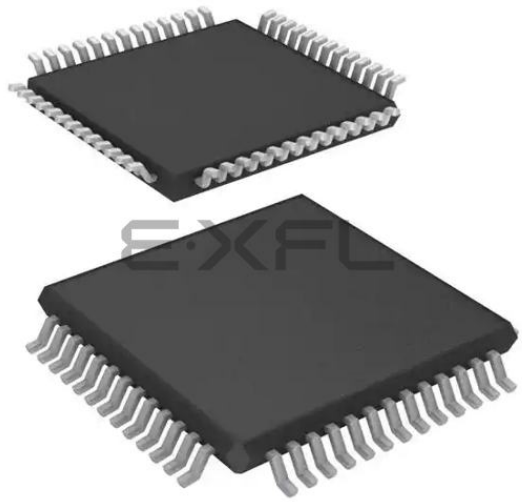


Welcome to E-XFL.COM

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)	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-TQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/upsd3312dv-40t6

Contents

Table 2. Pin definitions (continued)

Port pin	Signal name	80-Pin num.	52-Pin num.(1)	In/Out	Function		
					Basic	Alternate 1	Alternate 2
PA0		35	N/A	I/O	General I/O port pin		All Port A pins support: – PLD Macro-cell outputs, or – PLD inputs, or – Latched Address Out (A0-A7), or – Peripheral I/O mode
PA1		34	N/A	I/O	General I/O port pin		
PA2		32	N/A	I/O	General I/O port pin		
PA3		28	N/A	I/O	General I/O port pin		
PA4		26	N/A	I/O	General I/O port pin		
PA5		24	N/A	I/O	General I/O port pin		
PA6		22	N/A	I/O	General I/O port pin		
PA7		21	N/A	I/O	General I/O port pin		

1. N/A = Signal Not Available on 52-pin package.

Table 7. Logical instruction set (continued)

Mnemonic[1] and use		Description	Length/cycles
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.

Table 8. Data transfer instruction set

Mnemonic[1] and use		Description	Length/cycles
MOV	A, Rn	Move register to ACC	1 byte/1 cycle
MOV	A, direct	Move direct byte to ACC	2 byte/1 cycle
MOV	A, @Ri	Move indirect SRAM to ACC	1 byte/1 cycle
MOV	A, #data	Move immediate data to ACC	2 byte/1 cycle
MOV	Rn, A	Move ACC to register	1 byte/1 cycle
MOV	Rn, direct	Move direct byte to register	2 byte/2 cycle
MOV	Rn, #data	Move immediate data to register	2 byte/1 cycle
MOV	direct, A	Move ACC to direct byte	2 byte/1 cycle
MOV	direct, Rn	Move register to direct byte	2 byte/2 cycle
MOV	direct, direct	Move direct byte to direct	3 byte/2 cycle
MOV	direct, @Ri	Move indirect SRAM to direct byte	2 byte/2 cycle
MOV	direct, #data	Move immediate data to direct byte	3 byte/2 cycle
MOV	@Ri, A	Move ACC to indirect SRAM	1 byte/1 cycle

Table 8. Data transfer instruction set (continued)

Mnemonic[1] and use		Description	Length/cycles
MOV	@Ri, direct	Move direct byte to indirect SRAM	2 byte/2 cycle
MOV	@Ri, #data	Move immediate data to indirect SRAM	2 byte/1 cycle
MOV	DPTR, #data16	Load Data Pointer with 16-bit constant	3 byte/2 cycle
MOVC	A, @A+DPTR	Move code byte relative to DPTR to ACC	1 byte/2 cycle
MOVC	A, @A+PC	Move code byte relative to PC to ACC	1 byte/2 cycle
MOVX	A, @Ri	Move XDATA (8-bit addr) to ACC	1 byte/2 cycle
MOVX	A, @DPTR	Move XDATA (16-bit addr) to ACC	1 byte/2 cycle
MOVX	@Ri, A	Move ACC to XDATA (8-bit addr)	1 byte/2 cycle
MOVX	@DPTR, A	Move ACC to XDATA (16-bit addr)	1 byte/2 cycle
PUSH	direct	Push direct byte onto stack	2 byte/2 cycle
POP	direct	Pop direct byte from stack	2 byte/2 cycle
XCH	A, Rn	Exchange register with ACC	1 byte/1 cycle
XCH	A, direct	Exchange direct byte with ACC	2 byte/1 cycle
XCH	A, @Ri	Exchange indirect SRAM with ACC	1 byte/1 cycle
XCHD	A, @Ri	Exchange low-order digit indirect SRAM with ACC	1 byte/1 cycle

1. All mnemonics copyrighted ©Intel Corporation 1980.

Table 9. Boolean variable manipulation instruction set

Mnemonic[1] and use		Description	Length/cycles
CLR	C	Clear carry	1 byte/1 cycle
CLR	bit	Clear direct bit	2 byte/1 cycle
SETB	C	Set carry	1 byte/1 cycle
SETB	bit	Set direct bit	2 byte/1 cycle
CPL	C	Compliment carry	1 byte/1 cycle
CPL	bit	Compliment direct bit	2 byte/1 cycle

11 Dual data pointers

XDATA is accessed by the External Direct addressing mode, which uses a 16-bit address stored in the DPTR register. Traditional 8032 architecture has only one DPTR register. This is a burden when transferring data between two XDATA locations because it requires heavy use of the working registers to manipulate the source and destination pointers.

However, the UPSD33xx has two data pointers, one for storing a source address and the other for storing a destination address. These pointers can be configured to automatically increment or decrement after each data transfer, further reducing the burden on the 8032 and making this kind of data movement very efficient.

