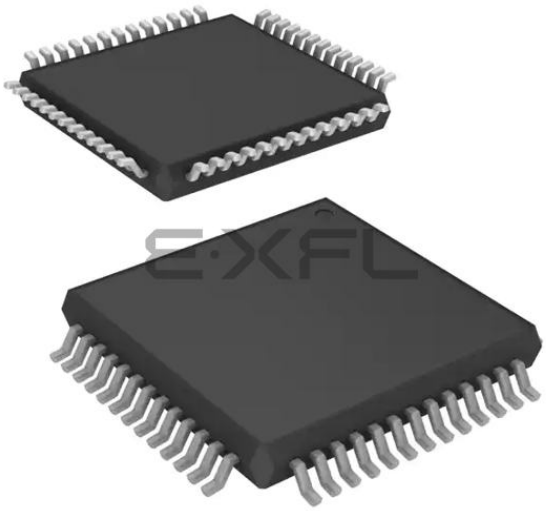


Welcome to E-XFL.COM

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	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/upsd3354dv-40t6

Table 2. Pin definitions

Port pin	Signal name	80-Pin num.	52-Pin num.(1)	In/Out	Function		
					Basic	Alternate 1	Alternate 2
MCUAD0	AD0	36	N/A	I/O	External bus Multiplexed address/data bus A0/D0		
MCUAD1	AD1	37	N/A	I/O	Multiplexed address/data bus A1/D1		
MCUAD2	AD2	38	N/A	I/O	Multiplexed address/data bus A2/D2		
MCUAD3	AD3	39	N/A	I/O	Multiplexed address/data bus A3/D3		
MCUAD4	AD4	41	N/A	I/O	Multiplexed address/data bus A4/D4		
MCUAD5	AD5	43	N/A	I/O	Multiplexed address/data bus A5/D5		
MCUAD6	AD6	45	N/A	I/O	Multiplexed address/data bus A6/D6		
MCUAD7	AD7	47	N/A	I/O	Multiplexed address/data bus A7/D7		
MCUA8	A8	51	N/A	O	External bus, Addr A8		
MCUA9	A9	53	N/A	O	External bus, Addr A9		
MCUA10	A10	55	N/A	O	External bus, Addr A10		
MCUA11	A11	57	N/A	O	External bus, Addr A11		
P1.0	T2 ADC0	52	34	I/O	General I/O port pin	Timer 2 Count input (T2)	ADC Channel 0 input (ADC0)
P1.1	T2X ADC1	54	35	I/O	General I/O port pin	Timer 2 Trigger input (T2X)	ADC Channel 1 input (ADC1)
P1.2	RxD1 ADC2	56	36	I/O	General I/O port pin	UART1 or IrDA Receive (RxD1)	ADC Channel 2 input (ADC2)
P1.3	TXD1 ADC3	58	37	I/O	General I/O port pin	UART or IrDA Transmit (TxD1)	ADC Channel 3 input (ADC3)

13.1 Individual interrupt sources

13.1.1 External interrupts Int0 and Int1

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

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

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

13.1.2 Timer 0 and 1 overflow interrupt

Timer 0 and Timer 1 interrupts are generated by the flag bits TF0 and TF1 when there is an overflow condition in the respective Timer/Counter register (except for Timer 0 in mode 3).

13.1.3 Timer 2 overflow interrupt

This interrupt is generated to the MCU by a logical OR of flag bits, TF2 and EXE2. The ISR must read the flag bits to determine the cause of the interrupt.

- TF2 is set by an overflow of Timer 2.
- EXE2 is generated by the falling edge of a signal on the external pin, T2X (pin P1.1).

13.1.4 UART0 and UART1 interrupt

Each of the UARTs have identical interrupt structure. For each UART, a single interrupt is generated to the MCU by the logical OR of the flag bits, RI (byte received) and TI (byte transmitted).

The ISR must read flag bits in the SFR named SCON0 for UART0, or SCON1 for UART1 to determine the cause of the interrupt.

13.1.5 SPI interrupt

The SPI interrupt has four interrupt sources, which are logically ORed together when interrupting the MCU. The ISR must read the flag bits to determine the cause of the interrupt.

A flag bit is set for: end of data transmit (TEISF); data receive overrun (RORISF); transmit buffer empty (TISF); or receive buffer full (RISF).

13.1.6 I^2C interrupt

The flag bit INTR is set by a variety of conditions occurring on the I^2C interface: received own slave address (ADDR flag); received general call address (GC flag); received STOP condition (STOP flag); or successful transmission or reception of a data byte.The ISR must read the flag bits to determine the cause of the interrupt.

13.1.7 ADC interrupt

The flag bit AINTF is set when an A-to-D conversion has completed.

