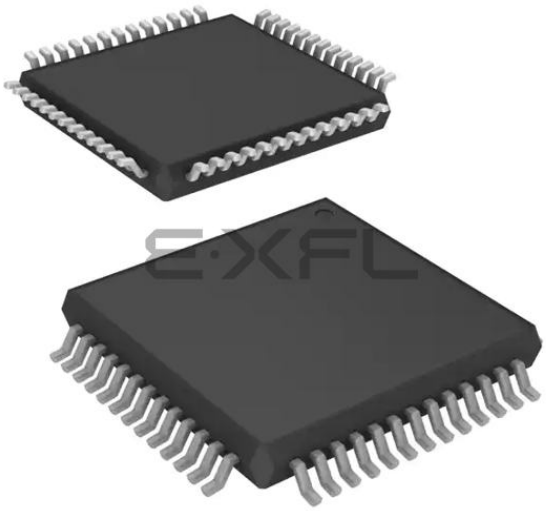


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

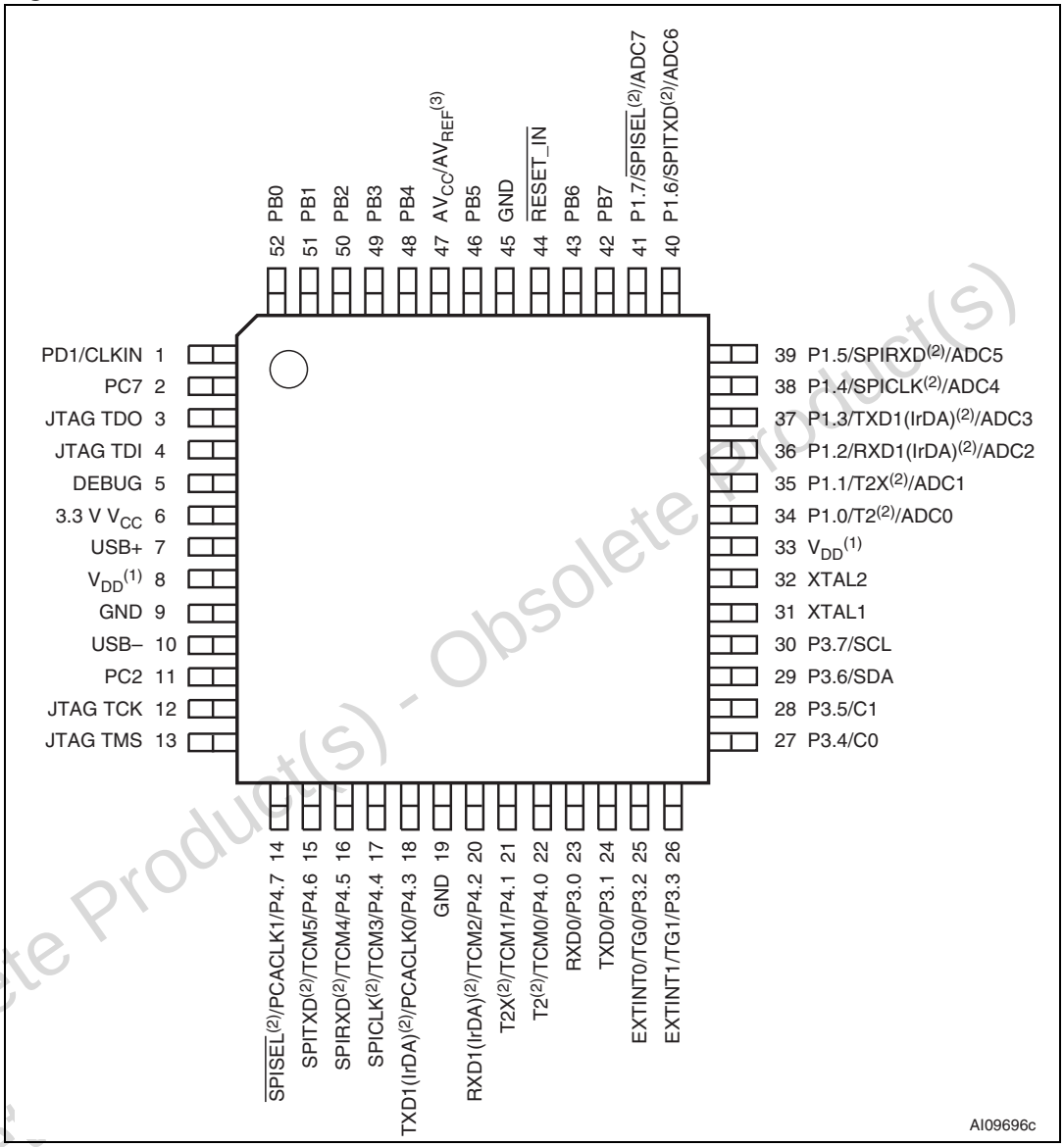
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	8032
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, IrDA, SPI, UART/USART, USB
Peripherals	LVD, POR, PWM, WDT
Number of I/O	35
Program Memory Size	288KB (288K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	32K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	52-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/upsd3454evb40t6