11.1 Data Pointer Control register, DPTC (85h)

By default, the DPTR register of the UPSD33xx will behave no different than in a standard 8032 MCU. The DPSEL0 Bit of SFR register DPTC shown in *Table 13 on page 56*, selects which one of the two “background” data pointer registers (DPTR0 or DPTR1) will function as the traditional DPTR register at any given time. After reset, the DPSEL0 Bit is cleared, enabling DPTR0 to function as the DPTR, and firmware may access DPTR0 by reading or writing the traditional DPTR register at SFR addresses 82h and 83h. When the DPSEL0 bit is set, then the DPTR1 register functions as DPTR, and firmware may now access DPTR1 through SFR registers at 82h and 83h. The pointer which is not selected by the DPSEL0 bit remains in the background and is not accessible by the 8032. If the DPSEL0 bit is never set, then the UPSD33xx will behave like a traditional 8032 having only one DPTR register.

To further speed XDATA to XDATA transfers, the SFR bit, AT, may be set to automatically toggle the two data pointers, DPTR0 and DPTR1, each time the standard DPTR register is accessed by a MOVX instruction. This eliminates the need for firmware to manually manipulate the DPSEL0 bit between each data transfer.

Detailed description for the SFR register DPTC is shown in *Table 13*.

Table 13. DPTC: Data Pointer Control register (SFR 85h, reset value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	AT	–	–	–	–	–	DPSEL0

Table 14. DPTC register bit definition

Bit	Symbol	R/W	Definition
7	–	–	Reserved
6	AT	R,W	0 = Manually Select Data Pointer 1 = Auto Toggle between DPTR0 and DPTR1
5-1	–	–	Reserved
0	DPSE0	R,W	0 = DPTR0 Selected for use as DPTR 1 = DPTR1 Selected for use as DPTR

Reading port pin vs. reading port latch

When firmware reads the GPIO ports, sometimes the actual port pin is sampled in hardware, and sometimes the port SFR latch is read and not the actual pin, depending on the type of MCU instruction used. These two data paths are shown in *Figure 16* through *Figure 18 on page 80*. SFR latches are read (and not the pins) only when the read is part of a *read-modify-write* instruction and the write destination is a bit or bits in a port SFR. These instructions are: ANL, ORL, XRL, JBC, CPL, INC, DEC, DJNZ, MOV, CLR, and SETB. All other types of reads to port SFRs will read the actual pin logic level and not the port latch. This is consistent with 8051 architecture.

