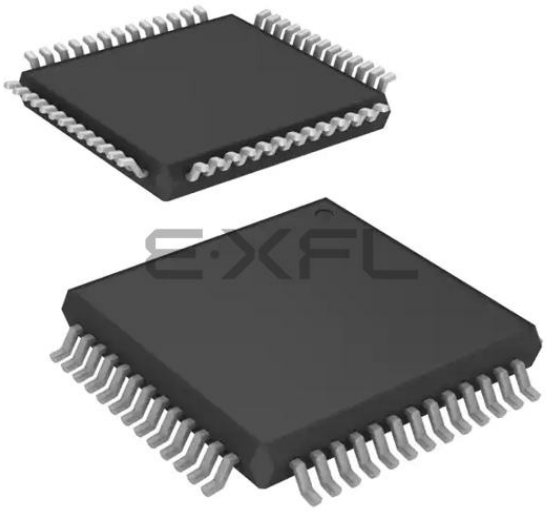


Welcome to E-XFL.COM

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"

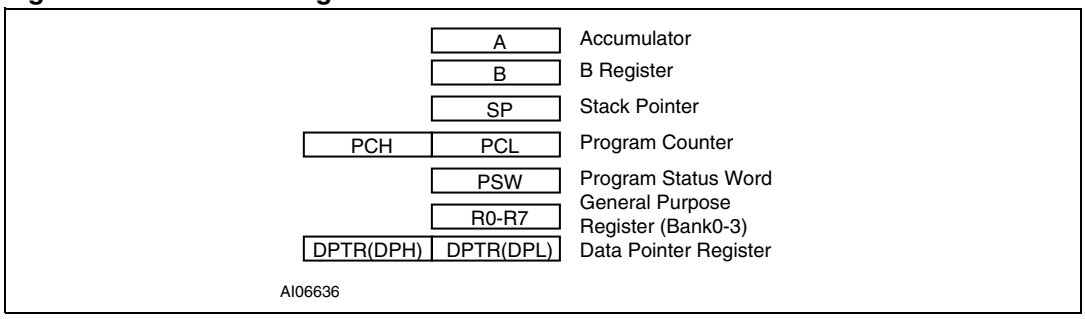
Details	
Product Status	Obsolete
Core Processor	8032
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	LVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	80KB (80K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data 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/upsd3312d-40t6

List of tables

7 8032 MCU registers

The UPSD33xx has the following 8032 MCU core registers, also shown in *Figure 10*.

Figure 10. 8032 MCU registers



7.1 Stack Pointer (SP)

The SP is an 8-bit register which holds the current location of the top of the stack. It is incremented before a value is pushed onto the stack, and decremented after a value is popped off the stack. The SP is initialized to 07h after reset. This causes the stack to begin at location 08h (top of stack). To avoid overlapping conflicts, the user must initialize the top of the stack to 20h if all four banks of registers R0 - R7 are used, and the user must initialize the top of stack to 30h if all of the 8032 bit memory locations are used.

7.2 Data Pointer (DPTR)

DPTR is a 16-bit register consisting of two 8-bit registers, DPL and DPH. The DPTR register is used as a base register to create an address for indirect jumps, table look-up operations, and for external data transfers (XDATA). When not used for addressing, the DPTR register can be used as a general purpose 16-bit data register.

Very frequently, the DPTR register is used to access XDATA using the External Direct addressing mode. The UPSD33xx has a special set of SFR registers (DPTC, DPTM) to control a secondary DPTR register to speed memory-to-memory XDATA transfers. Having dual DPTR registers allows rapid switching between source and destination addresses (see details in *Section 11: Dual data pointers on page 56*).

7.3 Program Counter (PC)

The PC is a 16-bit register consisting of two 8-bit registers, PCL and PCH. This counter indicates the address of the next instruction in program memory to be fetched and executed. A reset forces the PC to location 0000h, which is where the reset jump vector is stored.