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

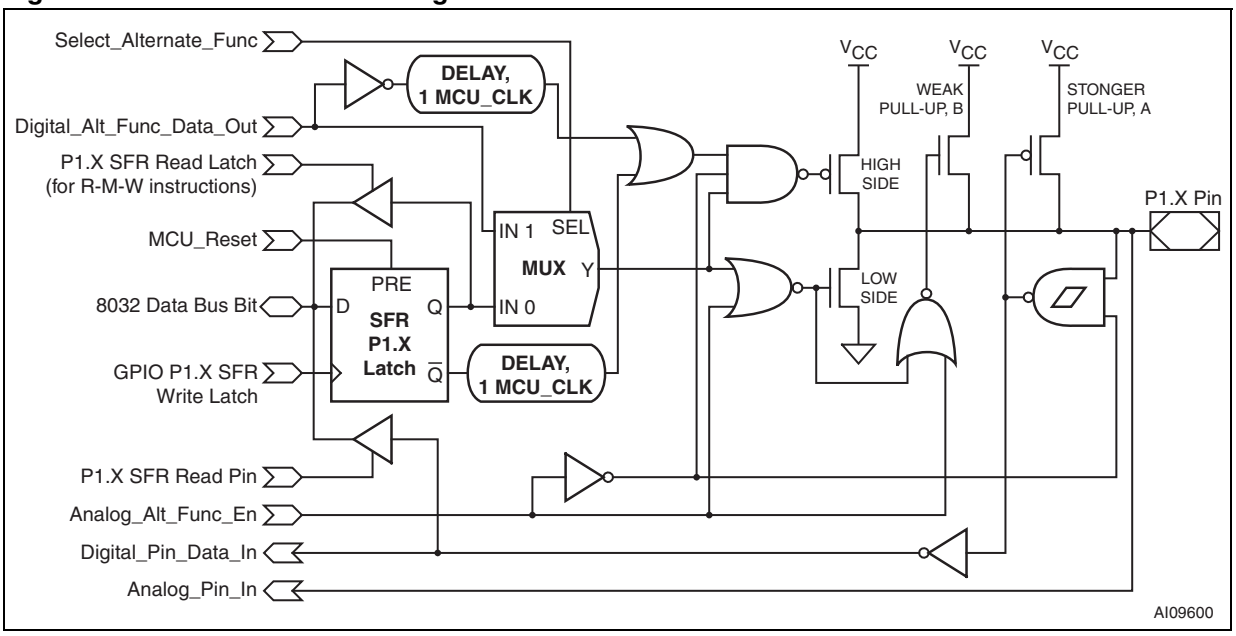


Figure 17. MCU I/O cell block diagram for Port 3

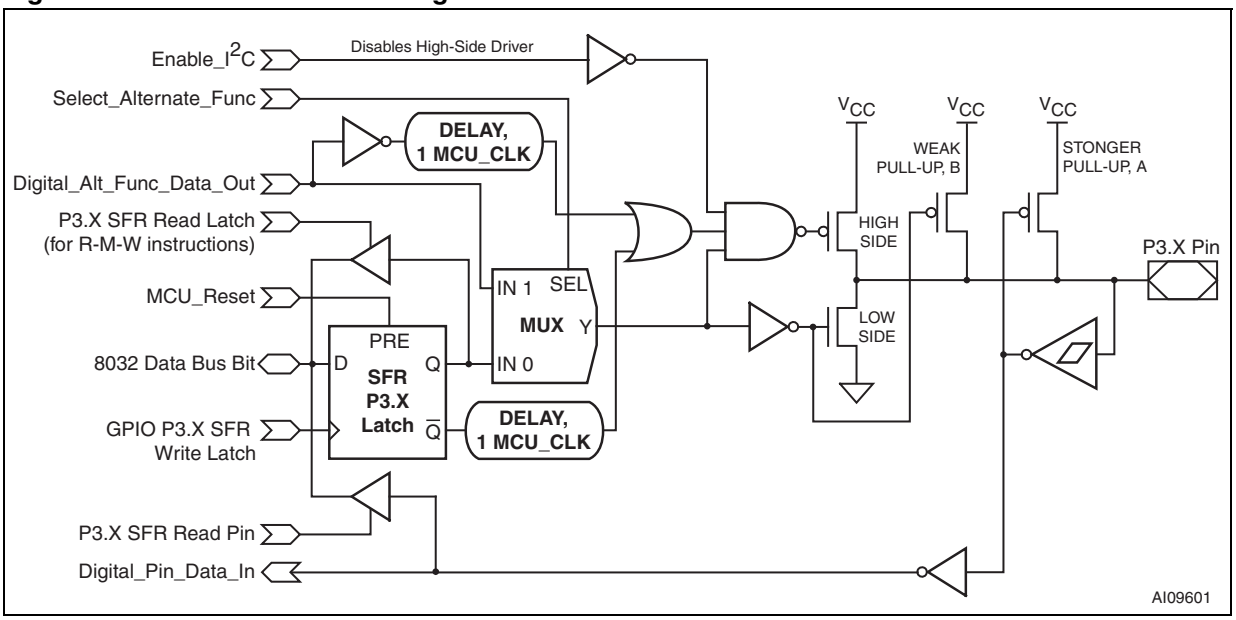


Figure 18. MCU I/O cell block diagram for Port 4

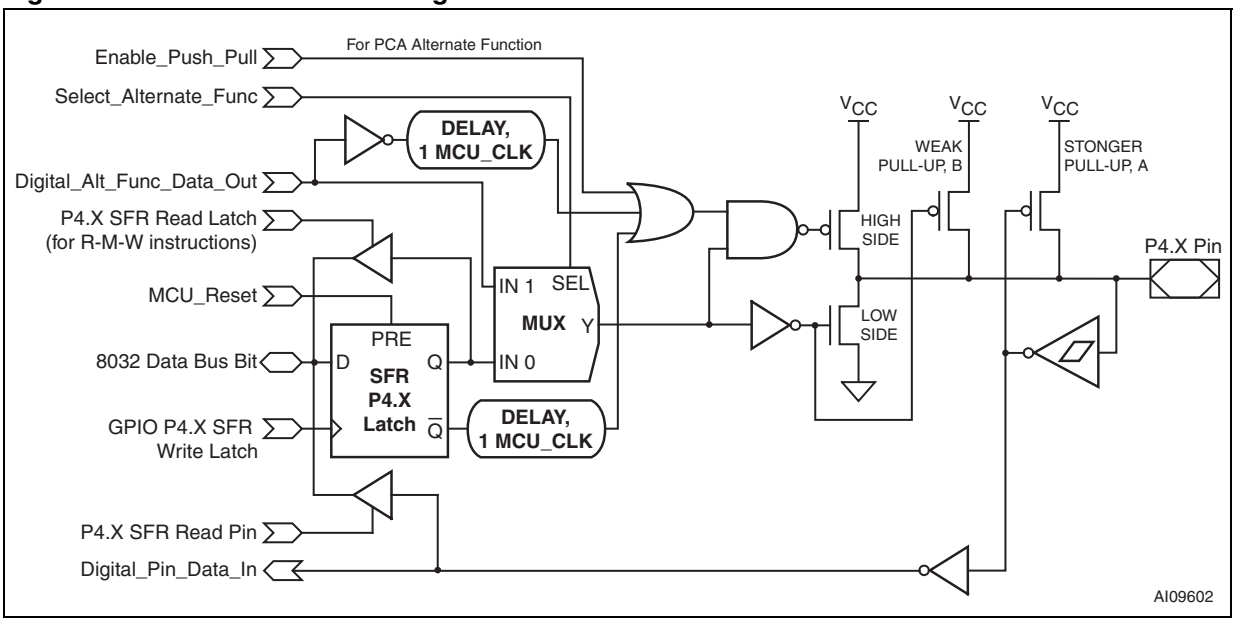


Table 33. P1: I/O Port 1 register (SFR 90h, reset value FFh)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0

Table 34. P1 register bit definition

Bit	Symbol	R/W	Function(1)
7	P1.7	R,W	**Port pin 1.7**
6	P1.6	R,W	**Port pin 1.6**
5	P1.5	R,W	**Port pin 1.5**
4	P1.4	R,W	**Port pin 1.4**
3	P1.3	R,W	**Port pin 1.3**
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.

In this example,

t_{MACH_CYC} = 100ns (4 MCU_CLK periods x 25ns)

$N_{OVERFLOW} = 2^{24}$ = 16777216 up-counts

WDT_{PERIOD} = 100ns X 16777216 = 1.67 seconds

The actual value will be slightly longer due to PFQ/BC.

19.5.1 Firmware example

The following 8051 assembly code illustrates how to operate the WDT. A simple statement in the reset initialization firmware enables the WDT, and then a periodic write to clear the WDT in the main firmware is required to keep the WDT from overflowing. This firmware is based on the example above (40 MHz f_{OSC}, CCON0 = 10h, BUSCON = C1h).