Figure 15. MCU module port pin function routing

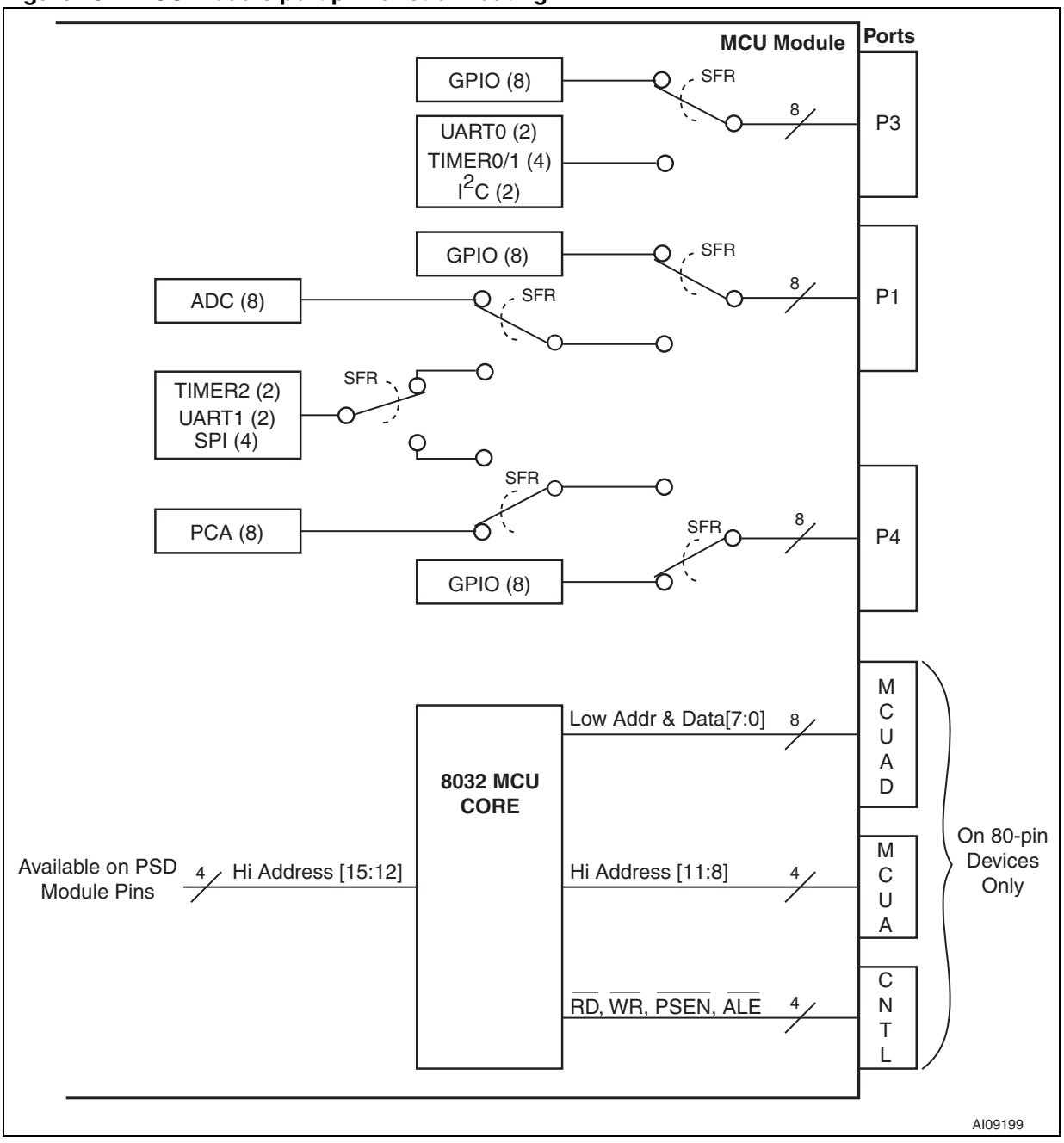


Figure 16. MCU I/O cell block diagram for Port 1

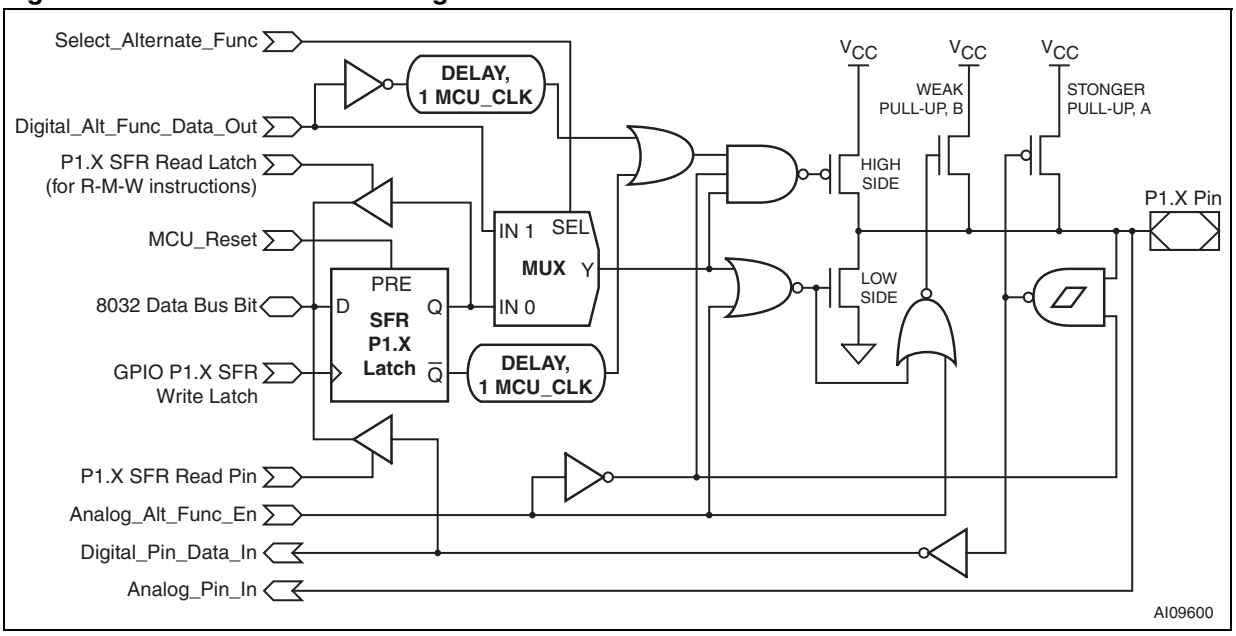


Figure 17. MCU I/O cell block diagram for Port 3

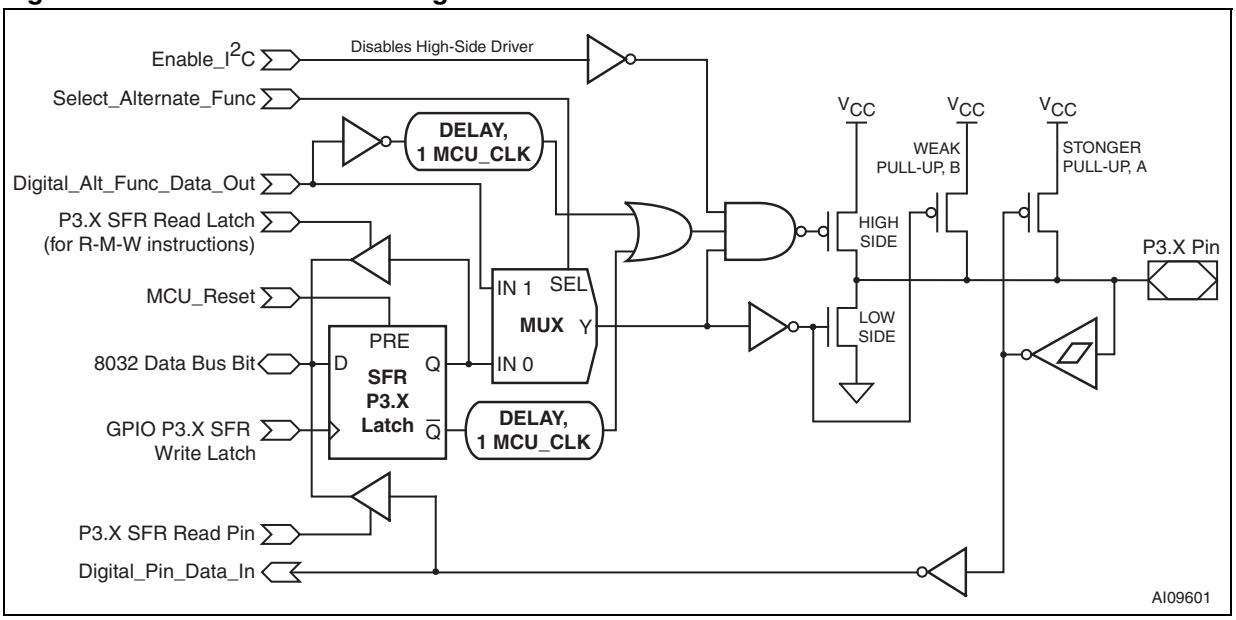


Table 61. Commonly used baud rates generated from Timer 2 (T2CON = 34h)

f_{OSC} MHz	Desired baud rate	Timer 2 SFRs		Resulting baud rate	Baud rate deviation
		RCAP2H (hex)	RCAP2L(hex)		
40.0	115200	FF	F5	113636	-1.36%
40.0	57600	FF	EA	56818	-1.36%
40.0	28800	FF	D5	29070	0.94%
40.0	19200	FF	BF	19231	0.16%
40.0	9600	FF	7E	9615	0.16%
36.864	115200	FF	F6	115200	0
36.864	57600	FF	EC	57600	0
36.864	28800	FF	D8	28800	0
36.864	19200	FF	C4	19200	0
36.864	9600	FF	88	9600	0
36.0	28800	FF	D9	28846	0.16%
36.0	19200	FF	C5	19067	-0.69%
36.0	9600	FF	8B	9615	0.16%
24.0	57600	FF	F3	57692	0.16%
24.0	28800	FF	E6	28846	0.16%
24.0	19200	FF	D9	19231	0.16%
24.0	9600	FF	B2	9615	0.16%
12.0	28800	FF	F3	28846	0.16%
12.0	9600	FF	D9	9615	0.16%
11.0592	115200	FF	FD	115200	0
11.0592	57600	FF	FA	57600	0
11.0592	28800	FF	F4	28800	0
11.0592	19200	FF	EE	19200	0
11.0592	9600	FF	DC	9600	0
3.6864	115200	FF	FF	115200	0
3.6864	57600	FF	FE	57600	0
3.6864	28800	FF	FC	28800	0
3.6864	19200	FF	FA	19200	0
3.6864	9600	FF	F4	9600	0
1.8432	19200	FF	FD	19200	0
1.8432	9600	FF	FA	9600	0