8 Special function registers (SFR)

A group of registers designated as Special Function register (SFR) is shown in *Table 5 on page 41*. SFRs control the operating modes of the MCU core and also control the peripheral interfaces and I/O pins on the MCU module. The SFRs can be accessed only by using the Direct Addressing method within the address range from 80h to FFh of internal 8032 SRAM. Sixteen addresses in SFR address space are both byte- and bit-addressable. The bit-addressable SFRs are noted in *Table 5*.

86 of a possible 128 SFR addresses are occupied. The remaining unoccupied SFR addresses (designated as “RESERVED” in *Table 5*) should not be written. Reading unoccupied locations will return an undefined value.

Note: There is a separate set of control registers for the PSD module, designated as csiop, and they are described in the Section 27: PSD module on page 164. The I/O pins, PLD, and other functions on the PSD module are NOT controlled by SFRs.

SFRs are categorized as follows:

- MCU core registers: IP, A, B, PSW, SP, DPTL, DPTH, DPTC, DPTM
- MCU module I/O port registers: P1, P3, P4, P1SFS0, P1SFS1, P3SFS, P4SFS0, P4SFS1
- Standard 8032 timer registers: TCON, TMOD, T2CON, TH0, TH1, TH2, TL0, TL1, TL2, RCAP2L, RCAP2H
- Standard serial interfaces (UART): SCON0, SBUF0, SCON1, SBUF1
- Power, clock, and bus timing registers: PCON, CCON0, BUSCON
- Hardware watchdog timer registers: WDKEY, WDRST
- Interrupt system registers: IP, IPA, IE, IEA
- Program counter array (PCA) control registers: PCACL0, PCACH0, PCACON0, PCASTA, PCACL1, PCACH1, PCACON1, CCON2, CCON3
- PCA capture/compare and PWM registers

  CAPCOML0, CAPCOMH0, TCMMODE0, CAPCOML1, CAPCOMH1, TCMMODE2, CAPCOML2, CAPCOMH2, TCMMODE2, CAPCOML3, CAPCOMH3, TCMMODE3, CAPCOML4, CAPCOMH4, TCMMODE4, CAPCOML5, CAPCOMH5, TCMMODE5, PWMF0, PMWF1
- SPI interface registers: SPICLKD, SPISTAT, SPITDR, SPIRDR, SPICON0, SPICON1
- I^2C interface registers: S1SETUP, S1CON, S1STA, S1DAT, S1ADR
- Analog-to-digital converter registers: ACON, ADCPS, ADAT0, ADAT1
- IrDA interface register: IRDACON

```
MOVX @R0,A                    ; Move into the accumulator the
                               ; XDATA that is pointed to by
                               ; the address contained in R0.
```

9.7 Indexed addressing

This mode is used for the MOVC instruction which allows the 8032 to read a constant from program memory (not data memory). MOVC is often used to read look-up tables that are embedded in program memory. The final address produced by this mode is the result of adding either the 16-bit PC or DPTR value to the contents of the accumulator. The value in the accumulator is referred to as an index. The data fetched from the final location in program memory is stored into the accumulator, overwriting the index value that was previously stored there. For example:

```
MOVC A, @A+DPTR               ; Move code byte relative to
                               ; DPTR into accumulator
MOVC A, @A+PC                 ; Move code byte relative to PC
                               ; into accumulator
```

9.8 Relative addressing

This mode will add the two's-compliment number stored in the second byte of the instruction to the program counter for short jumps within +128 or –127 addresses relative to the program counter. This is commonly used for looping and is very efficient since no additional bus cycle is needed to fetch the jump destination address. For example:

```
SJMP 34h                      ; Jump 34h bytes ahead (in program
                               ; memory) of the address at which
                               ; the SJMP instruction is stored. If
                               ; SJMP is at 1000h, program
                               ; execution jumps to 1034h.
```

9.9 Absolute addressing

This mode will append the 5 high-order bits of the address of the next instruction to the 11 low-order bits of an ACALL or AJUMP instruction to produce a 16-bit jump address. The jump will be within the same 2 Kbyte page of program memory as the first byte of the following instruction. For example:

```
AJMP 0500h                    ; If next instruction is located at
                               ; address 4000h, the resulting jump
                               ; will be made to 4500h.
```

