Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

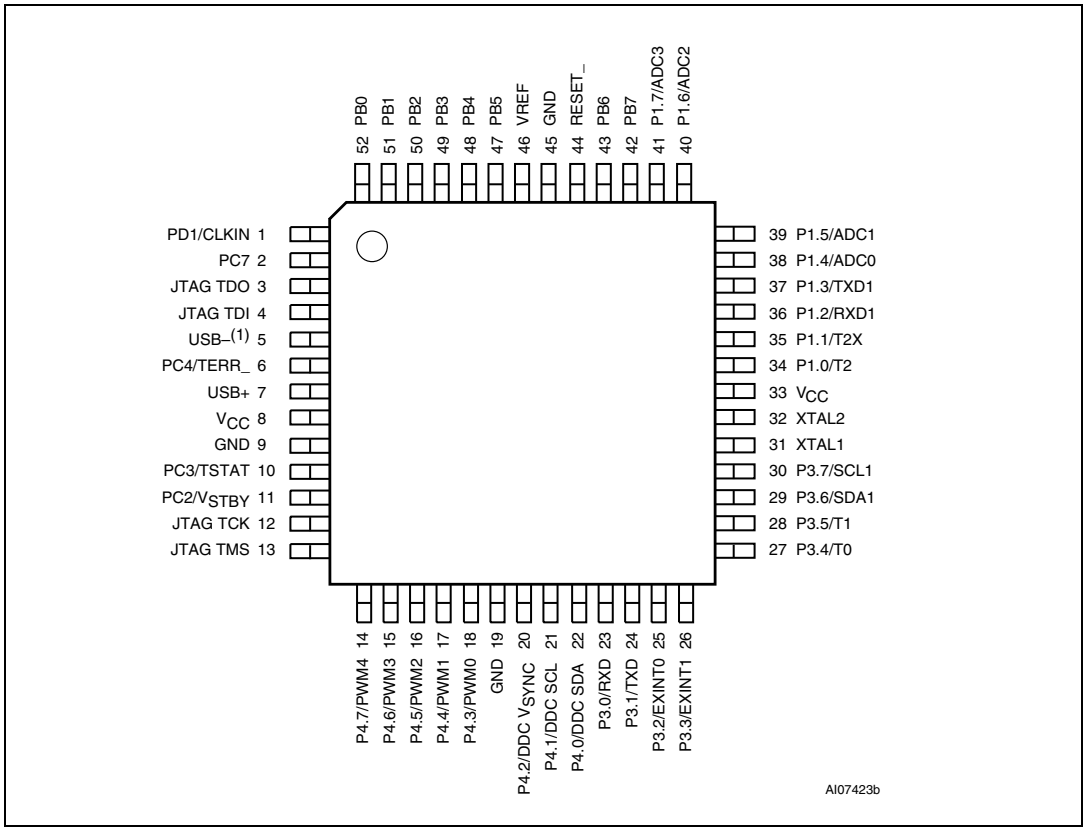
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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, UART/USART, USB
Peripherals	LVR, POR, PWM, WDT
Number of I/O	46
Program Memory Size	288KB (288K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 4x8b
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/upsd3234a-40u6

Contents

Figure 2. TQFP52 connections



1. Pull-up resistor required on pin 5 (2 kΩ for 3 V devices, 7.5 kΩ for 5 V devices) for all 52-pin devices, with or without USB function.

Table 2. 80-pin package pin description (continued)

Port pin	Signal name	Pin no.	In/ out	Function	
				Basic	Alternate
	AD7	47	I/O	Multiplexed Address/Data bus A7/D7	
P1.0	T2	52	I/O	General I/O port pin	Timer 2 Count input
P1.1	TX2	54	I/O	General I/O port pin	Timer 2 Trigger input
P1.2	RxD1	56	I/O	General I/O port pin	2nd UART Receive
P1.3	TxD1	58	I/O	General I/O port pin	2nd UART Transmit
P1.4	ADC0	59	I/O	General I/O port pin	ADC Channel 0 input
P1.5	ADC1	60	I/O	General I/O port pin	ADC Channel 1 input
P1.6	ADC2	61	I/O	General I/O port pin	ADC Channel 2 input
P1.7	ADC3	64	I/O	General I/O port pin	ADC Channel 3 input
	A8	51	O	External Bus, Address A8	
	A9	53	O	External Bus, Address A9	
	A10	55	O	External Bus, Address A10	
	A11	57	O	External Bus, Address A11	
P3.0	RxD0	75	I/O	General I/O port pin	UART Receive
P3.1	TxD0	77	I/O	General I/O port pin	UART Transmit
P3.2	EXINT0	79	I/O	General I/O port pin	Interrupt 0 input / Timer 0 gate control
P3.3	EXINT1	2	I/O	General I/O port pin	Interrupt 1 input / Timer 1 gate control
P3.4	T0	40	I/O	General I/O port pin	Counter 0 input
P3.5	T1	42	I/O	General I/O port pin	Counter 1 input
P3.6	SDA1	44	I/O	General I/O port pin	I^2C Bus serial data I/O
P3.7	SCL1	46	I/O	General I/O port pin	I^2C Bus clock I/O
P4.0	DDC SDA	33	I/O	General I/O port pin	
P4.1	DDC SCL	31	I/O	General I/O port pin	
P4.2	DDC V_{SYNC}	30	I/O	General I/O port pin	
P4.3	PWM0	27	I/O	General I/O port pin	8-bit Pulse Width Modulation output 0
P4.4	PWM1	25	I/O	General I/O port pin	8-bit Pulse Width Modulation output 1
P4.5	PWM2	23	I/O	General I/O port pin	8-bit Pulse Width Modulation output 2
P4.6	PWM3	19	I/O	General I/O port pin	8-bit Pulse Width Modulation output 3