Figure 49. USB packets in a USB transfer example . . . 153
Figure 50. IN and OUT bulk transfers . . . 154
Figure 51. Interrupt transfer . . . 155
Figure 52. Control transfer . . . 156
Figure 53. FIFOs with no pairing . . . 158
Figure 54. FIFO pairing example (1/2 IN paired and 3/4 OUT paired) . . . 159
Figure 55. Typical self powered example . . . 177
Figure 56. 10-bit ADC . . . 179
Figure 57. PCA0 block diagram . . . 181
Figure 58. Timer mode . . . 184
Figure 59. PWM mode - (x8), fixed frequency . . . 185
Figure 60. PWM mode - (x8) programmable frequency . . . 186
Figure 61. PSD module block diagram . . . 191
Figure 62. Memory page register . . . 194
Figure 63. Typical system memory map . . . 198
Figure 64. PSDsoft express memory mapping . . . 199
Figure 65. Mapping: split second Flash in half . . . 200
Figure 66. Mapping: all Flash in code space . . . 201
Figure 67. Mapping: small code / big data . . . 201
Figure 68. PSD module memory priority . . . 202
Figure 69. VM register control of memories . . . 204
Figure 70. VM register example corresponding to memory map example . . . 204
Figure 71. Data polling flowchart . . . 214
Figure 72. Data toggle flowchart . . . 215
Figure 73. DPLD and GPLD . . . 221
Figure 74. DPLD logic array . . . 223
Figure 75. GPLD: one OMC, one IMC, and one I/O port (typical pin, port A, B, or C) . . . 224
Figure 76. Detail of a single OMC . . . 226
Figure 77. OMC allocator . . . 227
Figure 78. Detail of a single IMC . . . 230
Figure 79. Detail of a single I/O port (typical of ports A, B, C) . . . 232
Figure 80. Simple PLD logic example . . . 237
Figure 81. Pin declarations in PSDsoft express for simple PLD example . . . 237
Figure 82. Using the design assistant in PSDsoft Express for simple PLD example . . . 238
Figure 83. Peripheral I/O mode . . . 239
Figure 84. Port A structure . . . 243
Figure 85. Port B structure . . . 244
Figure 86. Port C structure . . . 246
Figure 87. Port D structure . . . 247
Figure 88. Automatic power-down (APD) unit . . . 252
Figure 89. Power-down mode flowchart . . . 253
Figure 90. JTAG chain in UPSD34xx package . . . 258
Figure 91. Recommended 4-pin JTAG connections . . . 259
Figure 92. Recommended 6-pin JTAG connections . . . 261
Figure 93. Recommended JTAG connector . . . 262
Figure 94. Example of chaining UPSD34xx devices . . . 263
Figure 95. PLD ICC / frequency consumption (5 V range) . . . 265
Figure 96. PLD ICC / frequency consumption (3 V range) . . . 266
Figure 97. Switching waveforms – key . . . 270
Figure 98. External READ cycle (80-pin device only) . . . 275
Figure 99. External WRITE cycle (80-pin device only) . . . 276
Figure 100. Input to output disable / enable . . . 278

2 Pin descriptions

Figure 2. LQFP52 connections



1. 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.
2. These signals can be used on one of two different ports (Port 1 or Port 4) for flexibility. Default is Port1.
3. AV_{REF} and 3.3 V AV_{CC} are shared in the 52-pin package only. ADC channels must use 3.3 V as AV_{REF} for the 52-pin package.

Table 2. Pin definitions (continued)

Port pin	Signal name	80-pin No.	52-pin No.(1)	In/out	Function		
					Basic	Alternate 1	Alternate 2
PC7		5	2	I/O	General I/O port pin		PLD, Macrocell output, or PLD input
PD1	CLKIN	3	1	I/O	General I/O port pin		– PLD I/O – Clock input to PLD and APD
PD2	CSI	1	N/A	I/O	General I/O port pin		– PLD I/O – Chip select ot PSD module
USB+		11	7	I/O	USB D+ pin; 1.5 kΩ pull-up resistor is required.		
USB–		14	10	I/O	USB D– pin		
3.3 V-V_{CC}		10	6		V_{CC} - MCU module. connect AV_{CC} to V_{CC} if the ADC is not used.		
AV_{CC}		72	47		Analog VCC Input		
V_{DD} 3.3 V or 5 V		12	8		V_{DD} - PSD module V_{DD} - 3.3 V for 3 V V_{DD} - 5 V for 5 V		
V_{DD} 3.3 V or 5 V		50	33		V_{DD} - PSD module V_{DD} - 3.3 V for 3 V V_{DD} - 5 V for 5 V		
GND		13	9				
GND		29	19				
GND		69	45				
NC		51	N/A				
NC		53	N/A				
NC		55	N/A				
NC		57	N/A				

1. N/A = Signal not available on 52-pin package.

4 Memory organization

The 8032 MCU core views memory on the MCU module as “internal” memory and it views memory on the PSD module as “external” memory, see *Figure 5*

Internal memory on the MCU module consists of DATA, IDATA, and SFRs. These standard 8032 memories reside in 384 bytes of SRAM located at a fixed address space starting at address 0000h.

External memory on the PSD module consists of four types: main Flash (64 Kbyte, 128 Kbyte, or 256 Kbyte), a smaller secondary Flash (32 Kbyte), SRAM (4 Kbyte, 8 Kbyte or 32 Kbyte), and a block of PSD module control registers called csiop (256 bytes). These external memories reside at programmable address ranges, specified using the software tool PSDsoft Express. See the PSD module section of this document for more details on these memories.

External memory is accessed by the 8032 in two separate 64 Kbyte address spaces. One address space is for program memory and the other address space is for data memory. Program memory is accessed using the 8032 signal, $\overline{PSEN}$. Data memory is accessed using the 8032 signals, $\overline{RD}$ and $\overline{WR}$. If the 8032 needs to access more than 64 Kbytes of external program or data memory, it must use paging (or banking) techniques provided by the page register in the PSD module.

Note: When referencing program and data memory spaces, it has nothing to do with 8032 internal SRAM areas of DATA, IDATA, and SFR on the MCU module. Program and data memory spaces only relate to the external memories on the PSD module.

External memory on the PSD module can overlap the internal SRAM memory on the MCU module in the same physical address range (starting at 0000h) without interference because the 8032 core does not assert the $\overline{RD}$ or $\overline{WR}$ signals when accessing internal SRAM.