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 18. 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)
Reserved	8	0033h	–	–	–	–
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)

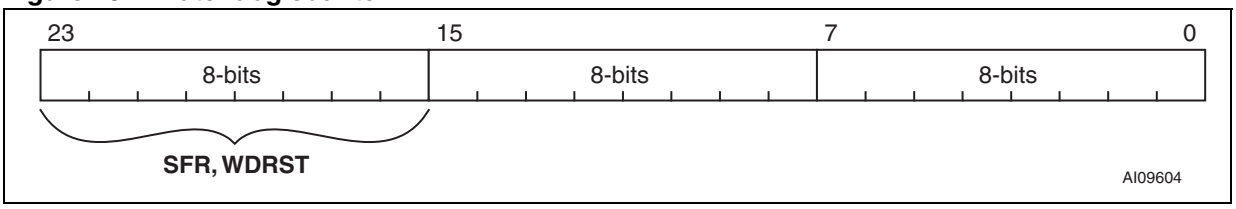
Table 32. PCON register bit definition (continued)

Bit	Symbol	R/W	Function
2	TCLK1	R,W	**Transmit Clock Flag (UART1)** (See *Table 58 on page 100* for flag description)
1	PD	R,W	**Activate Power-down mode** 0 = Not in Power-down mode 1 = Enter Power-down mode
0	IDL	R,W	**Activate Idle mode** 0 = Not in Idle mode 1 = Enter Idle mode

To prevent the WDT from timing out and generating a reset, firmware must repeatedly write some value to WDRST before the count reaches FFFFFh. Whenever WDRST is written, the upper 8 bits of the 24-bit counter are loaded with the written value, and the lower 16 bits of the counter are cleared to 0000h.

The WDT timeout period can be adjusted by writing a value other that 00h to WDRST. For example, if WDRST is written with 04h, then the WDT will start counting 040000h, 040001h, 040002h, and so on for each MCU machine cycle. In this example, the WDT timeout period is shorter than if WDRST was written with 00h, because the WDT is an up-counter. A value for WDRST should never be written that results in a WDT timeout period shorter than the time required to complete the longest code task in the application, else unwanted WDT overflows will occur.

Figure 20. Watchdog counter



The formula to determine WDT timeout period is:

$WDT_{PERIOD} = t_{MACH\_CYC} \times N_{OVERFLOW}$

$N_{OVERFLOW}$ is the number of WDT up-counts required to reach FFFFFFh. This is determined by the value written to the SFR, WDRST.

t_{MACH_CYC} is the average duration of one MCU machine cycle. By default, an MCU machine cycle is always 4 MCU_CLK periods for UPSD33xx, but the following factors can sometimes add more MCU_CLK periods per machine cycle:

- The number of MCU_CLK periods assigned to MCU memory bus cycles as determined in the SFR, BUSCON. If this setting is greater than 4, then machine cycles have additional MCU_CLK periods during memory transfers.
- Whether or not the PFQ/BC circuitry issues a stall during a particular MCU machine cycle. A stall adds more MCU_CLK periods to a machine cycle until the stall is removed.

t_{MACH_CYC} is also affected by the absolute time of a single MCU_CLK period. This number is fixed by the following factors:

Note: Frequency of the external crystal, resonator, or oscillator: (f_{OSC})

Note: Bit settings in the SFR CCON0, which can divide f_{OSC} and change MCU_CLK

As an example, assume the following:

1. f_{OSC} is 40 MHz, thus its period is 25ns.
2. CCON0 is 10h, meaning no clock division, so the period of MCU_CLK is also 25ns.
3. BUSCON is C1h, meaning the PFQ and BC are enabled, and each MCU memory bus cycle is 4 MCU_CLK periods, adding no additional MCU_CLK periods to MCU machine cycles during memory transfers.
4. Assume there are no stalls from the PFQ/BC. In reality, there are occasional stalls but their occurrence has minimal impact on WDT timeout period.
5. WDRST contains 00h, meaning a full 2^{24} up-counts are required to reach FFFFFh and generate a reset.

addressed leave their SM2 bits set and go on about their business, ignoring the coming data bytes.

