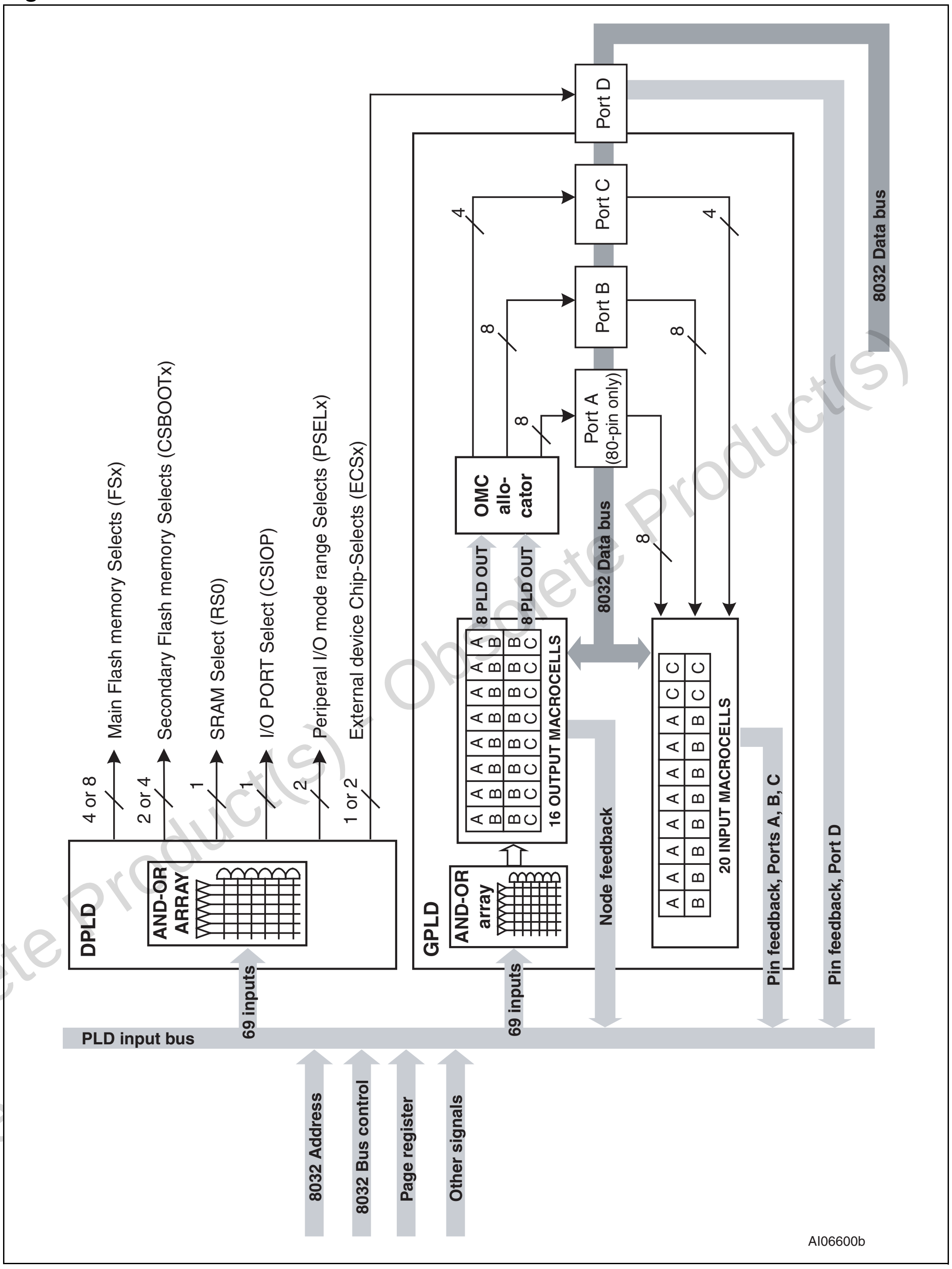
An instruction sequence must be followed exactly. Any invalid combination of instruction bytes or time-out between two consecutive bytes while addressing Flash memory resets the PSD module Flash logic into Read Array mode (where Flash memory is read like a ROM device). The Flash memories support instruction sequences summarized in *Table 163 on page 209*.