Figure 5. UPSD34xx memories

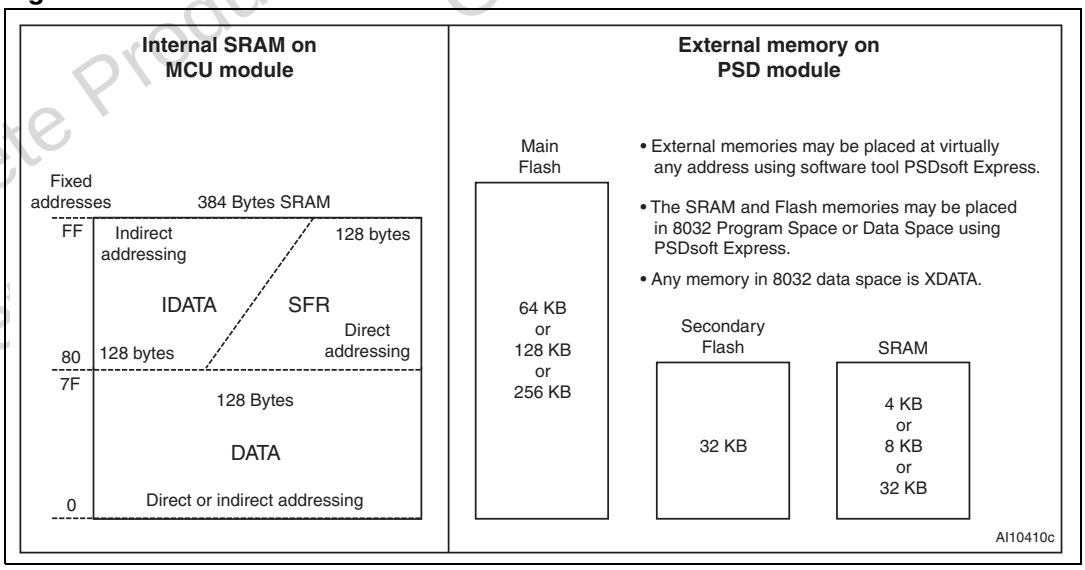


Table 6. Arithmetic instruction set (continued)

Mnemonic[1] and use		Description	Length/cycles
DEC	@Ri	Decrement indirect SRAM	1 byte/1 cycle
INC	DPTR	Increment Data Pointer	1 byte/2 cycle
MUL	AB	Multiply ACC and B	1 byte/4 cycle
DIV	AB	Divide ACC by B	1 byte/4 cycle
DA	A	Decimal adjust ACC	1 byte/1 cycle

1. All mnemonics copyrighted ©Intel Corporation 1980.

Table 7. Logical instruction set

Mnemonic[1] and use		Description	Length/cycles
Logical Instructions			
ANL	A, Rn	AND register to ACC	1 byte/1 cycle
ANL	A, direct	AND direct byte to ACC	2 byte/1 cycle
ANL	A, @Ri	AND indirect SRAM to ACC	1 byte/1 cycle
ANL	A, #data	AND immediate data to ACC	2 byte/1 cycle
ANL	direct, A	AND ACC to direct byte	2 byte/1 cycle
ANL	direct, #data	AND immediate data to direct byte	3 byte/2 cycle
ORL	A, Rn	OR register to ACC	1 byte/1 cycle
ORL	A, direct	OR direct byte to ACC	2 byte/1 cycle
ORL	A, @Ri	OR indirect SRAM to ACC	1 byte/1 cycle
ORL	A, #data	OR immediate data to ACC	2 byte/1 cycle
ORL	direct, A	OR ACC to direct byte	2 byte/1 cycle
ORL	direct, #data	OR immediate data to direct byte	3 byte/2 cycle
SWAP	A	Swap nibbles within the ACC	1 byte/1 cycle
XRL	A, Rn	Exclusive-OR register to ACC	1 byte/1 cycle
XRL	A, direct	Exclusive-OR direct byte to ACC	2 byte/1 cycle
XRL	A, @Ri	Exclusive-OR indirect SRAM to ACC	1 byte/1 cycle
XRL	A, #data	Exclusive-OR immediate data to ACC	2 byte/1 cycle
XRL	direct, A	Exclusive-OR ACC to direct byte	2 byte/1 cycle
XRL	direct, #data	Exclusive-OR immediate data to direct byte	3 byte/2 cycle
CLR	A	Clear ACC	1 byte/1 cycle
CPL	A	Compliment ACC	1 byte/1 cycle
RL	A	Rotate ACC left	1 byte/1 cycle
RLC	A	Rotate ACC left through the carry	1 byte/1 cycle
RR	A	Rotate ACC right	1 byte/1 cycle
RRC	A	Rotate ACC right through the carry	1 byte/1 cycle

1. All mnemonics copyrighted ©Intel Corporation 1980.

12 Debug unit

The 8032 MCU module supports run-time debugging through the JTAG interface. This same JTAG interface is also used for In-System Programming (ISP) and the physical connections are described in the PSD module section, *Section 28.6.1: JTAG ISP and JTAG debug on page 257*.

Debugging with a serial interface such as JTAG is a non-intrusive way to gain access to the internal state of the 8032 MCU core and various memories. A traditional external hardware emulator cannot be completely effective on the UPSD34xx because of the Pre-Fetch Queue and Branch Cache. The nature of the PFQ and BC hide the visibility of actual program flow through traditional external bus connections, thus requiring on-chip serial debugging instead.

Debugging is supported by Windows PC based software tools used for 8051 code development from 3rd party vendors listed at www.st.com/psm. Debug capabilities include:

- Halt or start MCU execution
- Reset the MCU
- Single step
- 3 match breakpoints
- 1 range breakpoint (inside or outside range)
- Program tracing
- Read or modify MCU core registers, DATA, IDATA, SFR, XDATA, and code
- External debug event pin, input or output

Some key points regarding use of the JTAG debugger.

  - The JTAG debugger can access MCU registers, data memory, and code memory while the MCU is executing at full speed by cycle-stealing. This means “watch windows” may be displayed and periodically updated on the PC during full speed operation. Registers and data content may also be modified during full speed operation.
  - There is no on-chip storage for Program Trace data, but instead this data is scanned from the UPSD34xx through the JTAG channel at run-time to the PC host for processing. As such, full speed program tracing is possible only when the 8032 MCU is operating below approximately one MIPS of performance. Above one MIPS, the program will not run real-time while tracing. One MIPS performance is determined by the combination of choice for MCU clock frequency, and the bit settings in SFR registers BUSCON and CCON0.
  - Breakpoints can optionally halt the MCU, and/or assert the external Debug Event pin.
  - Breakpoint definitions may be qualified with read or write operations, and may also be qualified with an address of code, SFR, DATA, IDATA, or XDATA memories.
  - Three breakpoints will compare an address, but the fourth breakpoint can compare an address and also data content. Additionally, the fouth breakpoint can be logically combined (AND/OR) with any of the other three breakpoints.
  - The Debug Event pin can be configured by the PC host to generate an output pulse for external triggering when a break condition is met. The pin can also be configured as an event input to the breakpoint logic, causing a break on the falling-edge of an external event signal. If not used, the Debug Event pin should be pulled

Table 28. CCON0 register bit definition (continued)

