Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	8032
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, IrDA, SPI, UART/USART, USB
Peripherals	LVD, POR, PWM, WDT
Number of I/O	46
Program Memory Size	288KB (288K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	32K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/upsd3454e-40u6

four MCU clocks). But it is also important to understand PFQ operation on multi-cycle instructions.

5.2 PFQ example, multi-cycle instructions

Let us look at a string of two-byte, two-cycle instructions in *Figure 8 on page 35*. There are three instructions executed sequentially in this example, instructions A, B, and C. Each of the time divisions in the figure is one machine-cycle of four clocks, and there are six phases to reference in this discussion. Each instruction is pre-fetched into the PFQ in advance of execution by the MCU. Prior to Phase 1, the PFQ has pre-fetched the two instruction bytes (A1 and A2) of Instruction A. During Phase one, both bytes are loaded into the MCU execution unit. Also in Phase 1, the PFQ is pre-fetching Instruction B (bytes B1 and B2) from program memory. In Phase 2, the MCU is processing Instruction A internally while the PFQ is pre-fetching Instruction C. In Phase 3, both bytes of instruction B are loaded into the MCU execution unit and the PFQ begins to pre-fetch bytes for the next instruction. In Phase 4 Instruction B is processed.

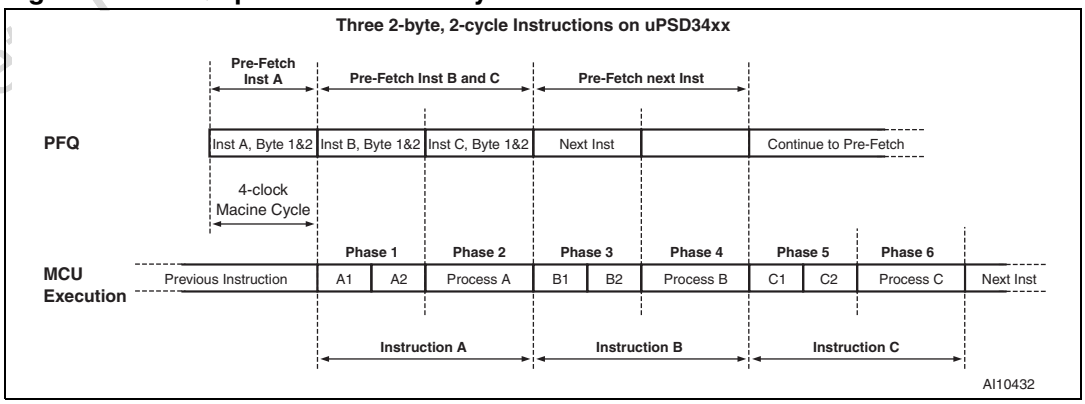
The UPSD34xx MCU instructions are an exact 1/3 scale of all standard 8032 instructions with regard to number of cycles per instruction. *Figure 9 on page 36* shows the equivalent instruction sequence from the example above on a standard 8032 for comparison.

5.3 Aggregate performance

The stream of two-byte, two-cycle instructions in *Figure 8 on page 35*, running on a 40 MHz, 5 V, UPSD34xx will yield 5 MIPs. And we saw the stream of one- or two-byte, one-cycle instructions in *Figure 6 on page 33*, on the same MCU yield 10 MIPs. Effective performance will depend on a number of things: the MCU clock frequency; the mixture of instructions types (bytes and cycles) in the application; the amount of time an empty PFQ stalls the MCU (mix of instruction types and misses on Branch Cache); and the operating voltage. A 5 V UPSD34xx device operates with four memory wait states, but a 3.3 V device operates with five memory wait states yielding 8 MIPS peak compared to 10 MIPs peak for 5 V device. The same number of wait states will apply to both program fetches and to data READ/WRITEs unless otherwise specified in the SFR named BUSCON.

In general, a 3X aggregate performance increase is expected over any standard 8032 application running at the same clock frequency.

Figure 8. PFQ operation on multi-cycle instructions



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
- USB interface registers
  UADDR, UPAIR, WE0-3, UIF0-3, UCTL, USTA, USEL, UCON, USIZE, UBASEH, UBASEL, USCI, USCV

Table 5. SFR memory map with direct address and reset value

SFR addr (hex)	SFR name	Bit name and <bit address>								Reset value (hex)	Reg. descr. with link
		7	6	5	4	3	2	1	0		
80	RESERVED										
81	SP	SP[7:0]								07	*Section 7.1*
82	DPL	DPL[7:0]								00	*Section 7.2*
83	DPH	DPH[7:0]								00	
84	RESERVED										
85	DPTC	–	AT	–	–	–	DPSEL[2:0]			00	*Table 13*
86	DPTM	–	–	–	–	MD1[1:0]		MD0[1:0]		00	*Table 15*
87	PCON	SMOD0	SMOD1	–	POR	RCLK1	TCLK1	PD	IDLE	00	*Table 33*
88(1)	TCON	TF1 <8Fh>	TR1 <8Eh>	TF0 <8Dh>	TR0 <8Ch>	IE1 <8Bh>	IT1 <8Ah>	IE0 <89h>	IT0 <88h>	00	*Table 56*
89	TMOD	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	00	*Table 58*
8A	TL0	TL0[7:0]								00	*Section 20.1*
8B	TL1	TL1[7:0]								00	
8C	TH0	TH0[7:0]								00	
8D	TH1	TH1[7:0]								00	
8E	P1SFS0	P1SFS0[7:0]								00	*Table 43*
8F	P1SFS1	P1SFS1[7:0]								00	*Table 44*
90(1)	P1	P1.7 <97h>	P1.6 <96h>	P1.5 <95h>	P1.4 <94h>	P1.3 <93h>	P1.2 <92h>	P1.1 <91h>	P1.0 <90h>	FF	*Table 35*
91	P3SFS	P3SFS[7:0]								00	*Table 41*
92	P4SFS0	P4SFS0[7:0]								00	*Table 46*
93	P4SFS1	P4SFS1[7:0]								00	*Table 47*