Note: In UPSD323xx devices, the stack resides in on-chip RAM, and grows upwards. The PUSH instruction first increments the Stack Pointer (SP), then copies the byte into the stack. PUSH and POP use only direct addressing to identify the byte being saved or restored, but the stack itself is accessed by indirect addressing using the SP register. This means the stack can go into the Upper 128 bytes of RAM, if they are implemented, but not into SFR space.

The Data Transfer instructions include a 16-bit MOV that can be used to initialize the Data Pointer (DPTR) for look-up tables in Program Memory.

The XCH A, <byte> instruction causes the Accumulator and ad-dressed byte to exchange data. The XCHD A, @Ri instruction is similar, but only the low nibbles are involved in the exchange. To see how XCH and XCHD can be used to facilitate data manipulations, consider first the problem of shifting and 8-digit BCD number two digits to the right. *Table 8* shows how this can be done using XCH instructions. To aid in understanding how the code works, the contents of the registers that are holding the BCD number and the content of the Accumulator are shown alongside each instruction to indicate their status after the instruction has been executed.

After the routine has been executed, the Accumulator contains the two digits that were shifted out on the right. Doing the routine with direct MOVs uses 14 code bytes. The same operation with XCHs uses only 9 bytes and executes almost twice as fast. To right-shift by an odd number of digits, a one-digit must be executed. *Table 9* shows a sample of code that will right-shift a BCD number one digit, using the XCHD instruction. Again, the contents of the registers holding the number and of the accumulator are shown alongside each instruction.

Table 6. Data transfer instructions that access internal data memory space

Mnemonic	Operation	Addressing modes			
		Dir.	Ind.	Reg.	Imm.
MOV A,<src>	A = <src>	X	X	X	X
MOV <dest>,A	<dest> = A	X	X	X	
MOV <dest>,<src>	<dest> = <src>	X	X	X	X
MOV DPTR,#data16	DPTR = 16-bit immediate constant				X
PUSH <src>	INC SP; MOV “@SP”,<src>	X			
POP <dest>	MOV <dest>,”@SP”; DEC SP	X			
XCH A,<byte>	Exchange contents of A and <byte>	X	X	X	
XCHD A,@Ri	Exchange low nibbles of A and @Ri		X		

First, pointers R1 and R0 are set up to point to the two bytes containing the last four BCD digits. Then a loop is executed which leaves the last byte, location 2EH, holding the last two digits of the shifted number. The pointers are decremented, and the loop is repeated for location 2DH. The CJNE instruction (Compare and Jump if Not equal) is a loop control that will be described later. The loop executed from LOOP to CJNE for R1 = 2EH, 2DH, 2CH, and 2BH. At that point the digit that was originally shifted out on the right has propagated to location 2AH. Since that location should be left with 0s, the lost digit is moved to the Accumulator.

The LJMP instruction encodes the destination address as a 16-bit constant. The instruction is 3 bytes long, consisting of the opcode and two address bytes. The destination address can be anywhere in the 64K Program Memory space.

The AJMP instruction encodes the destination address as an 11-bit constant. The instruction is 2 bytes long, consisting of the opcode, which itself contains 3 of the 11 address bits, followed by another byte containing the low 8 bits of the destination address. When the instruction is executed, these 11 bits are simply substituted for the low 11 bits in the PC. The high 5 bits stay the same. Hence the destination has to be within the same 2K block as the instruction following the AJMP.

In all cases the programmer specifies the destination address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the destination address into the correct format for the given instruction. If the format required by the instruction will not support the distance to the specified destination address, a “Destination out of range” message is written into the List file.

The JMP @A+DPTR instruction supports case jumps. The destination address is computed at execution time as the sum of the 16-bit DPTR register and the Accumulator. Typically. DPTR is set up with the address of a jump table. In a 5-way branch, for ex-ample, an integer 0 through 4 is loaded into the Accumulator. The code to be executed might be as follows:

```
MOV DPTR,#JUMP TABLE
MOV A,INDEX_NUMBER
RL A
JMP @A+DPTR
```

The RL A instruction converts the index number (0 through 4) to an even number on the range 0 through 8, because each entry in the jump table is 2 bytes long:

```
JUMP TABLE:
AJMP CASE 0
AJMP CASE 1
AJMP CASE 2
AJMP CASE 3
AJMP CASE 4
```

Table 13 shows a single “CALL addr” instruction, but there are two of them, LCALL and ACALL, which differ in the format in which the subroutine address is given to the CPU. CALL is a generic mnemonic which can be used if the programmer does not care which way the address is encoded.

The LCALL instruction uses the 16-bit address format, and the subroutine can be anywhere in the 64K Program Memory space. The ACALL instruction uses the 11-bit format, and the subroutine must be in the same 2K block as the instruction following the ACALL.

In any case, the programmer specifies the subroutine address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the address into the correct format for the given instructions.

Subroutines should end with a RET instruction, which returns execution to the instruction following the CALL.