Bit	Symbol	R/W	Definition
3	CPUAR	R,W	Automatic MCU Clock Recovery 0 = There is no change of CPUPS[2:0] when an interrupt occurs. 1 = Contents of CPUPS[2:0] automatically become 000b whenever any interrupt occurs.
2:0	CPUPS	R,W	MCUCLK Pre-Scaler 000b: $f_{MCU} = f_{OSC}$ (Default after reset) 001b: $f_{MCU} = f_{OSC}/2$ 010b: $f_{MCU} = f_{OSC}/4$ 011b: $f_{MCU} = f_{OSC}/8$ 100b: $f_{MCU} = f_{OSC}/16$ 101b: $f_{MCU} = f_{OSC}/32$ 110b: $f_{MCU} = f_{OSC}/1024$ 111b: $f_{MCU} = f_{OSC}/2048$

Table 29. CCON1 PLL control register (SFR FAh, reset value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PLLM[3:0]				PLLD[3:0]			

Table 30. CCON1 register bit definition

Bit	Symbol	R/W	Definition
7:4	PLLM[3:0]	R,W	Lower 4 bits of the 5-bit PLLM[4:0] Multiplier (Default after reset: PLLM = 00h) PLLM[4] is in the CCON0 register.
3:0	PLLD[3:0]	R,W	4-bit PLL Divider (Default after reset: PLLD = 0h)

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]

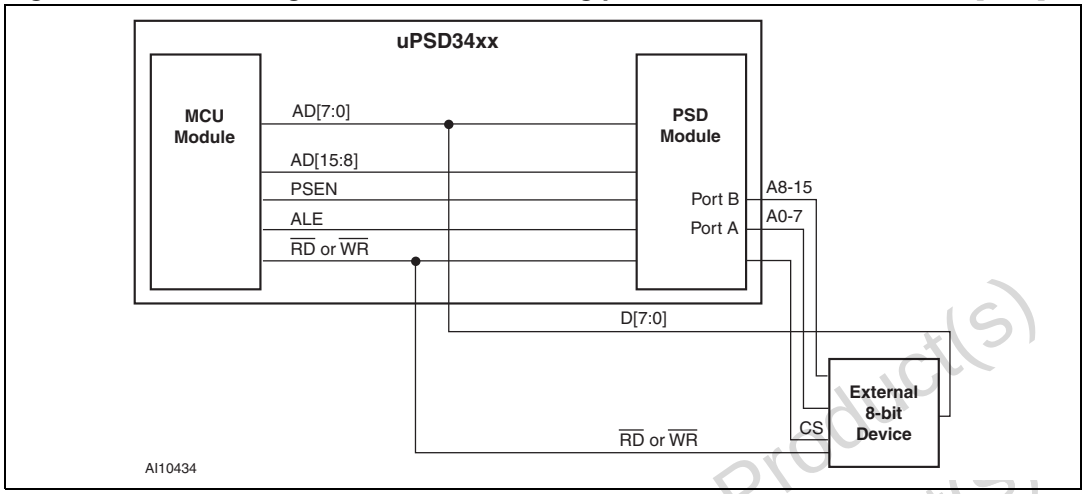
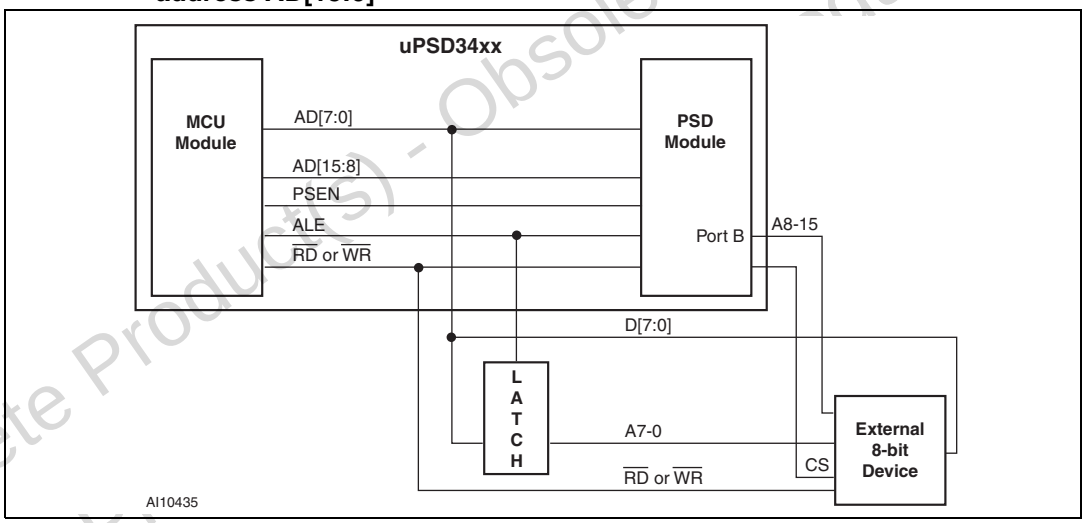


Figure 20. Connecting external devices using port A and an external latch for address AD[15:0]



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.