For example, in the reset initialization firmware (the function that executes after a jump to the reset vector):

```
MOV AE, #AA                    ; enable WDT by writing value to
                                  ; WDKEY other than 55h
```

Somewhere in the flow of the main program, this statement will execute periodically to reset the WDT before it's timeout period of 1.67 seconds. For example:

```
MOV A6, #00                    ; reset WDT, loading 000000h.
                                  ; Counting will automatically
                               ; resume as long as 55h in not in
                                           ; WDKEY
```

Table 50. WDKEY: Watchdog Timer Key register (SFR AEh, reset value 55h)

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

Table 51. WDKEY register bit definition

Bit	Symbol	R/W	Definition
[7:0]	WDKEY	W	55h disables the WDT from counting. 55h is automatically loaded in this SFR after any reset condition, leaving the WDT disabled by default. Any value other than 55h written to this SFR will enable the WDT, and counting begins.

Table 52. WDRST: Watchdog Timer Reset Counter register (SFR A6h, reset value 00h)

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

20.5.4 Mode 3

Timer 1 in mode 3 simply holds its count. The effect is the same as setting TR1 = 0.

Timer 0 in mode 3 establishes TL0 and TH0 as two separate counters. The logic for mode 3 on Timer 0 is shown in *Figure 23 on page 99*. TL0 uses the Timer 0 control Bits: C/$\overline{T}$, GATE, TR0, and TF0, as well as the pin EXTINT0. TH0 is locked into a timer function (counting at a rate of 1/12 f_{OSC}) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the “Timer 1“ interrupt flag.

Mode 3 is provided for applications requiring an extra 8-bit timer on the counter (see *Figure 23 on page 99*). With Timer 0 in mode 3, a UPSD33xx device can look like it has three Timer/Counters (not including the PCA). When Timer 0 is in mode 3, Timer 1 can be turned on and off by switching it out of and into its own mode 3, or can still be used by the serial port as a baud rate generator, or in fact, in any application not requiring an interrupt.

Table 56. TMOD: Timer Mode register (SFR 89h, reset value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GATE	C/$\overline{T}$	M[1:0]		GATE	C/$\overline{T}$	M[1:0]	

Table 57. TMOD register bit definition

Bit	Symbol	R/W	Timer	Definition (T/C is abbreviation for timer/counter)
7	GATE	R,W	Timer 1	**Gate control.** When GATE = 1, T/C is enabled only while pin EXTINT1 is '1' and the flag TR1 is '1.' When GATE = 0, T/C is enabled whenever the flag TR1 is '1.'
6	C/$\overline{T}$	R,W		**Counter or Timer function select.** When C/$\overline{T}$ = 0, function is timer, clocked by internal clock. C/$\overline{T}$ = 1, function is counter, clocked by signal sampled on external pin, C1.
[5:4]	M[1:0]	R,W		**Mode Select.** 00b = 13-bit T/C. 8 bits in TH1 with TL1 as 5-bit pre-scaler. 01b = 16-bit T/C. TH1 and TL1 are cascaded. No pre-scaler. 10b = 8-bit auto-reload T/C. TH1 holds a constant and loads into TL1 upon overflow. 11b = Timer Counter 1 is stopped.

Figure 24. Timer 2 in Capture mode

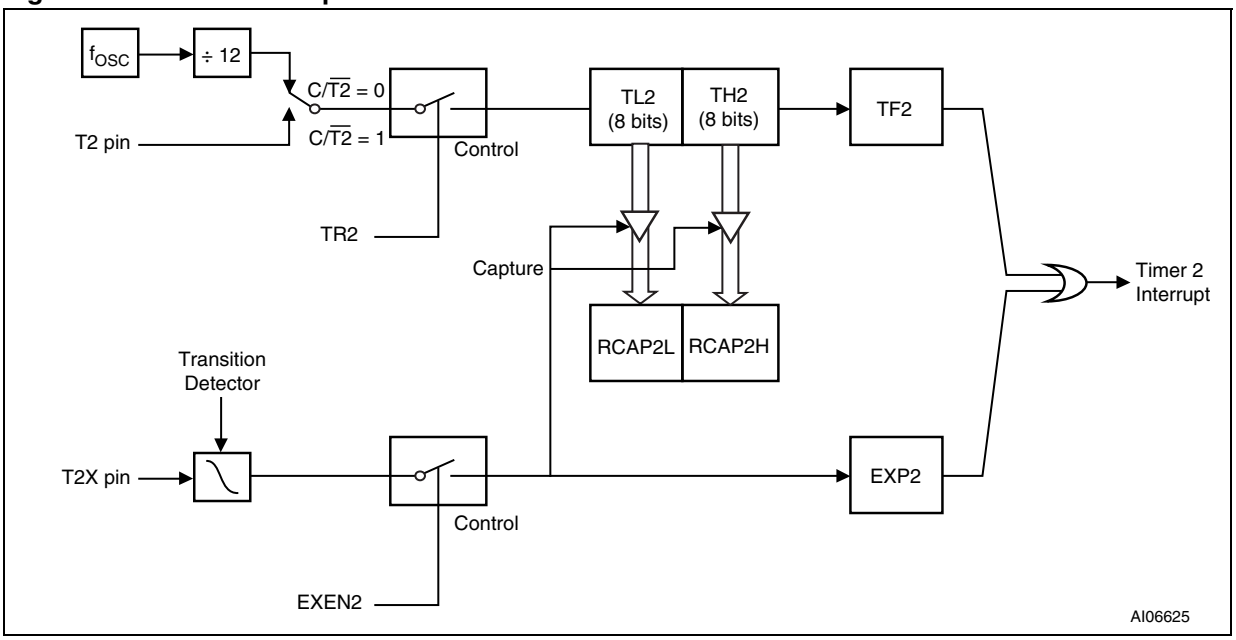
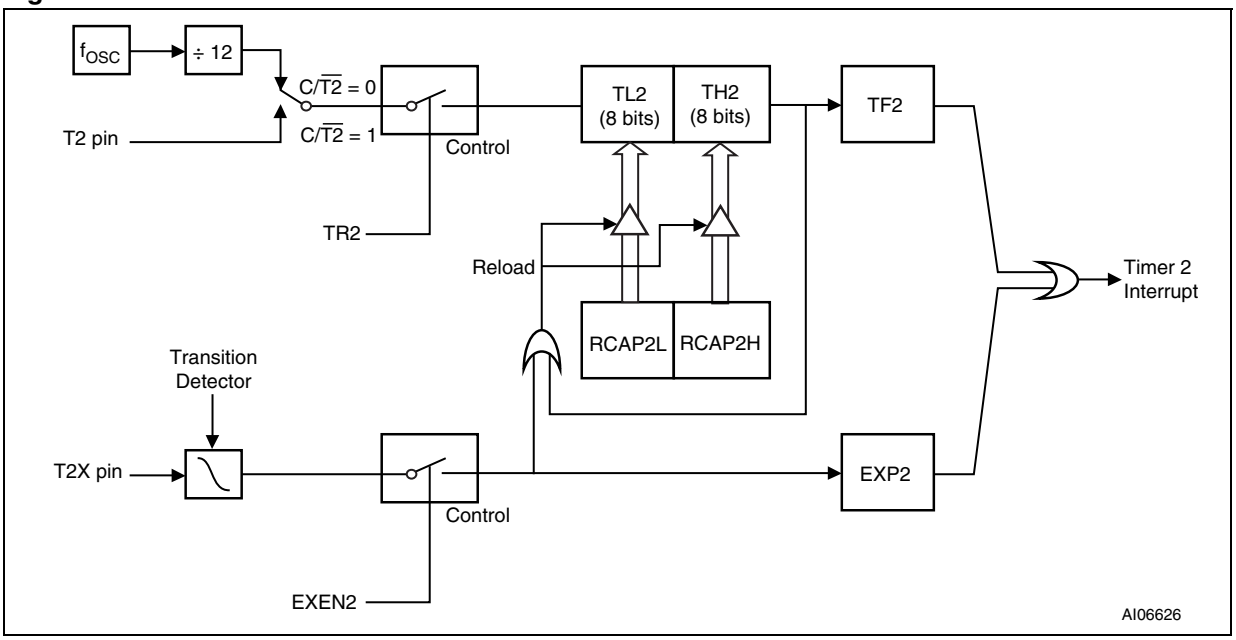