RETI is used to return from an interrupt service routine. The only difference between RET and RETI is that RETI tells the interrupt control system that the interrupt in progress is done.

If there is no interrupt in progress at the time RETI is executed, then the RETI is functionally identical to RET.

Table 13. Unconditional Jump instructions

Mnemonic	Operation
JMP addr	Jump to addr
JMP @A+DPTR	Jump to A+DPTR
CALL addr	Call Subroutine at addr
RET	Return from subroutine
RETI	Return from interrupt
NOP	No operation

Table 14 shows the list of conditional jumps available to the UPSD323xx device user. All of these jumps specify the destination address by the relative offset method, and so are limited to a jump distance of -128 to +127 bytes from the instruction following the conditional jump instruction. Important to note, however, the user specifies to the assembler the actual destination address the same way as the other jumps: as a label or a 16-bit constant.

There is no Zero Bit in the PSW. The JZ and JNZ instructions test the Accumulator data for that condition.

The DJNZ instruction (Decrement and Jump if Not Zero) is for loop control. To execute a loop N times, load a counter byte with N and terminate the loop with a DJNZ to the beginning of the loop, as shown below for N = 10:

```
MOV COUNTER,#10
LOOP: (begin loop)
  •
  •
  •
(end loop)
DJNZ COUNTER, LOOP
(continue)
```

The CJNE instruction (Compare and Jump if Not Equal) can also be used for loop control as in *Table 9*. Two bytes are specified in the operand field of the instruction. The jump is executed only if the two bytes are not equal. In the example of *Table 9* Shifting a BCD Number One Digits to the Right, the two bytes were data in R1 and the constant 2Ah. The initial data in R1 was 2Eh.

Every time the loop was executed, R1 was decremented, and the looping was to continue until the R1 data reached 2Ah.

Another application of this instruction is in “greater than, less than” comparisons. The two bytes in the operand field are taken as unsigned integers. If the first is less than the second, then the Carry Bit is set (1). If the first is greater than or equal to the second, then the Carry Bit is cleared.

2.16 Machine cycles

A machine cycle consists of a sequence of six states, numbered S1 through S6. Each state time lasts for two oscillator periods. Thus, a machine cycle takes 12 oscillator periods or 1µs if the oscillator frequency is 12MHz. Refer to *Table 13: State sequence in UPSD323xx devices*.

Each state is divided into a Phase 1 half and a Phase 2 half. State Sequence in UPSD323xx devices shows that retrieve/execute sequences in states and phases for various kinds of instructions.

Normally two program retrievals are generated during each machine cycle, even if the instruction being executed does *not* require it. If the instruction being executed does not need more code bytes, the CPU simply ignores the extra retrieval, and the Program Counter is not incremented.

Execution of a one-cycle instruction (*Figure 13: State sequence in UPSD323xx devices*) begins during State 1 of the machine cycle, when the opcode is latched into the Instruction Register. A second retrieve occurs during S4 of the same machine cycle. Execution is complete at the end of State 6 of this machine cycle.

The MOVX instructions take two machine cycles to execute. No program retrieval is generated during the second cycle of a MOVX instruction. This is the only time program retrievals are skipped. The retrieve/execute sequence for MOVX instruction is shown in *Figure 13* (d).

Table 14. Conditional jump instructions

Mnemonic	Operation	Addressing modes			
		Dir.	Ind.	Reg.	Imm.
JZ rel	Jump if A = 0	Accumulator only			
JNZ rel	Jump if A ≠ 0	Accumulator only			
DJNZ <byte>,rel	Decrement and jump if not zero	X		X	
CJNE A,<byte>,rel	Jump if A ≠ <byte>	X			X
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X	

9.3 Watchdog timer overflow

The Watchdog timer generates an internal reset when its 22-bit counter overflows. See Watchdog Timer section for details.

9.4 USB reset

The USB reset is generated by a detection on the USB bus $\overline{\text{RESET}}$ signal. A single-end zero on its upstream port for 4 to 8 times will set RSTF Bit in UISTA register. If Bit 6 (RSTE) of the UIEN Register is set, the detection will also generate the $\overline{\text{RESET}}$ signal to reset the CPU and other peripherals in the MCU.

11 Timer/counters (Timer 0, Timer 1 and Timer 2)

The UPSD323xx devices has three 16-bit Timer/Counter registers: Timer 0, Timer 1 and Timer 2. All of them can be configured to operate either as timers or event counters and are compatible with standard 8032 architecture.

In the “Timer” function, the register is incremented every machine cycle. Thus, one can think of it as counting machine cycles. Since a machine cycle consists of 6 CPU clock periods, the count rate is 1/6 of the CPU clock frequency or 1/12 of the oscillator frequency (f_{OSC}).

In the “Counter” function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0 or T1. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 f_{OSC} clock periods) to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the f_{OSC}. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full cycle. In addition to the “Timer” or “Counter” selection, Timer 0 and Timer 1 have four operating modes from which to select.

11.1 Timer 0 and Timer 1

The “Timer” or “Counter” function is selected by control bits C/ T in the Special Function Register TMOD. These Timer/Counters have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for Timers/ Counters. Mode 3 is different. The four operating modes are de-scribed in the following text.