Table 66. SCON1 register bit definition (continued)

Bit	Symbol	R/W	Definition
1	TI	R,W	**Transmit Interrupt flag.** Causes interrupt at end of 8th bit time when transmitting in mode 0, or at beginning of stop bit transmission in other modes. Must clear flag with firmware.
0	RI	R,W	**Receive Interrupt flag.** Causes interrupt at end of 8th bit time when receiving in mode 0, or halfway through stop bit reception in other modes (see SM2 for exception). Must clear this flag with firmware.

21.3 UART baud rates

The baud rate in mode 0 is fixed:

Mode 0 Baud Rate = f_{OSC} / 12

The baud rate in mode 2 depends on the value of the bit SMOD in the SFR named PCON. If SMOD = 0 (default value), the baud rate is 1/64 the oscillator frequency, f_{OSC}. If SMOD = 1, the baud rate is 1/32 the oscillator frequency.

Mode 2 Baud Rate = (2^{SMOD} / 64) x f_{OSC}

Baud rates in modes 1 and 3 are determined by the Timer 1 or Timer 2 overflow rate.

21.3.1 Using Timer 1 to generate baud rates

When Timer 1 is used as the baud rate generator (bits RCLK = 0, TCLK = 0), the baud rates in modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

Mode 1,3 Baud Rate =
(2^{SMOD} / 32) x (Timer 1 overflow rate)

The Timer 1 Interrupt should be disabled in this application. The Timer itself can be configured for either “timer” or “counter” operation, and in any of its 3 running modes. In the most typical applications, it is configured for “timer” operation, in the Auto-reload mode (high nibble of the SFR TMOD = 0010B). In that case the baud rate is given by the formula:

Mode 1,3 Baud Rate =
(2^{SMOD} / 32) x (f_{OSC} / (12 x [256 – (TH1)]))

Table 67 on page 111 lists various commonly used baud rates and how they can be obtained from Timer 1.

Table 79. S1ADR register bit definition

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

23.12 I²C START sample setting (S1SETUP)

The S1SETUP register (*Table 80 on page 132*) determines how many times an I²C bus START condition will be sampled before the SIOE validates the START condition, giving the SIOE the ability to reject noise or illegal transmissions.

Because the minimum duration of an START condition varies with I²C bus speed (f_{SCL}), and also because the UPSD33xx may be operated with a wide variety of frequencies (f_{OSC}), it is necessary to scale the number of samples per START condition based on f_{OSC} and f_{SCL}.

In Slave mode, the SIOE recognizes the beginning of a START condition when it detects a 1-to-0 transition on the SDA bus line while the SCL line is high (see *Figure 38 on page 124*). The SIOE must then validate the START condition by sampling the bus lines to ensure SDA remains low and SCL remains high for a minimum amount of hold time, t_{HLDSTA}. Once validated, the SIOE begins receiving the address byte that follows the START condition.

If the EN_SS Bit (in the S1SETUP register) is not set, then the SIOE will sample only once after detecting the 1-to-0 transition on SDA. This single sample is taken $1/f_{OSC}$ seconds after the initial 1-to-0 transition was detected. However, more samples should be taken to ensure there is a valid START condition.

To take more samples, the SIOE should be initialized such that the EN_SS Bit is set, and a value is written to the SMPL_SET[6:0] field of the S1SETUP register to specify how many samples to take. The goal is to take a good number of samples during the minimum START condition hold time, t_{HLDSTA}, but no so many samples that the bus will be sampled after t_{HLDSTA} expires.

Table 82 describes the relationship between the contents of S1SETUP and the resulting number of I²C bus samples that SIOE will take after detecting the 1-to-0 transition on SDA of a START condition.

Note: ***Important:*** *Keep in mind that the time between samples is always $1/f_{OSC}$.*

The minimum START condition hold time, t_{HLDSTA}, is different for the three common I²C speed categories per *Table 83 on page 133*.

Table 80. S1SETUP: I²C START Condition Sample Setup register (SFR DBh, reset value 00h)

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

Table 107. PCA Status register PCASTA (SFR 0A5h, Reset Value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OVF1	INTF5	INTF4	INTF3	OVF0	INTF2	INTF1	INTF0

Table 108. PCASTA register bit definition

Bit	Symbol	Function
7	OFV1	**PCA1 Counter OverFlow flag** Set by hardware when the counter rolls over. OVF1 flags an interrupt if Bit EOVFI in PCACON1 is set. OVF1 may be set with either hardware or software but can only be cleared with software.
6	INTF5	**TCM5 Interrupt flag** Set by hardware when a match or capture event occurs. Must be clear with software.
5	INTF4	**TCM4 Interrupt flag** Set by hardware when a match or capture event occurs. Must be clear with software.
4	INTF3	**TCM3 Interrupt flag** Set by hardware when a match or capture event occurs. Must be clear with software.
3	OVF0	**PCA0 Counter OverFlow flag** Set by hardware when the counter rolls over. OVF0 flags an interrupt if Bit EOVFI in PCACON0 is set. OVF1 may be set with either hardware or software but can only be cleared with software.
2	INTF2	**TCM2 Interrupt flag** Set by hardware when a match or capture event occurs. Must be clear with software.
1	INTF1	**TCM1 Interrupt flag** Set by hardware when a match or capture event occurs. Must be clear with software.
0	INTF0	**TCM0 Interrupt flag** Set by hardware when a match or capture event occurs. Must be clear with software.

and PSEL1) used to enable a special data bus repeater function on Port A, referred to as Peripheral I/O mode. There are 69 DPLD input signals which include: 8032 address and control signals, Page register outputs, PSD module Port pin inputs, and GPLD logic feedback.