Table 5. SFR memory map with direct address and reset value (continued)

SFR addr (hex)	SFR name	Bit name and <bit address>								Reset value (hex)	Reg. descr. with link
		7	6	5	4	3	2	1	0		
B0(1)	P3	P3.7 <B7h>	P3.6 <B6h>	P3.5 <B5h>	P3.4 <B4h>	P3.3 <B3h>	P3.2 <B2h>	P3.1 <B1h>	P3.0 <B0h>	FF	Table 37
B1	CAPCOMH1	CAPCOMH1[7:0]								00	Table 143
B2	CAPCOML2	CAPCOML2[7:0]								00	
B3	CAPCOMH2	CAPCOMH2[7:0]								00	
B4	PWMF0	PWMF0[7:0]								00	
B5	RESERVED										
B6	RESERVED										
B7	IPA	PADC	PSPI	PPCA	PS1	–	–	PI2C	–	00	Table 24
B8(1)	IP	–	–	PT2 <BDh>	PS0 <BCh>	PT1 <BBh>	PX1 <BAh>	PT0 <B9h>	PX0 <B8h>	00	Table 22
B9	RESERVED										
BA	PCACL1	PCACL1[7:0]								00	Table 143
BB	PCACH1	PCACH1[7:0]								00	
BC	PCACON1	–	EN_PCA	EOVF1	PCA_IDL	–	–	CLK_SEL[1:0]		00	Table 150
BD	TCMMODE3	EINTF	E_COMP	CAP_PE	CAP_NE	MATCH	TOGGLE	PWM[1:0]		00	Table 154
BE	TCMMODE4	EINTF	E_COMP	CAP_PE	CAP_NE	MATCH	TOGGLE	PWM[1:0]		00	
BF	TCMMODE5	EINTF	E_COMP	CAP_PE	CAP_NE	MATCH	TOGGLE	PWM[1:0]		00	
C0(1)	P4	P4.7 <C7h>	P4.6 <C6h>	P4.5 <C5h>	P4.4 <C4h>	P4.3 <C3h>	P4.2 <C2h>	P4.1 <C1h>	P4.0 <C0h>	FF	Table 39

Table 5. SFR memory map with direct address and reset value (continued)

SFR addr (hex)	SFR name	Bit name and <bit address>								Reset value (hex)	Reg. descr. with link
		7	6	5	4	3	2	1	0		
C1	CAPCOML3	CAPCOML3[7:0]								00	Table 143
C2	CAPCOMH3	CAPCOMH3[7:0]								00	
C3	CAPCOML4	CAPCOML4[7:0]								00	
C4	CAPCOMH4	CAPCOMH4[7:0]								00	
C5	CAPCOML5	CAPCOML5[7:0]								00	
C6	CAPCOMH5	CAPCOMH5[7:0]								00	
C7	PWMF1	PWMF1[7:0]								00	
C8(1)	T2CON	TF2 <CFh>	EXF2 <CEh>	RCLK <CDh>	TCLK <CCh>	EXEN2 <CBh>	TR2 <CAh>	C/$\overline{T2}$ <C9h>	CP/$\overline{RL2}$ <C8h>	00	Table 60
C9	RESERVED										
CA	RCAP2L	RCAP2L[7:0]								00	Section 20.1
CB	RCAP2H	RCAP2H[7:0]								00	
CC	TL2	TL2[7:0]								00	
CD	TH2	TH2[7:0]								00	
CE	IRDACON	–	IRDA_EN	BIT_PULS	CDIV4	CDIV3	CDIV2	CDIV1	CDIV0	0F	Table 70
D0(1)	PSW	CY <D7h>	AC <D6h>	F0 <D5h>	RS[1:0] <D4h, D3h>		OV <D2h>	–	P <D0>	00	Section 7.7
D1	RESERVED										
D2	SPICLKD	SPICLKD[5:0]						–	–	04	Table 93
D3	SPISTAT	–	–	–	BUSY	TEISF	RORISF	TISF	RISF	02	Table 95
D4	SPITDR	SPITDR[7:0]								00	Table 91
D5	SPIRDR	SPIRDR[7:0]								00	
D6	SPICON0	–	TE	RE	SPIEN	SSEL	FLSB	SPO	–	00	Table 89
D7	SPICON1	–	–	–	–	TEIE	RORIE	TIE	RIE	00	Table 91
D8(1)	SCON1	SM0 <DF	SM1 <DE>	SM2 <DD>	REN <DC>	TB8 <DB>	RB8 <DA>	TI <D9>	RI <D8>	00	Table 67
D9	SBUF1	SBUF1[7:0]								00	Section 21
DA	RESERVED										
DB	S1SETUP	SS_EN	SMPL_SET[6:0]							00	Table 84
DC	S1CON	CR2	EN1	STA	STO	ADDR	AA	CR1	CR0	00	Table 75