SM2 has no effect in mode 0, and in mode 1, SM2 can be used to check the validity of the stop bit. In a mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

21.2 Serial port control registers

The SFR SCON0 controls UART0, and SCON1 controls UART1, shown in *Table 63* and *Table 65 on page 109*. These registers contain not only the mode selection bits, but also the 9th data bit for transmit and receive (bits TB8 and RB8), and the UART Interrupt flags, TI and RI.

Table 63. SCON0: Serial Port UART0 Control register (SFR 98h, reset value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Table 64. SCON0 register bit definition

Bit	Symbol	R/W	Definition
7	SM0	R,W	**Serial Mode Select**, See *Table 62 on page 107*. Important, notice bit order of SM0 and SM1. **[SM0:SM1] = 00b**, mode 0 **[SM0:SM1] = 01b**, mode mode 1 **[SM0:SM1] = 10b**, mode 2 **[SM0:SM1] = 11b**, mode 3
6	SM1	R,W	
5	SM2	R,W	**Serial Multiprocessor Communication Enable.** **Mode 0**: SM2 has no effect but should remain 0. **Mode 1**: If SM2 = 0 then stop bit ignored. SM2 =1 then RI active if stop bit = 1. **Mode 2 and 3**: Multiprocessor Comm Enable. If SM2=0, 9th bit is ignored. If SM2=1, RI active when 9th bit = 1.
4	REN	R,W	**Receive Enable.** If REN=0, UART reception disabled. If REN=1, reception is enabled
3	TB8	R,W	TB8 is assigned to the 9th transmission bit in mode 2 and 3. Not used in mode 0 and 1.
2	RB8	R,W	**Mode 0**: RB8 is not used. **Mode 1**: If SM2 = 0, the RB8 is the level of the received stop bit. **Mode 2 and 3**: RB8 is the 9th data bit that was received in mode 2 and 3.

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
```

27.2.3 Specifying the memory map with PSDsoft Express

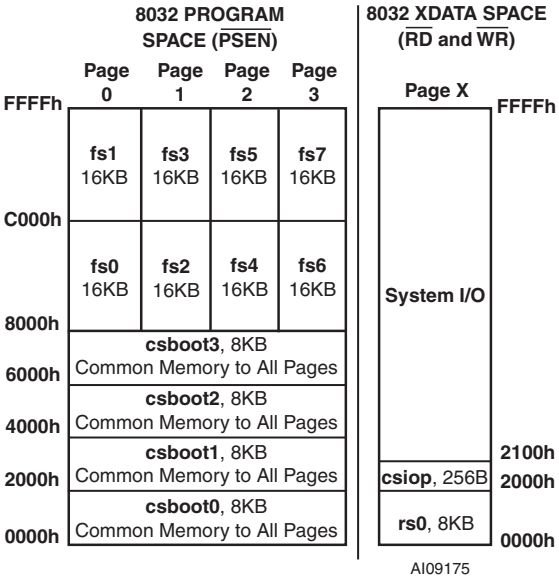
The memory map example shown in *Figure 52* is implemented using PSDsoft Express in a point-and-click environment. PSDsoft Express will automatically generate Hardware Definition Language (HDL) statements of the ABEL language for the DPLD, such as those shown in *Table 114 on page 172*.

Specifying these equations using PSDsoft Express is very simple. For example, *Figure 53 on page 173*, shows how to specify the chip-select equation for the 16 Kbyte Flash memory segment, fs4. Notice fs4 is on memory page 1. This specification process is repeated for all other Flash memory segments, the SRAM, the csiop register block, and any external chip select signals that may be needed.

Table 114. HDL statement example generated from PSDsoft for memory map

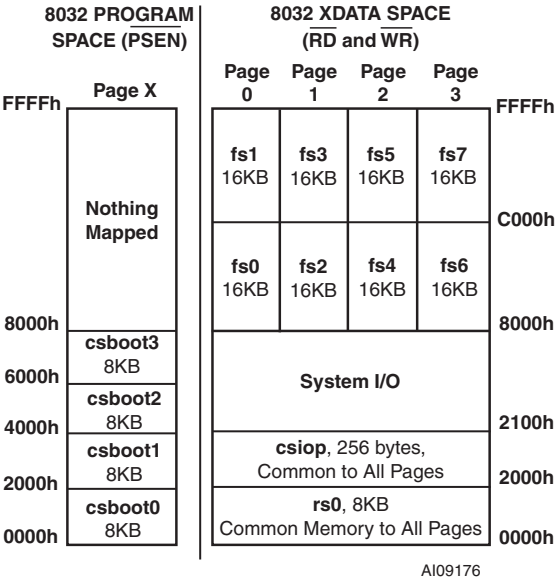
```
rs0 =    ((address ≥ ^h0000)    &    (address ≤ ^h1FFF));
csiop =  ((address ≥ ^h2000)    &    (address ≤ ^h20FF));
fs0 =    ((address ≥ ^h0000)    &    (address ≤ ^h3FFF));
fs1 =    ((address ≥ ^h4000)    &    (address ≤ ^h7FFF));
fs2 =    ((page == 0)    &    (address ≥ ^h8000)    &    (address ≤ ^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 55. Mapping: all Flash in code space