Figure 24. Timer/counter mode 0: 13-bit counter

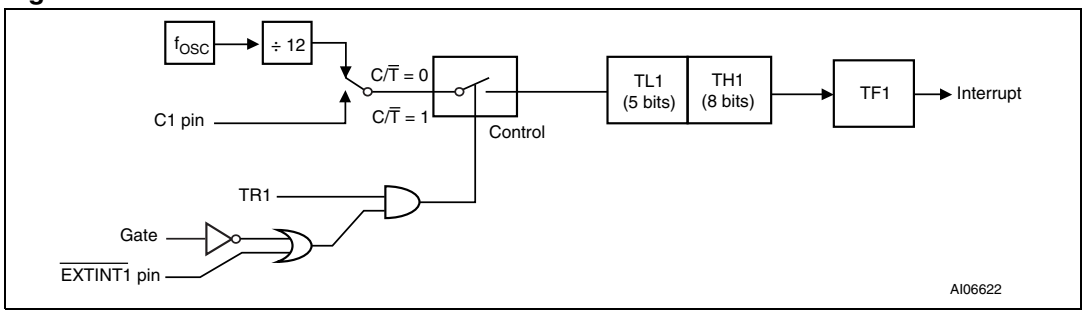


Figure 25. Timer/counter mode 2: 8-bit Auto-reload

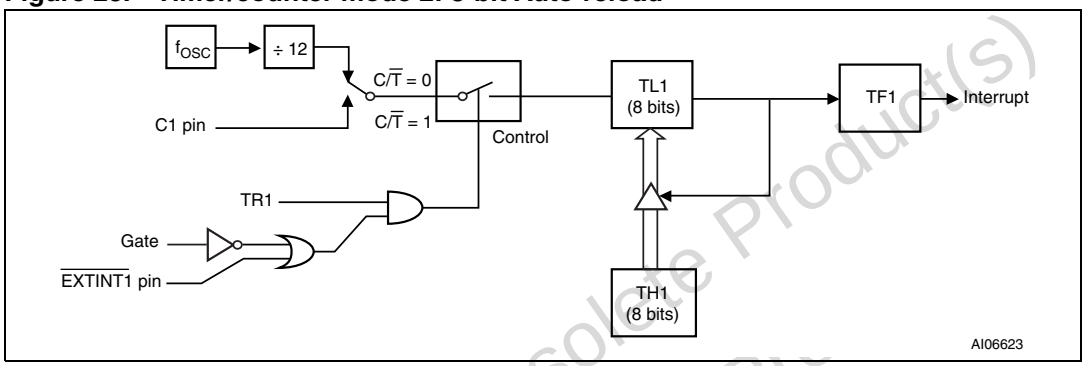
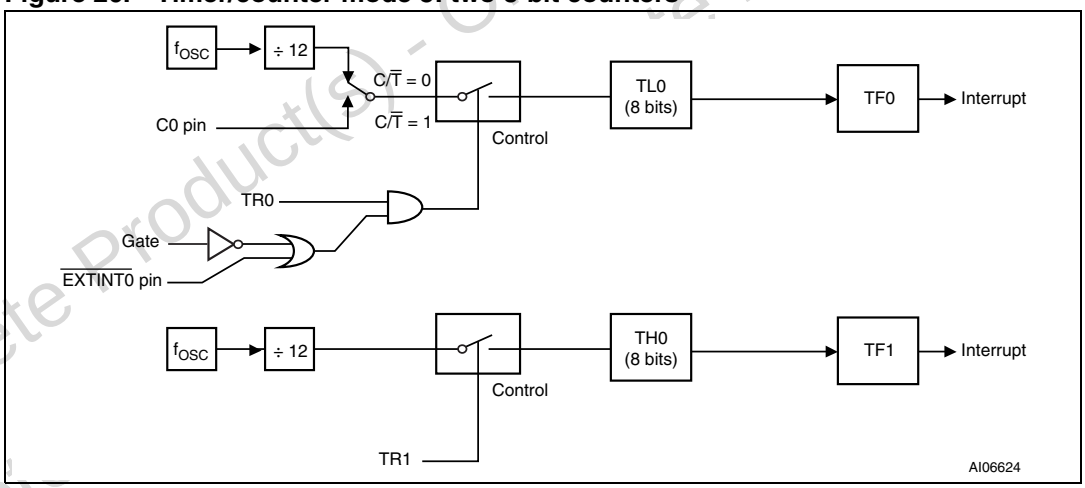


Figure 26. Timer/counter mode 3: two 8-bit counters



20.6 Timer 2

Timer 2 can operate as either an event timer or as an event counter. This is selected by the bit C/$\overline{T2}$ in the SFR named, T2CON (*Table 60 on page 101*). Timer 2 has three operating modes selected by bits in T2CON, according to *Table 62 on page 102*. The three modes are:

- Capture mode
- Auto re-load mode
- Baud rate generator mode

21.5 More about UART mode 1

Refer to the block diagram in *Figure 32 on page 115*, and timing diagram in *Figure 33 on page 115*.

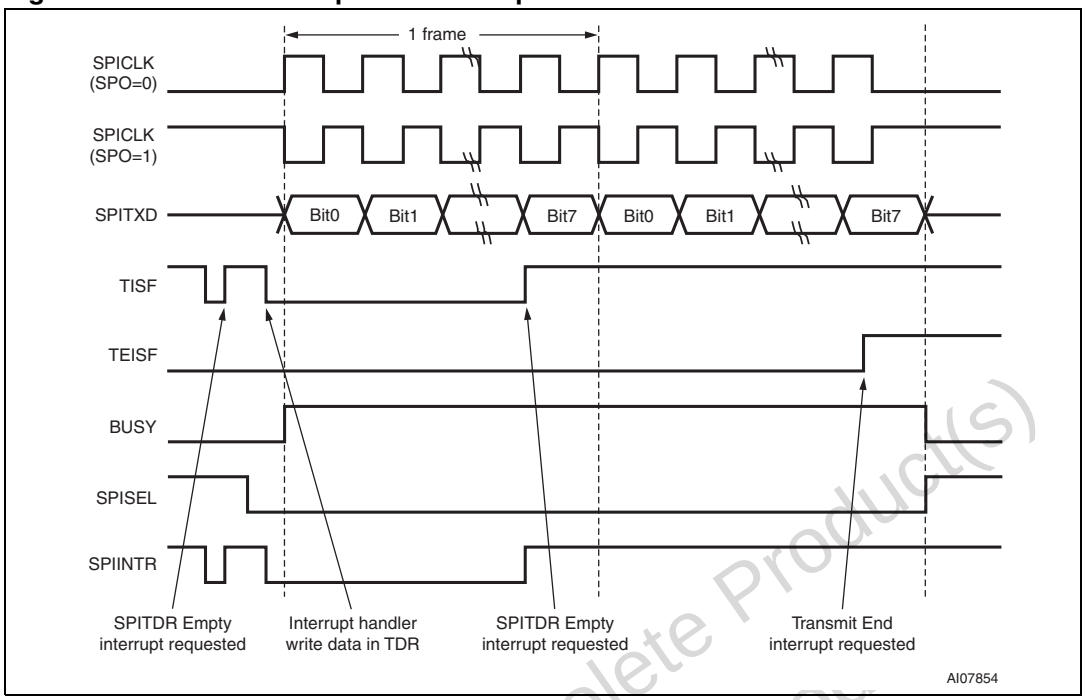
Transmission is initiated by any instruction which writes to SBUF. At the end of a write operation to SBUF, a '1' is loaded into the 9th position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission actually starts at the end of the MCU the machine cycle following the next rollover in the divide-by-16 counter. Thus, the bit times are synchronized to the divide-by-16 counter, not to the writing of SBUF. Transmission begins with activation of SEND which puts the start bit at pin TxD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to pin TxD. The first shift pulse occurs one bit time after that. As data bits shift out to the right, zeros are clocked 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 and then deactivates SEND, and sets the interrupt flag, TI. This occurs at the 10th divide-by-16 rollover after a write to SBUF.