9 8032 addressing modes

The 8032 MCU uses 11 different addressing modes listed below:

- Register
- Direct
- Register indirect
- Immediate
- External direct
- External indirect
- Indexed
- Relative
- Absolute
- Long
- Bit

9.1 Register addressing

This mode uses the contents of one of the registers R0 - R7 (selected by the last three bits in the instruction opcode) as the operand source or destination. This mode is very efficient since an additional instruction byte is not needed to identify the operand. For example:

```
MOV A, R7                ; Move contents of R7 to accumulator
```

9.2 Direct addressing

This mode uses an 8-bit address, which is contained in the second byte of the instruction, to directly address an operand which resides in either 8032 DATA SRAM (internal address range 00h-07Fh) or resides in 8032 SFR (internal address range 80h-FFh). This mode is quite fast since the range limit is 256 bytes of internal 8032 SRAM. For example:

```
MOV A, 40h               ; Move contents of DATA SRAM
                         ; at location 40h into the accumulator
```

9.3 Register indirect addressing

This mode uses an 8-bit address contained in either register R0 or R1 to indirectly address an operand which resides in 8032 IDATA SRAM (internal address range 80h-FFh). Although 8032 SFR registers also occupy the same physical address range as IDATA, SFRs will not be accessed by register Indirect mode. SFRs may only be accesses using Direct address mode. For example:

```
MOV A, @R0               ; Move into the accumulator the
                         ; contents of IDATA SRAM that is
                         ; pointed to by the address
                         ; contained in R0.
```

Table 34. PCON register bit definition (continued)

Bit	Symbol	R/W	Function
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

Table 36. P1 register bit definition (continued)

Bit	Symbol	R/W	Function(1)
2	P1.2	R,W	Port pin 1.2
1	P1.1	R,W	Port pin 1.1
0	P1.0	R,W	Port pin 1.0

1. Write '1' or '0' for pin output. Read for pin input, but prior to READ, this bit must have been set to '1' by firmware or by a reset event.

Table 37. P3: I/O port 3 register (SFR B0h, reset value FFh)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0

Table 38. P3 register bit definition

Bit	Symbol	R/W	Function(1)
7	P3.7	R,W	Port pin 3.7
6	P3.6	R,W	Port pin 3.6
5	P3.5	R,W	Port pin 3.5
4	P3.4	R,W	Port pin 3.4
3	P3.3	R,W	Port pin 3.3
2	P3.2	R,W	Port pin 3.2
1	P3.1	R,W	Port pin 3.1
0	P3.0	R,W	Port pin 3.0

1. Write '1' or '0' for pin output. Read for pin input, but prior to READ, this bit must have been set to '1' by firmware or by a reset event.

Table 39. P4: I/O port 4 register (SFR C0h, reset value FFh)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4.7	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0

Table 40. P4 register bit definition

Bit	Symbol	R/W	Function(1)
7	P4.7	R,W	Port pin 4.7
6	P4.6	R,W	Port pin 4.6
5	P4.5	R,W	Port pin 4.5
4	P4.4	R,W	Port pin 4.4
3	P4.3	R,W	Port pin 4.3
2	P4.2	R,W	Port pin 4.2
1	P4.1	R,W	Port pin 4.1
0	P4.0	R,W	Port pin 4.0

1. Write '1' or '0' for pin output. Read for pin input, but prior to READ, this bit must have been set to '1' by firmware or by a reset event.

18 MCU bus interface

The MCU module has a programmable bus interface which is a modified 8032 bus with 16 multiplexed address and data lines. The bus supports four types of data transfer (16- or 8-bit), each transfer is to/from a memory location external to the MCU module:

- Code Fetch cycle using the PSEN signal: fetch a 16-bit code word for filling the pre-fetch queue. The CPU fetches a code byte from the PFQ for execution;
- Code Read cycle using PSEN: read a 16-bit code word using the MOVC (Move Constant) instruction. The code word is routed directly to the CPU and by-pass the PFQ;
- XDATA Read cycle using the RD signal: read a data byte using the MOVX (Move eXternal) instruction; and
- XDATA Write cycle using the WR signal: write a data byte using the MOVX instruction

18.1 PSEN bus cycles

In a PSEN bus cycle, the MCU module fetches the instruction from the 16-bit program memory in the PSD module. The multiplexed address/data bus AD[15:0] is connected to the PSD module for 16-bit data transfer. The UPSD34xx does not support external PSEN cycles and cannot fetch instruction from other external program memory devices.

18.2 READ or WRITE bus cycles

In an XDATA READ or WRITE bus cycle, the MCU's multiplexed AD[15:0] bus is connected to the PSD module, but only the lower bytes AD[7:0] are used for the 8-bit data transfer. The AD[7:0] lines are also connected to pins in the 80-pin package for accessing external devices. If the high address byte A[15:8] is needed for external devices, Port B in the PSD module can be configured to provide the latched A[15:8] address outputs.

18.3 Connecting external devices to the MCU bus

The UPSD34xx supports 8-bit only external I/O or Data memory devices. The READ and WRITE data transfer is carried out on the AD[7:0] bus which is available in the 80-pin package. The address lines can be brought out to the external devices in one of three ways:

1. Configure Ports B and A of the PSD module in Address Output mode, as shown in *Figure 19*;
2. Use Port B together with an external latch, as shown in *Figure 20 on page 87*. The external latch latches the low address byte from the AD[7:0] bus with the ALE signal.This configuration is for design where Port A is needed for CPLD functions; and
3. Configure the microcell in the CPLD to output any address line to any of the CPLD output pins. This is the most flexible implementation but requires the use of CPLD resources.