- Program a byte
- Unlock Sequence Bypass
- Erase memory by array or by sector
- Suspend or resume a sector erase
- Reset to Read Array mode

The first two bytes of an instruction sequence are 8032 bus WRITE operations to “unlock” the Flash array, followed by writing a command byte. The bus operations consist of writing the data AAh to address X555h during the first bus cycle and data 55h to address XAAAh during the second bus cycle. 8032 address signals A12-A15 are “Don’t care” during the instruction sequence during WRITE cycles. However, the appropriate sector select signal (*FSx or CSBOOTx*) from the DPLD must be active during the entire instruction sequence to complete the entire 8032 address (this includes the page number when memory paging is used). Ignoring A12-A15 means the user has more flexibility in memory mapping. For example, in many traditional Flash memories, instruction sequences must be written to addresses AAAAh and 5555h, not XAAAh and X555h like supported on the PSD module. When the user has to write to AAAAh and 5555h, the memory mapping options are limited.

The Main Flash and Secondary Flash memories each have the same instruction set shown in *Table 163 on page 209*, but the sector select signals determine which memory array will receive and execute the instructions.

**Figure 73. DPLD and GPLD**



### 28.5.27 Decode PLD (DPLD)

The DPLD (*Figure 74 on page 223*) generates the following memory decode signals:

- Eight Main Flash memory sector select signals (FS0-FS7) with three product terms each
- Four Secondary Flash memory sector select signals (CSBOOT0-CSBOOT3) with three product terms each
- One SRAM select signal (RS0) with two product terms
- One select signal for the base address of 256 PSD module device control and status registers (csiop) with one product term
- Two external chip-select output signals for Port D pins, each with one product term (52-pin devices only have one pin on Port D)
- Two chip-select signals (PSEL0, PSEL1) used to enable the 8032 data bus repeater function (Peripheral I/O mode) for Port A on 80-pin devices. Each has one product term.

A product term indicates the logical OR of two or more inputs. For example, three product terms in a DPLD output means the final output signal is capable of representing the logical OR of three different input signals, each input signal representing the logical AND of a combination of the 69 PLD inputs.

Using the signal FS0 for example, the user may create a 3-product term chip select signal that is logic true when any one of three different address ranges are true... FS0 = address range 1 OR address range 2 OR address range 3.

The phrase “one product term” is a bit misleading, but commonly used in this context. One product term is the logical AND of two or more inputs, with no OR logic involved at all, such as the CSIOP signal in *Figure 74 on page 223*.

### 28.5.50 Port C structure

Port C supports the following operating modes on pins PC2, PC3, PC4, PC7:

- MCU I/O Mode
- GPLD Output Mode from Output Macrocells MCELLBC2, MCELLBC3, MCELLBC4, MCELLBC7
- GPLD Input Mode to Input Macrocells IMCC2, IMCC3, IMCC4, IMCC7

See *Figure 86 on page 246* for detail.

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

- Pins PC3 and PC4 support TSTAT and $\overline{\text{TERR}}$ status indicators, to reduce the amount of time required for JTAG ISP programming. These two pins must be used together for this function, adding to the four standard JTAG signals. When TSTAT and $\overline{\text{TERR}}$ are used, it is referred to as “6-pin JTAG”. PC3 and PC4 cannot be used for other functions if they are used for 6-pin JTAG. See *Section 28.6.1: JTAG ISP and JTAG debug on page 257* for details.
- PC3 can be used as an output to indicate when a Flash memory program or erase operation has completed. This is specified in PSDsoft Express as *Section 28.5.13: Ready/Busy (PC3) on page 215*.

The remaining four pins (TDI, TDO, TCK, TMS) on Port C are dedicated to the JTAG function and cannot be used for any other function. See *Section 28.6.1: JTAG ISP and JTAG debug on page 257*.