Reception is initiated by a detected 1-to-0 transition at the pin RxD. For this purpose RxD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written into the input shift register. Resetting the divide-by-16 counter aligns its rollovers with the boundaries of the incoming bit times. The 16 states of the counter divide each bit time into 16ths. At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RxD. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for noise rejection. If the value accepted during the first bit time is not '0,' the receive circuits are reset and the unit goes back to looking for another '1'-to-'0' transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the reset of the rest of the frame will proceed. As data bits come in from the right, '1s' shift out to the left. When the start bit arrives at the left-most position in the shift register (which in mode 1 is a 9-bit register), it flags the RX Control unit to do one last shift, load SBUF and RB8, and set the receive interrupt flag RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

1. RI = 0, and
2. Either SM2 = 0, or the received stop bit = 1.

If either of these two conditions are not met, the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB8, the 8 data bits go into SBUF, and RI is activated. At this time, whether the above conditions are met or not, the unit goes back to looking for a '1'-to-'0' transition on pin RxD.

Figure 46. SPI transmit operation example



24.4 SPI SFR registers

Six SFR registers control the SPI interface:

- SPICON0 (*Table 89*) for interface control
- SPICON1 (*Table 91*) for interrupt control
- SPITDR (SFR D4h, Write only) holds byte to transmit
- SPIRDR (SFR D5h, Read only) holds byte received
- SPICLKD (*Table 93*) for clock divider
- SPISTAT (*Table 95 on page 149*) holds interface status

The SPI interface functional block diagram (*Figure 47*) shows these six SFRs. Both the transmit and receive data paths are double-buffered, meaning that continuous transmitting or receiving (back-to-back transfer) is possible by reading from SPIRDR or writing data to SPITDR while shifting is taking place. There are a number of flags in the SPISTAT register that indicate when it is full or empty to assist the 8032 MCU in data flow management. When enabled, these status flags will cause an interrupt to the MCU.

Table 98. UPSD34xx supported endpoints

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

25.3.3 FIFO pairing

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

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

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

- Non-pairing FIFOs Example

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

- USB endpoint control register (UCON)

  The Endpoint selected by the USB endpoint select register (see *Table 124 on page 172*) determines the direction and FIFO (IN or OUT) that is controlled by the USB endpoint control register (see *Table 126*). The USB endpoint control register is used to control the selected Endpoint and provides some status about that Endpoint.

Table 126. USB endpoint control register (UCON 0F1h, reset value 08h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	–	–	–	Enable	STALL	TOGGLE	BSY

Table 127. UCON register bit definition

Bit	Symbol	R/W	Definition
7	–	–	Reserved
6	–	–	Reserved
5	–	–	Reserved
4	–	–	Reserved
3	ENABLE	R/W	Selected FIFO Enable Bit. *Note: All FIFOs for each endpoint is enabled after a reset.*
2	STALL	R/W	Stall Control Bit When this bit is set, the Endpoint returns a STALL handshake whenever it receives an IN or OUT token.
1	TOGGLE	R	Data Toggle Bit – Endpoint IN Case The state of this bit determines the type of data packet (0=DATA0 or 1=DATA1) that will be sent during the next IN transaction. This bit is managed by the USB SIE. It is only toggled when an ACK is properly received during the IN transaction. In some cases it may be necessary for firmware to clear this bit. In this case, see the Important Notes section at the end of this data sheet. – Endpoint OUT Case The state of this bit indicates the type of data packet PID that the USB SIE expects to receive with the next OUT transaction (0=DATA0, 1=DATA1). If the Data Toggle for the next OUT transaction received is not as expected, the USB SIE assumes the host is retransmitting the last packet. In this case, an ACK is sent but no interrupt is generated since the original transmission of the packet was OK. This bit is managed by the USB SIE. It is only toggled when an OUT packet is properly received. In some cases it may be necessary for firmware to clear this bit. In this case, see the Important Notes section at the end of this data sheet. *Important notes:* *1. Disabling and enabling the USB SIE using the USBEN bit the UCTL register clears the TOGGLE bit for both directions of all endpoints.* *2. Revision A silicon: See the Important Notes section at the end of the data sheet that explains the workaround for clearing this data toggle bit.* *3. Revision B silicon: Disabling and Enabling the selected FIFO using the ENABLE bit in this register clears the data toggle bit for the selected endpoint's FIFO.*

- USB setup command index and value registers (USCI and USCV)

  When a Setup/Data packet is received over the USB, the 8 bytes of data received are stored in a command buffer. The USB setup command index register (see *Table 134*) determines which one of the eight bytes in the buffer is read using the USB setup command value register (see *Table 136*).

Table 134. USB setup command index register (USCI 0F5h, reset value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	–	–	–	–	USCI[2:0]		

Table 135. USCI register bit definition

Bit	Symbol	R/W	Definition
7:3	–	–	Reserved
2:0	USCI[2:0]	R/W	Index to access one of the 8 bytes of USB Setup Command Data received with the last Setup transaction

Table 136. USB setup command value register (USCV 0F6h, reset value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USCV[7:0]							

Table 137. USCV register bit definition

Bit	Symbol	R/W	Definition
7:0	USCV	R/W	The nth byte of the 8 bytes of USB Setup Command Data received with the last Setup transaction. The nth byte that is read from this register is specified by the index value in the USCI register.