Figure 25. Timer 2 in Auto-Reload mode



23.2 Communication flow

I²C data flow control is based on the fact that all I²C compatible devices will drive the bus lines with open-drain (or open-collector) line drivers pulled up with external resistors, creating a wired-AND situation. This means that either bus line (SDA or SCL) will be at a logic '1' level only when no I²C device is actively driving the line to logic '0.' The logic for handshaking, arbitration, synchronization, and collision detection is implemented by each I²C device having:

1. The ability to hold a line low against the will of the other devices who are trying to assert the line high.
2. The ability of a device to detect that another device is driving the line low against its will.

Assert high means the driver releases the line and external pull-ups passively raise the signal to logic '1.' Holding low means the open-drain driver is actively pulling the signal to ground for a logic '0.'

For example, if a Slave device cannot transmit or receive a byte because it is distracted by and interrupt or it has to wait for some process to complete, it can hold the SCL clock line low. Even though the Master device is generating the SCL clock, the Master will sense that the Slave is holding the SCL line low against the will of the Master, indicating that the Master must wait until the Slave releases SCL before proceeding with the transfer.

Another example is when two Master devices try to put information on the bus simultaneously, the first one to release the SDA data line looses arbitration while the winner continues to hold SDA low.

Two types of data transfers are possible with I²C depending on the R/$\overline{W}$ bit, see *Figure 38 on page 124*.

1. **Data transfer from Master Transmitter to Slave Receiver (R/$\overline{W}$ = 0).** In this case, the Master generates a START condition on the bus and it generates a clock signal on the SCL line. Then the Master transmits the first byte on the SDA line containing the 7-bit Slave address plus the R/$\overline{W}$ bit. The Slave who owns that address will respond with an acknowledge bit on SDA, and all other Slave devices will not respond. Next, the Master will transmit a data byte (or bytes) that the addressed Slave must receive. The Slave will return an acknowledge bit after each data byte it successfully receives. After the final byte is transmitted by the Master, the Master will generate a STOP condition on the bus, or it will generate a RE-START condition and begin the next transfer. There is no limit to the number of bytes that can be transmitted during a transfer session.
2. **Data transfer from Slave Transmitter to Master Receiver (R/$\overline{W}$ = 1).** In this case, the Master generates a START condition on the bus and it generates a clock signal on the SCL line. Then the Master transmits the first byte on the SDA line containing the 7-bit Slave address plus the R/$\overline{W}$ bit. The Slave who owns that address will respond with an acknowledge bit on SDA, and all other Slave devices will not respond. Next, the addressed Slave will transmit a data byte (or bytes) to the Master. The Master will return an acknowledge bit after each data byte it successfully receives, unless it is the last byte the Master desires. If so, the Master will not acknowledge the last byte and from this, the Slave knows to stop transmitting data bytes to the Master. The Master will then generate a STOP condition on the bus, or it will generate a RE-START condition and begin the next transfer. There is no limit to the number of bytes that can be transmitted during a transfer session.

Table 84. S1SETUP examples for various I²C bus speeds and oscillator frequencies

I²C Bus speed, f_{SCL}	Parameter	Oscillator frequency, f_{OSC}				
		6 MHz	12 MHz	24 MHz	33 MHz	40 MHz
Standard	Recommended S1SETUP Value	93h	A7h	CFh	EEh	FFh
	Number of Samples	20	40	80	111	128
	Time Between Samples	166.6ns	83.3ns	41.6ns	30ns	25ns
	Total Sampled Period	3332ns	3332ns	3332ns	3333ns	3200ns
Fast	Recommended S1SETUP Value	82h	85h	8Bh	90h	93h
	Number of Samples	3	6	12	17	20
	Time Between Samples	166.6ns	83.3ns	41.6ns	30ns	25ns
	Total Sampled Period	500ns	500ns	500ns	510ns	500ns
High	Recommended S1SETUP Value	(1)	80	82	83	84
	Number of Samples	-	1	3	4	5
	Time Between Samples	-	83.3ns	41.6ns	30ns	25ns
	Total Sampled Period	-	83.3	125ns	120ns	125ns

1. Not compatible with high Speed I²C.

23.13 I²C operating sequences

The following pseudo-code explains hardware control for these I²C functions on the UPSD33xx:

- Initialize the Interface
- Function as Master-Transmitter
- Function as Master-Receiver
- Function as Slave-Transmitter
- Function as Slave-Receiver
- Interrupt Service Routine

Full C code drivers for the UPSD33xx I²C interface, and other interfaces are available from the web at *www.st.com/mcu*.

Initialization after a UPSD33xx reset

```
Ensure pins P3.6 and P3.7 are GPIO inputs
```

- `SFR P3.7 = 1 and SFR P3.6 = 1`

```
Configure pins P3.6 and P3.7 as I2C
```

- `SFR P3SFS.6 = 1 and P3SFS.7 = 1`

```
Set I2C clock prescaler to determine fSCL
```

- `SFR S1CON.CR[2:0] = desired SCL freq.`

Set bus START condition sampling

- SFR S1SETUP[7:0] = number of samples

Enable individual I2C interrupt and set priority

- SFR IEA.I2C = 1
- SFR IPA.I2C = 1 if high priority is desired

Set the Device address for Slave mode

- SFR S1ADR = XXh, desired address

Enable SIOE (as Slave) to return an ACK signal

- SFR S1CON.AA = 1

Master-transmitter

Disable all interrupts

- SFR IE.EA = 0

Set pointer to global data xmit buffer, set count

- *xmit_buf = *pointer to data
- buf_length = number of bytes to xmit

Set global variables to indicate Master-Xmitter

- I2C_master = 1, I2C_xmitter = 1

Disable Master from returning an ACK

- SFR S1CON.AA = 0

Enable I2C SIOE

- SFR S1CON.INI1 = 1

Transmit Address and R/W bit = 0 to Slave

- Is bus not busy? (SFR S1STA.BBUSY = 0?)
  <If busy, then test until not busy>
- SFR S1DAT[7:0] = Load Slave Address & FEh
- SFR S1CON.STA = 1, send START on bus
  <bus transmission begins>