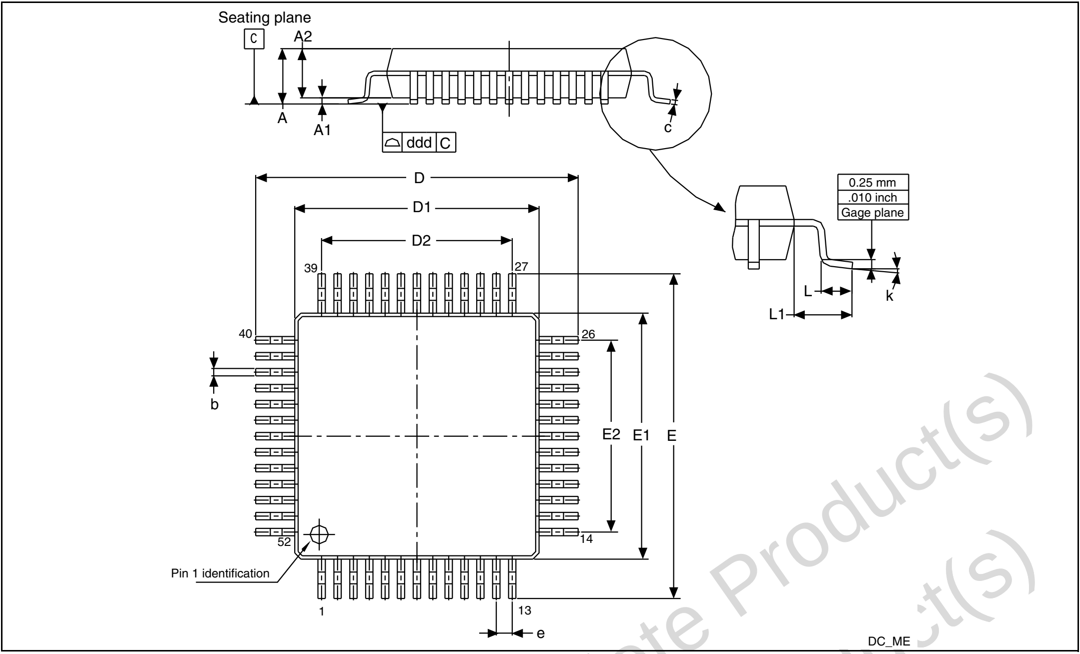
Port C also supports the Open Drain output drive type options on pins PC2, PC3, PC4, and PC7 using the csiop Drive Select registers.

Table 212. PSD module DC characteristics (with 5 V $V_{DD}$)

| Symbol | Parameter | | Test condition (in addition to those in *Table 211 on page 271*) | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| $V_{IH}$ | Input high voltage | | $4.5\ V < V_{DD} < 5.5\ V$ | 2 | | $V_{DD}$ +0.5 | V |
| $V_{IL}$ | Input low voltage | | $4.5\ V < V_{DD} < 5.5\ V$ | –0.5 | | 0.8 | V |
| $V_{LKO}$ | $V_{DD}$ (min) for Flash erase and program | | | 2.5 | | 4.2 | V |
| $V_{OL}$ | Output low voltage | | $I_{OL}$ = 20 µA, $V_{DD}$ = 4.5 V | | 0.01 | 0.1 | V |
| | | | $I_{OL}$ = 8 mA, $V_{DD}$ = 4.5 V | | 0.25 | 0.45 | V |
| $V_{OH}$ | Output high voltage | | $I_{OH}$ = –20 µA, $V_{DD}$ = 4.5 V | 4.4 | 4.49 | | V |
| | | | $I_{OH}$ = –2 mA, $V_{DD}$ = 4.5 V | 2.4 | 3.9 | | V |
| $I_{SB}$ | Standby supply current for power-down mode | | $\overline{CSI} > V_{DD} - 0.3\ V$ [(1)(2)] | | 120 | 250 | µA |
| $I_{LI}$ | Input leakage current | | $V_{SS} < V_{IN} < V_{DD}$ | –1 | ±0.1 | 1 | µA |
| $I_{LO}$ | Output leakage current | | $0.45 < V_{OUT} < V_{DD}$ | –10 | ±5 | 10 | µA |
| $I_{CC}$ (DC)[(3)] | Operating supply current | PLD only | PLD_TURBO = Off, f = 0 MHz [(3)] | | 0 | | µA/PT |
| | | | PLD_TURBO = On, f = 0 MHz | | 400 | 700 | µA/PT |
| | | Flash memory | During Flash memory WRITE/Erase Only | | 15 | 30 | mA |
| | | | Read only, f = 0 MHz | | 0 | 0 | mA |
| | | SRAM | f = 0 MHz | | 0 | 0 | mA |
| $I_{CC}$ (AC)[(3)] | PLD AC adder | | | | | (4) | |
| | Flash memory AC adder | | | | 1.5 | 2.5 | mA/MHz |
| | SRAM AC adder | | | | 1.5 | 3.0 | mA/MHz |