Table 144. CCON2 register bit definition (SFR 0FBh, reset value 10h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	–	–	PCA0CE	PCA0PS3	PCA0PS2	PCA0PS1	PCA0PS0

Table 145. CCON2 register bit definition

Bit	Symbol	R/W	Definition
4	PCA0CE	R/W	PCA0 Clock Enable 0 = PCA0CLK is disabled 1 = PCA0CLK is enabled (default)
3:0	PCA0PS [3:0]	R/W	PCA0 Prescaler $f_{PCA0CLK} = f_{OSC} / (2 \text{ ^ } PCA0PS[3:0])$ Divisor range: 1, 2, 4, 8, 16... 16384, 32768

Table 146. CCON3 register bit definition (SFR 0FCh, reset value 10h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	–	–	PCA1CE	PCA1PS3	PCA1PS2	PCA1PS1	PCA1PS0

Table 147. CCON3 register bit definition

Bit	Symbol	R/W	Definition
4	PCA1CE	R/W	PCA1 Clock Enable 0 = PCA1CLK is disabled 1 = PCA1CLK is enabled (default)
3:0	PCA1PS [3:0]	R/W	PCA1 Prescaler $f_{PCA1CLK} = f_{OSC} / (2 \text{ ^ } PCA1PS[3:0])$ Divisor range: 1, 2, 4, 8, 16... 16384, 32768

27.3 Operation of TCM modes

Each of the TCM in a PCA block supports four modes of operation. However, an exception is when the TCM is configured in PWM Mode with programmable frequency. In this mode, all TCM in a PCA block must be configured in the same mode or left to be not used.

27.4 Capture mode

The CAPCOM registers in the TCM are loaded with the counter values when an external pin input changes state. The user can configure the counter value to be loaded by positive edge, negative edge or any transition of the input signal. At loading, the TCM can generate an interrupt if it is enabled.

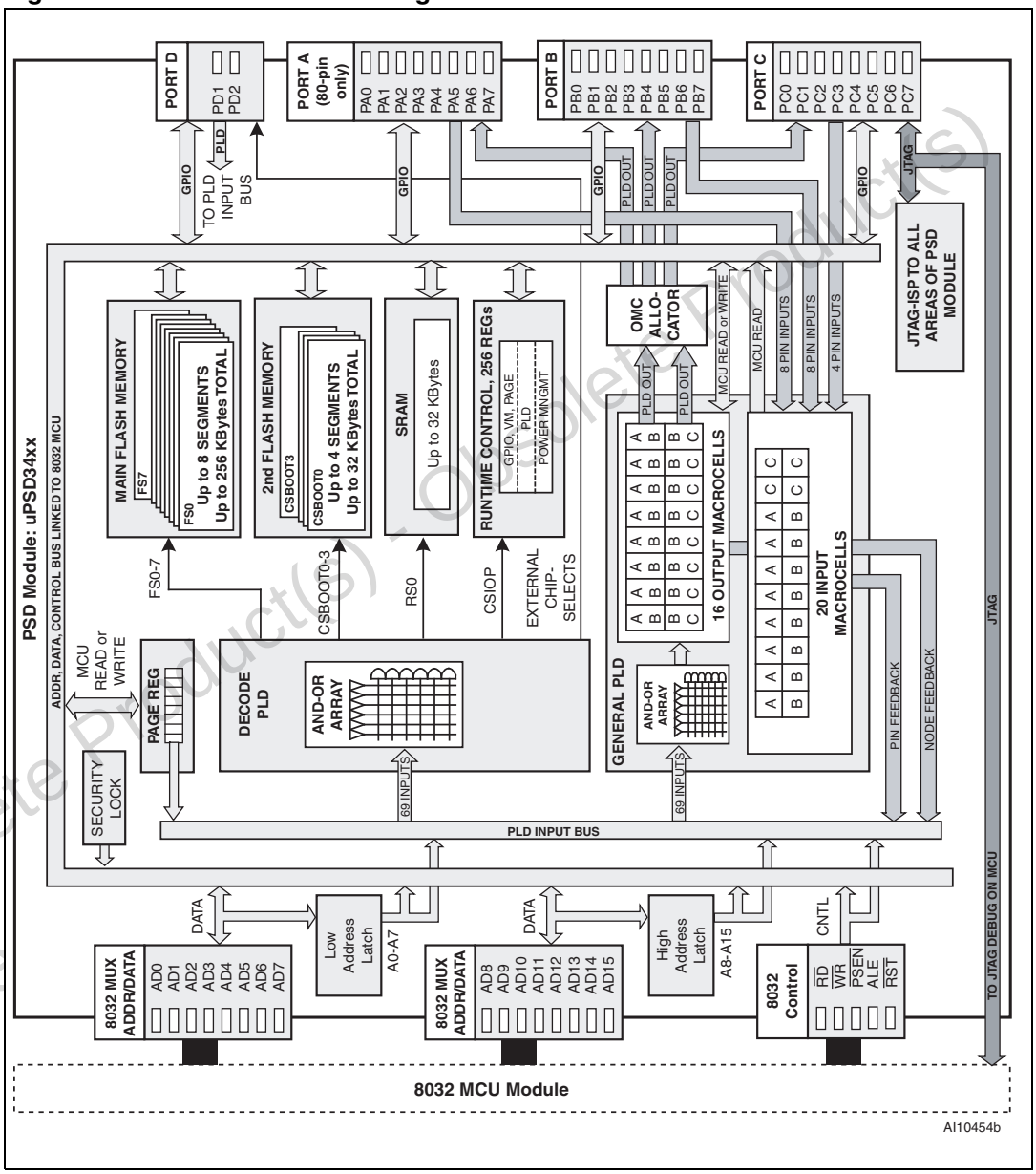
27.5 Timer mode

The TCM modules can be configured as software timers by enable the comparator. The user writes a value to the CAPCOM registers, which is then compared with the 16-bit counter. If there is a match, an interrupt can be generated to CPU.

28 PSD module

The PSD module is stacked with the MCU module to form the UPSD34xx, see *Section 3: Hardware description on page 28*. Details of the PSD module are shown in *Figure 61*. The two separate modules interface with each other at the 8032 Address, Data, and Control interface blocks in *Figure 61*.

Figure 61. PSD module block diagram



memory contents through its 8-bit data bus even while the security bit is set. The 8032 can read the status of the security bit at run-time (but it cannot change it) by reading the csiop register defined in *Table 166*.