- *Figure 56 on page 175* Place the larger main Flash memory into XDATA space and the smaller secondary Flash into program space for systems that need a large amount of Flash for data recording or large look-up tables, and not so much Flash for 8032 firmware.

Figure 56. Mapping: small code / big data



It is also possible to “reclassify” the Flash memories during runtime, moving the memories between XDATA memory space and program memory space on-the-fly. This essentially means that the user can override the initial setting during run-time by writing to a csiop register (the VM register). This is useful for IAP, because standard 8051 architecture does not allow writing to program space. For example, if the user wants to update firmware in main Flash memory that is residing in program space, the user can temporarily “reclassify” the main Flash memory into XDATA space to erase and rewrite it while executing IAP code

accessible for a new READ or WRITE operation. The operation is finished when two successive READs yield the same value for DQ6.

DQ6 may also be used to indicate when an erase operation has completed. During an erase operation, DQ6 will toggle from '0' to '1' and '1' to ’0’ until the erase operation is complete, then DQ6 stops toggling. The erase is finished when two successive READs yield the same value of DQ6. The correct sector select signal, FSx or CSBOOTx, must be active during the entire procedure.

DQ6 is valid after the fourth instruction byte WRITE operation (for program instruction sequence) or after the sixth instruction byte WRITE operation (for erase instruction sequence).

If all the Flash memory sectors selected for erasure are protected, DQ6 toggles to ’0’ for about 100µs, then returns value of D6 of the previously addressed byte.

Error Flag (DQ5)

During a normal program or erase operation, the Error Flag Bit (DQ5) is to ’0’. This bit is set to ’1’ when there is a failure during Flash memory byte program, sector erase, or bulk erase operations.

In the case of Flash memory programming, DQ5 Bit indicates an attempt to program a Flash memory bit from the programmed state of 0, to the erased state of 1, which is not valid. DQ5 may also indicate a particular Flash cell is damaged and cannot be programmed.

In case of an error in a Flash memory sector erase or byte program operation, the Flash memory sector in which the error occurred or to which the programmed byte belongs must no longer be used. Other Flash memory sectors may still be used. DQ5 is reset after a Reset Flash instruction sequence.