Ports A and B in the PSD module can be configured in the PSDsoft to provide latched MCU address A[7:0] and A[15:8] (see *Section 28.5: PSD module detailed operation on page 207* for details on how to enable Address Output mode). The latched address outputs on the ports are pin configurable. For example, Port B pins PB[2:0] can be enabled to provide

Table 50. BUSCON register bit definition

Bit	Symbol	R/W	Definition
7	EPFQ	R,W	Enable pre-fetch queue 0 = PFQ is disabled 1 = PFQ is enabled (default)
6	EBC	R,W	Enable branch cache 0 = BC is disabled 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 00b: 4 clock periods 01b: 5 clock periods 10b: 6 clock periods (default) 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 00b: 4 clock periods 01b: 5 clock periods 10b: 6 clock periods (default) 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 00b: 3 clock periods - exception, for MOVC instructions this setting results 4 clock periods 01b: 4 clock periods 10b: 5 clock periods 11b: 6 clock periods (default)

Figure 30. UART mode 0, block diagram

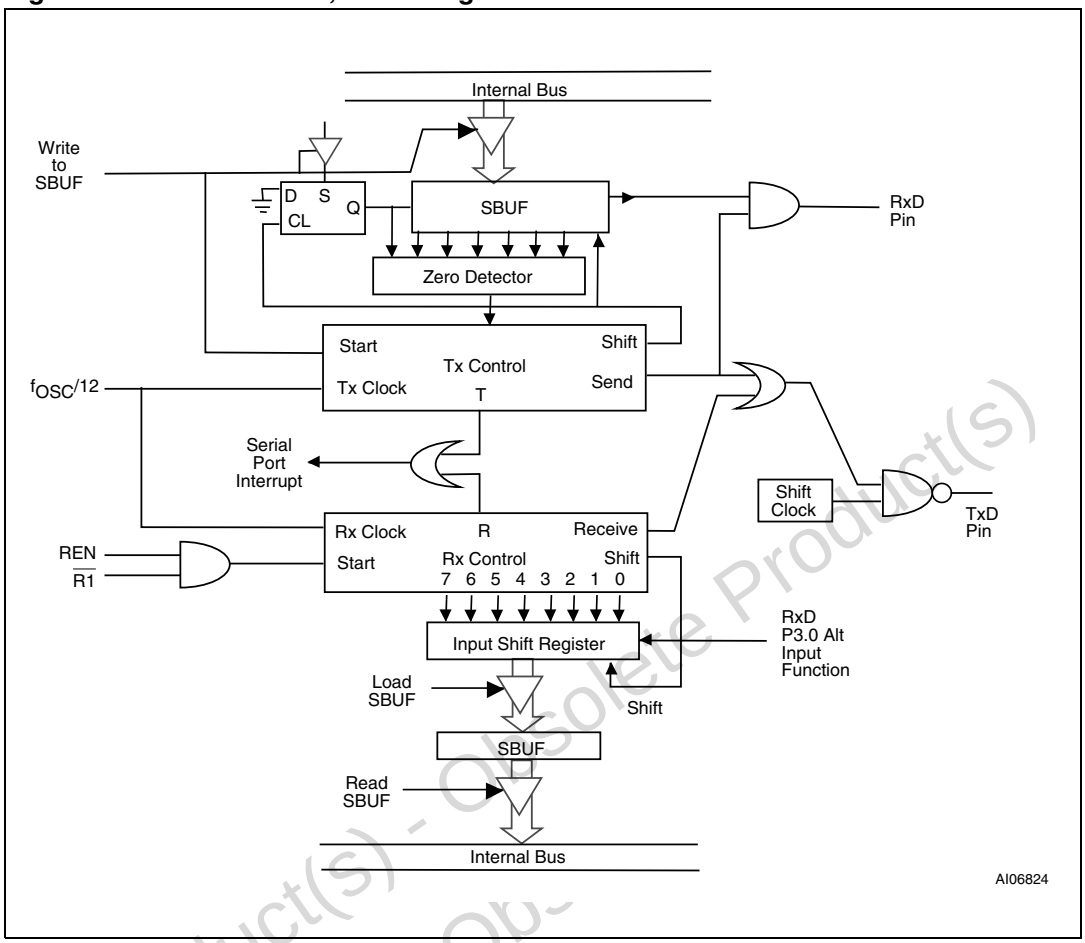
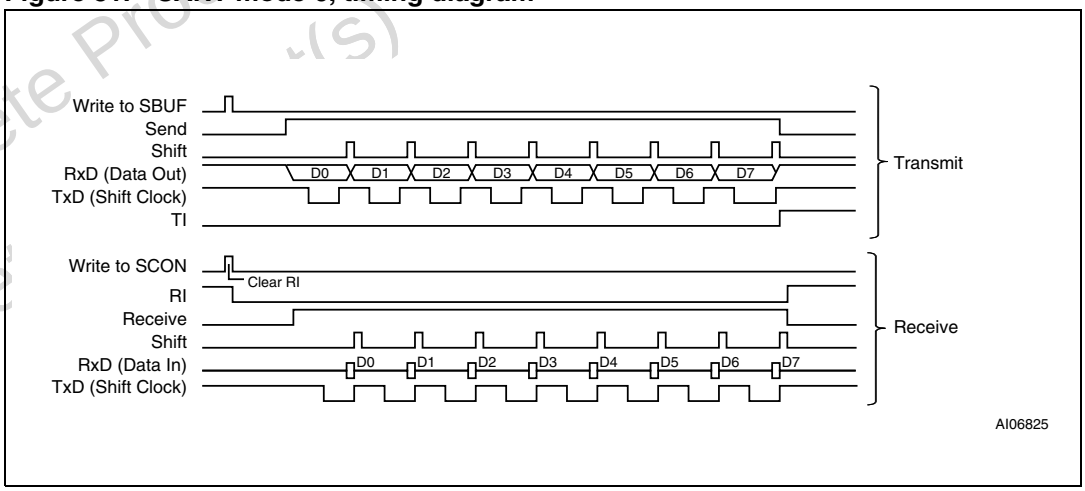