Table 165. Main Flash memory protection register definition (address = csiop + offset C0h)(1)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sec7_Prot	Sec6_Prot	Sec5_Prot	Sec4_Prot	Sec3_Prot	Sec2_Prot	Sec1_Prot	Sec0_Prot

1. *Bit definitions:*
Sec<i>_Prot 1 = Flash memory sector <i> is write protected, 0 = Flash memory sector <i> is not write protected.

Table 166. Secondary Flash memory protection/security register definition (csiop + offset C2h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3(1)	Bit 2(1)	Bit 1(1)	Bit 0(1)
Security_Bit (2)	not used	not used	not used	Sec3_Prot	Sec2_Prot	Sec1_Prot	Sec0_Prot

1. *Sec<i>_Prot 1 = Flash memory sector <i> is write protected, 0 = Flash memory sector <i> is not write protected.*
2. Security_Bit = 1, device is secured, 0 = not secured.

28.5.25 PLDs

The PSD module contains two PLDs: the Decode PLD (DPLD), and the General PLD (GPLD), as shown in *Figure 73 on page 221*. Both PLDs are fed by a common PLD input signal bus, and additionally, the GPLD is connected to the 8032 data bus.

PLD logic is specified using PSDsoft Express and programmed into the PSD module using the JTAG ISP channel. PLD logic is non-volatile and available at power-up. PLDs may not be programmed by the 8032. The PLDs have selectable levels of performance and power consumption.

The DPLD performs address decoding, and generates select signals for internal and external components, such as memory, registers, and I/O ports. The DPLD can generate External Chip-Select (ECS1-ECS2) signals on Port D.

The GPLD can be used for logic functions, such as loadable counters and shift registers, state machines, encoding and decoding logic. These logic functions can be constructed from a combination of 16 Output Macrocells (OMC), 20 Input Macrocells (IMC), and the AND-OR Array.

Routing of the 16 OMCs outputs can be divided between pins on three Ports A, B, or C by the OMC Allocator as shown in *Figure 77 on page 227*. Eight of the 16 OMCs that can be routed to pins on Port A or Port B and are named MCELLAB0-MCELLAB7. The other eight OMCs to be routed to pins on Port B or Port C and are named MCELLBC0-MCELLBC7. This routing depends on the pin number assignments that are specified in PSDsoft Express for “PLD Outputs” in the Pin Definition section. OMC outputs can also be routed internally (not to pins) used as buried nodes to create shifters, counters, etc.

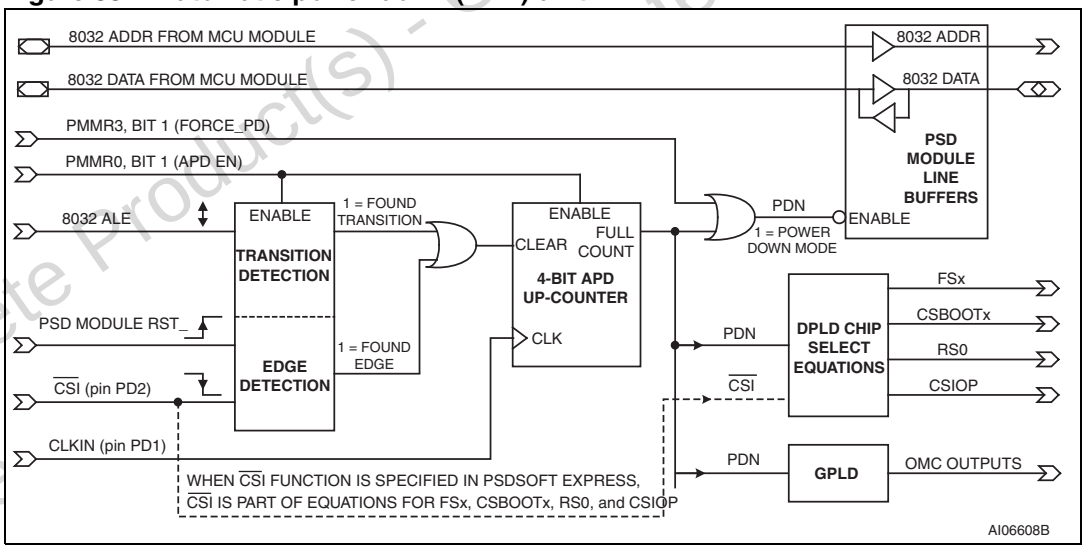
The AND-OR Array is used to form product terms. These product terms are configured from the logic definitions entered in PSDsoft Express. A PLD Input Bus consisting of 69 signals is connected to both PLDs. Input signals are shown in *Table 167*, both the true and compliment versions of each of these signals are available at inputs to each PLD.

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



32 Package mechanical information

In order to meet environmental requirements, ST offers these devices in different grades of ECOPACK® packages, depending on their level of environmental compliance. ECOPACK® specifications, grade definitions and product status are available at: *www.st.com*. ECOPACK® is an ST trademark.

implemented then this extra and unexpected data packet would result in a communication breakdown.

Workaround

Revision A and B - In the USB ISR, when an INx (x = the endpoint number of the IN FIFO) interrupt is detected, the IN FIFOs respective busy bit should be unconditionally cleared. The UPSD3400 USB firmware implements this workaround.

34.7 IN FIFO pairing operation

Description

When FIFO pairing is used on IN endpoints, an erroneous resend of a data packet may occur. See the "Erroneous Resend of Data Packet" note as it also applies when IN FIFO pairing is used.

Impact on application

See the "Erroneous Resend of Data Packet" note as the impact is the same when IN FIFO pairing is used.

Workaround

Revision A and B - See the "Erroneous Resend of Data Packet" note as the workaround is the same when IN FIFO pairing is used.

34.8 OUT FIFO pairing operation

Description

When data packets are received from the host and FIFO pairing is used, the paired FIFOs may get out of order.

Impact on application

The received data packets are read out of order compared to the way they were sent from the host. If the workaround is not implemented, the out of order packets would result in a communication breakdown.

Workaround

Revision A and B - In the USB ISR, when an OUTx (x = the endpoint number of the OUT FIFO) interrupt is detected, the OUT FIFOs respective busy bit should be unconditionally cleared. The UPSD3400 USB firmware implements this workaround.

34.9 Missing ACK to host retransmission of SETUP packet

Description

If a host does not properly receive the ACK (due to noise) from the UPSD3400 in response to a SETUP packet, it will resend the SETUP packet but the UPSD3400 will not respond