1. Internal Power-down mode is active.
2. PLD is in non-Turbo mode, and none of the inputs are switching.
3. $I_{OUT}$ = 0 mA.
4. Please see *Figure 95 on page 265* for the PLD current calculation.

**Figure 113. LQFP52 – 52-lead plastic thin, quad, flat package outline**



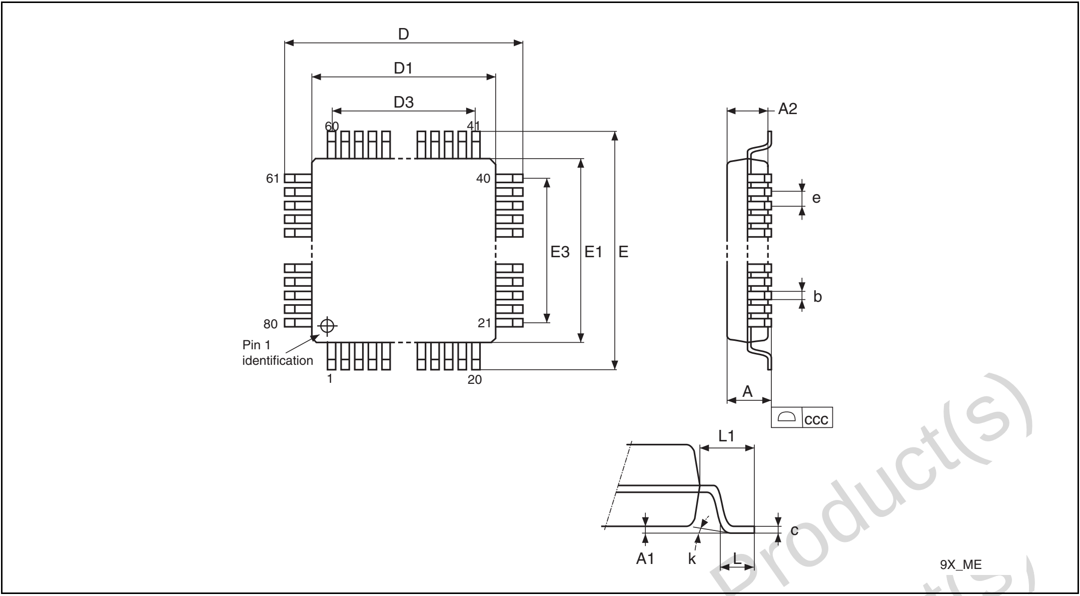
1. Drawing is not to scale.