27.1.10 PLD #2, General PLD (GPLD)

This programmable logic is used to create both combinatorial and sequential general purpose logic (see *Figure 64 on page 199*). The GPLD contains 16 Output Macrocells (OMCs) and 20 Input Macrocells (IMCs). Output Macrocell registers are unique in that they have direct connection to the 8032 data bus allowing them to be loaded and read directly by the 8032 at runtime through OMC registers in csiop. This direct access is good for making small peripheral devices (shifters, counters, state machines, etc.) that are accessed directly by the 8032 with little overhead. There are 69 GPLD inputs which include: 8032 address and control signals, Page register outputs, PSD module Port pin inputs, and GPLD feedback.

27.1.11 OMCs

There are two banks of eight OMCs inside the GPLD, MCELLAB, and MCELLBC, totalling 16 OMCs all together. Each individual OMC is a base logic element consisting of a flip-flop and some AND-OR logic (*Figure 65 on page 201*). The general structure of the GPLD with OMCs is similar in nature to a 22V10 PLD device with the familiar sum-of-products (AND-OR) construct. True and compliment versions of 69 input signals are available to the inputs of a large AND-OR array. AND-OR array outputs feed into an OR gate within each OMC, creating up to 10 product-terms for each OMC. Logic output of the OR gate can be passed on as combinatorial logic or combined with a flip-flop within in each OMC to realize sequential logic. OMC outputs can be used as a buried nodes driving internal feedback to the AND-OR array, or OMC outputs can be routed to external pins on Ports A, B, or C through the OMC Allocator.

27.1.12 OMC allocator

The OMC allocator (*Figure 66 on page 202*) will route eight of the OMCs from MCELLAB to pins on either Port A or Port B, and will route eight of the OMCs from MCELLBC to pins on either Port B or Port C, based on what is specified in PSDsoft Express.

27.1.13 IMCs

Inputs from pins on Ports A, B, and C are routed to IMCs for conditioning (clocking or latching) as they enter the chip, which is good for sampling and debouncing inputs. Alternatively, IMCs can pass port input signals directly to PLD inputs without clocking or latching (*Figure 67 on page 205*). The 8032 may read the IMCs asynchronously at any time through IMC registers in csiop.

Note: The JTAG signals TDO, TDI, TCK, and TMS on Port C do not route through IMCs, but go directly to JTAG logic.

Table 117. Flash memory instruction sequences(1)(2)(3)

Instr. sequence	Bus cycle 1	Bus cycle 2	Bus cycle 3	Bus cycle 4	Bus cycle 5	Bus cycle 6	Bus cycle 7	Link
Read Memory Contents (Read Array mode)	Read byte from any valid Flash memory addr							*Section 2 7.4.4*
Program (write) a Byte to Flash memory	Write AAh to X555h (*unlock*)	Write 55h to XAAAh (*unlock*)	Write A0h to X555h (*command*)	Write data byte to address				*Section 2 7.4.9*
Bypass Unlock	Write AAh to X555h (*unlock*)	Write 55h to XAAAh (*unlock*)	Write 20h to X555h (*command*)					*Section 2 7.4.13*
Program a Byte to Flash memory with Bypassed Unlock	Write A0h to XXXXh (*command*)	Write data byte to address						*Section 2 7.4.13*
Reset Bypass Unlock	Write 90h to XXXXh (*command*)	Write 00h to XXXXh (*command*)						*Section 2 7.4.13*
Flash Bulk Erase(3)	Write AAh to X555h (*unlock*)	Write 55h to XAAAh (*unlock*)	Write 80h to X555h (*command*)	Write AAh to X555h (*unlock*)	Write 55h to XAAAh (*unlock*)	Write 10h to X555h (*command*)		*Section 2 7.4.15*
Flash Sector Erase	Write AAh to X555h (*unlock*)	Write 55h to XAAAh (*unlock*)	Write 80h to X555h (*command*)	Write AAh to X555h (*unlock*)	Write 55h to XAAAh (*unlock*)	Write 30h to desired Sector (*command*)	Write 30h to another Sector (*command*)	*Section 2 7.4.16*
Suspend Sector Erase	Write B0h to address that activates FSx or CSBOOTx where erase is in progress (*command*)							*Section 2 7.4.17*

1. All values are in hexadecimal, X = Don't care
2. 8032 addresses A12 through A15 are "Don't care" during the instruction sequence decoding. Only address bits A0-A11 are used during decoding of Flash memory instruction sequences. The individual sector select signal (FS0 - FS7 or CSBOOT0-CSBOOT3) which is active during the instruction sequence determines the complete address.
3. Directing this command to any individual sector within a Flash memory array will invoke the bulk erase of all Flash memory sectors within that array.

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.

logic where more product terms may be available. The following list summarizes how product terms are allocated to each OMC, as shown in *Table 122 on page 203*.