27.4.8 Erase timeout flag (DQ3)

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

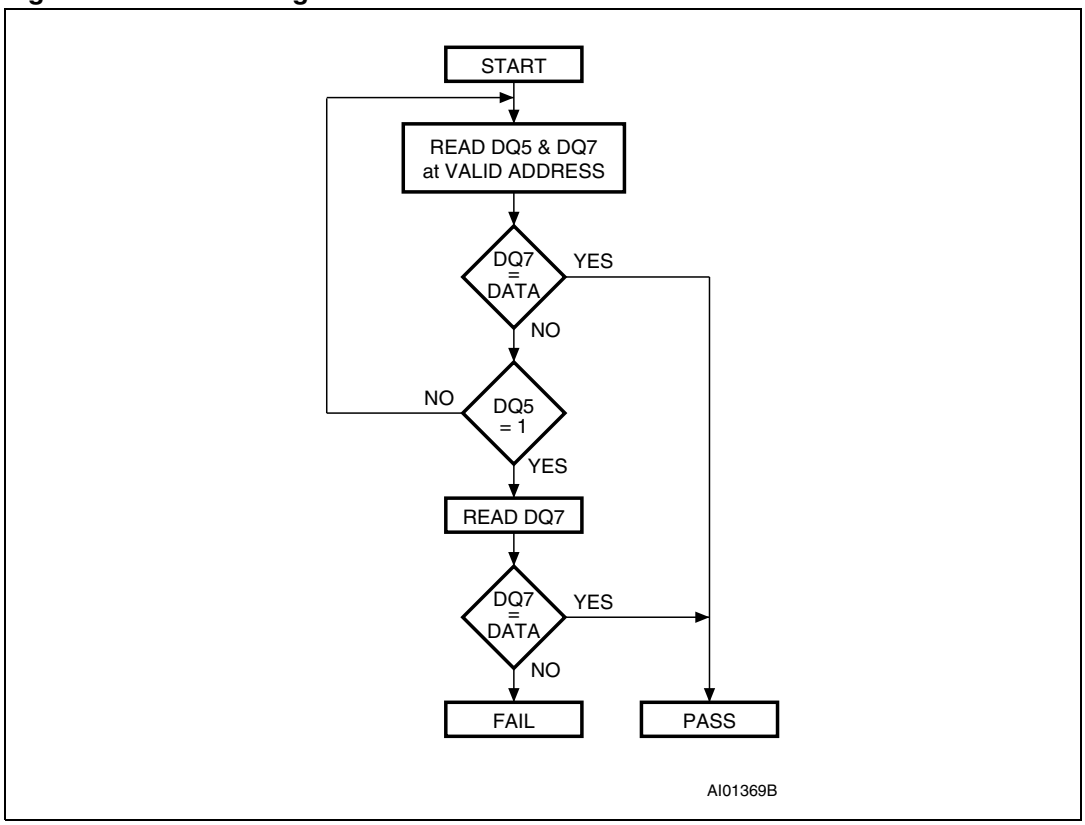
27.4.9 Programming Flash memory

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

When using the Data Polling method during an erase operation, *Figure 60 on page 188* still applies. However, the Data Polling Flag Bit (DQ7) is '0' until the erase operation is complete. A ’1’ on the Error Flag Bit (DQ5) indicates a timeout condition on the Erase cycle, a ’0’ indicates no error. The 8032 can read any location within the sector being erased to get the Data Polling Flag Bit (DQ7) and the Error Flag Bit (DQ5).

PSDsoft Express generates ANSI C code functions for implementation of these Data Polling algorithms.

Figure 60. Data Polling flowchart



27.4.11 Data toggle

Checking the Toggle Flag Bit (DQ6) is another method of determining whether a program or erase operation is in progress or has completed. *Figure 61 on page 189* shows the Data Toggle algorithm.

When the 8032 issues a program instruction sequence, the embedded algorithm within the Flash memory array begins. The 8032 then reads the location of the byte to be programmed in Flash memory to check status. The Toggle Flag Bit (DQ6) of this location toggles each time the 8032 reads this location until the embedded algorithm is complete. The 8032 continues to read this location, checking the Toggle Flag Bit (DQ6) and monitoring the Error Flag Bit (DQ5). When the Toggle Flag Bit (DQ6) stops toggling (two consecutive reads yield the same value), then the embedded algorithm is complete. If the Error Flag Bit (DQ5) is '1,' the 8032 should test the Toggle Flag Bit (DQ6) again, since the Toggle Flag Bit (DQ6) may have changed simultaneously with the Error Flag Bit (DQ5) (see *Figure 61 on page 189*).