Table 36. Control register (TCON)

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Table 37. Description of the TCON bits

Bit	Symbol	Function
7	TF1	Timer 1 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine
6	TR1	Timer 1 run control bit. Set/cleared by software to turn Timer/Counter on or off
5	TF0	Timer 0 overflow flag. Set by hardier on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine
4	TR0	Timer 0 run control bit. Set/cleared by software to turn Timer/Counter on or off
3	IE1	Interrupt 1 Edge Flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed
2	IT1	Interrupt 1 Type Control Bit. Set/cleared by software to specify falling-edge/low-level triggered external interrupt

Table 42. Timer/counter2 operating modes

Mode	T2CON			T2MOD DECN	T2CON EXEN	P1.1 T2EX	Remarks	Input clock	
	RxCLK or TxCLK	CP/$\overline{RL2}$	TR2					Internal	External (P1.0/T2)
16-bit Auto-reload	0	0	1	0	0	x	Reload upon overflow	$f_{OSC}/12$	MAX $f_{OSC}/24$
	0	0	1	0	1	-	Reload trigger (falling edge)		
	0	0	1	1	x	0	Down counting		
	0	0	1	1	x	1	Up counting		
16-bit Capture	0	1	1	x	0	x	16-bit Timer/Counter (only up counting)	$f_{OSC}/12$	MAX $f_{OSC}/24$
	0	1	1	x	1	-	Capture (TH1,TL2) → (RCAP2H,RCAP2L)		
Baud Rate Generator	1	x	1	x	0	x	No overflow interrupt request (TF2)	$f_{OSC}/12$	MAX $f_{OSC}/24$
	1	x	1	x	1	-	Extra External Interrupt (Timer 2)		
Off	x	x	0	x	x	x	Timer 2 stops	—	—

1. ↓ = falling edge

Figure 24. Timer 2 in Capture mode

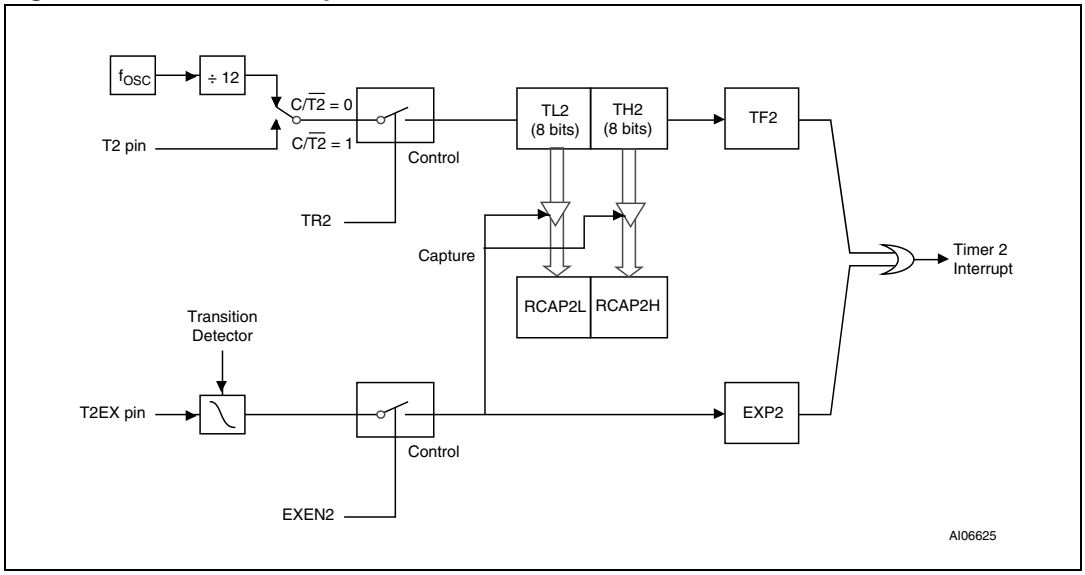


Table 44. Description of the SCON bits (continued)

Bit	Symbol	Function
1	TI	Transmit Interrupt Flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the Stop bit in the other modes, in any serial transmission. Must be cleared by software
0	RI	Receive Interrupt Flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the Stop bit in the other modes, in any serial reception (except for SM2). Must be cleared by software

12.2.1 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 Bit SMOD = 0 (which is the value on reset), the baud rate is 1/64 the oscillator frequency. If SMOD = 1, the baud rate is 1/32 the oscillator frequency.

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

In the UPSD323xx devices, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate.

12.2.2 Using Timer 1 to generate baud rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

Modes 1 and 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 TMOD = 0010B). In that case the baud rate is given by the formula:

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

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 Interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the Timer 1 Interrupt to do a 16-bit software reload. *Figure 21* lists various commonly used baud rates and how they can be obtained from Timer 1.

12.2.3 Using Timer/counter 2 to generate baud rates

In the UPSD323xx devices, Timer 2 selected as the baud rate generator by setting TCLK and/or RCLK (see *Figure 21* Timer/ Counter 2 Control Register (T2CON)).

Note: The baud rate for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer into its Baud Rate Generator mode.

The RCLK and TCLK Bits in the T2CON register configure UART 1. The RCLK1 and TCLK1 Bits in the PCON register configure UART 2.