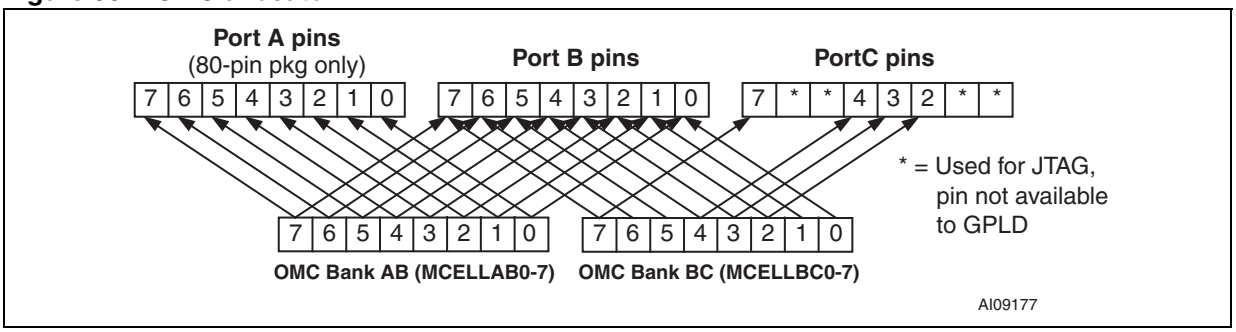
- MCELLAB0-MCELLAB7 each have three native product terms and may borrow up to six more
- MCELLBC0-MCELLBC3 each have four native product terms and may borrow up to five more
- MCELLBC4-MCELLBC7 each have four native product terms and may borrow up to six more.

Native product terms come from the AND-OR Array. Each OMC may borrow product terms only from certain other OMCs, if they are not in use. Product term allocation does not add any propagation delay to the logic. The fitter report generated by PSDsoft Express will show any PT allocation that has occurred.

If an equation requires more product terms than are available to it through PT allocation, then “external” product terms are required, which consumes other OMCs. This is called product term expansion and also happens automatically in PSDsoft Express as needed. PT expansion causes additional propagation delay because an additional OMC is consumed by the expansion process and it’s output is rerouted (or fed back) into the AND-OR array. The user can examine the fitter report generated by PSDsoft Express to see resulting PT allocation and PT expansion (expansion will have signal names, such as ‘*.fb_0’ or ‘*.fb_1’). PSDsoft Express will always try to fit the logic design first by using PT allocation, and if that is not sufficient then PSDsoft Express will use PT expansion.

Product term expansion may occur in the DPLD for complex chip select equations for Flash memory sectors and for SRAM, but this is a rare occurrence. If PSDsoft Express does use PT expansion in the DPLD, it results in an approximate 15ns additional propagation delay for that chip select signal, which gives 15ns less time for the memory to respond. Be aware of this and consider adding a wait state to the 8032 bus access (using the SFR named, BUSCON), or lower the 8032 clock frequency to avoid problems with memory access time.

Figure 66. OMC allocator



The Port Data Buffer (PDB) provides feedback to the 8032 and allows only one source at a time to be read when the 8032 reads various csiop registers. There is one PDB for each port pin enabling the 8032 to read the following on a pin-by-pin basis:

1. MCU I/O signal direction setting (csiop Direction reg)
2. Pin drive type setting (csiop Drive Select reg)
3. Latched Addr Out mode setting (csiop Control reg)
4. MCU I/O pin output setting (csiop Data Out reg)
5. Output Enable of pin driver (csiop Enable Out reg)
6. MCU I/O pin input (csiop Data In reg)

A port pin's output enable signal is controlled by a two input OR gate whose inputs come from: a product term of the AND-OR array; the output of the csiop Direction register. If an output enable from the AND-OR Array is not defined, and the port pin is not defined as an OMC output, and if Peripheral I/O mode is not used, then the csiop Direction register has sole control of the OE signal.

As shown in *Figure 68 on page 208*, a physical port pin is connected to the I/O Port logic and is also separately routed to an IMC, allowing the 8032 to read a port pin by two different methods (MCU I/O input mode or read the IMC).

27.4.36 Port operating modes

I/O Port logic has several modes of operation. *Table 125 on page 204* summarizes which modes are available on each port. Each of the port operating modes are described in following sections. Some operating modes can be defined using PSDsoft Express, and some by the 8032 writing to the csiop registers at run-time, and some require both. For example, PLD I/O, Latched Address Out, and Peripheral I/O modes must be defined in PSDsoft Express and programmed into the device using JTAG, but an additional step must happen at run-time to activate Latched Address Out mode and Peripheral I/O mode, but not needed for PLD I/O. In another example, MCU I/O mode is controlled completely by the 8032 at run-time and only a simple pin name declaration is needed in PSDsoft Express for documentation.

Table 131 on page 209 summarizes what actions are needed in PSDsoft Express and what actions are required by the 8032 at run-time to achieve the various port functions.