Enable All Interrupts and go do something else

- SFR IE.EA = 1

Master-receiver

Disable all interrupts

- SFR IE.EA = 0

Set pointer to global data recv buffer, set count

- *recv_buf = *pointer to data
- buf_length = number of bytes to recv

Set global variables to indicate Master-Xmitter

- I2C_master = 1, I2C_xmitter = 0

Table 87. SPICON1: SPI Interface Control register 1 (SFR D7h, Reset Value 00h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	–	–	–	TEIE	RORIE	TIE	RIE

Table 88. SPICON1 register bit definition

Bit	Symbol	R/W	Definition
7-4	–	–	Reserved
3	TEIE	RW	**Transmission End Interrupt Enable** 0 = Disable Interrupt for Transmission End 1 = Enable Interrupt for Transmission End
2	RORIE	RW	**Receive Overrun Interrupt Enable** 0 = Disable Interrupt for Receive Overrun 1 = Enable Interrupt for Receive Overrun
1	TIE	RW	**Transmission Interrupt Enable** 0 = Disable Interrupt for SPITDR empty 1 = Enable Interrupt for SPITDR empty
0	RIE	RW	**Reception Interrupt Enable** 0 = Disable Interrupt for SPIRDR full 1 = Enable Interrupt for SPIRDR full

Table 89. SPICLKD: SPI Prescaler (Clock Divider) register (SFR D2h, Reset Value 04h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DIV128	DIV64	DIV32	DIV16	DIV8	DIV4	–	–

Table 90. SPICLKD register bit definition

Bit	Symbol	R/W	Definition
7	DIV128	RW	0 = No division 1 = Divide f_{OSC} clock by 128
6	DIV64	RW	0 = No division 1 = Divide f_{OSC} clock by 64
5	DIV32	RW	0 = No division 1 = Divide f_{OSC} clock by 32
4	DIV16	RW	0 = No division 1 = Divide f_{OSC} clock by 16
3	DIV8	RW	0 = No division 1 = Divide f_{OSC} clock by 8
2	DIV4	RW	0 = No division 1 = Divide f_{OSC} clock by 4
1-0	Not Used	–	

Table 100. 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 101. CCON3 register (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 102. 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

26.3 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.

26.4 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.

26.5 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.

27.2.3 Specifying the memory map with PSDsoft Express

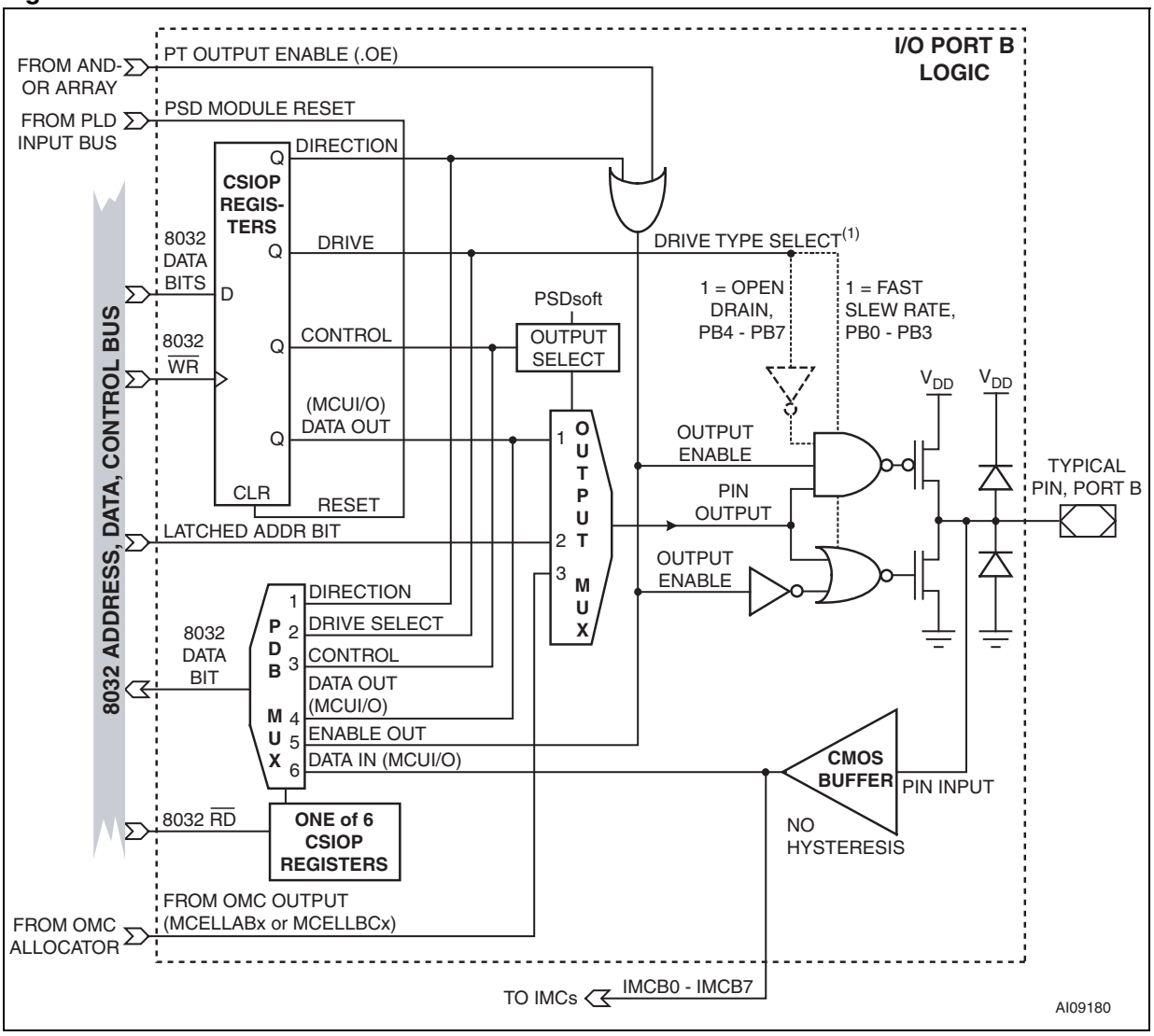
The memory map example shown in *Figure 52* is implemented using PSDsoft Express in a point-and-click environment. PSDsoft Express will automatically generate Hardware Definition Language (HDL) statements of the ABEL language for the DPLD, such as those shown in *Table 114 on page 172*.