The Baud Rate Generator mode is similar to the Auto-reload Mmode, in that a roll over in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

Table 49. PWM SFR memory map (continued)

SFR addr	Reg name	Bit register name								Reset value	Comments
		7	6	5	4	3	2	1	0		
B2	PSCL0H									00	Prescaler 0 High (8-bit)
B3	PSCL1L									00	Prescaler 1 Low (8-bit)
B4	PSCL1H									00	Prescaler 1 High (8-bit)

PWMCON register bit definition:

- PWML = PWM 0-3 polarity control
- PWMP = PWM 4 polarity control
- PWME = PWM enable (0 = disabled, 1= enabled)
- CFG3..CFG0 = PWM 0-3 Output (0 = Open Drain; 1 = Push-Pull)
- CFG4 = PWM 4 Output (0 = Open Drain; 1 = Push-Pull)

14.2 Programmable period 8-bit PWM

The PWM 4 channel can be programmed to provide a PWM output with variable pulse width and period. The PWM 4 has a 16-bit Prescaler, an 8-bit Counter, a Pulse Width Register, and a Period Register. The Pulse Width Register defines the PWM pulse width time, while the Period Register defines the period of the PWM. The input clock to the Prescaler is $f_{OSC}/2$. The PWM 4 channel is assigned to Port 4.7.

Figure 36. Programmable 4-channel PWM block diagram

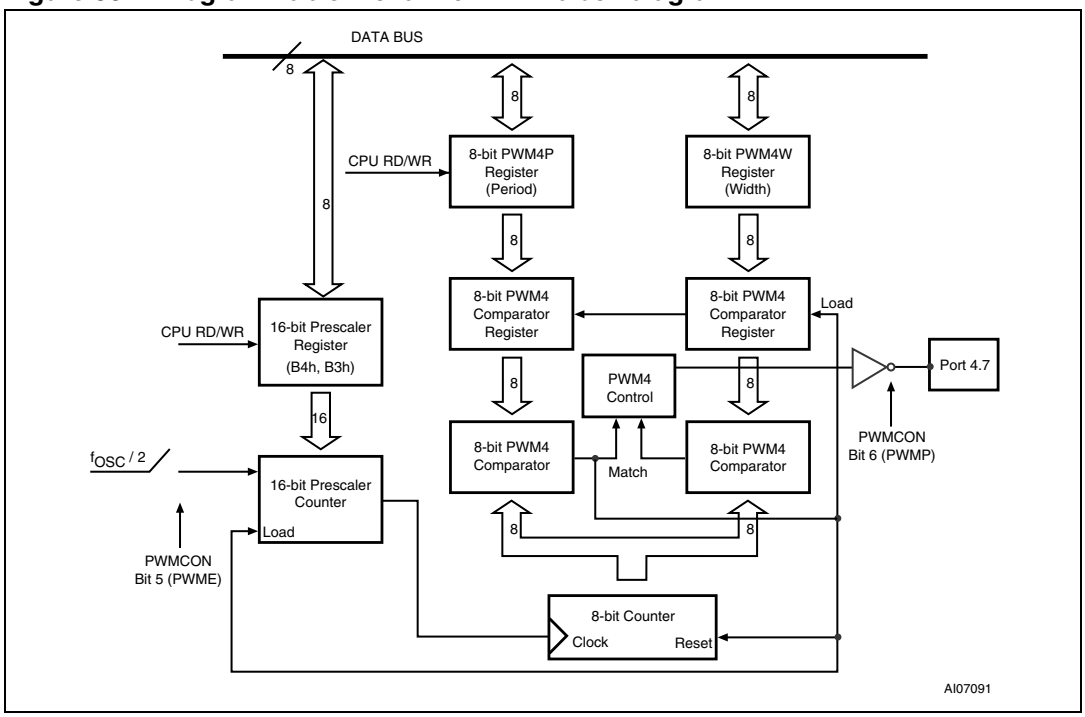


Table 50. Serial control register (SxCON: S1CON, S2CON)

7	6	5	4	3	2	1	0
CR2	ENII	STA	STO	ADDR	AA	CR1	CR0

Table 51. Description of the SxCON bits

Bit	Symbol	Function
7	CR2	This bit along with Bits CR1and CR0 determines the serial clock frequency when SIO is in the Master mode.
6	ENII	Enable IIC. When ENI1 = 0, the IIC is disabled. SDA and SCL outputs are in the high impedance state.
5	STA	START flag. When this bit is set, the SIO H/W checks the status of the I²C-bus and generates a START condition if the bus free. If the bus is busy, the SIO will generate a repeated START condition when this bit is set.
4	STO	STOP flag. With this bit set while in Master mode a STOP condition is generated. When a STOP condition is detected on the I²C-bus, the I²C hardware clears the STO flag. **Note:** This bit have to be set before 1 cycle interrupt period of STOP. That is, if this bit is set, STOP condition in Master mode is generated after 1 cycle interrupt period.
3	ADDR	This bit is set when address byte was received. Must be cleared by software.
2	AA	Acknowledge enable signal. If this bit is set, an acknowledge (low level to SDA) is returned during the acknowledge clock pulse on the SCL line when: • Own slave address is received • A data byte is received while the device is programmed to be a Master Receiver • A data byte is received while the device is a selected Slave Receiver. When this bit is reset, no acknowledge is returned. SIO release SDA line as high during the acknowledge clock pulse.
1	CR1	These two bits along with the CR2 Bit determine the serial clock frequency when SIO is in the Master mode.
0	CR0	