Figure 69. Simple PLD logic example

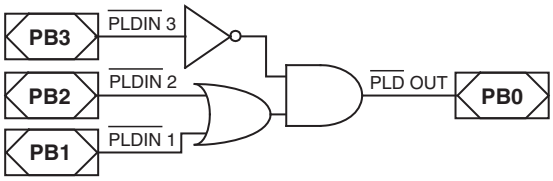


Figure 70. Pin declarations in PSDsoft Express for simple PLD example

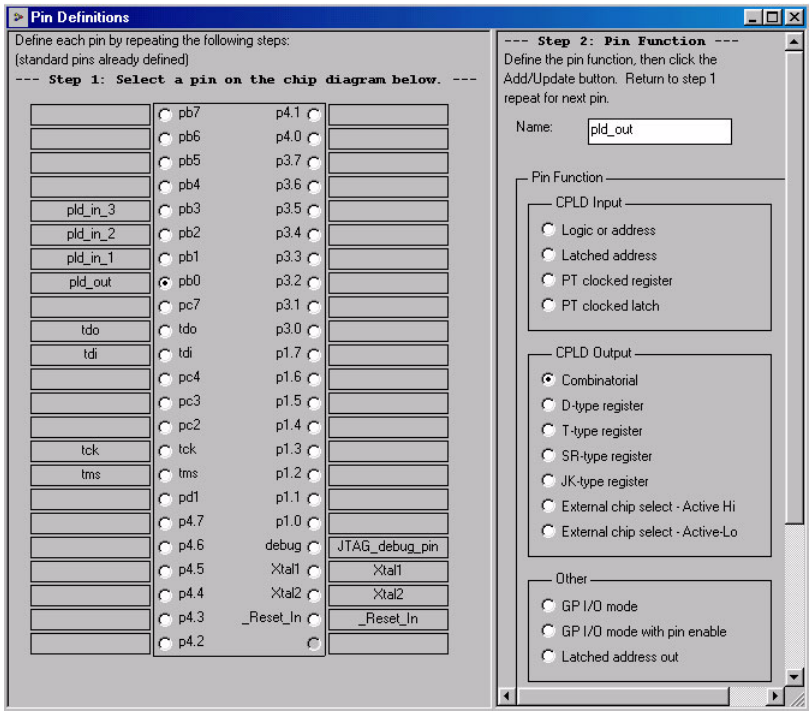
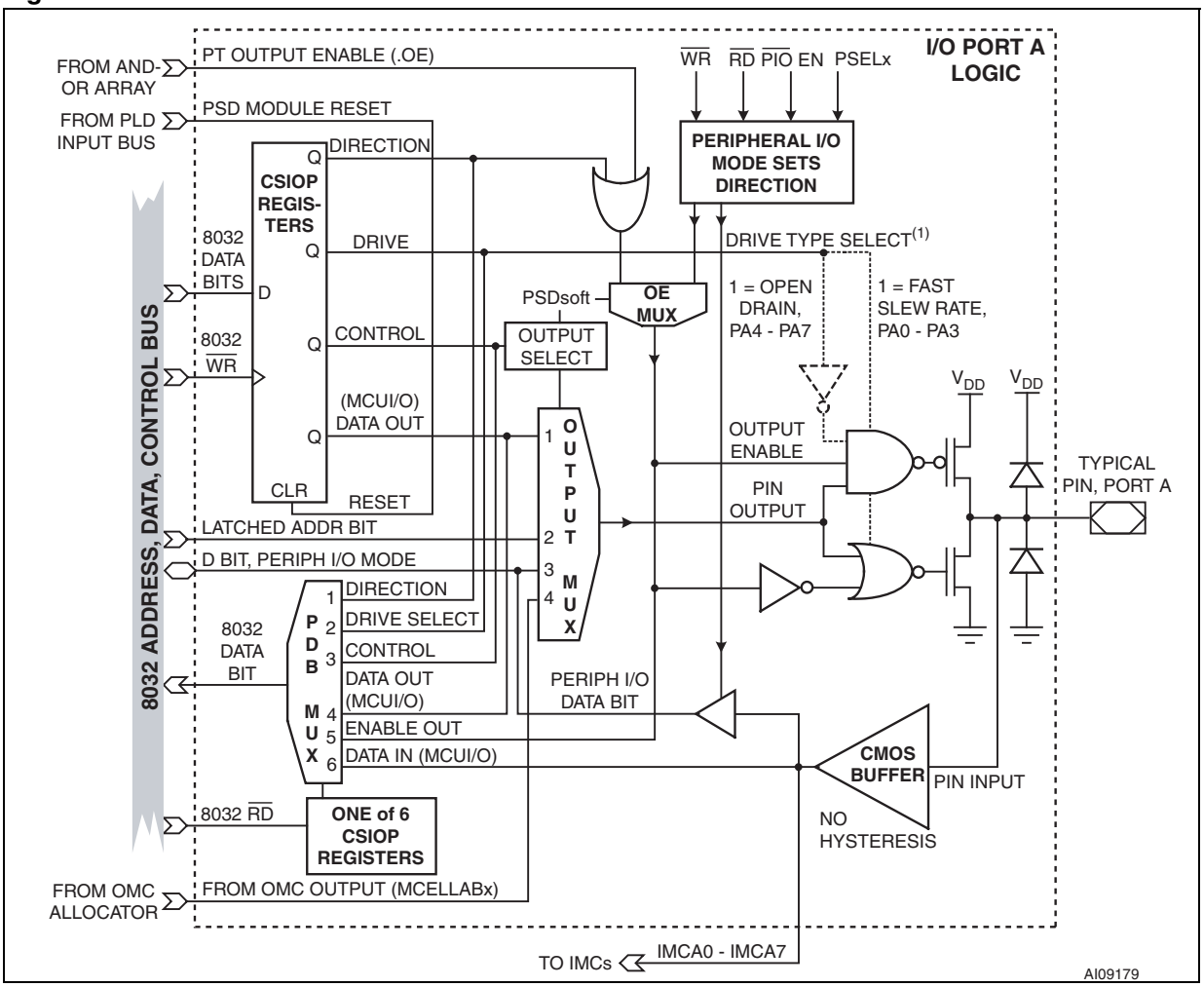


Figure 73. Port A structure



1. Port pins PA0-PA3 are capable of Fast Slew Rate output drive option. Port pins PA4-PA7 are capable of Open Drain output option.

27.4.48 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 74 on page 220* for details.