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

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

```
rs0 =    ((address ≥ ^h0000)    &    (address ≤ ^h1FFF));
csiop =  ((address ≥ ^h2000)    &    (address ≤ ^h20FF));
fs0 =    ((address ≥ ^h0000)    &    (address ≤ ^h3FFF));
fs1 =    ((address ≥ ^h4000)    &    (address ≤ ^h7FFF));
fs2 =    ((page == 0)    &    (address ≥ ^h8000)    &    (address ≤ ^hBFFF));
fs3 =    ((page == 0)    &    (address ≥ ^hC000)    &    (address ≤ ^hFFFF));
fs4 =    ((page == 1)    &    (address ≥ ^h8000)    &    (address ≤ ^hBFFF));
fs5 =    ((page == 1)    &    (address ≥ ^hC000)    &    (address ≤ ^hFFFF));
fs6 =    ((page == 2)    &    (address ≥ ^h8000)    &    (address ≤ ^hBFFF));
fs7 =    ((page == 2)    &    (address ≥ ^hC000)    &    (address ≤ ^hFFFF));
csboot0 =  ((address ≥ ^h8000)    &    (address ≤ ^h9FFF));
csboot1 =  ((address ≥ ^hA000)    &    (address ≤ ^hBFFF));
csboot2 =  ((address ≥ ^hC000)    &    (address ≤ ^hDFFF));
csboot3 =  ((address ≥ ^hE000)    &    (address ≤ ^hFFFF));
```

Figure 74. Port B structure



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.

27.4.49 Port C structure

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

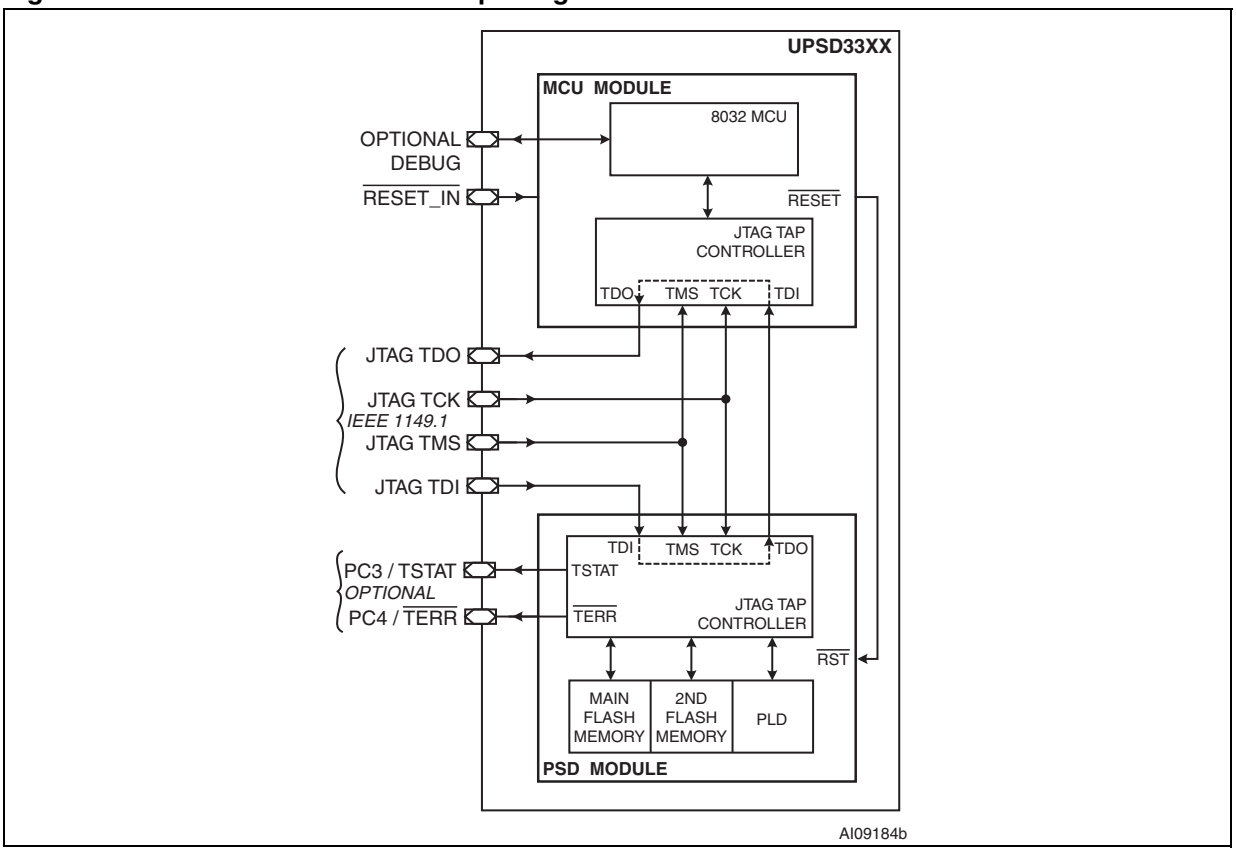
- MCU I/O mode
- GPLD Output mode from Output Macrocells MCELLBC2, MCELLBC3, MCELLBC4, MCELLBC7
- GPLD Input mode to Input Macrocells IMCC2, IMCC3, IMCC4, IMCC7

See *Figure 75 on page 222* for detail.

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

- Pins PC3 and PC4 support TSTAT and $\overline{\text{TERR}}$ status indicators, to reduce the amount of time required for JTAG ISP programming. These two pins must be used together for this function, adding to the four standard JTAG signals. When TSTAT and $\overline{\text{TERR}}$ are

Figure 79. JTAG chain in UPSD33xx package



27.5.3 In-system programming

The ISP function can use two different configurations of the JTAG interface:

- 4-pin JTAG: TDI, TDO, TCK, TMS
- 6-pin JTAG: Signals above plus TSTAT, $\overline{\text{TERR}}$