**Table 237. LQFP52 – 52-lead plastic thin, quad, flat package mechanical data**

| Symbol | millimeters | | | inches[1] | | |
|---|---|---|---|---|---|---|
| | Typ | Min | Max | Typ | Min | Max |
| A | | | 1.60 | | | 0.063 |
| A1 | | 0.05 | 0.15 | | 0.002 | 0.0059 |
| A2 | | 1.35 | 1.45 | | 0.0531 | 0.0571 |
| b | | 0.22 | 0.38 | | 0.0087 | 0.015 |
| C | | 0.09 | 0.2 | | 0.0035 | 0.0079 |
| D | 12 | | | 0.4724 | | |
| D1 | 10 | | | 0.3937 | | |
| D2 | 7.8 | | | 0.3071 | | |
| E | 12 | | | 0.4724 | | |
| E1 | 10 | | | 0.3937 | | |
| E2 | 7.8 | | | 0.3071 | | |
| e | 0.65 | | | 0.0256 | | |
| L | | 0.45 | 0.75 | | 0.0177 | 0.0295 |
| L1 | 1 | | | 0.0394 | | |
| k | | 0° | 7° | | 0° | 7° |
| ddd | 0.100 | | | 0.0039 | | |

1. Values in inches are converted from mm and rounded to 4 decimal digits.

Figure 114. LQFP80 – 80-lead plastic thin, quad, flat package outline



1. Drawing is not to scale.

Table 238. LQFP80 – 80-lead plastic thin, quad, flat package mechanical data

| Symbol | millimeters | | | inches[(1)] | | |
|---|---|---|---|---|---|---|
| | Typ | Min | Max | Typ | Min | Max |
| A | | | 1.600 | | | 0.0630 |
| A1 | | 0.050 | 0.150 | | 0.0020 | 0.0059 |
| A2 | 1.400 | 1.350 | 1.450 | 0.0551 | 0.0531 | 0.0571 |
| b | 0.220 | 0.170 | 0.270 | 0.0087 | 0.0067 | 0.0106 |
| C | | 0.090 | 0.200 | | 0.0035 | 0.0079 |
| D | 14.000 | | | 0.5512 | | |
| D1 | 12.000 | | | 0.4724 | | |
| D3 | 9.500 | | | 0.3740 | | |
| E | 14.000 | | | 0.5512 | | |
| E1 | 12.000 | | | 0.4724 | | |
| E3 | 9.500 | | | 0.3740 | | |
| e | 0.500 | | | 0.0197 | | |
| L | 0.600 | 0.450 | 0.750 | 0.0236 | 0.0177 | 0.0295 |
| L1 | 1.000 | | | 0.0394 | | |
| k | | 0° | 7° | | 0° | 7° |
| ccc | 0.080 | | | 0.0031 | | |

1. Values in inches are converted from mm and rounded to 4 decimal digits.

### Workaround

Revision A and B - When a USB reset is detected, the USB SIE's registers must be initialized appropriately by the firmware. The 3400 USB firmware examples clear USB SIE's registers if USB reset is detected.

## 34.4 Data toggle

### Description

The data toggle bit is read only.

### Impact on application

The IN FIFO data toggle bit is controlled exclusively by the USB SIE; therefore, it is not possible to change the state of the data toggle bit by firmware.

### Workaround

Revision A - For cases where the data toggle bit must be reset, such as after a Clear Feature/Stall request, sending the subsequent data on that endpoint twice results in getting the data toggle bit back to the state that it should be.

Revision B - A change in silicon was made so that the data toggle bit is reset by disabling and then enabling the respective endpoint's FIFO.

## 34.5 USB FIFO accessibility

### Description

The USB FIFO is only accessible by firmware and not by a JTAG debugger.

### Impact on application

Using a JTAG debugger, it is not possible to view the USB FIFO's contents in a memory dump window.

### Workaround

Revision A and B - None identified at this time.

## 34.6 Erroneous resend of data packet

### Description

When a data packet is sent the respective IN FIFO busy bit is not automatically cleared by the USB SIE. This can cause a data packet to be erroneously resent to the host in response to an IN PID immediately after the first correct transmission of this data packet.

### Impact on application

Since the Data Toggle in the retransmitted data packet is toggled from when the data was first sent, the host will treat this packet as valid. If the identified workaround is not