Table 52. Selection of the serial clock frequency SCL in Master mode

CR2	CR1	CR0	f_{OSC} divisor	Bit rate (kHz) at f_{OSC}			
				12 MHz	24 MHz	36 MHz	40 MHz
0	0	0	16	375	750	X	X
0	0	1	24	250	500	750	833
0	1	0	30	200	400	600	666
0	1	1	60	100	200	300	333
1	0	0	120	50	100	150	166
1	0	1	240	25	50	75	83

Table 64. Description of the UADR Bits

Bit	Symbol	R/W	Function
7	USBEN	R/W	USB Function Enable Bit. When USBEN is clear, the USB module will not respond to any tokens from host. $\overline{\text{RESET}}$ clears this bit.
6 to 0	UADD6 to UADD0	R/W	Specify the USB address of the device. $\overline{\text{RESET}}$ clears these bits.

Table 65. USB interrupt enable register (UIEN: 0E9h)

7	6	5	4	3	2	1	0
SUSPNDI	RSTE	RSTFIE	TXD0IE	RXD0IE	TXD1IE	EOPIE	RESUMI

Table 66. Description of the UIEN bits

Bit	Symbol	R/W	Function
7	SUSPNDI	R/W	Enable SUSPND Interrupt
6	RSTE	R/W	Enable USB Reset; also resets the CPU and PSD modules when bit is set to '1.'
5	RSTFIE	R/W	Enable RSTF (USB Bus Reset Flag) Interrupt
4	TXD0IE	R/W	Enable TXD0 Interrupt
3	RXD0IE	R/W	Enable RXD0 Interrupt
2	TXD1IE	R/W	Enable TXD1 Interrupt
1	EOPIE	R/W	Enable EOP Interrupt
0	RESUMI	R/W	Enable USB Resume Interrupt when it is the Suspend mode

Table 67. USB interrupt status register (UISTA: 0E8h)

7	6	5	4	3	2	1	0
SUSPND	—	RSTF	TXD0F	RXD0F	TXD1F	EOPF	RESUMF

Table 68. Description of the UISTA bits

Bit	Symbol	R/W	Function
7	SUSPND	R/W	USB Suspend Mode Flag. To save power, this bit should be set if a 3ms constant idle state is detected on USB bus. Setting this bit stops the clock to the USB and causes the USB module to enter Suspend mode. Software must clear this bit after the Resume flag (RESUMF) is set while this Resume Interrupt Flag is serviced
6	—	—	Reserved
5	RSTF	R	USB Reset Flag. This bit is set when a valid $\overline{\text{RESET}}$ signal state is detected on the D+ and D- lines. When the RSTE bit in the UIEN Register is set, this reset detection will also generate an internal reset signal to reset the CPU and other peripherals including the USB module.

Table 72. Description of the UCON1 bits

Bit	Symbol	R/W	Function
7	TSEQ1	R/W	Endpoint 1/ Endpoint 2 Transmit Data Packet PID. (0=DATA0, 1=DATA1) This bit determines which type of data packet (DATA0 or DATA1) will be sent during the next IN transaction directed to Endpoint 1 or Endpoint 2. Toggling of this bit must be controlled by software. $\overline{\text{RESET}}$ clears this bit.
6	EP12SEL	R/W	Endpoint 1/ Endpoint 2 Transmit Selection. (0=Endpoint 1, 1=Endpoint 2) This bit specifies whether the data inside the registers UDT1 are used for Endpoint 1 or Endpoint 2. If all the conditions for a successful Endpoint 2 USB response to a hosts IN token are satisfied (TXD1F=0, TX1E=1, STALL2=0, and EP2E=1) except that the EP12SEL Bit is configured for Endpoint 1, the USB responds with a NAK handshake packet. $\overline{\text{RESET}}$ clears this bit.
5	TX1E	R/W	Endpoint1 / Endpoint2 Transmit Enable. This bit enables a transmit to occur when the USB Host Controller send an IN token to Endpoint 1 or Endpoint 2. The appropriate endpoint enable bit, EP1E or EP2E Bit in the UCON2 register, should also be set. Software should set the TX1E Bit when data is ready to be transmitted. It must be cleared by software when no more data needs to be transmitted. If this bit is '0' or TXD1F is set, the USB will respond with a NAK handshake to any Endpoint 1 or Endpoint 2 directed IN token. $\overline{\text{RESET}}$ clears this bit.
4	FRESUM	R/W	Force Resume. This bit forces a resume state (“K” on non-idle state) on the USB data lines to initiate a remote wake-up. Software should control the timing of the forced resume to be between 10ms and 15ms. Setting this bit will not cause the RESUMF Bit to set.
3 to 0	TP1SIZ3 to TP1SIZ0	R/W	The number of transmit data bytes. These bits are cleared by $\overline{\text{RESET}}$.