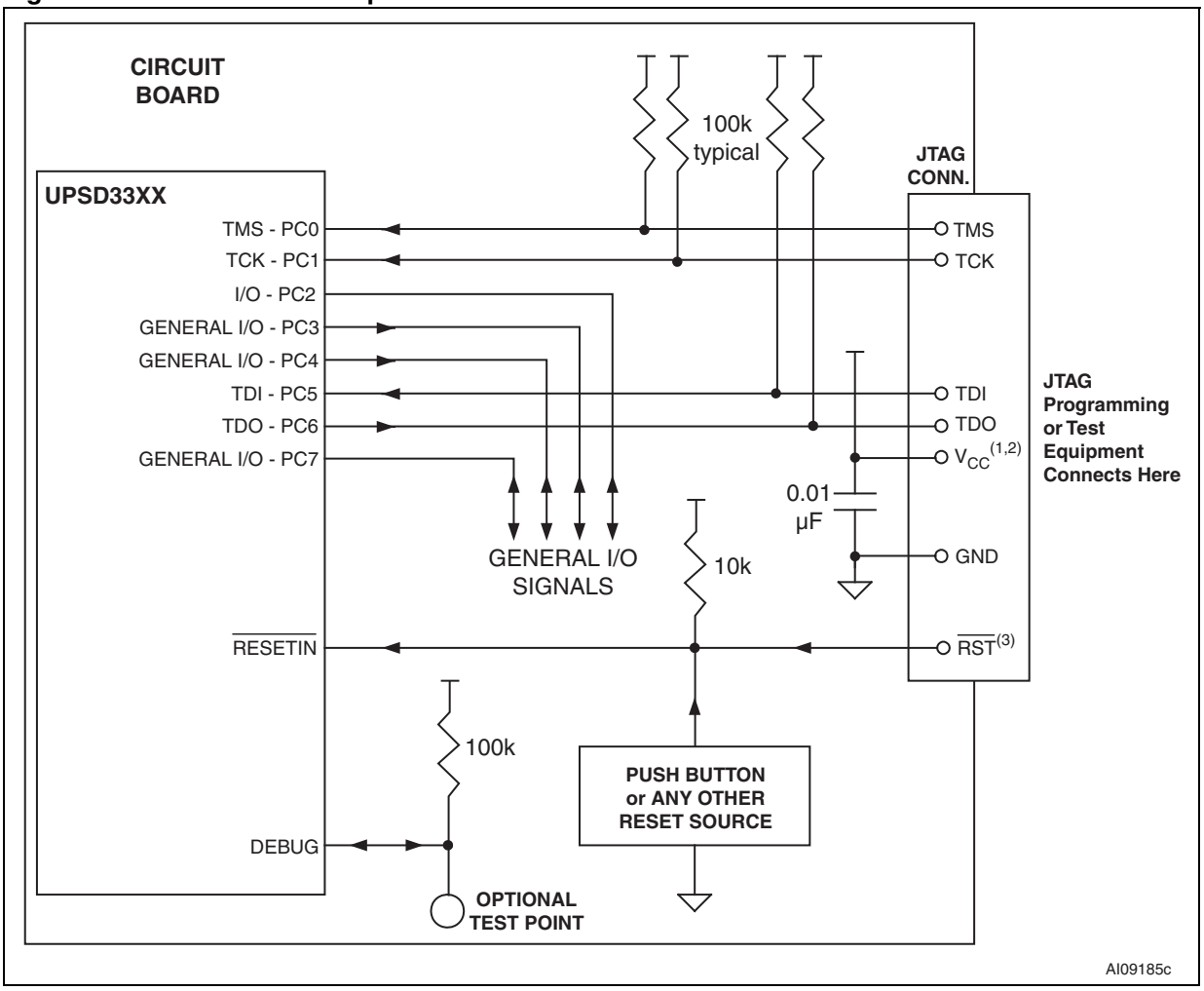
At power-up, the four basic JTAG signals are all inputs, waiting for a command to appear on the JTAG bus from programming or test equipment. When the enabling command is received, TDO becomes an output and the JTAG channel is fully functional. The same command that enables the JTAG channel may optionally enable the two additional signals, TSTAT and $\overline{\text{TERR}}$.

27.5.4 4-pin JTAG ISP (default)

The four basic JTAG pins on Port C are enabled for JTAG operation at all times. These pins may not be used for other I/O functions. There is no action needed in PSDsoft Express to configure a device to use 4-pin JTAG, as this is the default condition. No 8032 firmware is needed to use 4-pin ISP because all ISP functions are controlled from the external JTAG program/test equipment. *Figure 80 on page 236* shows recommended connections on a circuit board to a JTAG program/test tool using 4-pin JTAG. It is required to connect the $\overline{\text{RST}}$ output signal from the JTAG program/test equipment to the $\overline{\text{RESET\_IN}}$ input on the UPSD33xx. The $\overline{\text{RST}}$ signal is driven by the equipment with an Open Drain driver, allowing other sources (like a push button) to drive $\overline{\text{RESET\_IN}}$ without conflict.

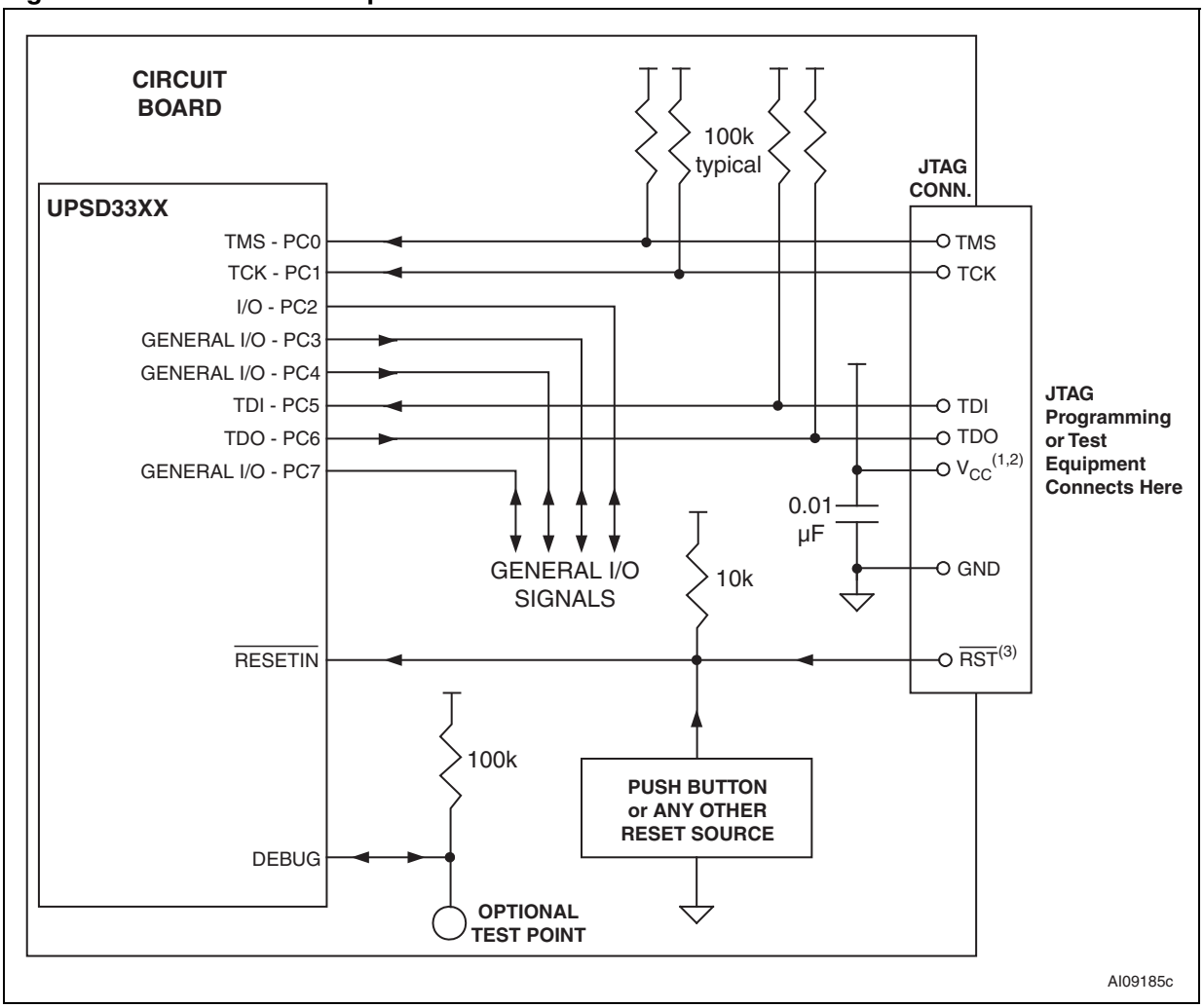
Note: The recommended pull-up resistors and decoupling capacitor are illustrated in Figure 80.