Figure 31. UART mode 0, timing diagram



Is this the next to last byte to receive from Slave?

If this is the next to last byte, do not allow Master to ACK on next interrupt.

- S1CON.AA = 0, don't let Master return ACK
- Exit ISR, now ready to recv last byte from Slv

If this is not next to last byte, let Master send ACK to Slave

<S1CON.AA is already 1>

- Exit ISR, ready to recv more bytes from Slave

Else If mode is Slave-Transmitter:

Is this Intr from SIOE detecting a Stop on bus?

If Yes, a Stop was detected:

- S1DAT = dummy, write to release bus
- Exit ISR, Master needs no more data bytes

If No, a Stop was not detected, continue:

ACK recvd from Master? (status.ACK_RESP=0?)

If No, an ACK was not received:

- S1DAT = dummy, write to release bus
- Exit ISR, Master needs no more data bytes

If Yes, ACK was received, then continue:

- S1DAT = xmit_buf[buffer_index], transmit byte
- Exit ISR, transmit next byte on next interrupt

Else If mode is Slave-Receiver:

Is this Intr from SIOE detecting a Stop on bus?

If Yes, a Stop was detected:

- recv_buf[buffer_index] = S1DAT, get last byte
- Exit ISR, Master has sent last byte

If No, a Stop was not detected, continue:

Determine if this Interrupt is from receiving an address or a data byte from a Master.

Is (S1CON.ADDR = 1 and S1CON.AA =1)?

If No, intr is from receiving data, goto C:

If Yes, intr is from an address, continue:

- slave_is_adressed = 1, local variable set true

<indicates Master selected this slave>

- S1CON.ADDR = 0, clear address match flag

Determine if R/W bit indicates transmit or receive.