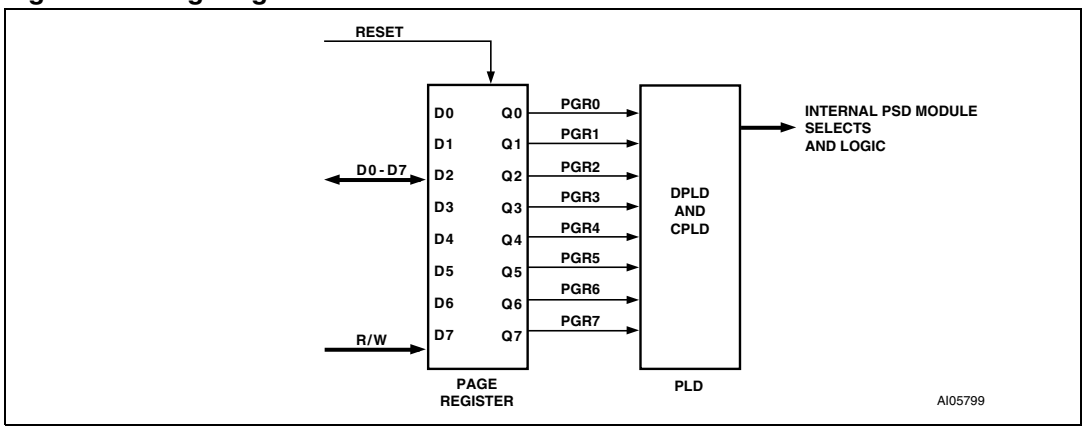
Table 73. USB control register (UCON2: 0ECh)

7	6	5	4	3	2	1	0
—	—	—	SOUT	EP2E	EP1E	STALL2	STALL1

Table 74. Description of the UCON2 bits

Bit	Symbol	R/W	Function
7 to 5	—	—	Reserved
4	SOUT	R/W	Status out is used to automatically respond to the OUT of a control READ transfer
3	EP2E	R/W	Endpoint2 enable. $\overline{\text{RESET}}$ clears this bit

Figure 56. Page register



AND Array, ORed with the Direction Register output. The pin is enabled upon Power-up if no output enable equation is defined and if the pin is declared as a PLD output in PSDsoft Express.

If the Output Macrocell (OMC) output is declared as an internal node and not as a port pin output in the PSDabel file, the port pin can be used for other I/O functions. The internal node feedback can be routed as an input to the AND Array.

23.6 Input macrocells (IMC)

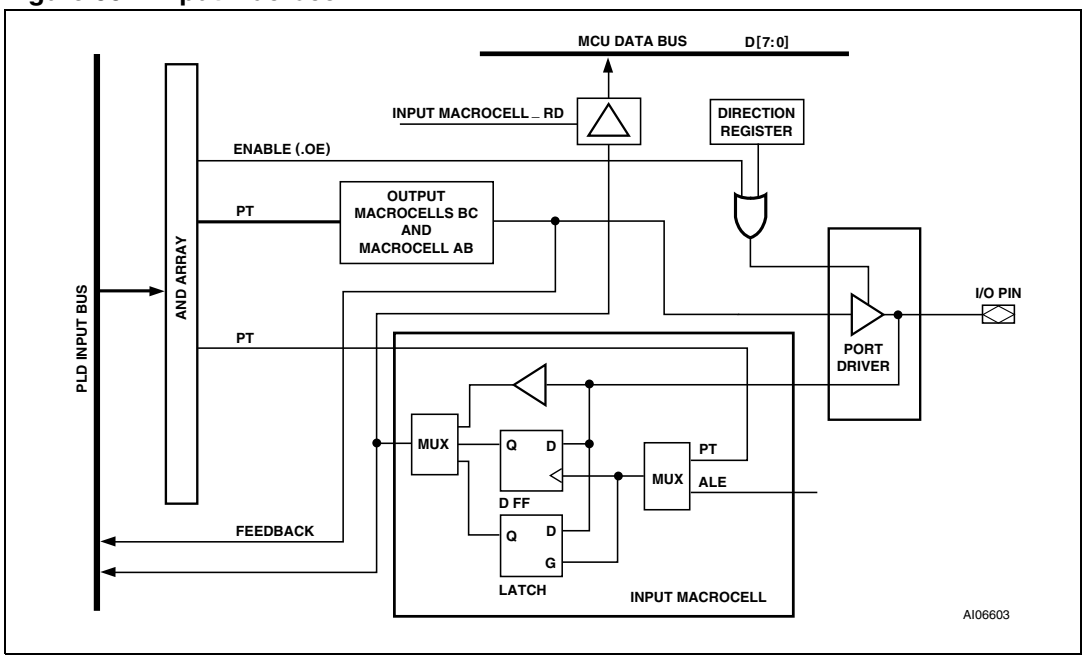
The CPLD has 20 Input Macrocells (IMC), one for each pin on Ports A and B, and 4 on Port C. The architecture of the Input Macrocells (IMC) is shown in *Figure 60*. The Input Macrocells (IMC) are individually configurable, and can be used as a latch, register, or to pass incoming Port signals prior to driving them onto the PLD input bus. The outputs of the Input Macrocells (IMC) can be read by the MCU through the internal data bus.

The enable for the latch and clock for the register are driven by a multiplexer whose inputs are a product term from the CPLD AND Array or the MCU Address Strobe (ALE). Each product term output is used to latch or clock four Input Macrocells (IMC). Port inputs 3-0 can be controlled by one product term and 7-4 by another.

Configurations for the Input Macrocells (IMC) are specified by equations written in PSDsoft (see Application Note *AN1171*). Outputs of the Input Macrocells (IMC) can be read by the MCU via the IMC buffer.

See *Section 24: I/O ports (PSD module)*.