Figure 77. Automatic Power-down (APD) unit

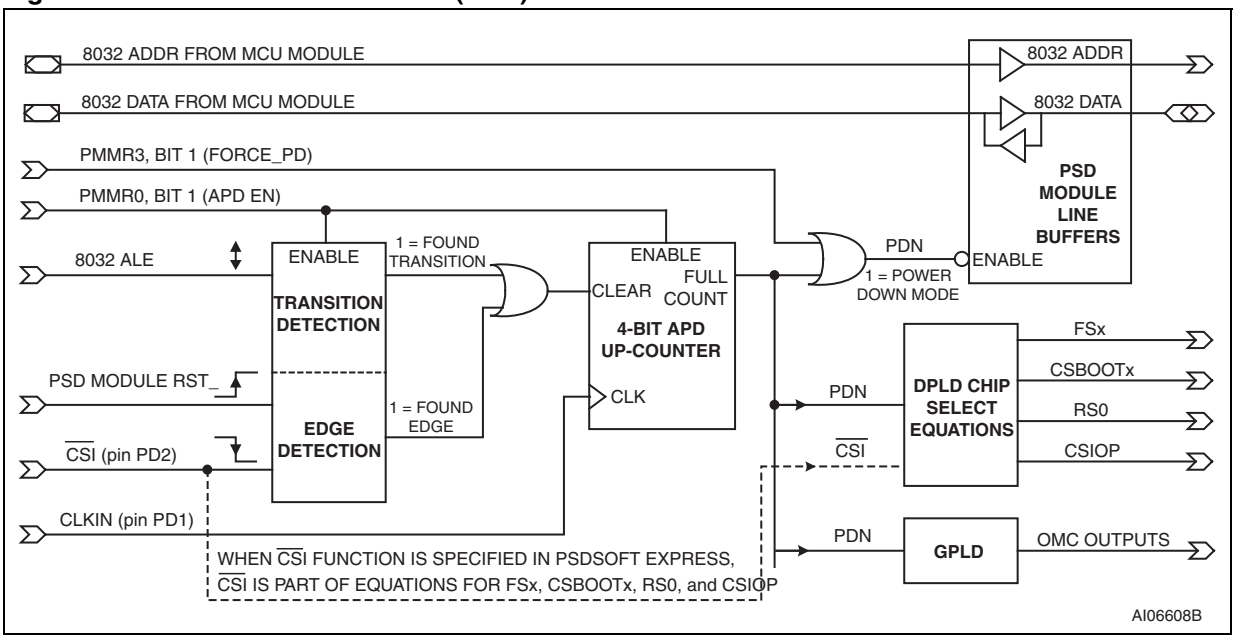


Figure 78. Power-down mode flowchart

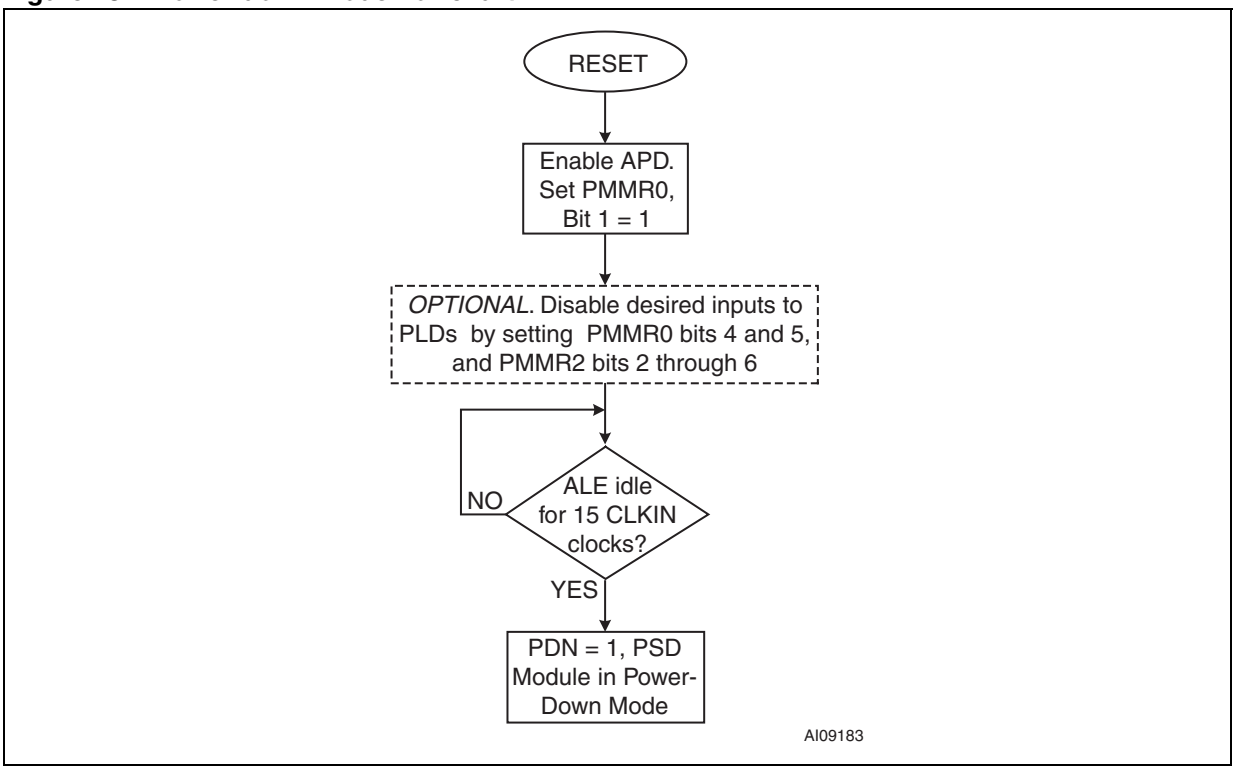


Figure 99. External clock cycle

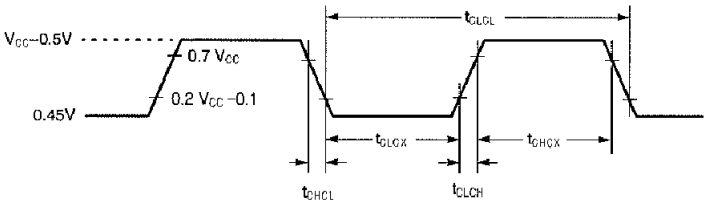


Figure 100. PSD module AC measurement I/O waveform

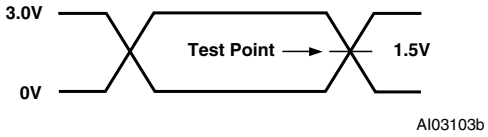


Figure 101. PSD module AC measurement load circuit

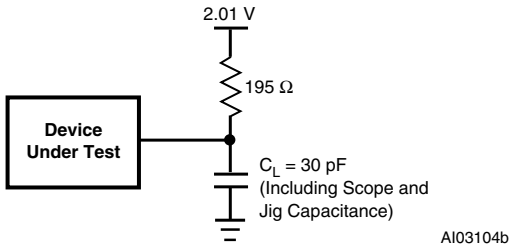


Table 190. I/O pin capacitance

Symbol	Parameter[1]	Test condition	Typ.[2]	Max.	Unit
C_{IN}	Input capacitance (for input pins)	V_{IN} = 0 V	4	6	pF
C_{OUT}	Output capacitance (for input/output pins)[3]	V_{OUT} = 0 V	8	12	pF

1. Sampled only, not 100% tested.
2. Typical values are for T_A = 25 °C and nominal supply voltages.
3. Maximum for MCU Address and Data lines is 20 pF each.