Figure 80. Recommended 4-pin JTAG connections



1. For 5 V UPSD33xx 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 UPSD33xx 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 $\overline{\text{RESETIN}}$.

Figure 81. Recommended 6-pin JTAG connections



1. For 5 V UPSD33xx 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 UPSD33xx 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.

Figure 87. External $\overline{PSEN}$/READ cycle (80-pin device only)

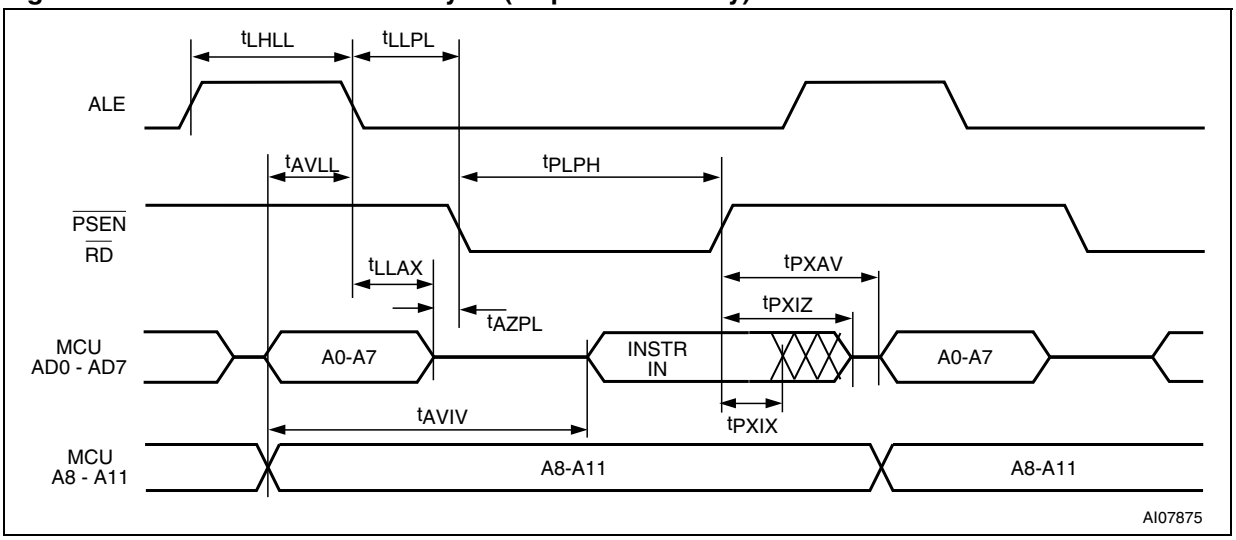


Table 169. External $\overline{PSEN}$ or READ cycle AC Characteristics (3 V or 5 V device)

Symbol	Parameter	40 MHz oscillator(1)		Variable oscillator $1/t_{CLCL}$ = 8 to 40 MHz		Unit
		Min	Max	Min	Max	
t_{LHLL}	ALE pulse width	17		$t_{CLCL} - 8$		ns
t_{AVLL}	Address setup to ALE	13		$t_{CLCL} - 12$		ns
t_{LLAX}	Address hold after ALE	7.5		$0.5t_{CLCL} - 5$		ns
t_{LLPL}	ALE to $\overline{PSEN}$ or $\overline{RD}$	7.5		$0.5t_{CLCL} - 5$		ns
t_{PLPH}	$\overline{PSEN}$ or $\overline{RD}$ pulse width(2)	40		$nt_{CLCL} - 10$		ns
t_{PXIX}	Input instruction/data hold after $\overline{PSEN}$ or $\overline{RD}$	2		2		ns
t_{PHIZ}	Input instruction/data float after $\overline{PSEN}$ or $\overline{RD}$		10.5		$0.5t_{CLCL} - 2$	ns
t_{PXAV}	Address hold after $\overline{PSEN}$ or $\overline{RD}$	7.5		$0.5t_{CLCL} - 5$		ns
t_{AVIV}	Address to valid instruction/data in(2)		70		$mt_{CLCL} - 5$	ns
t_{AZPL}	Address float to $\overline{PSEN}$ or $\overline{RD}$	–2		–2		ns

1. BUSCON register is configured for 4 PFQCLK.
2. Refer to *Table 170* for “n” and “m” values.

Table 182. Program, WRITE and Erase times (5 V, 3 V PSD modules)

Symbol	Parameter	Min.	Typ.	Max.	Unit
	Flash Program		8.5		s
	Flash Bulk Erase(1) (pre-programmed)		3(2)	10	s
	Flash Bulk Erase (not pre-programmed)		5		s
t_{WHQV3}	Sector Erase (pre-programmed)		1	10	s
t_{WHQV2}	Sector Erase (not pre-programmed)		2.2		s
t_{WHQV1}	Byte Program		14	150	µs
	Program/Erase cycles (per sector)	100,000			cycles
	PLD Program/Erase cycles	1000			cycles
t_{WHWLO}	Sector Erase timeout		100		µs
t_{Q7VQV}	DQ7 Valid to output (DQ7-DQ0) valid (data polling)(3)			30	ns

1. Programmed to all zero before erase.
2. Typical after 100K program/erase cycle is 5 seconds.
3. The polling status, DQ7, is valid t_{Q7VQV} time units before the data byte, DQ0-DQ7, is valid for reading.

Figure 99. External clock cycle

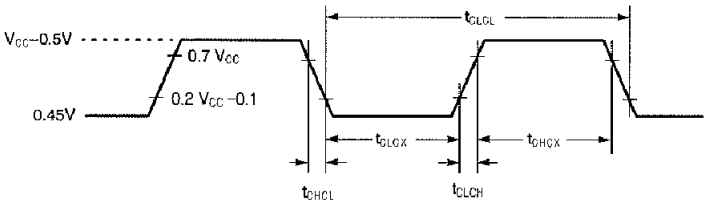


Figure 100. PSD module AC measurement I/O waveform

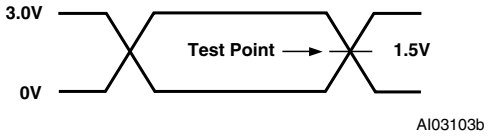


Figure 101. PSD module AC measurement load circuit

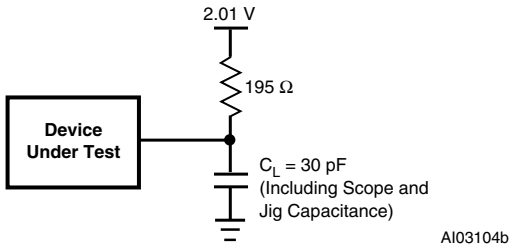


Table 190. I/O pin capacitance

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.