Figure 44. SPI full-duplex data exchange

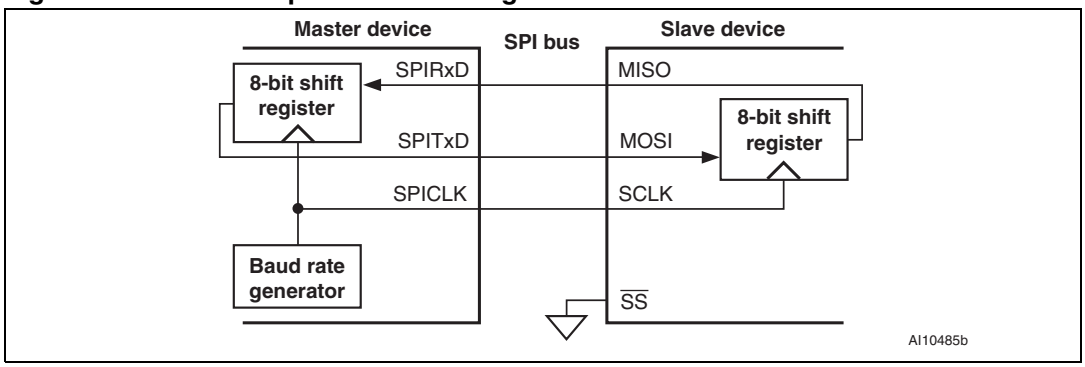


Figure 45. SPI receive operation example

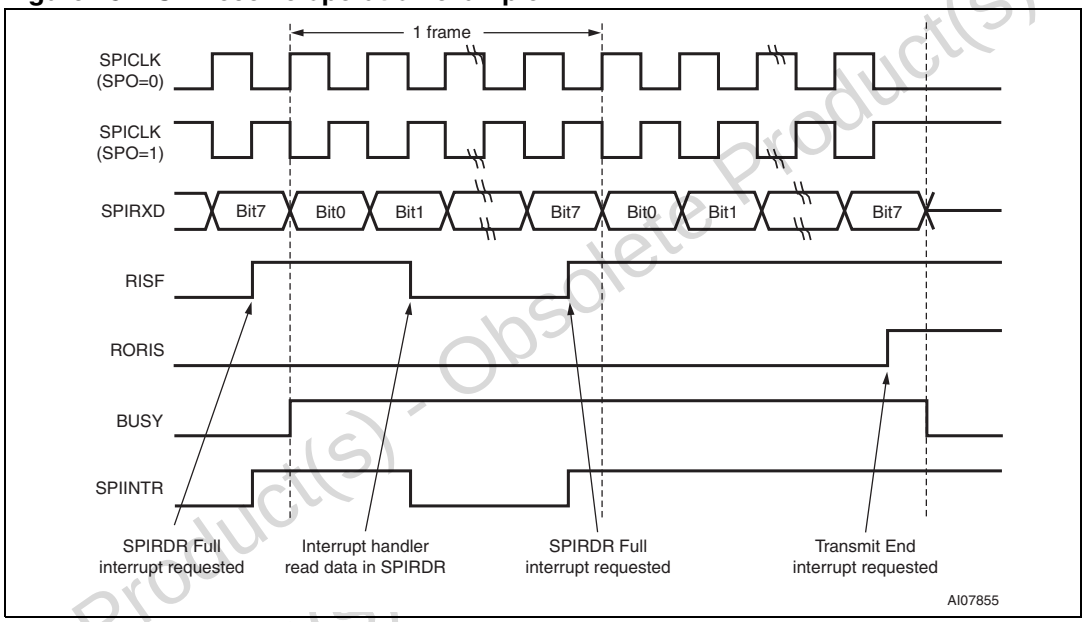


Table 162. CSIOP registers and their offsets (in hexadecimal) (continued)

Register name	Port A (80-pin)	Port B	Port C	Port D	Other	Description	Link
Drive Select	08h	09h	16h	17h		Write to configure port pins as either CMOS push-pull or Open Drain on some pins, while selecting high slew rate on other pins. Read to check status. Default output type is CMOS push-pull.	*Table 192 on page 241*
Input Macrocells	0Ah	0Bh	18h			Read to obtain logic state of IMCs. No WRITEs.	*Table 173 on page 230*
Enable Out	OCh	0Dh	1Ah	1Bh		Read state of output enable logic on each I/O port driver. 1 = driver output is enabled, 0 = driver is off, and it is in high impedance state. No WRITEs.	*Table 196 on page 242*
Output Macrocells AB (MCELLAB)					20h	Read logic state of MCELLAB outputs (bank of eight OMCs). Write to load MCELLAB flip-flops.	*Table 169 on page 228*
Output Macrocells BC (MCELLBC)					21h	Read logic state of MCELLBC outputs (bank of eight OMCs). Write to load MCELLBC flip-flops.	*Table 170 on page 228*
Mask Macrocells AB					22h	Write to set mask for MCELLAB. Logic '1' blocks READs/WRITEs of OMC. Logic '0' will pass OMC value. Read to check status.	*Table 171 on page 229*
Mask Macrocells BC					23h	Write to set mask for MCELLBC. Logic '1' blocks READs/WRITEs of OMC. Logic '0' will pass OMC value. Read to check status.	*Table 172 on page 229*
Main Flash Sector Protection					C0h	Read to determine Main Flash Sector Protection Setting (non-volatile) that was specified in PSDsoft Express. No WRITEs.	*Table 165 on page 219*
Security Bit and Secondary Flash Sector Protection					C2h	Read to determine if PSD module device Security Bit is active (non-volatile) Logic 1 = device secured. Also read to determine Secondary Flash Protection Setting (non-volatile) that was specified in PSDsoft. No WRITEs.	*Table 166 on page 219*

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.

28.5.49 Port B structure

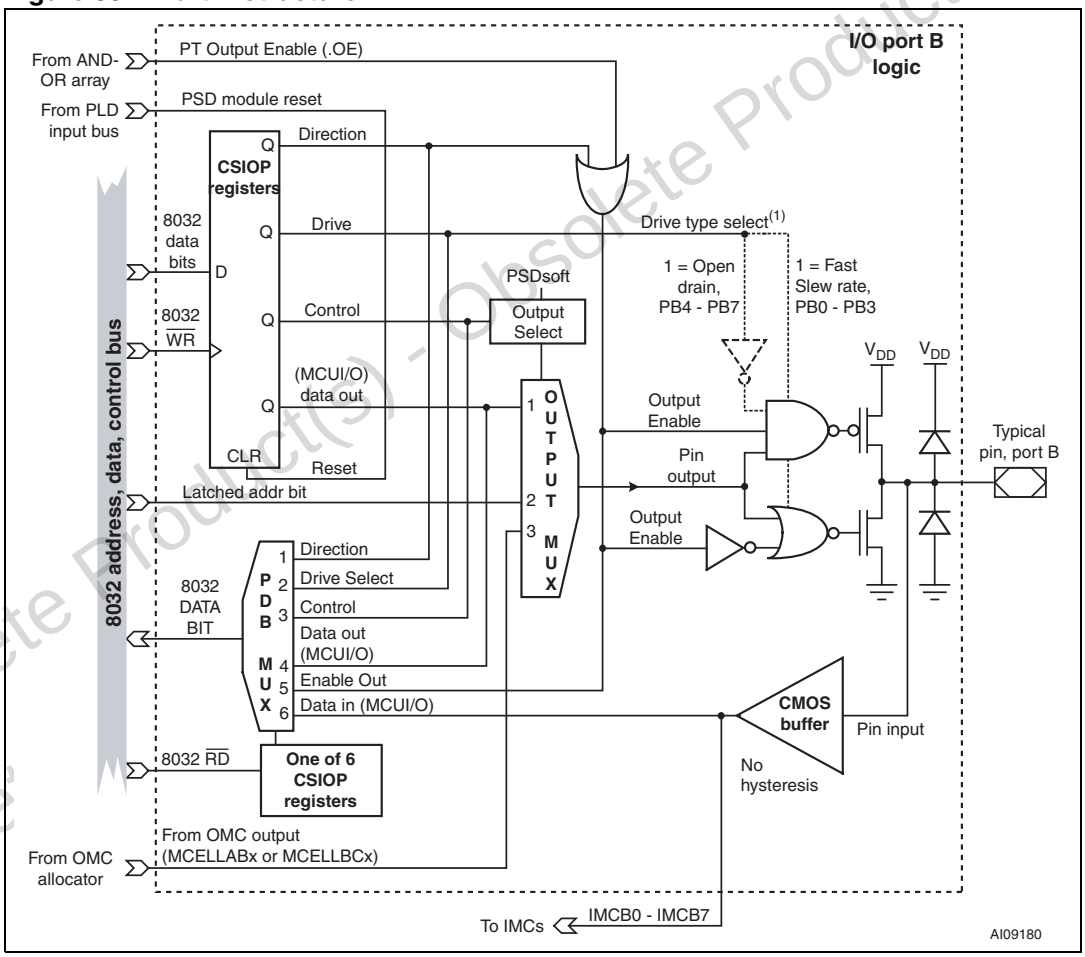
Port B supports the following operating modes:

- MCU I/O Mode
- GPLD Output Mode from Output Macrocells MCELLABx, or MCELLBCx (OMC allocator routes these signals)
- GPLD Input Mode to Input Macrocells IMCBx
- Latched Address Output Mode

Port B also supports Open Drain/Slew Rate output drive type options using the csiop Drive Select registers. Pins PB0-PB3 can be configured to fast slew rate, pins PB4-PB7 can be configured to Open Drain Mode.

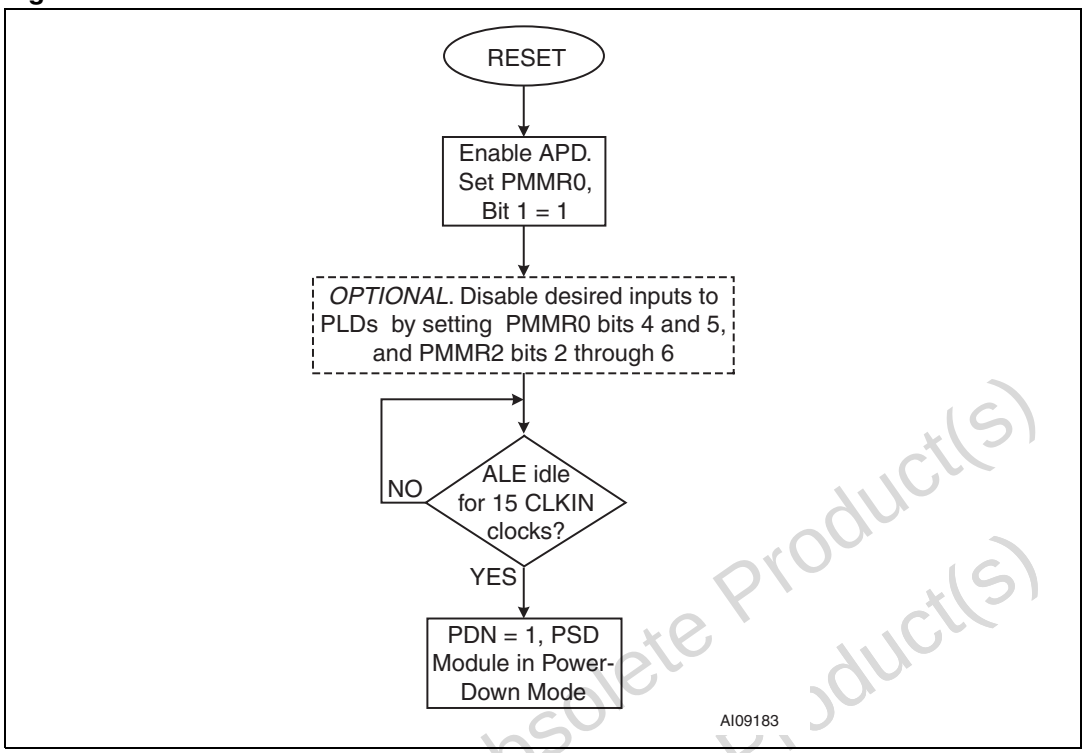
See *Figure 85* for detail.

Figure 85. Port B structure



Note: 1 Port pins PB0-PB3 are capable of Fast Slew Rate output drive option. Port pins PB4-PB7 are capable of Open Drain output option.