27.4.26 Decode PLD (DPLD)

The DPLD (*Figure 63 on page 197*) 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 63 on page 197*.

Table 122. 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
MCELLBC3	Port B3 or C3	4	5	D3
MCELLBC4	Port B4 or C4	4	6	D4
MCELLBC5	Port B5	4	6	D5
MCELLBC6	Port B6	4	6	D6
MCELLBC7	Port B7 orC7	4	6	D7

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

27.4.31 Loading and reading OMCs

Each of the two OMC groups (eight OMCs each) occupies a byte in csiop space, named MCELLAB and MCELLBC (see *Table 123* and *Table 124 on page 204*). When the 8032 writes or reads these two OMC registers in csiop it is accessing each of the OMCs through it's 8-bit data bus, with the bit assignment shown in *Table 122 on page 203*. Sometimes it is important to know the bit assignment when the user builds GPLD logic that is accessed by the 8032. For example, the user may create a 4-bit counter that must be loaded and read by the 8032, so the user must know which nibble in the corresponding csiop OMC register the firmware must access. The fitter report generated by PSDsoft Express will indicate how it assigned the OMCs and data bus bits to the logic. The user can optionally force PSDsoft Express to assign logic to specific OMCs and data bus bits if desired by using the 'PROPERTY' statement in PSDsoft Express. Please see the PSDsoft Express User's Manual for more information on OMC assignments.

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

27.4.51 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, which puts the memories into standby mode (~ zero DC current) when 8032 address signals are not changing. As soon as a transition occurs on any address input, the affected 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 it’s lowest current consumption levels. If enabled, the APD counter will timeout when there is a lack of 8032 bus activity for an extended amount of time (8032 asleep). After timeout 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.

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

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.

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 memories, placing them 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 84 on page 242* (5 V) and *Figure 85 on page 243* (3.3 V).

Blocking bits

Significant power savings can be achieved by blocking 8032 bus control signals ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, 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 csiop PMMR registers. Current consumption of the PLDs is directly related to the composite frequency of all transitions on PLD inputs, so blocking certain PLD inputs can significantly lower PLD operating frequency and power consumption (resulting in a lower frequency on the graphs of *Figure 84 on page 242* and *Figure 85 on page 243*).

Note: It is recommended to prevent unused inputs from floating on Ports A, B, C, and D by pulling them up to V_{DD} with a weak external resistor (100 KΩ), or by setting the csiop Direction register to "output" at run-time for all unused inputs. This will prevent the CMOS input buffers of unused input pins from drawing excessive current.

The csiop PMMR register definitions are shown in *Table 154* through *Table 156 on page 226*.

Table 154. Power Management Mode register PMMR0 (address = csiop + offset B0h)(1)

Bit num.	Bit name	Value	Description
Bit 0	X	0	Not used, and should be set to zero.
Bit 1	APD Enable	0	Automatic Power-down (APD) counter is disabled.
		1	APD counter is enabled
Bit 2	X	0	Not used, and should be set to zero.
Bit 3	PLD Turbo Disable	0 = on	PLD Turbo mode is on
		1 = off	PLD Turbo mode is off, saving power.
Bit 4	Blocking bit, CLKIN to PLDs(2)	0 = on	CLKIN (pin PD1) to the PLD Input Bus is not blocked. Every transition of CLKIN powers-up the PLDs.
		1 = off	CLKIN input to PLD Input Bus is blocked, saving power. But CLKIN still goes to APD counter.
Bit 5	Blocking bit, CLKIN to OMCs only(2)	0 = on	CLKIN input is not blocked from reaching all OMC's common clock inputs.
		1 = off	CLKIN input to common clock of all OMCs is blocked, saving power. But CLKIN still goes to APD counter and all PLD logic besides the common clock input on OMCs.
Bit 6	X	0	Not used, and should be set to zero.
Bit 7	X	0	Not used, and should be set to zero.