Figure 60. Input macrocell



24.8 Port configuration registers (PCR)

Each Port has a set of Port Configuration Registers (PCR) used for configuration. The contents of the registers can be accessed by the MCU through normal READ/WRITE bus cycles at the addresses given in *Table 84*. The addresses in *Table 84* are the offsets in hexadecimal from the base of the CSIOP register.

The pins of a port are individually configurable and each bit in the register controls its respective pin. For example, Bit 0 in a register refers to Bit 0 of its port. The three Port Configuration Registers (PCR), shown in *Table 96*, are used for setting the Port configurations. The default Power-up state for each register in *Table 96* is 00h.

24.8.1 Control register

Any bit reset to '0' in the Control Register sets the corresponding port pin to MCU I/O mode, and a '1' sets it to Address Out mode. The default mode is MCU I/O. Only Ports A and B have an associated Control Register.

24.8.2 Direction register

The Direction Register, in conjunction with the output enable (except for Port D), controls the direction of data flow in the I/O Ports. Any bit set to '1' in the Direction Register causes the corresponding pin to be an output, and any bit set to '0' causes it to be an input. The default mode for all port pins is input.

Figure 63 and *Figure 64* show the Port Architecture diagrams for Ports A/B and C, respectively. The direction of data flow for Ports A, B, and C are controlled not only by the direction register, but also by the output enable product term from the PLD AND Array. If the output enable product term is not active, the Direction Register has sole control of a given pin's direction.

An example of a configuration for a Port with the three least significant bits set to output and the remainder set to input is shown in *Table 99*. Since Port D only contains two pins (shown in *Figure 66*), the Direction Register for Port D has only two bits active.

24.8.3 Drive Select register

The Drive Select Register configures the pin driver as Open Drain or CMOS for some port pins, and controls the slew rate for the other port pins. An external pull-up resistor should be used for pins configured as Open Drain.

A pin can be configured as Open Drain if its corresponding bit in the Drive Select Register is set to a '1.' The default pin drive is CMOS.

Note: The slew rate is a measurement of the rise and fall times of an output. A higher slew rate means a faster output response and may create more electrical noise. A pin operates in a high slew rate when the corresponding bit in the Drive Register is set to '1.' The default rate is slow slew.

Table 100 shows the Drive Register for Ports A, B, C, and D. It summarizes which pins can be configured as Open Drain outputs and which pins the slew rate can be set for.

Table 117. Major parameters (continued)

Parameters/conditions/ comments	5 V test conditions	5.0 V value	3.3 V test conditions	3.3 V value	Unit
Standby current, typical (Power-down mode, requires reset to exit mode; without Low-Voltage Detect (LVD) Supervisor)	180 µA with LVD	110	100 µA with LVD	60	µA
I/O sink/source current Ports A, B, C, and D	V_{OL} = 0.25 V (max); V_{OH} = 3.9 V (min)	I_{OL} = 8 (max); I_{OH} = –2 (min)	V_{OL} = 0.15 V (max); V_{OH} = 2.6 V (min)	I_{OL} = 4 (max); I_{OH} = –1 (min)	mA
PLD macrocells (For registered or combinatorial logic)	–	16	–	16	–
PLD inputs (Inputs from pins, macrocell feedback, or MCU addresses)	–	69	–	69	–
PLD outputs (Output to pins or internal feedback)	–	16	–	16	–
PLD propagation delay, typical (PLD input to output, Turbo mode)	–	15	–	22	ns

Table 131. CPLD macrocell asynchronous clock mode timing (5 V devices)

Symbol	Parameter	Conditions	Min	Max	PT aloc	Turbo off	Slew rate	Unit
f_{MAXA}	Maximum frequency external feedback	$1/(t_{SA}+t_{COA})$		38.4				MHz
	Maximum frequency internal feedback (f_{CNTA})	$1/(t_{SA}+t_{COA}-10)$		62.5				MHz
	Maximum frequency pipelined data	$1/(t_{CHA}+t_{CLA})$		71.4				MHz
t_{SA}	Input setup time		7		+ 2	+ 10		ns
t_{HA}	Input hold time		8					ns
t_{CHA}	Clock input high time		9			+ 10		ns
t_{CLA}	Clock input low time		9			+ 10		ns
t_{COA}	Clock to output delay			21		+ 10	– 2	ns
t_{ARDA}	CPLD array delay	Any macrocell		11	+ 2			ns
t_{MINA}	Minimum clock period	$1/f_{CNTA}$	16					ns

Table 132. CPLD macrocell asynchronous clock mode timing (3 V devices)

Symbol	Parameter	Conditions	Min.	Max.	PT aloc	Turbo off	Slew rate	Unit
f_{MAXA}	Maximum frequency external feedback	$1/(t_{SA}+t_{COA})$		21.7				MHz
	Maximum frequency internal feedback (f_{CNTA})	$1/(t_{SA}+t_{COA}-10)$		27.8				MHz
	Maximum frequency pipelined data	$1/(t_{CHA}+t_{CLA})$		33.3				MHz
t_{SA}	Input setup time		10		+ 4	+ 20		ns
t_{HA}	Input hold time		12					ns
t_{CHA}	Clock input high time		17			+ 20		ns
t_{CLA}	Clock input low time		13			+ 20		ns
t_{COA}	Clock to output delay			36		+ 20	– 6	ns
t_{ARD}	CPLD array delay	Any macrocell		25	+ 4			ns
t_{MINA}	Minimum clock period	$1/f_{CNTA}$	36					ns