Figure 89. Power-down mode flowchart



28.5.55 Chip select input ($\overline{CSI}$)

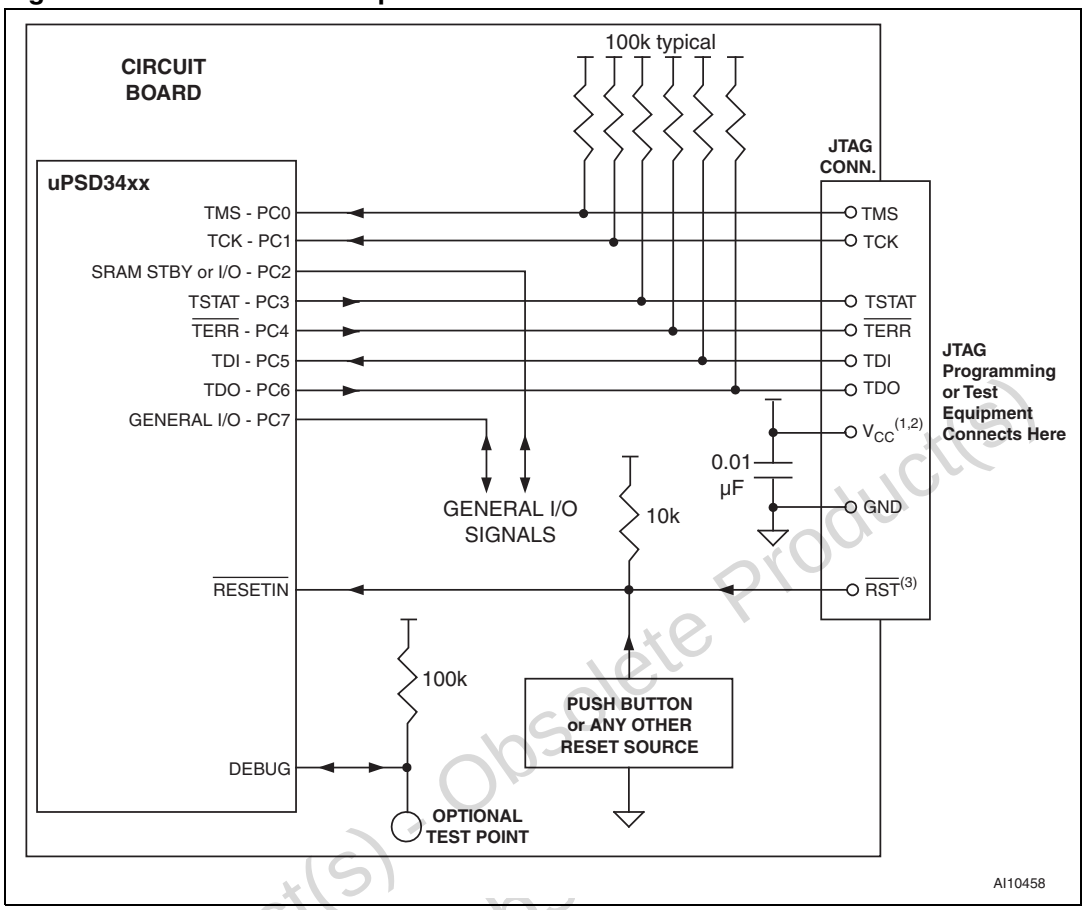
Pin PD2 of Port D can optionally be configured in PSDsoft Express as the PSD module Chip Select Input, $\overline{CSI}$, which is an active-low logic input. By default, pin PD2 does not have the $\overline{CSI}$ function.

When the $\overline{CSI}$ function is specified in PSDsoft Express, the $\overline{CSI}$ signal is automatically included in DPLD chip select equations for FSx, CSBOOTx, RS0, and CSIOP. When the $\overline{CSI}$ pin is driven to logic '0' from an external device, all of these memories will be available for READ and WRITE operations. When $\overline{CSI}$ is driven to logic '1,' none of these memories are available for selection, regardless of the address activity from the 8032, reducing power consumption. The state of the PLD and port I/O pins are not changed when $\overline{CSI}$ goes to logic '1' (disabled).

28.5.56 PLD non-turbo mode

The power consumption and speed of the PLDs are controlled by the Turbo Bit (Bit 3) in the csiop PMMR0 register. By setting this bit to logic '1,' the Turbo mode is turned off and both PLDs consume only standby current when ALL PLD inputs have no transitions for an extended time (65 ns for 5 V devices, 100 ns for 3.3 V devices), significantly reducing current consumption. The PLDs will latch their outputs and go to standby, drawing very little current. When Turbo mode is off, PLD propagation delay time is increased as shown in the AC specifications for the PSD module. Since this additional propagation delay also effects the DPLD, the response time of the memories on the PSD module is also lengthened by that same amount of time. If Turbo mode is off, the user should add an additional wait state to the 8032 BUSCON SFR register if the 8032 clock frequency is higher that a particular value. Please refer to *Table 51 on page 90* in the MCU module section.

Figure 92. Recommended 6-pin JTAG connections



1. For 5 V UPSD34xx devices, pull-up resistors and V_{CC} pin on the JTAG connector should be connected to 5 V system V_{DD}.
2. For 3.3 V UPSD34xx devices, pull-up resistors and V_{CC} pin on the JTAG connector should be connected to 3.3 V system V_{CC}.
3. This signal is driven by an Open-Drain output in the JTAG equipment, allowing more than one source to activate RESET_IN.

28.6.6 Recommended JTAG connector

There is no industry standard JTAG connector. STMicroelectronics recommends a specific JTAG connector and pinout for uPSD3xxx so programming and debug equipment will easily connect to the circuit board. The user does not have to use this connector if there is a different connection scheme.

The recommended connector scheme can accept a standard 14-pin ribbon cable connector (2 rows of 7 pins on 0.1” centers, 0.025” square posts, standard keying) as shown in *Figure 93*. See the STMicroelectronics “FlashLINK, FL-101 User Manual” for more information.

Table 209. AC signal behavior symbols for timing

Letter	Meaning
t	Time
L	Logic level low or ALE
H	Logic level high
V	Valid
X	No longer a valid logic level
Z	Float
PW	Pulse width

Note: Example: t_{AVLX} = time from address valid to ALE invalid.

Figure 97. Switching waveforms – key