1. All the bits of this register are cleared to zero following Power-up. Subsequent Reset ($\overline{RST}$) pulses do not clear the registers.
2. Blocking bits should be set to logic '1' only if the signal is not needed in a DPLD or GPLD logic equation.

27.4.53 Forced Power-down (FDP)

An alternative to APD is FPD. The resulting power-savings is the same, but the PDN signal in *Figure 77 on page 229* is set and Power-down mode is entered immediately when firmware sets the FORCE_PD Bit to logic '1' in the csiop *register* PMMR3 (Bit 1). FPD will override APD counter activity when FORCE_PD is set. No external clock source for the APD counter is needed. The FORCE_PD Bit is cleared only by a reset condition.

Caution must be used when implementing FPD because code memory goes off-line as soon as PSD module Power-down mode is entered, leaving the MCU with no instruction stream to execute.

The MCU module must put itself into Power-down mode after it puts the PSD module into Power-down mode. How can it do this if code memory goes off-line? The answer is the Pre-Fetch Queue (PFQ) in the MCU module. By using the instruction scheme shown in the 8051 assembly code example in *Table 157 on page 228*, 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.

Table 157. 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 93. Input macrocell timing (product term clock)

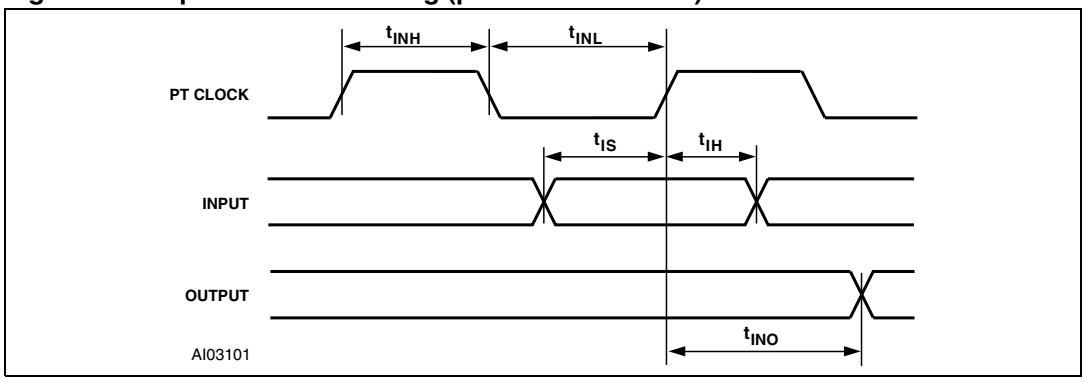


Table 180. Input macrocell timing (5 V PSD module)

Symbol	Parameter	Conditions	Min	Max	PT Aloc	Turbo Off	Unit
t_{IS}	Input setup time	(1)	0				ns
t_{IH}	Input hold time	(1)	15			+ 10	ns
t_{INH}	NIB input high time	(1)	9				ns
t_{INL}	NIB input low time	(1)	9				ns
t_{INO}	NIB input to combinatorial delay	(1)		34	+ 2	+ 10	ns

1. Inputs from Port A, B, and C relative to register/ latch clock from the PLD. ALE/AS latch timings refer to t_{AVLX} and t_{LXAX}.

Table 181. Input macrocell timing (3V PSD module)

Symbol	Parameter	Conditions	Min	Max	PT Aloc	Turbo Off	Unit
t_{IS}	Input setup time	(1)	0				ns
t_{IH}	Input hold time	(1)	25			+ 15	ns
t_{INH}	NIB input high time	(1)	12				ns
t_{INL}	NIB input low time	(1)	12				ns
t_{INO}	NIB input to combinatorial delay	(1)		43	+ 4	+ 15	ns

1. Inputs from Port A, B, and C relative to register/ latch clock from the PLD. ALE/AS latch timings refer to t_{AVLX} and t_{LXAX}.